A recommender system for hotels using Yelp data

1st gwwb64
Department of Computer Science
Durham University
Durham, United Kingdom

***Abstract*—This paper investigates the creation of a high-accuracy hybrid recommender system (RS) for generating recommendations of hotels and hotel bars from the Yelp dataset [1].**

I. Introduction

A. Domain & Aims

The domain of hotels and hotel bars was chosen since it is familiar meaning that some expert knowledge is available when determining which sort of features are important or which predictions may be relevant. Secondly, the dataset is primarily comprised of restaurants with hotels being a much smaller fraction. The limited dataset will make training models faster and will also help in memory-based methods due to lower memory requirements. However, this very advantage may also have a downside as less data potentially means less accurate modelling. Being forced to make an RS with both limited and sparse data makes the problem more interesting. The time-span chosen is as far back as possible since limiting the already small dataset further is unwise. Also, hotels do not change that regularly so reviews are likely to remain relevant for a long time. The final RS should be capable of predicting better than a baseline collaborative filter.

B. Related Work

Much research has been done on Collaborative Filtering (CF) where predictions are based on items rated by other users. Highly-rated items by similar users are likely to be relevant to the target user. CF approaches can be memory-based or model-based, but for large datasets, the best processing speeds can be achieved via model-based approaches such as Matrix Factorization [2]. While the dataset has many more users than items and item-based nearest-neighbour methods [3] are simple to implement, they are unsuitable for our RS due to their large memory requirements and slow processing speed. A common type of such Matrix Factorization (MF) model is known as "SVD". Items and users are mapped into a lower-dimensional latent space of fixed vector size. Predictions are made by taking the dot product between an item vector and a user vector. The preferred measure of accuracy for recommendation algorithms is Root Mean Square Error (RMSE) since it penalises large errors [2]. CF techniques can suffer from cold-start problems both for new users and new items [4]. For MF algorithms, this can be overcome by taking averages for items and users in the model. However, for new items especially, this production may be highly inaccurate, particularly when the new item is different to many existing items. While they tend to overspecialize, Content-Based Filtering (CBF) systems are not susceptible to the first-rater problem [5] and have the added benefit of being easy to explain for non-specialist users. CBFs often use normalisation techniques or TF-IDF to ensure documents are treated fairly [5]. TF-IDF is not useful for our work since our features will be binary values for a fixed set of features. Ideally an RS should maximise its advantages and reduce the likelihood that it will produce wrong predictions when edge cases are encountered. A hybrid system can combine multiple RS with different combination strategies such as weighting, mixing or switching [6]. Determining weights for a weighted hybrid may be done by empirical study of a particular dataset. Switching is done with a confidence metric of some sort but choosing a valid metric is an active area of research. Some papers [7] have the RS calculate its own confidence in its predictions as the confidence metric. [8] shows a way to measure the novelty of items in an RS.

II. Methods

A. Data description, preparation & feature selection

The data are organised into 6 files with names of the form *yelp_academic_dataset_⟨file_for_dataset⟩.json* where ⟨file_for_dataset⟩ is one of *business*, *checkin*, *covid_features*, *review*, *tip* and *user*. Each file contains data about the respective element organised as a list of json objects where one row in the dataset is represented by one line in the file (as opposed to an array of objects). The first step is to filter all the output data into files of the same name as the originals but in a folder called *bnb_filtered_data*. The business data (item) is first filtered using the categories provided by the establishment owner. If the item categories include *Bed & Breakfast*, *Hotels* or *Hotel bar*, the corresponding item will be saved in the filtered folder. If not, it will be discarded as we are not interested in other establishments. For each of the other files, for every file that is linked to an item, e.g. reviews, any data elements which do not have a *business_id* in the filtered business list will be discarded before being saved to the filter folder corresponding file. Any time a data point has a *user_id*, it denotes that this data point corresponds to an interaction with a business by the user with that id. A list of users that did interact is maintained and the users that had no interactions are discarded. The filtered files are created mostly to speed up data processing so gigabytes of data do not need to be loaded into memory to run the RS. The main data points of interest were the reviews each user gave businesses, particularly the *stars* field which is a rating between 1 and 5. This was used by both our CF and CBF. In addition, the *attributes* field in each business was used. Unfortunately, these attributes are not well standardised, containing JSON strings with fields which can be nested JSON strings, string True/False or other string values. These must be parsed into an attribute/item matrix for CBF. While these are owner-defined features so may not always be reliable, it is in the owner's best interests to ensure these are accurate at all times otherwise user ratings may fall.

B. Hybrid Scheme

Two separate hybrid schemes were trialled in this experiment, with a hybrid consisting of a CBF and CF. The first was *weighting* and the second was *switching*. Since the weighting scheme was simpler, this was tried first with equal weights assigned to both the CBF and CF. The poor RMSE performance of this led to attempts at tweaking the weights with little to no (or negative) improvement. As such, the weighting technique was abandoned in the final implementation but has been kept for comparison with weighting 0.25 for the CBF and 0.75 for the CF. The switching technique was implemented using the same idea as in [7] where the recommendations are used to determine confidence. In our system, predicted values for all businesses for the active user were compared to the ground truth where it was available via RMSE. A 95% confidence interval was then calculated for this error on both the CBF and CF using equation:

$$\gamma_f = 1.96 * \sqrt{\left|\frac{\epsilon_f * (1-\epsilon_f)}{N_f}\right|} \tag{1}$$

where ϵ_f is the RMSE of filter f, N_f is the number of items the user has reviewed and f is one of the CBF or CF. The RS with the lowest γ value was chosen as the sole RS for the output of the hybrid. Division by zero is not an issue here since only users with at least one recommendation for a business end up in the training data.

C. Recommendation Algorithms

The first RS system is a CF using SVD. As mentioned in I-B, latent spaces for users and items must be generated. Initially, these latent spaces are assigned small float values and then stochastic gradient descent is performed to minimise the error by iterating over all ratings for many epochs. At the same time, biases for users and items are also calculated. The equations from [2] are as follows:

$$\begin{aligned}
\hat{r}_{ui} &= \mu + b_i + b_u + q_i \cdot p_u \\
e_{ui} &= r_{ui} - \hat{r}_{ui} \\
b_u &= b_u + \gamma \cdot (e_{ui} - \lambda \cdot b_u) \\
b_i &= b_i + \gamma \cdot (e_{ui} - \lambda \cdot b_i) \\
q_i &= q_i + \gamma \cdot (e_{ui} \cdot p_u - \lambda \cdot q_i) \\
p_u &= p_u + \gamma \cdot (e_{ui} \cdot q_i - \lambda \cdot p_u)
\end{aligned} \tag{2}$$

where μ is the mean of all items, γ is the learning rate and λ is the regularization to apply. In our *recommender.py* file, a couple of sample implementations were created. *train_svd* uses the full equations and *train_svd_u* has no biases. The final CF uses SVD in the *scikit-surprise* Python module [9] which supports k-fold crossover with training and test sets as well as a Grid Search which allows for turning of hyperparameters γ and λ as well as the number of epochs to use for training and the number of latent factors. Our final model uses 40 factors over 50 epochs with $\gamma = 0.005$ and $\lambda = 0.05$. These may not be ideal parameters but running a Grid Search for 100s of parameter options in total is not viable.

The second RS is a simple CBF [5]. A user profile is constructed by taking the dot product of every business for a certain feature with the user's business ratings. Ratings can then predicted for a given item by taking the dot product of all the item's features with the user profile.

D. Evaluation Methods

Each RS and the hybrid had RMSE calculated to determine the accuracy of ratings compared to ground truth. The precision, recall and F1 score were calculated, with a threshold of 3.5 rating denoting relevancy, to show how relevant predicted item are. The novelty of each RS was calculated since this helps ensure a discovery process for items which is good for businesses. All calculated values were averaged across the test set. Testing across the entire dataset may lead to over-fitting and additionally takes far too long with our implementation. However, with SVD CFs, you cannot really have a test set since the model must be trained for all users/items with ratings to obtain latent factors. To balance the time it would take and the difficulty in constructing test sets, our tests were performed using a seeded random sample of 1000 elements. Our implementation of precision and recall is based on reference implementations in the Surprise FAQ [9] though it was adapted for Numpy and our custom grading implementation.

III. Implementation

A. Input interface

The input interface is a simple CLI. The user is able to login by typing their integer user id after a prompt that lists available user ids. This would not be suitable in a production environment, but for this experiment, it is fine. The ability to 'login' as multiple users allows for immediate comparisons to be made between recommendations. Once logged in, the user is presented with a list of states with hotels in the dataset. The user can enter a state to choose where they would like to travel to.

B. Recommendation Algorithm

Upon state entry, the system starts by generating the business attribute matrix and reading all ratings the user has given to businesses. Businesses, attributes and ratings are passed to the CBF. Importantly, our CBF implementation uses 2 normalisation steps. Firstly, attributes are normalised within the business/attribute matrix which improves the quality of predictions by preventing some businesses with lots of attributes having large weights. The second normalisation step is an interpolation after predicting ratings. This is just to rescale ratings between 1 and 5 to make comparison easier and to allow easy use of the weighted hybrid technique (despite it not making the final implementation). The CF is passed the model loaded from file if it exists (or it is created if not), the internal user id and all businesses to generate for. Both RS will predict ratings for all businesses. These are passed into the switching hybrid which calculates the prior confidence interval for the predictions for the user and then picks which RS values to use. A filtering step is then performed to remove any recommendations which are currently closed due to covid-19 by setting their rating to -1 if the closure end time is beyond the current time. The top 10 predictions are then chosen with filtering by state. This acts as a context-aware [10] RS step. If 10 predictions cannot be found for the state (e.g. NY), the remaining recommendations will be filled with other items the user may like.

C. *Output interface*

The output is simply a list of the 10 items the user may be interested in. If there are fewer than 10 in the requested state, the available ones will be displayed first followed by further separate recommendations that the user may also like. The output also contains a brief explanation regarding the method for generating the list. The list is technically in order with more relevant items being shown higher in the list. The output doesn't mention this, nor provide a corresponding predicted score since scores for recommendations are usually near the maximum anyway and this information does nothing for the user experience and may overwhelm the user with text.

IV. Evaluation

Our switched hybrid RS system performed better than the baseline CF by around 14% in the RMSE metric. Precision, recall and the F1 average of these were slightly improved in the hybrid RS. The weighted hybrid performed worse in RMSE than CF hence its abandonment. Our hybrid saw a large reduction in the range of values generated by the 95% confidence interval, hence greater certainty on predictions. The F1 score being around 65% shows that the model performs adequately for providing relevant recommendations. The hybrid greatly improves the novelty compared to CF which is good for the hotel domain, especially during peak demand times where popular hotels may already be booked. This may also be good from an ethical standpoint because more in-demand hotels will likely be more expensive so the RS does not attempt to force users into spending more money (especially in cases where the platform makes money via commission). The switching hybrid in [6] did not perform well, however, the switching method in theirs ordered the RS and only fell to a lower if sufficient confidence could not be established. Ours always chooses the most confident prediction which tends to be the CF based on experimentation with the CBF only used in a fraction of cases. Algorithm opacity is a potential ethical issue. This system tries to counteract this by providing an explanation to the user. Its use of a CBF makes explanations easier but the exact decisions the hybrid takes are not easily explainable. Leaking private

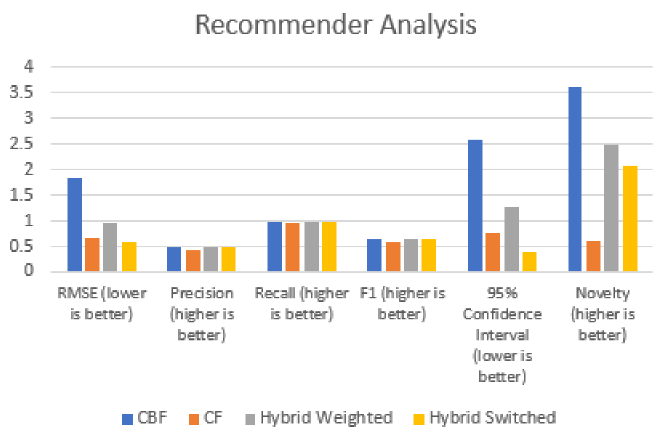

Fig. 1. Bar chart showing comparison between RS systems for different metrics

data in a breach is always a risk. Luckily, the dataset used was anonymised. Our implementation does not collect any further data on users. However, de-anonymisation is an active area of research and especially via the friends lists, there may be some way to reconstruct user IDs via social graphs.

V. Conclusion

Allowing the user to choose a location is a key part of travel RS. Unfortunately, it was not possible to directly grade the final output shown to the user for a given state due to some states having so few rated hotels. Metrics would have been nonsense since filtering by state introduces large bias. An improved system would use more data for states such as NY to accurately measure the entire RS in grading, including post-filtering. The RS could also be expanded by using the user's friends list to recommend items friends have liked.

References

[1] (2021) Yelp dataset. [Online]. Available: https://www.yelp.com/dataset
[2] Y. Koren and R. Bell, *Advances in Collaborative Filtering*. Boston, MA: Springer US, 2011, pp. 145–186. [Online]. Available: https://doi.org/10.1007/978-0-387-85820-3_5
[3] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, "Item-based collaborative filtering recommendation algorithms," *Proceedings of ACM World Wide Web Conference*, vol. 1, 08 2001.
[4] B. Schafer, D. Frankowski, J. Herlocker, and S. Sen, "Collaborative filtering recommender systems," 01 2007.
[5] P. Lops, M. de Gemmis, and G. Semeraro, *Content-based Recommender Systems: State of the Art and Trends*, 01 2011, pp. 73–105.
[6] R. Burke and Robin, "Hybrid web recommender systems," vol. 4321, 01 2007.
[7] L. Ardissono, D. Petrelli, and K. Kuflik, "Personalization in cultural heritage: The road travelled and the one ahead," *User Modeling and User-Adapted Interaction*, vol. 22, pp. 73–99, 05 2012.
[8] M. Kaminskas and D. Bridge, "Diversity, serendipity, novelty, and coverage: A survey and empirical analysis of beyond-accuracy objectives in recommender systems," *ACM Trans. Interact. Intell. Syst.*, vol. 7, no. 1, Dec. 2016. [Online]. Available: https://doi.org/10.1145/2926720
[9] (2021) Scikit surprise. [Online]. Available: http://surpriselib.com/
[10] G. Adomavicius and A. Tuzhilin, *Context-Aware Recommender Systems*. Boston, MA: Springer US, 2011, pp. 217–253. [Online]. Available: https://doi.org/10.1007/978-0-387-85820-3_7