Book Rating Prediction using Recommender Systems

Course project for CSE 6240: Web Search and Text Mining, Spring 2022

Abhishek Gawande
Georgia Tech
Atlanta, USA
agawande7@gatech.edu

Amandeep Singh
Georgia Tech
Atlanta, USA
asingh788@gatech.edu

Sai Shanbhag
Georgia Tech
Atlanta, USA
sshanbhag33@gatech.edu

Siddharth Sen
Georgia Tech
Atlanta, USA
ssen79@gatech.edu

ABSTRACT

Over the past decade, with the ever-rising adoption of e-readers, recommender systems have become the primary tools bibliophiles use for discovering and engaging with new content. The ability to provide good and accurate recommendations that pique the interest of the consumers can have a significant impact on not only increasing the popularity of reading as a recreational activity but also motivating authors to write more books while at the same time helping businesses meet their growth targets by improving sales. In this project, we have built a recommender system for books belonging to the Poetry genre using approaches in collaborative filtering and neural networks. The dataset used for this project comes from the book cataloging website Goodreads, and it has data on over 36K books and 377K users.

KEYWORDS

datasets, neural networks, recommender systems, collaborative filtering

1 INTRODUCTION

The goal of the project is to build a recommender system for books. For this exercise, we have used the dataset containing books related to the genre "Poetry" from Goodreads, which is a website that allows users to freely search their database for books and book reviews. Through this project, we have implemented a recommendation system, for recommending poetry books using multiple approaches based on collaborative filtering and other models leveraging deep learning methods. We shall cover three implementations, item-item collaborative filtering, neural collaborative filtering as well as wide-and-deep model for recommender systems.

We have evaluated and compared the results obtained from each model. Our first approach, which serves as a baseline for the remaining models, is Item-Item Collaborative Filtering. Here, we directly used the ratings given by users for different books for predicting user ratings. We later improved upon the predictions obtained from this model by generating embeddings for users and items using Matrix Factorization, and then later trained MLP and NeuMF for predicting the final rating given by a user for a particular book. The Wide-and-Deep model approach gave us the lowest Mean Squared Absolute error, which was our metric for measuring performance.

Through this exercise, we have been able to build a recommendation system that can reasonably predict the likelihood of a user liking or disliking a book. This enables us to provide meaningful recommendations to users for their future reading choices according to each user's taste and reading preferences, however eclectic they may be.

2 LITERATURE SURVEY/BASELINES

2.1 Item-to-Item Collaborative Filtering [3]

The idea behind this approach is that rather than matching the user to similar customers, item-to-item collaborative filtering matches each of the user's purchased and rated items to similar items, then combines those similar items into a recommendation list. Cosine similarity is used to calculate the similarity between products, and a similar-items table is built. Given a similar-items table, the algorithm finds items similar to each of the user's purchases and ratings, aggregates those items, and then recommends the most popular or correlated items. This computation is very quick, depending only on the number of items the user purchased or rated.

The main advantages of this algorithm are that it is scalable over very large customer bases and product catalogs, is able to react immediately to changes in a user's data and makes accurate recommendations for all users regardless of the number of purchases or ratings.

2.2 Neural Collaborative Filtering [2]

This paper works on improving the traditional collaborative filtering approach used as the basis for recommender systems by incorporating techniques based on neural networks. By replacing the inner product of the latent features of users and items with a neural architecture that can learn an arbitrary function from data, the paper presents a general framework named Neural Collaborative Filtering (NCF), which is generic and can express and generalize matrix factorization under its framework proposed three instantiations – GMF, MLP and NeuMF – that model user-item interactions in different ways.

This paper has been very well received in the community as it complements the mainstream shallow models for collaborative filtering, opening a new avenue of research possibilities for recommendations based on deep learning.

2.3 Wide and deep learning for recommender systems [1]

This paper focuses on developing a new recommender system framework that combines the benefits of memorization and generalization. This is achieveed by jointly training wide linear models and deep neural networks. Wide linear models can effectively memorize while deep neural networks can effectively generalize. Memorization can be defined as learning the frequent co-occurrence of items and Generalization explores new feature combinations that have never or rarely occurred in the past. Recommendations based on memorization are directly relevant to items that the users have already bought while generalization tends to improve the diversity of recommended items. The authors evaluated their framework on user and app data collected from Google Play within a period. Their experiments showed that the Wide and Deep model significantly improved app acquisitions over wide-only and deep-only models.

Overall, this paper relates to our project through its attempt at building a novel recommender system framework that can be trained on our project's dataset.

3 DATASET DESCRIPTION

3.1 Data Preparation

We have taken our datasets from the following website: Goodread Datasets [4][5]. The dataset was collected in late 2017 from the Goodreads catalog by scraping users' public shelves i.e., everyone can see it on the web without login. User IDs and review IDs were anonymized.

We have selected the Poetry genre with the following datasets: books metadata, user data and user-book interactions (shelves), and reviews data. We have combined interactions and reviews to get the book ratings by a user as reviews provide the ratings and interactions indicate whether the user has read the book or not. This allowed us to filter fraud reviews and provide legitimate ratings. Also, in case of updates, we have taken the last given rating.

Since we are aiming to provide personalized recommendations based on user ratings, this data provides necessary ratings as well as the metadata for sufficient books and users.

3.2 Raw Data Statistics

# books	36,514
# users	377,799
# shelves	2,734,350
# read	1,313,610
# rate	1,229,059
# review	155,414
# shelve/book	74.88
# shelve/user	7.24
# read/book	35.98
# read/user	3.48
# rate/book	33.66
# rate/user	3.25
# review/book	4.26
# review/user	0.41

The table above encapsulates a wide range of data statistics, such as number of reviews per book, reviews of user, books read, books rated and so on.

The ratings given by users range from 1-5, and the dataset spans multiple languages as shown in the figures below.

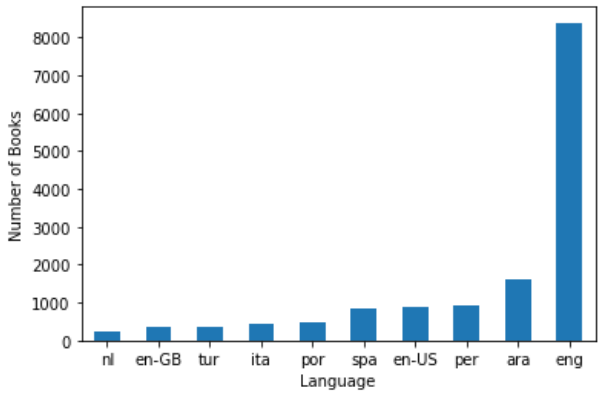


Figure 1: Number of Books per Language

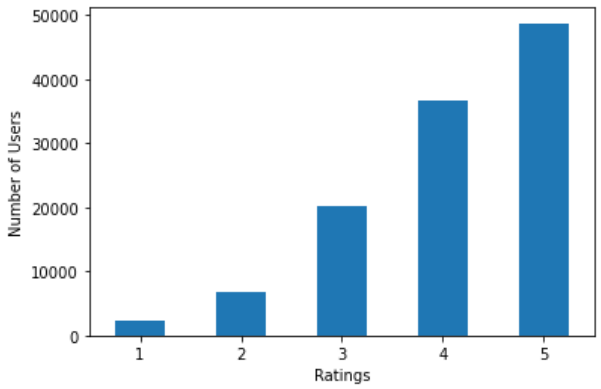


Figure 2: Number of Books per User Rating

4 EXPERIMENTAL SETTINGS

We shall be working on books, reviews, and user interactions data obtained from the Goodreads dataset for books related to the poetry genre. Our goal was to predict the prospective ratings of unread books for a user. For training purposes, we divided the data into training and test sets with a split ratio of 80:20. We then evaluate our model using MSE (Mean Square Error) and MAE (Mean Absolute Error) in the prediction of book-rating in test data. We used Google Colab to train and run our models, with an Nvidia K80 GPU, 12GB RAM, and a dual core server CPU.

5 METHODS

5.1 Item-item Collaborative Filtering

Item-item collaborative filtering is a form of collaborative filtering for recommender systems based on the similarity between items

calculated using people's ratings of those items via the user-item rating matrix. Our dataset contained rows where each record represents one rating for a particular book given by a specified user. Since we did not have available any pre-trained embedding for computing similarity scores for user and item pairs, we decided to directly use the user-item rating matrix for computing the similarity scores using cosine similarity.

While splitting the data into training and test sets, we noticed that there were some books in the dataset which contained only one rating. Since we do not have embeddings for the books and our cosine similarity computation needs to directly come from the rating matrix, we cannot keep any such items in the test set without them being present in the training set. Hence, all books with only one rating were by default added to the train dataset. For the remaining data, we randomly selected one rating instance for each book and added it to the test dataset while all the remaining instances were added to the training data. This approach led to roughly a 90:10 split between train and test data.

We then generated the user-item rating dictionary and item-user rating dictionary based on the training dataset. To execute the recommendations scores and calculate the predicted rating of user i on item j based on the item-item collaborative filtering, we used the similarity scores and computed the predicted rating on item i for a user u, by computing the weighted sum of the ratings given by the user on the other items i.

This method utilizes ratings given by users to items, and since we have sufficient ratings/book, this method will serve adequately as a baseline from which we can compare our more complicated models. For evaluation of our model, we used Mean Absolute Error (L1) and Mean Squared Error (L2). The reported scores are in the conclusion section.

5.2 Neural Collaborative Filtering

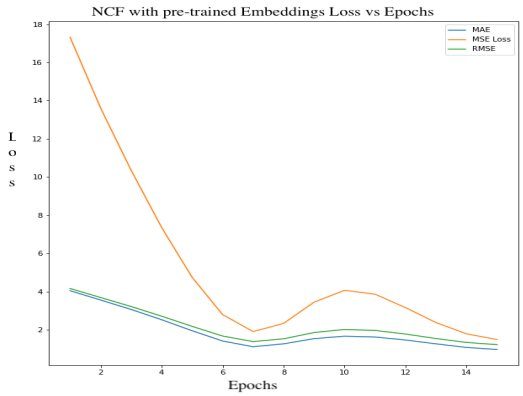
NCF has been one of the recent breakthrough papers in the world of recommendation systems, as it has shown significant improvements in recommendation performance over pre-existing methods. Therefore, we believe that this is a suitable model for our attempt to build a rating prediction system for books. We have implemented three different variations of this model, and the measurement metrics are in the conclusion section.

5.2.1 NCF with Matrix Factorization.

Matrix factorization can be used to generate latent features when multiplying two different kinds of entities. We have used Collaborative filtering as an application of matrix factorization to identify the relationship between readers and book entities. In this approach, we first used rating adjacency matrix factorization to create reader and book embeddings. We then combined these two to create input embeddings which are then passed to a Multi-Layered Perceptron network. Model architecture is as follows:

```
MLP(
  (fc1): Linear(in_features=30, out_features=512, bias=True)
  (fc2): Linear(in_features=512, out_features=512, bias=True)
  (output): Linear(in_features=512, out_features=1, bias=True)
  (relu): ReLU()
)
```

```
==========================================================
Layer (type:depth-idx)                   Param #
==========================================================
MLP                                      --
├─Linear: 1-1                            15,872
├─Linear: 1-2                            262,656
├─Linear: 1-3                            513
├─ReLU: 1-4                              --
==========================================================
Total params: 279,041
Trainable params: 279,041
Non-trainable params: 0
==========================================================
```



5.2.2 Neural Matrix Factorization (NeuMF).

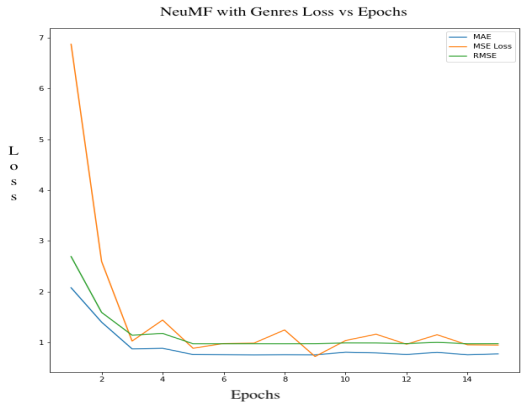
Rather than using matrix factorization, we next trained reader and book embeddings as part of the MLP network. This ensures better embeddings as compared to previous models which are trained dynamically through loss optimization. Model Architecture is as follows:

```
MLP(
  (embedding_user): Embedding(267821, 32)
  (embedding_item): Embedding(36182, 32)
  (output): Linear(in_features=8, out_features=1, bias=True)
  (relu): ReLU()
  (fc_layers): ModuleList(
    (0): Linear(in_features=64, out_features=64, bias=True)
    (1): Linear(in_features=64, out_features=32, bias=True)
    (2): Linear(in_features=32, out_features=16, bias=True)
    (3): Linear(in_features=16, out_features=8, bias=True)
  )
)
```

```
==========================================================
Layer (type:depth-idx)                   Param #
==========================================================
MLP                                      --
├─Embedding: 1-1                         8,570,272
├─Embedding: 1-2                         1,157,824
├─Linear: 1-3                            9
├─ReLU: 1-4                              --
├─ModuleList: 1-5                        --
│    └─Linear: 2-1                       4,160
│    └─Linear: 2-2                       2,080
│    └─Linear: 2-3                       528
│    └─Linear: 2-4                       136
==========================================================
Total params: 9,735,009
Trainable params: 9,735,009
Non-trainable params: 0
==========================================================
```

5.2.3 NeuMF with Book Genres.
Since book ratings are influenced by their Genres as well, we further included this information as part of the embedding input to be passed in MLP architecture. Since a book can be classified under multiple genres, we used the user genre classification as the basis to create this embedding. Model Architecture is as follows:

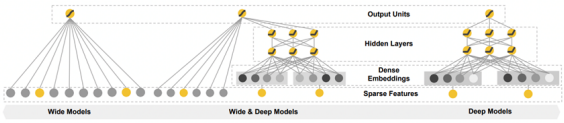
```
MLP(
  (embedding_user): Embedding(267821, 32)
  (embedding_item): Embedding(36182, 32)
  (output): Linear(in_features=8, out_features=1, bias=True)
  (relu): ReLU()
  (fc_layers): ModuleList(
    (0): Linear(in_features=74, out_features=32, bias=True)
    (1): Linear(in_features=32, out_features=16, bias=True)
    (2): Linear(in_features=16, out_features=8, bias=True)
    (3): Linear(in_features=8, out_features=8, bias=True)
  )
)

==========================================================================
Layer (type:depth-idx)                   Param #
==========================================================================
MLP                                      --
├─Embedding: 1-1                         8,570,272
├─Embedding: 1-2                         1,157,824
├─Linear: 1-3                            9
├─ReLU: 1-4                              --
├─ModuleList: 1-5                        --
│    └─Linear: 2-1                       2,400
│    └─Linear: 2-2                       528
│    └─Linear: 2-3                       136
│    └─Linear: 2-4                       72
==========================================================================
Total params: 9,731,241
Trainable params: 9,731,241
Non-trainable params: 0
==========================================================================
```



5.3 Wide and Deep Learning for Recommender Systems

We followed the implementation of Wide-and-Deep model provided by the API in this GitHub Repository.

The Wide-and-Deep learning framework jointly trains a feed-forward neural network with embeddings (the deep component) and a linear model with non-linear transformations (the wide component). The wide component is a generalized linear model which uses cross-product transformed user and item one-hot vector as its feature set. The deep component uses the high-dimensional user and item features (the one-hot vectors) and converts them into a low-dimensional and dense real-valued vector, often referred to as an embedding vector. We set the dimensionality of these embeddings to 32 for user embeddings and 16 for item embeddings. The embedding vectors are randomly initialized and then the values are trained to minimize the final loss function during model training. These low-dimensional dense embedding vectors are then fed into the hidden layers of the neural network in the forward pass.



It's important to highlight the distinction between ensemble learning and joint training. In an ensemble, individual models are trained independently of each other, and their predictions are combined at training time but only at inference time. In contrast, joint training takes both the wide and deep parts as well as the weights of their sum into account at training time and optimizes all parameters simultaneously.

User-provided ratings for the books along with the book-genre feature were used to train the wide-and-deep model. The genre

	userID	itemID	rating	genre
0	0	5949	4	[1461.0, 4635.0, 1304.0, 0.0, 3669.0, 0.0, 82....
1	0	5831	4	[1394.0, 2837.0, 567.0, 0.0, 3028.0, 0.0, 57.0...
2	0	21931	5	[0.0, 1523.0, 169.0, 0.0, 9400.0, 0.0, 206.0, ...
3	1	21931	3	[0.0, 1523.0, 169.0, 0.0, 9400.0, 0.0, 206.0, ...
4	2	17215	4	[0.0, 548.0, 51.0, 0.0, 4954.0, 0.0, 99.0, 571...

feature for a book captures the number of users who tagged a particular book as belonging to a certain genre. We had 10 total genres in our dataset which were ['genres.history, historical fiction, biography', 'genres.fiction', 'genres.fantasy, paranormal', 'genres.mystery, thriller, crime', 'genres.poetry', 'genres.romance', 'genres.non-fiction', 'genres.children', 'genres.young-adult', 'genres.comics, graphic'].

We used the following hyperparameters and feature sets for the model:

	Wide Model	Deep Model
Feature Set	User and Item one-hot vectors	• Deep, lower-dimensional embedding vectors for each user and item • Book Genre Feature
Hyper-Parameters	• Adagrad Optimizer • Learning Rate = 0.0621	• Adagrad Optimizer • Learning Rate = 0.1 • Hidden units = [512, 128, 64] • Dropout rate = 0.8 • Batch normalization (Batch size = 32) • User embedding vector size = 32 • Item embedding vector size = 16

6 CONCLUSION AND FUTURE SCOPE

In our project, we explored different approaches to predict prospective book ratings using Goodreads Reviews Dataset. We have used 3 broad approaches to tackle this problem. Beginning with item-item collaborative filtering approach, we continuously improved our model results with Neural Collaborative Filtering approaches and Wide-and-Deep model. The summary of the Model results is as follows:

Model	MAE	MSE
Item-Item Collaborative Filtering	2.851	11.518
NCF with Matrix Factorization	0.97	1.426
Neural Matrix Factorization (NeuMF)	0.784	1.002
NeuMF with Book Genres	0.779	0.964
Wide-and-Deep Recommendation Model	0.717	0.774

As we can see, the Wide-and-Deep recommendation model provides the best results, marginally better than NCF models and significantly improved over Item-Item CF.

For future work, we can improve upon our model in the following ways:

- We can create embeddings using methods such as node2vec to leverage graph structural information.
- We can also use BERT to include book metadata and descriptions along with readers' browsing behaviors to further amplify the level of details and information

Though we have used the Goodreads dataset, our model results and approaches can be used by companies to generate prospective ratings which could then be used in personalized recommendation systems.

7 CONTRIBUTION

All team members have contributed a similar amount of effort.

REFERENCES

[1] Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, Rohan Anil, Zakaria Haque, Lichan Hong, Vihan Jain, Xiaobing Liu, and Hemal Shah. 2016. Wide & Deep Learning for Recommender Systems. *CoRR* abs/1606.07792 (2016). arXiv:1606.07792 http://arxiv.org/abs/1606.07792

[2] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural Collaborative Filtering. *CoRR* abs/1708.05031 (2017). arXiv:1708.05031 http://arxiv.org/abs/1708.05031

[3] G. Linden, B. Smith, and J. York. 2003. Amazon.com recommendations: item-to-item collaborative filtering. *IEEE Internet Computing* 7, 1 (2003), 76–80. https://doi.org/10.1109/MIC.2003.1167344

[4] Mengting Wan and Julian J. McAuley. 2018. Item recommendation on monotonic behavior chains. In *Proceedings of the 12th ACM Conference on Recommender Systems, RecSys 2018, Vancouver, BC, Canada, October 2-7, 2018*, Sole Pera, Michael D. Ekstrand, Xavier Amatriain, and John O'Donovan (Eds.). ACM, 86–94. https://doi.org/10.1145/3240323.3240369

[5] Mengting Wan, Rishabh Misra, Ndapa Nakashole, and Julian J. McAuley. 2019. Fine-Grained Spoiler Detection from Large-Scale Review Corpora. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, Anna Korhonen, David R. Traum, and Lluís Màrquez (Eds.). Association for Computational Linguistics, 2605–2610. https://doi.org/10.18653/v1/p19-1248