Using collaborative filtering for restaurant recommendations

cghh83

1 Introduction

With the increasingly high number of restaurants in towns and cities, it has become tiresome and tedious to choose where to eat or get takeaway. Many recommender systems have been developed to tackle this issue, with takeaway companies such as Deliveroo and JustEat providing examples of this for picking delivery options, and Google Maps showing restaurants in certain areas for you to pick. Whilst these recommender systems are useful in their separate scenarios, this paper aims to develop a recommender system to help recommend restaurants to a user covering both takeaway and eat-in destinations. This was done using the Yelp dataset [1], due to the dataset containing over 150,000 businesses to choose from and nearly 7 million reviews. Using collaborative filtering (CF), a personalised recommender system for these restaurants can be developed. This was selected due to the size of this dataset, as the number of user reviews available would allow for reliable predictions for new users to the system. The collaborative filtering method selected was SVD++ [2] (Singular Value Decomposition), as it is suited to large datasets and allows for new users to be added to the model in one training step. This paper will compare this personalised recommender system with a non-personalised one, and assess the differences between the two.

2 Related works

In a proposed solution to collaborative filtering for restaurant recommendations [5], a Pearson correlation-based algorithm is used to propose new predictions based on user similarity. This method is very computation heavy and doesn't allow for comparison to all users, as the proposed SVD++ model does.

3 Methodology

3.1 Data Description

The Yelp dataset [1] contains information about restaurants in 11 metropolitan areas, complete with images, user reviews and detailed attributes of the business. It is split into businesses, reviews, users, tips, and check-in data. The dataset was selected due to the large sample size of the data, reducing issues with data sparsity and increasing the accuracy of the model trained on this data.

Business data contains information such as an address, opening hours, geographic information, categories (such as cuisine), and attributes. A rating is given for each business, which was used for the non-personalised system. Reviews contain the id of the user and business, the number of stars given, ratings of the review given by other users, and text. Users contain an id, the number of reviews they have made, and rankings such as compliments they have been given and the average number of ratings they give.

3.2 Data preparation and feature selection

The data was in JSON format. Converting this to CSV made it more suitable for use in python libraries through pandas, and decreases the loading time as it does not need to be parsed every time the system is loaded.

In the business dataset, the important elements were ID for use as a key and all other information for display in the application. Reviews contain the number of stars, the user ID, the business ID, and more information about how useful the review is according to other users. Ratings from reviews were normalised using z-score normalisation as shown. Normalising these values removes bias towards items with a higher average rating, allowing the system to give predictions on all restaurants equally.

$$z_{ui} = \frac{r_{ui} - \overline{r_i}}{\sigma_i}$$

To fit the dataset to the domain of restaurants, the businesses needed to be filtered to only include those in the food and drink sector. This was done by selecting items with specific attributes. These attributes are 'RESTAURANTS', 'FOOD', 'BAR', 'BEER', 'WINE', 'COFFEE', 'TEA', 'CAFE', 'BREWERY', 'BARS', 'RESTAURANT', 'PUBS' and 'PUB'.

Limitations in training the model include the time and memory required. To overcome this, the number of users was cut down. Whilst cutting down the reviews per user was another potential solution, this was dismissed as it would reduce the authenticity of a comparison to the respective user. 100,000 randomly selected users were then used, as this created a large enough basis for accurate comparisons whilst not sacrificing variety in potential predictions.

For standardization across the system, including the model parameters, business and user IDs were re-initialised as numeric keys from 1 to n, then saved to the csv files to provide a consistent identifier.

Implicit data can be used with the SVD++ model to provide more accurate results. One part of this will be in

the form of the matrix R, with each value y_{ui} being 1 if the user u has rated item i and 0 if not.

3.3 Methods and techniques

The method chosen for the personalised recommender system was SVD++ [2], implemented with PyTorch [3] [4] as this allows for a model which can be easily extended, both in prediction accuracy and adding new users or businesses.

The model predicts a user's rating of item i by capturing the overall user interest in the latent factors of i. Latent factors of items are stored in vector q_i, where elements of q_i are positive values representing the extent to which item i possesses the latent factors. Vector p_u stores similar values for the extent user u has interest in the same latent factors. Predictions are also based on the biases b_u and b_i, which are deviations the user or item has from their respective averages. The overall mean rating μ is also used in the prediction. Additionally to these values, implicit data can be used to give a more accurate prediction. The data used for this is the matrix R, and factor vectors y_i, which are stored in the model.

Vectors q_i, p_u, y_i, b_u, and b_i are stored in embeddings in the model, with base weights initialised with a Gaussian distribution of $\mu = 0$ and $\sigma = 0.1$ and biases being initialised with a uniform distribution over the range $[-0.01, 0.01]$. This prevents the model from exploding, as loss gradients may become too large. These values are then used to give the predicted value of a user's rating on an item.

$$\hat{r}_{ui} = \mu + b_i + b_u + q_i^T \cdot (p_u + |\boldsymbol{R}(u)|^{-\frac{1}{2}} \sum_{j \in R(u)} y_j)$$

Predictions are done in batches to decrease computation time, with a batch size of 64 selected. The loss is then computed on each item in the batch using the absolute error.

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

For each training step, the loss is back propagated into a stochastic gradient descent (SGD) optimiser to move the parameters away from the gradient.

$$\begin{aligned} b_u &\leftarrow b_u + \gamma \cdot (e_{ui} - \lambda \cdot b_u) \\ b_i &\leftarrow b_i + \gamma \cdot (e_{ui} - \lambda \cdot b_i) \\ q_i &\leftarrow q_i + \gamma \cdot (e_{ui} \cdot p_u - \lambda_2 \cdot q_u) \\ p_u &\leftarrow p_u + \gamma \cdot (e_{ui} \cdot q_i - \lambda \cdot p_u) \\ \forall j \in \boldsymbol{R}(u) : y_j &\leftarrow y_j + \gamma \cdot (e_{ui} \cdot |\boldsymbol{R}(u)|^{-\frac{1}{2}} \cdot q_i - \lambda \cdot y_j) \end{aligned}$$

As part of the dataset was needed for evaluation, a test-train split was used, with the train set split again to create a validation set. Due to the size of the dataset, a small portion was needed for testing to give an accurate representation of the model's performance, therefore 20% was selected for the test size, and 10% of the resulting train set was used for validation. The validation set was used during training to measure the model's convergence after every iteration over the dataset, and the test set was used for the evaluation of both the personalised and non-personalised systems.

3.4 Evaluation Metrics

For evaluation, the test set outlined above was used in an offline experiment, for ease of conduct and comparison. The same test split was used to measure both the personalised and non-personalised systems, to ensure a fair evaluation between them.

The first metric to be measured was the root-mean-squared error (RMSE). This was chosen as larger errors disproportionately impact the mean, making it sensitive to outliers. Therefore, a consistent quality of results can be measured for the recommender systems, and a low deviation from the test set ratings would confirm the success of the model's ability to accurately predict a user's preference in restaurants. RMSEs were taken by comparing the model predictions to the standardised reviews in the test set.

The second metric to be measured was rank. The ranking was measured to ensure the user was being predicted restaurants that they were more likely to rate highly, and not being shown ones that they would dislike. This was measured by taking the users in the test set and ranking the businesses that they had reviewed, then using ranking metrics to evaluate how similar the model and non-personalised systems,R_s, were to the user's preferences, R_u. Rankings were taken proportionally within the size of predictions, as the model predicts for all businesses, whilst the reviews were limited to a small number. The metrics used for measuring the rank were Spearman's rank correlation coefficient (SRCC) and mean reciprocal rank (MRR). SRCC was used to determine if the model was predicting a positive or negative correlation with the user's reviews, and if so how much of a correlation. On the other hand, MRR was specifically used to evaluate how similar those rankings were to the user's reviews.

$$\rho_{R_s,R_u} = \frac{cov(R_s, R_u)}{\sigma_{R_s}\sigma_{R_u}}$$

4 Experimentation

Hyper-parameters for the model were experimented with to find optimal values for the dataset. The two values tested were weight decay, λ, and the learning rate, γ. Experiments were carried out with averages taken across different seeded test-train splits so the parameters selected were not overfitting to a specific train set.

As shown in fig. 1, the rolling average of loss over different learning rates gave an optimal value of $\gamma = 0.0034$ and weight decay $\lambda = 0.002$. Both these parameters were then used for the training of the model.

5 Implementation

For use in the system, all predictions are generated for the user and the top 8 are presented with details such as their average rating, name and a limited number of categories (as shown in Fig. 2). The number of predictions has been chosen to provide enough range of options without providing an overwhelming amount of information. The user can then select more predictions on a page basis. A user can select a restaurant to view it in more detail, loading from the business dataset, as shown in Fig. 3. Options to go back to

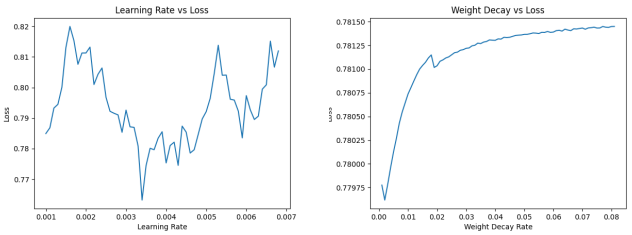

Fig. 1: Average loss for learning rate (left) and variable weight decay (right) across seeded test-train splits over 5 epochs of training.

the list of predictions or rate a business are then available. If the user rates the item, the model is trained on that rating until an absolute error of 1 is predicted, or a maximum number of steps. This is to allow the model to converge on the user's sentiment, without slowing down the user experience. When a new user is introduced to the system, the

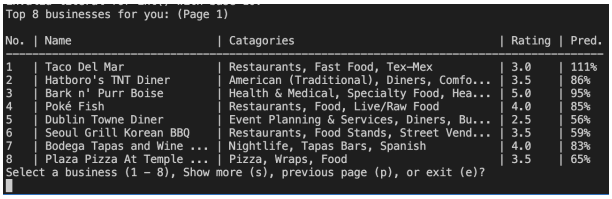

Fig. 2: List of predictions for a user.

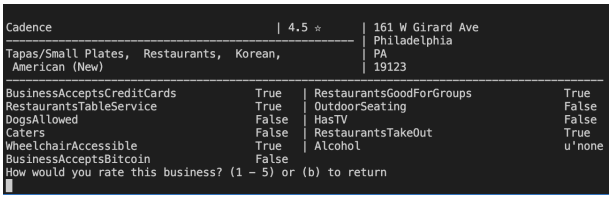

Fig. 3: View of business in the system.

model is expanded to accommodate this. The embeddings in the model are taken, stored, and then recreated in the new dimensions so that the old weights can be copied into their original indexes in an expanded embedding. The SGD optimiser's parameters are also expanded in the same way. The matrix R is expanded by one row to accommodate the new user's ratings.

For the non-personalised system, recommendations are given to restaurants that have the highest ratings. An improvement on this would be to offer predictions based on currently highly rated restaurants rather than of all time, it would have required too much data processing due to the size of the dataset. The display for this is the same as the personalised system, however, ratings do not impact the current user's recommendations.

6 Evaluation

6.1 Results

As discussed in section 3.4, the metrics were evaluated on the test set, with results shown in table 1.

Model	RMSE	SRCC	MRR
Personalised	0.86104	0.52510	0.2155827576
Non-personalised	0.92310	-0.04668	2.58980983e-6

TABLE 1: The three evaluation metrics selected: evaluated on the test set of reviews.

6.2 Comparison

The RMSE for the personalised system was 6.72% lower than the non-personalised, showing more precise predictions and a lack of outliers skewing the average error. As the personalised model gave a much higher prediction accuracy, it shows an ability to confidently predict restaurants that a user would enjoy or not enjoy.

The Spearman's rank correlation coefficient showed a weak positive correlation between the personalised rankings and the user's actual rankings, this shows that the model was predicting the user's hierarchical taste in restaurants, but not strongly. On the other hand, the non-personalised system displayed a negative correlation in ranking. In comparison between the two, the personalised system provided predictions that ranked, weakly, in the same way the user would. Whereas, the non-personalised did not.

The MMR ratings showed the personalised system having a much better correlation to the user's review rankings than the non-personalised, however still very low.

7 Conclusion

7.1 Summary

Overall, the success of the personalised system is shown in comparison to the non-personalised. The SVD++ model provides a much lower RMSE, showing the model accurately predicts user ratings, therefore providing predictions the user would likely rate highly. This is substantiated by the ranking correlation, as rankings predicted by the model correlated well to the ones in the test set. These correlations also provide evidence the personalised system created a good ranking of predictions, as the positive correlation shown by the SRCC indicates the restaurant prediction hierarchy matched that of the user's interest.

7.2 Limitations

Limitations of the proposed personalised model include the weak correlation to rankings, as predictions may have been lower down than the users. Another is the speed of the system when loading it. Whilst the system is very quick to predict, startup time is increased massively by loading the implicit data into memory. Allocating and then creating the data takes a large amount of time.

7.3 Future work

The SVD model was used because of how easily expandable it is. Model parameters could be tuned more to fit the dataset better using cross-validation rather than checking all values. The predictions and model parameters could be expanded to use a time-sensitive SVD++ model, as the reviews in the dataset have timestamps on them, this wouldn't be too intensive to implement and would help to decrease the RMSE.

REFERENCES

[1] Yelp, *Yelp Dataset*, https://www.yelp.com/dataset
[2] V. Klema and A. Laub, *"The singular value decomposition: Its computation and some applications"* in IEEE Transactions on Automatic Control, vol. 25, no. 2, pp. 164-176, April 1980, doi: 10.1109/TAC.1980.1102314.
[3] Paszke, A. et al., 2019. *PyTorch: An Imperative Style, High-Performance Deep Learning Library* in Advances in Neural Information Processing Systems 32. Curran Associates, Inc., pp. 8024–8035. Available at: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.
[4] nbsps, 2022, *Recommender System WITH PyTorch*, https://github.com/nbsps/RS-Models/tree/24c5e69f584349348a3f166cd8fbe85b0c2aba60.
[5] Achmad Arif Munaji and Andi Wahju Rahardjo Emanuel, *"Restaurant Recommendation System Based on User Ratings with Collaborative Filtering"* in IOP Conference Series: Materials Science and Engineering, Feb. 2021. https://dx.doi.org/10.1088/1757-899X/1077/1/012026.