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 appropriate recommendations to new users with the ability of CF to find peer users with unexpected shared preferences. This is appropriate for restaurants since a user may be in the mood for a specific cuisine and only want to see suggestions relevant to that, which is where the KB system comes in. The CF algorithm will then be able to sort these restaurants provided by the KB system by their predicted rating by the user, 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 $business.json$ file contains business data including location data, attributes, and cuisine. $review.json$ contains full review text data including the $user\_id$ of the review and the $business\_id$ that the review is for. The file $user.json$ holds every user's first name along with other data and metadata which I don't require [5]. The data files $checkin.json$, $tip.json$, and $photos.json$ are not used in my system. 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 [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. So I converted every file I used in the system into a typical JSON file, allowing for extremely fast data lookups in the system. I filtered the list of all Yelp businesses to restaurants and then further by selecting only those which were still open and fit into one or more of the top 12 cuisines in terms of number of restaurants because it's enough to provide a wide variety of options to the user without being so many that the user is overwhelmed. This left 20,565 restaurants which could be recommended to the user, which needed to be split into their respective metropolitan areas. The reviews were further filtered to only those which had 5 or more 'useful' votes by other users, to ensure I was only using high quality reviews which others found helpful. This left around 120,000 reviews, which were then converted to a CSV file ($reviews.csv$) 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 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. 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.

Measuring the accuracy of usage predictions is done using precision, calculated by:

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

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.

Coverage of the system is calculated by:

$$Prediction\ coverage = \frac{|I_p|}{|I|} \tag{3}$$

where I_p is the set of restaurants which can be recommended and I is the set of all restaurants. In my system the coverage is 0.32 ($\frac{20{,}565}{63{,}944}$), this is the most appropriate metric since the data for previously viewed items does not exist, ruling out novelty and diversity metrics.

Explainability is measured holistically, by looking at the explanations given throughout the system and how well they describe the purpose of the recommendations, how accurately they match the mechanisms used to generate recommendations, how much they improve the system's transparency, and how they help users to make decisions more efficiently and effectively. Generally the explainability of this system is quite high with regards to all four of these criteria.

III. Implementation

A. Input interface

The system has a command line interface, and recognises the active user by their unique user ID. Some example IDs are provided for testing and demonstration purposes. Only explicit user data is gathered, making 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 leave a rating, which will update their future recommendations.

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:

$$\widehat{r}_{ui} = \mu + b_u + b_i + q_i^T p_u \tag{4}$$

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

$$\sum_{r_{ui}\in R_{train}} (r_{ui} - \widehat{r}_{ui})^2 + \lambda(b_i^2 + b_u^2 + ||q_i||^2 + ||p_u||^2) \tag{5}$$

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 using an ASCII table. The top 8 recommendations are output since this is enough variety for the user to find something new to try, without being too overwhelming. Since the user will be interacting

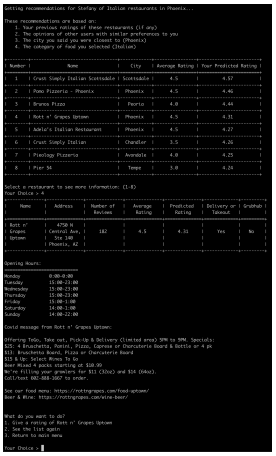

Fig. 1. System output

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 collects and uses, further adds to the explainability and transparency.

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, and quality of experience, in descending order of importance and 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

One ethical issue resulting from data collection is whether people who use Yelp are aware that some of their personal data is being made publicly available as part of a large dataset. An approach to solve this issue would be an opt-in for whether a user wants to allow their data to be used in this way. Another issue is a business which has received unfair negative reviews, my recommender system would be less likely to suggest this restaurant to a user, causing the owner to unfairly lose business. This can be solved by moderation of reviews and allowing business owners to appeal negative reviews. A third issue arising from this system is that users now don't need to visit the Yelp website to find restaurant recommendations, meaning that Yelp will lose ad-revenue due to reduced site-traffic. Yelp could address this issue by introducing additional useful features to their website which are not available in this system, such that users would want to visit Yelp.com for their recommendations.

V. Conclusion

A. Limitations

Currently the system cannot consider the user's preferences in terms of pricing of a restaurant, and it also does not have precise location information, only the metropolitan area the user is in or near to. This means that how central a restaurant is to the city centre currently has no impact on the recommendations.

B. Further developments

The system could be developed further by implementing the idea of particular cuisines being more similar to some than others. The user could also specify an acceptable price range that they are looking for, and the system could make recommendations based on estimated or known demographic information of the user.

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