Cascading Hybrid Recommender System

*

hvts63

I. Introduction

A. Domain

This system was built to recommend restaurants to users looking to try new places to eat. Ideally, the user would be using a desktop or laptop to be able to easily type inputs into the system.

B. Related Work Review

Hybrid recommenders [1], [2] enable us to combine different types of recommendation algorithms and allows to combine their benefits when generating and scoring candidates. As a result hybrid approaches can often outperform traditional algorithms on evaluation metrics.

Deep learning approaches have been successfully applied to recommendation applications [3], [4]. However, these recommenders are often black box methods and offer little explainability. Recently, more explainable models have been developed such as the Explainable Restricted Boltzmann Machine [5] and the Explainable Collaborative Autoencoder [6]. These models are able to match the performance of their black box counterparts whilst scoring much higher on explainability metrics.

C. Purpose

This recommender system is designed to suggest new restaurants to users based on restaurants they have previously liked and restaurants that their friends have liked. In addition, the system should be able to allow users to update their preferences by rating restaurants from their recommendations that they subsequently try. As this system was built during the Covid-19 pandemic, it was also required to offer users recommendations only for restaurants that are offering takeout delivery services or in partnership with Grubhub to deliver food.

II. Methods

A. Data Description

The recommender was trained and evaluated on the Yelp Academic Dataset [8]. From this dataset, I used the review, business, covid and user data for feature selection and discarded the tips and checkin files.

B. Data Preparation and Feature Extraction

We first merged the business and covid files to get restaurants offering delivery or Grubhub services due to Covid-19. All non-restaurant businesses and closed restaurants were removed from the data. We chose a timeframe from 1st December 2019 and removed all restaurants not reviewed after this date and all users that were inactive from this data. All reviews were aggregated by restaurant and removed all punctuation. We then applied snowball stemming to the review text. We only considered users with more than 20 total reviews who had rated at least one restaurant in our business subset. Any user with no friends in the dataset was then removed. The final dataset consisted of 3088 users and 4055 restaurants. A user-item matrix was computed from the reviews and this matrix was normalised, dividing all entries by the maximum rating. We also computed an additional explainability vector for each user in this matrix. This was calculated based on the equation used in [5]. However, instead of using the k nearest neighbours to a user to calculate the explainability scores, we decided to use the user's friends' ratings. For each user, we used the following formula to calculate the explainability for each restaurant in the resulting dataset:

$$e_i = \frac{\sum_{u \in F} r_{ui}}{|F| R_{max}} \tag{1}$$

In this equation, F is the set of the user's friends, r_{ui} is the friend's rating for item i and R_{max} is the maximum rating for that item amongst all friends in F.

C. Hybrid Scheme

We use a cascade hybrid scheme. For this technique, we use a primary recommender to generate the candidates. This recommender then scores the candidates. In addition, we then use a secondary recommender to further rank these candidates by generating scores and using the secondary scores as a tie breaker for the initial ranking. We chose to use content based filtering and collaborative filtering for our recommenders. This allows us to minimise the cold start problem compared to a collaborative filtering scheme as our dataset is very sparse. However, we can then use a collaborative scheme which are often more accurate to generate scores in the event of a tie from the first recommender to produce a more accurate ranking.

D. Recommendation techniques

For our content based recommender, we compute the TF-IDF vectors for each restaurant using the aggregated reviews for each restaurant. We then compute a user profile vector for the user by summing the product of their rating and the TF-IDF vector for each restaurant they have rated. We then compute the cosine similarity between the user profile vector for each item vector and return the k nearest neighbours as our candidates with their cosine similarity as the score. As the semantic content of all restaurant reviews are likely to be quite similar, I decided on using TF-IDF (which does not capture the text's semantics) over a word2vec model as it is faster to compute. Cosine similarity was also chosen as it is fast to compute and works well with sparse data.

The collaborative recommender is based on the explainable autoencoder model with a single hidden layer of 50 neurons. For each user, the model takes as input a vector of the user's ratings for each item and an explainability vector. For each restaurant, its explainability value is calculated similar to the method used for the explainable RBM [5]. However, the explainability for our recommender is calculated using the user's friends in the dataset rather than finding the k most similar other users. The dataset was split into a training set and a test set. The autoencoder has a hidden layer of 50 neurons and was trained for 100 epochs on the training set.

E. Evaluation Methods

I evaluated the accuracy of rating predictions using the root mean squared error (RMSE), which is calculated as follows:

$$RMSE = \sqrt{\frac{1}{\tau} \sum_{(u,i) \in \tau} (\hat{r_{ui}} - r_{ui})^2} \tag{2}$$

where τ is the set of test users, r_{ui} is the user u's rating for item i and $\hat{r_{ui}}$ is the predicted rating. We chose this metric as the scores generated by both the autoencoder and the hybrid should be between 0 and 1 and so any value generated outside of this range should be penalised heavily.

The accuracy of usage predictions was measured using precision and recall. We decided that any normalised rating greater than 0.6 would be a positive rating otherwise it would be negative. The precision was then calculated according to the following formula:

$$precision = \frac{TP}{TP + FP} \tag{3}$$

where TP are the true positives, FP are the false positives. A high precision means that more of the recommendations made to the user will be restaurants that they like. This is important as only up to 20 restaurants are presented to the user so any false positives will damage the user experience.

Coverage was measured using prediction coverage. This is defined by the equation:

$$Prediction\ coverage = \frac{|I_p|}{|I|} \tag{4}$$

where I is the set of all items in the dataset and I_p is the set of all items which generated a prediction in the dataset. Users should be able to view any restaurants that match their interests and so measuring prediction coverage will enable is to detect whether the system can offer a diverse range of restaurants across all users whilst still being able to offer a narrow range of restaurants for a specific user's tastes.

Explainability was measured using Mean Explainability Precision (MEP) [7], which is the ratio of explainable recommended items to all items recommended and can be calculated as follows:

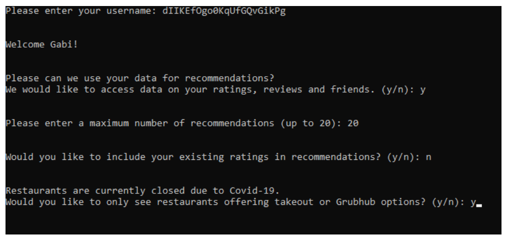
$$MEP = \frac{1}{\tau} \sum_{u \in \tau} \frac{|I_{exp} \cap I_{rec}|}{I_{rec}} \tag{5}$$

Users should be able to understand why they were recommended a restaurant and MEP is an easily interpretable measure of explainability.

III. Implementation

A. Input Interface

User input is accepted through the command line. After an initial login, the user can answer most questions using 'y' (yes) or 'n' (no) with the exception needing to provide a numerical value for ratings and the maximum number of recommendations (figure 1).



```
Please enter your username: dIIKEfOgo0KqUfGQvGikPg


Welcome Gabi!


Please can we use your data for recommendations?
We would like to access data on your ratings, reviews and friends. (y/n): y


Please enter a maximum number of recommendations (up to 20): 20


Would you like to include your existing ratings in recommendations? (y/n): n


Restaurants are currently closed due to Covid-19.
Would you like to only see restaurants offering takeout or Grubhub options? (y/n): y
```

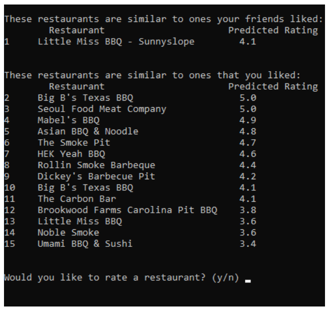
Fig. 1. Interface for user input

B. Recommendation Algorithm

The hybrid recommender is comprised of two algorithms: a k nearest neighbour content based recommender and an explainable collaborative autoencoder. The content based recommender generates k candidates and scores them. A vector of the users rating is then input into the autoencoder along with its explainability vector. Any recommendations that received the same score (rounded to 0.1) then use their scores from the autoencoder when output to the terminal. Only recommendations with a normalised score above 0.6 are shown to the user and so the final number of recommendations is often less than k. If the explainability score is greater than 0, we add the restaurant to a list of restaurants liked by the user's friends. All other restaurants are added to a list of restaurants similar to ones the user has previously liked.

C. Output Interface

The system requires a user to login with their user id. This can be any id from the user_id field of the users_covid_data file. Upon login, the system asks the user to consent to their data being used for recommendations. A user can then choose whether they want to include restaurants they have already rated, whether to only include restaurants offering delivery or Grubhub services during Covid-19 and the maximum number of recommendations (k for the k nearest neighbours algorithm). The list of recommendations is generated and ranked according to predicted score. The recommendations are then split into a list of restaurants that the user's friends liked and a list of restaurants similar to ones that the user has already liked (figure 2). The user can then add an actual rating to any restaurant in either list and the system will then calculate the new recommendations.



```
These restaurants are similar to ones your friends liked:
        Restaurant                            Predicted Rating
1     Little Miss BBQ - Sunnyslope              4.1


These restaurants are similar to ones that you liked:
        Restaurant                            Predicted Rating
2     Big B's Texas BBQ                         5.0
3     Seoul Food Meat Company                   5.0
4     Mabel's BBQ                               4.9
5     Asian BBQ & Noodle                        4.8
6     The Smoke Pit                             4.7
7     HEK Yeah BBQ                              4.6
8     Rollin Smoke Barbeque                     4.4
9     Dickey's Barbecue Pit                     4.2
10    Big B's Texas BBQ                         4.1
11    The Carbon Bar                            4.1
12    Brookwood Farms Carolina Pit BBQ          3.8
13    Little Miss BBQ                           3.6
14    Noble Smoke                               3.6
15    Umami BBQ & Sushi                         3.4


Would you like to rate a restaurant? (y/n) _
```

Fig. 2. An output set of recommendations for a user

IV. Evaluation Results

TABLE I
Evaluation results for hybrid recommender against our explainable autoencoder baseline

Metrics	**Hybrid**	**Autoencoder**
RMSE	0.250	0.703
Precision	0.6	0.005
Prediction Coverage	0.61	0.76
MEP	0.01	0.005

The hybrid model and autoencoder were evaluated offline on a test set of users consisting of 30% of the users from the original dataset. For the nearest neighbour algorithm in our hybrid, we set k to 20. The lower score for prediction coverage four our hybrid recommender is due to the smaller coverage of the content based recommender (which is used to generate the candidates) compared to the autoencoder. However, the cascading scheme significantly improved the accuracy of rating predictions and precision compared to our baseline. [2] proposes a group hybrid recommender that also combines content-based and collaborative filtering. When evaluated for recommendations for single users, this approach performed worse than our hybrid method and was only able to achieve a precision of 0.4297. However for group sizes between 2 and 4, this approach was able to achieve precision scores above 0.8 which is significantly higher than our score.

A. Ethical Issues

Whilst we ask the user for permission to use their data for generating recommendations, we do not necessarily have the permission of the user's friends to use their data. A potential solution would be to only use recommendations from the content based recommender if not all of the user's friends have given their consent for their data to be used. As the system uses a deep learning method, it is likely to be subject to bias in the dataset. For example, the system may rate vegetarian restaurants lower as people often rate restaurants which serve meat. To mitigate this, we could offer user feedback on recommendations and provide more detailed explanations for each recommendation. Another potential issue with the recommender system is that it tends to recommend a lot of restaurants of the same type (e.g. steak restaurants) as these are what the user has rated however, a use may be looking for a more diverse range of suggestions. This could lead to user behaviour manipulation as the recommender system reinforces the user's taste in restaurants. A solution would be to offer the user controls over how diverse they want their recommendations to be.

V. Conclusion

Our hybrid recommender was able to outperform the explainable autoencoder in all metrics except prediction coverage however.

A. Limitations

In practice, the sparsity of the dataset and the fact that most users only have a small number of friends meant that 99% of recommendations made by the hybrid model were unexplainable. A better approach might have been to use the approach of the explainable RBM and consider the k most similar users for each user with sufficiently large k. Using this method would also have allowed us to train our autoencoder on a much larger dataset as we would not need to discard users with no friends in the dataset.

B. Further Developments

To improve our recommender, we could consider additional contextual information for making recommendations such as only suggesting restaurants whose opening hours correspond to the time at which a user requests recommendations.

References

[1] . P. Seitlinger, D. Kowald, S. Kopeinik, I. Hasani-Mavriqi, E. Lex, and T. Ley, "Attention please! a hybrid resource recommender mimicking attention-interpretation dynamics," in Proc. of WWW'15. International World Wide Web Conferences Steering Committee, 2015, pp. 339–345.

[2] Kaššák, O., Kompan, M., Bieliková, M.: Personalized hybrid recommendation for group of users: top-N multimedia recommender. Inf. Process. Manag. 52(3), 459–477 (2016)

[3] H.-T. Cheng, L. Koc, J. Harmsen, T. Shaked, T. Chandra, H. Aradhye, G. Anderson, G. Corrado, W. Chai, M. Ispir, et al. Wide & deep learning for recommender systems. arXiv preprint arXiv:1606.07792, 2016.

[4] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural collaborative filtering. In Proceedings of the WWW. 173–182. https://doi.org/10.1145/3038912.3052569

[5] Behnoush Abdollahi and Olfa Nasraoui. 2016. Explainable Restricted Boltzmann Machines for Collaborative Filtering. 2016 ICML Workshop on Human

[6] Pegah Sagheb Haghighi, Olurotimi Seton, and Olfa Nasraoui. 2020. An Explainable Autoencoder For Collaborative Filtering Recommendation. ArXiv abs/2001.04344 (2020). Interpretability in Machine Learning (WHI 2016) Whi (2016). arXiv:1606.07129

[7] Behnoush Abdollahi and Olfa Nasraoui. 2016. Explainable Matrix Factorization for Collaborative Filtering. WWW (Companion Volume) (2016), 5–6. https: //doi.org/10.1145/2872518.2889405

[8] https://www.yelp.com/dataset