Recommender Systems Report

L3 Computer Science
Durham University
Durham, UK

I. Introduction

A. Domain of Application

The domain of this system is the hospitality industry, specifically restaurants. These restaurants all received reviews in the 2019 year.

B. Related Work & Aim

More than half of the recommendation approaches in literature (up to 2015) applied content-based filtering (55%), with a mere 18% applying collaborative filtering and 16% applying graph-based approaches. The last 11% consists of more niche approaches, including hybrid recommenders [1]. The intergration of multiple recommendation schemes to produce a single hybrid system was first introduced by Burke [2] in 1999, where he suggested an approach to combining both knowledge-based filtering and collaborative filtering to create a single model. This initial discussion of combining multiple schemes lead to the development of several hybrid architectures. Burke developed his own hybrid recommender system, EntreeC, that was used to recommend restaurants [3].

The systems aim is to accurately predict how users in the YELP dataset [5] would rate unvisited restaurants based of previous reviews, in order to be able to provide recommendations for them. Users should be able to explicitly select certain COVID regulations they want their recommendations to comply to.

II. Methods

A. Data description

The system uses the Yelp Dataset to create recommendations - more specifically it uses 3 of the 7 files provided in the Dataset. The first of these 3 files is the business.json, which contains a collection of business attributes, categories and locations. The second file it uses is the reviews.json, which contains information on how users rated businesses. The final file it uses is the COVID-19 data, which contains the different measures businesses have taken to ensure functionality and the safety of their customers during the COVID-19 pandemic.

B. Data preparation and feature selection

We begin by externally preprocessing the reviews.json dataset. The text in the review data is discarded, keeping only the star rating and date. Doing so leads to a huge reduction in file size, which leads to significant increases in data-loading times.

The chosen business category for my system is Restaurants, therefore we filter the datasets based on this business category. We then select reviews only made in 2019.

As one of the chosen algorithmic techniques is Singular Value Decomposition (SVD), the data is filtered based on the number of reviews, whereby users with a combined total of less than 15 reviews are removed from the dataset. Whilst the data will still be extremely sparse, it will better allow the SVD algorithm to identify latent features which it can categorise data into [4].

The second algorithmic technique involves the system comparing restaurants using cosine similarity, therefore we select the features that we want to compare. We selected 38 different features from the categories section of the business.json, with each feature encompassing characteristics that would be good for comparison. These features described the restaurants cuisine, e.g. Italian, American, etc. amongst other things such as if the serve alcohol.

C. Hybrid scheme

The system consists of 2 separate recommender systems, one which uses collaborative filtering, and the other which uses content-based filtering. We combine the two using a weighted scheme, whereby both recommenders produce scores for each restaurant, which are then combined together to produce overall scores.

D. Recommendation techniques/algorithms

In the collaborative filtering recommender, we use the singular value decomposition (SVD) algorithm proposed by Simon Funk in the Netflix Prize [6]. This is a model-based approach to collaborative filtering that incorporates matrix factorization and stochastic gradient descent. It is known to be effective on large datasets, significantly reducing the problems posed by a cold-start. The intuition behind this algorithm is to transform both users and items into a single k-dimensional latent feature space. By doing so, we can infer user preferences and consequently predict user-item ratings based on these k latent features. We can model users and items as vectors, where:

$$\begin{aligned} \text{user } u &\rightarrow p_u \in \mathbb{R}^k \\ \text{item } i &\rightarrow q_i \in \mathbb{R}^k \end{aligned} \tag{1}$$

With these vectors, we can formularise an equation to make user-item predictions, $\hat{r}_{ui}$, ensuring we capture divergences of user u and item i from the average by adding biases,

$$\hat{r}_{ui} = \mu + b_u + b_i + q_i^T p_u \tag{2}$$

The parameter μ represents an overall average rating, and parameters b_u and b_i both represent our biases. We can now treat this as an optimization problem, where we tune these parameters in order to minimize the regularised squared error [2] in a given test set, i.e. minimise the difference in predicted user-item ratings from actual user-item ratings. This can be achieved through stochastic gradient descent. Let

$$e_{ui} = r_{ui} - \hat{r}_{ui} \tag{3}$$

indicate the difference in actual and predicted ratings. Stochastic gradient descent performs as such:

$$\begin{aligned} b_u &\leftarrow b_u + \gamma(e_{ui} - \lambda b_u) \\ b_i &\leftarrow b_i + \gamma(e_{ui} - \lambda b_i) \\ p_u &\leftarrow p_u + \gamma(e_{ui} \cdot q_i - \lambda p_u) \\ q_i &\leftarrow q_i + \gamma(e_{ui} \cdot p_u - \lambda q_i) \end{aligned} \tag{4}$$

where γ is the learning rate and λ the regularization term. We can predict a users interest in item features,

$$q_i^T p_u = q_i^T \cdot p_u \tag{5}$$

This represents lower k-rank approximations of vectors contained in our original matrix, which allow us to identify user-item predictions, $q_i^T p_u$ that give high scores. We can then proceed to recommend these high scoring items i to our user u.

For the content-based filtering recommender, we model our corpus, i.e. our set of restaurants, as a vector space model, where the dimensionality of our vector space model is determined by our pre-selected dictionary of restaurant features. These features are derived directly from the restaurant attributes feature provided in the YELP dataset, and mainly consist of restaurant genres, such as Italian, Indian, etc. This, in turn, provides us with a matrix R such that, $R_{ij} = 1$ if restaurant j contains feature i, else $R_{ij} = 0$

We then apply K-Nearest-Neighbour to our vector space, measuring the cosine similarity of all restaurant vectors,

$$\frac{A \cdot B}{||A|| \times ||B||} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}} \tag{6}$$

These similarity scores are mapped to a new item-item matrix, containing the cosine similarity of each restaurant to each-other. We can now select items to recommend to the user based on the correlation between the attributes [7] of their previous positive rated restaurant reviews and the attributes of restaurants they have not yet visited. Unvisited restaurants with the highest cosine similarity rating are recommended to the user, as we can think of these restaurants as being the 'nearest neighbour'.

E. Evaluation methods

The system underwent a series of offline experiments. These were conducted on a random user (index 500), who had reviewed a total of 16 restaurants. The first of these experiments was to evaluate the accuracy of ratings predictions by calculating the Root Mean Squared Error (RMSE),

$$\text{RMSE} = \sqrt{\frac{1}{|\tau|} \sum_{(u,i) \in \tau} (\hat{r}_{ui} - r_{ui})^2} \tag{7}$$

The next experiment involved investigating the accuracy of usage predictions. This was done by exploring whether the system would recommend restaurants the user had already reviewed, had the system not been aware that they had previously been. This was modelled as a confusion matrix, and the precision is calculated as

$$\frac{TP}{TP + FP} \tag{8}$$

where TP is an acronym for true positive, FP for false positive. We then proceeded to calculate the novelty of the items recommended, using the equation,

$$\text{Novelty}(R) = \frac{\sum_{i \in R} -\log_2 p(i)}{|R|} \tag{9}$$

where $p(i)$ represents the proportion of users who rated the restaurant i and $|R|$ the number of recommendations. The results of these experiments will be compared to a baseline system, which is simply a content-based recommender, and the Fiducia system [8], a recommender with a very similar domain, found in literature.
The explainability of our recommendations can be evaluated by discussing the underlying techniques in our system, such as the algorithms used, and how much transparency the given algorithm provides.

III. Implementation

A. Input interface

The system uses an extremely simple command line interface, whereby you either enter 'R' to view recommendations, or 'E' to edit a user profile to add more reviews, leading to the generation of a different set of recommendations. You will be prompted for which user you would like to view/edit. Users can explicitly enter their location information and chosen COVID guidelines when viewing recommendations.

B. Recommendation algorithm

The recommendation algorithm consists of a weighted hybrid scheme, combining collaborative filtering and content-based filtering. Each filter produces a score out of 5 for each user-item pair, these are then added and divided by 2 to produce a final score. Our recommendations are made up of the restaurants with the highest final score. It then takes explicit user input to filter our recommendations based on their location and requested COVID regulations.

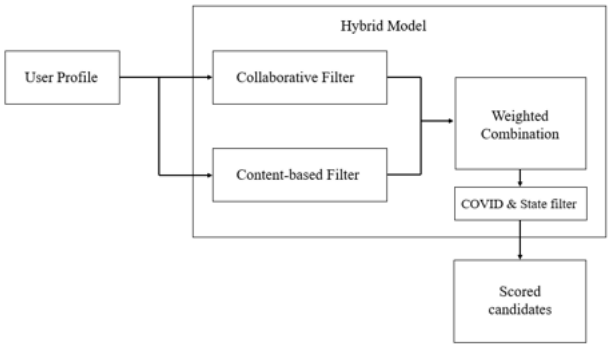

Fig. 1. Hybrid Recommender System Diagram

C. Output interface

The output is a list of 10 recommendations, where the first entry is the highest scored recommendation and the last is the lowest.

IV. Evaluation results

A. Accuracy of rating predictions

The hybrid system gave an RMSE of 1.19 and the baseline system achieved an RMSE of 1.98. This clearly indicates the hybrid model was more accurate. The Fiducia recommender achieves an RMSE of 1.01 and therefore is indeed more accurate at predicting ratings.

B. Accuracy of usage predictions

5 positive reviews (4+ stars) out of a total of 16 reviews were removed. Our system was able to produce 2 true positives and 8 false positives in the first 10 recommendations, giving an usage precision rating of 0.2. The baseline model gets an identical rating of 0.2, whilst the Fiducia system achieves a much higher precision of 0.74.

C. Novelty

The hybrid system produces novel recommendations (-7.15), with its top 10 recommendations making up restaurants that have total review counts from the range 57-1221. With this being said, the baseline also produces fairly novel recommendations (-6.44). I was unable to compare with the novelty of the Fiducia model as it is not provided.

D. Explainability

Whilst the baseline uses a content-based recommender that is trivial to explain, the hybrid incorporates SVD, which can be difficult to summarise to users without them being aware of the underlying mathematics involved in matrix factorization. The Fiducia model uses matrix factorization, and so also incurs similar issues to the hybrid model in terms of explainability.

E. Ethical issues

An immediate ethical concern for our system is the algorithm used in the collaborative filter. The collaborative filter uses SVD matrix factorization. Consequently, it is difficult to be fully transparent on how recommendations are generated without users being aware of the underlying principles behind SVD. We can summarise SVD to users in terms of underlying ”latent” features, and this is indeed communicated to users by the system after the recommendations have been produced.

Data filtering may serve as another ethical implication of our hybrid recommender system. In response to the COVID pandemic, it is important that we recognise the underlying risk involved in the hospitality industry, and so filter out recommendations based on their accessibility and safety during the pandemic. Our hybrid recommender system explicitly asks multiple COVID-related questions in order to provide a set of recommendations that meet the users safety requirements.

Privacy risk is another area we must consider. Users may be unaware of how we infer their preferences from the data they share. When recommendations are produced, the system should communicate how their data was collected for the intention of producing recommendations, and how their data is used to produce recommendations, as-well as communicating that all explicit information requested about COVID is not stored.

V. Conclusion

Due to the sparsity of the data, it is difficult to capture the underlying trends in the data. Consequently, SVD is the best approach. Our hybrid model perform better than our baseline model, but worse than the Fiducia model. A possible further development could include linking the system up to external API that, when given postal code data, returns the location, i.e. latitude, city. This data could then be used to further influence our recommendations, as realistically, a user would want to be recommended a restaurant that is within a suitable distance of them.

References

[1] Akhtar, Nikhat Agarwal, Devendra. (2015). A Literature Review of Empirical Studies of Recommendation Systems. International Journal of Applied Information Systems (IJAIS),New York, USA
[2] Burke, Robin. (2000). Integrating Knowledge-based and Collaborative-filtering Recommender Systems.
[3] Burke, Robin. (2002) Hybrid Recommender Systems: Survey and Experiments. User Model User-Adap Inter 12, 331–370
[4] Wen H., Ding G., Liu C., Wang J. (2014) Matrix Factorization Meets Cosine Similarity: Addressing Sparsity Problem in Collaborative Filtering Recommender System. In: Chen L., Jia Y., Sellis T., Liu G. (eds) Web Technologies and Applications
[5] YELP, YELP dataset, 2020. https://www.yelp.com/dataset [accessed 2021]
[6] Funk, Simon. (2006). Netflix Update: Try This at Home
[7] van Meteren, R., van Someren, M. (2000). Using Content-Based Filtering for Recommendation
[8] Mansi G., Agarwal A., Thukral D., Chakraborty T. (2019) Fiducia: A Personalized Food Recommender System for Zomato.