Recommendation systems project

Ludovica Benini

Introduction

The goal of the project was to develop a book recommendation system by comparing different approaches to the problem. Four models were tested: an SVD model, a neural collaborative filtering model, a content-based model, and a hybrid model that combines the SVD and content-based approaches. To ensure a consistent ground truth for evaluating the results, all the models were designed to predict a given user rating for a given item.

The objective was to determine which method has the lowest prediction error, measured using RMSE and MSE, and whether a hybrid approach offers a significant improvement over traditional methods.

The code can be found in this repository. The models were implemented and compared in recommendation.ipynb.

Dataset and Data cleaning

The dataset is called *Amazon Books Reviews*, and it is available on Kaggle[1]. It consists of two files, one with books' details and the other containing the ratings. The books' preserved details were description and categories. In the ratings file I was interested in the rating score, the user, and the title of the rated book. The ratings range between 1 and 5.

The data cleaning process involved eliminating rows with missing or incorrect descriptions, as these were the primary focus of the content analysis. A data visualization (figure 1) was then performed to examine the distribution of review counts across users. This revealed that a large number of users had only a handful of reviews, making them not suitable for the prediction task. Consequently, I decided to retain only users with **at least 9 reviews** in the dataset.

[1] https://www.kaggle.com/datasets/mohamedbakhet/amazon-books-reviews

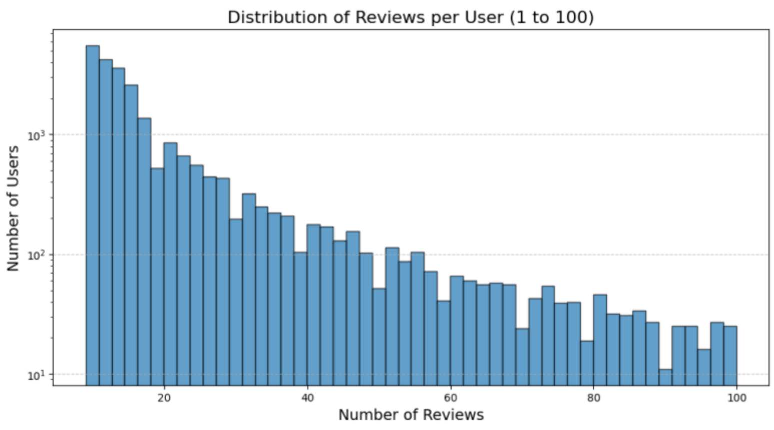

Figure 1. Histogram of the number of reviews per user, restricted up until 100 reviews for better visualization.

Subsequently, I used a **label encoder** to create numerical representations of both books and users. This process can be found in the notebook clean_data.ipynb.
The final dataset, containing **604,906 records**, was split into training (81%), validation (9%), and test (10%) sets.

Matrix Factorization - SVD

Singular Value Decomposition (SVD) is a matrix factorization technique widely used in recommender systems. According to the paper[2] published by the winners of the Netflix Prize:[3]

> "In its basic form, matrix factorization characterizes both items and users by vectors of factors inferred from item rating patterns. A high correspondence between item and user factors results in a recommendation."

In particular, the SVD-inspired algorithm as implemented by Funk,[4] learns the latent factors of users and items by:

- Factorizing the rating matrix into two lower-dimensional matrices: (users × latent factors) and (items × latent factors).
- Predicting ratings using the dot product of these matrices.
- Computing the error between predicted and actual ratings.
- Using stochastic gradient descent (SGD) to iteratively adjust the factor matrices and minimize the error.

The model was implemented using the **Surprise** library[5], which is designed for building recommender systems. After experimenting with different algorithms available in the library (detailed in the surprise.ipynb notebook), SVD proved to be the most effective.

[2] https://dl.acm.org/doi/10.1109/mc.2009.263
[3] https://en.wikipedia.org/wiki/Netflix_Prize
[4] https://sifter.org/~simon/journal/20061211.html
[5] https://surpriselib.com/

To optimize the model's performance, hyperparameter tuning was conducted, resulting in a trained model that achieved an **MSE of 0.4419** and an **RMSE of 0.6648** on the validation set.

Neural Collaborative Filtering (NCF)

While Matrix Factorization assumes linear relations between users and items (usually modelled through a dot product), **Neural Collaborative Filtering (NCF)** introduces non-linearity by leveraging neural networks to model the relationships between the latent factors of users and items.
To evaluate whether this approach could improve the performance of the recommendation system, I implemented a NFC model using **PyTorch** library and this article[6] as reference. Two embedding layers were employed to create randomly initialized embeddings for users and items. These embeddings were trained during the training process to capture the latent features users and items, using ratings as the ground truth. Over the course of training, embeddings for users and items with similar interaction patterns (based on ratings) converged to more similar representations.

To train the model a sequence of fully linear layers combined with activation functions was used. This feed-forward network takes the concatenated user and item embeddings as input and outputs a predicted rating or score.
The hidden dimension of the fully connected layers was included as a hyperparameter to allow for more exploration in optimizing the model's performances.

To optimize the training process, the well-known **Adam optimizer** was employed with a learning rate of 0.001 and a minimum **weight decay** of 0.00001. The loss function used during training is Mean Squared Error (**MSE**), while the final evaluation metric applied to the validation and test sets is Root Mean Squared Error (**RMSE**). This distinction was made because RMSE is a more intuitive and interpretable metric for assessing how much the predictions deviate from the ground truth ratings, as it is in the same unit as the target variable. Meanwhile, MSE performs better in terms of convergence during the training process due to its linear gradient, which facilitates stable and efficient optimization. [7]
In addition, to avoid wasting computation, I added an early stopping mechanism that stops training if the validation loss doesn't improve for 8 epochs[8].
After a training of 28 epochs, the results on the validation set are a **MSE of 0.56083** and a **RMSE of 0.74888.**

Hybrid Model

[6] https://pureai.substack.com/p/recommender-systems-with-pytorch

[7] "MSE vs RMSE: Which Is the Right Error Metric for Machine Learning? - Machine Learning Site," *Machinelearningsite.com* (blog), August 25, 2024, https://machinelearningsite.com/mse-vs-rmse/.
[8] https://stackoverflow.com/questions/71998978/early-stopping-in-pytorch

The third proposed model is a hybrid approach that combines the **SVD model**, which showed the best performance among the collaborative filtering models, with a **content-based model**. The goal is to determine whether the content-based model can enhance the performance of the SVD model, as suggested by the results discussed in the paper[9].

The content-based model relies on embeddings generated from the books' descriptions and categories. Two embedding systems were tested:
• Pre-trained embeddings from Stanford's GloVe project.[10]
• A **Word2Vec** model trained on the training set's descriptions and categories, which were previosuly converted into bigrams.
The embeddings were implemented using Python's **Gensim** library.[11] First, a new column was created by merging the book descriptions and categories. The text in this column was preprocessed to extract individual words, passed through a bigram converter trained on the train set, and then transformed into embeddings. This transformation was performed using a function that applies one of the embedding models (selected as a parameter) to each word in the text. The resulting word embeddings were then **averaged** to obtain a final representation for each book, following the approach outlined in the paper.[12]
To integrate the embeddings into the hybrid recommendation system, a **similarity matrix** is calculated using the cosine similarity between all the book embeddings.

The function that implements the hybrid recommendation first checks the number of ratings available for the input book. If the number is below a certain threshold, the hybrid system is used; otherwise, it returns the SVD prediction. This approach, as described in the paper,[13] leverages content-based filtering to address the cold-start problem, which reduces the efficiency of collaborative filtering.
If the hybrid modality is activated, the function finds all the books the user has rated and calculates the similarity between them and the target book. The predicted rating is then computed using the formula described in the article:[14]

[9] Mohammad Maghsoudi Mehrabani, Hamid Mohayeji, and Ali Moeini, "A Hybrid Approach to Enhance Pure Collaborative Filtering Based on Content Feature Relationship," *ArXiv.org*, 2020, https://arxiv.org/abs/2005.08148.
[10] https://nlp.stanford.edu/projects/glove/
[11] https://radimrehurek.com/gensim/index.html
[12] Mohammad Maghsoudi Mehrabani, Hamid Mohayeji, and Ali Moeini, "A Hybrid Approach to Enhance Pure Collaborative Filtering Based on Content Feature Relationship," *ArXiv.org*, 2020, https://arxiv.org/abs/2005.08148.
[13] Mohammad Maghsoudi Mehrabani, Hamid Mohayeji, and Ali Moeini, "A Hybrid Approach to Enhance Pure Collaborative Filtering Based on Content Feature Relationship," *ArXiv.org*, 2020, https://arxiv.org/abs/2005.08148.
[14] Sumeet Agrawal, "Item-Based Collaborative Filtering - Sumeet Agrawal - Medium," *Medium* (blog), October 13, 2021, https://medium.com/%40Sumeet_Agrawal/item-based-collaborative-filtering-4e64f65ae6ea.

$$\mathrm{R}(m,u) = \frac{\sum_j S(m,j) \cdot R(j,u)}{\sum_j S(m,j)}$$

Where:

- m is the current input book
- j is a book in the set of books rated by the user
- *S(x, y)* is the similarity function between 2 books x and y
- *R(b, u)* is the rating of a user u of a certain book b

The result is then weighted through an **alpha parameter** with the SVD prediction. A grid search is performed on the validation set to find the best parameters for:

- alpha,
- the minimum number of ratings for the input item for the hybrid system to be activated
- the model used for embeddings.

The results are:
- an alpha of 0.9,
- a minimum amount of rating of 10
- the use of the Word2Vec trained model for embedding.

This configuration of the hybrid model obtains a **RMSE of 0.6671** on the validation set. Since it has obtained the best performance among the models, I tested it on test set, obtaining an **RMSE of 0.6638** and a **MSE of 0.4407**.

Table 1. Results of all the models on validation set.

Model	RMSE	MSE
SVD	0.6682	0.4465
CNF	0.7488	0.5608
CB	2.0268	4.1079
Hybrid	0.6671	0.4450

Conclusions

As shown in Table 1, the best-performing models were the SVD and the hybrid. The hybrid model slightly outperformed the SVD, with an **improvement of only 0.0011** points in RMSE and 0.00125 points in MSE, which is a minimal difference. Given the relatively poor performance of the standalone content-based model (RMSE: 2.0268, MSE: 4.1079), the Hybrid model's score could potentially be improved by enhancing the content-based component, such as by finding different way of pre-processing the training data and using more complex models than Word2Vec.

When comparing the performance of SVD and CNF, it's important to note that the comparison here is not entirely fair. Due to computational constraints—since the Surprise library allows SVD to be trained significantly faster than the CNF implementation in PyTorch—the SVD model was trained for 300 epochs, whereas the NCF model was only trained for 28.

In conclusion, the results of both the SVD and hybrid models indicate that, on average, the error in predicting a rating is approximately 0.6. This is a solid outcome on a scale of 1 to 5.

Bibliography

Agrawal, Sumeet. "Item-Based Collaborative Filtering - Sumeet Agrawal - Medium." *Medium* (blog), October 13, 2021. https://medium.com/%40Sumeet_Agrawal/item-based-collaborative-filtering-4e64f65ae6ea.

Funk, Simon. "Netflix Update: Try This at Home." *Sifter.org* (blog), December 11, 2006. https://sifter.org/~simon/journal/20061211.html.

Lee, Dr Ernesto. 2024. "Building a Content-Based Recommender System with Python and Google Colab." *Medium* (blog). November 14, 2024. https://drlee.io/building-a-content-based-recommender-system-with-python-and-google-colab-c753c9bdd449.

Mehrabani, Mohammad Maghsoudi, Hamid Mohayeji, and Ali Moeini. 2020. "A Hybrid Approach to Enhance Pure Collaborative Filtering Based on Content Feature Relationship." *ArXiv.org*. https://arxiv.org/abs/2005.08148.

“MSE vs RMSE: Which Is the Right Error Metric for Machine Learning? - Machine Learning Site.” 2024b. *Machinelearningsite.com* (blog). August 25, 2024. https://machinelearningsite.com/mse-vs-rmse/.

Vivekecoder. 2024. “Movie-Recommendation-System.” GitHub. 2024. https://github.com/Vivekecoder/Movie-Recommendation-System.