Collaborative Filtering for Movie Recommendation

Comparing Classical and Neural Algorithms

Aditya Srivastava
Dept. of Computer Science
University of Colorado
Boulder
aditya.srivastava@colorado.edu

Harsh Gupta
Dept. of Computer Science
University Of Colorado
Boulder
harsh.gupta@colorado.edu

Akimun Jannat Alvina
Dept. of Electrical Engineering
University of Colorado
Denver
akimunjannat.alvina@ucdenver.edu

Niharika Narasimhiah Govinda
Dept. of Applied Mathematics
University of Colorado
Boulder
nina3335@colorado.edu

ABSTRACT

With the rising consumption of digital media, personalized content recommendation has become essential to maintain user engagement. Thus, the development of effective recommendation systems has become paramount, with collaborative filtering methods emerging as one of the key methods for doing so. In this paper, we propose the use of the MovieLens dataset to compare the classical matrix factorization algorithm to a newer neural algorithm and evaluate it on popularly used metrics.

1 Introduction

Viewers today have unlimited access to a wide array of films and shows, bringing in a new era of cinematic choice. While this abundance offers an exciting number of viewing opportunities, it also presents a considerable challenge for viewers in that they must traverse this vast library successfully and find content that suits their own tastes and preferences. Thus, recommendation systems have emerged as a key element in improving user experiences at a time when information and entertainment are becoming increasingly conflated. These systems, sometimes known as “recommender systems”, have developed into a crucial component of a variety of online platforms, from streaming services to e-commerce websites. Movie suggestion has distinguished itself as a compelling and well-accepted use case among the large range of applications for recommender systems and massively impacts all stakeholders, i.e. the viewers consuming content, the media houses producing movies, and the platform itself driving engagement. These points served as our motivation to pursue this project.

Recommendation systems can be generally classified into three types - content-based, collaborative filtering based, and approaches that are hybrids of both.

Content-based systems look at the attributes of objects (such as books, movies, and articles) and past user preferences. They then offer suggestions for products with features the viewer has previously liked. Collaborative filtering recommendation systems leverage the collective choices of users to make personalized predictions. These algorithms discover trends, correlations, and affinities among users by examining user interactions like movie ratings and viewing histories. Hybrid recommendation systems combine the strengths of both content-based filtering and collaborative filtering methods. By leveraging multiple recommendation techniques, they aim to provide more accurate and diverse suggestions.

In this paper we focus on collaborative filtering methods, and explore two algorithms, Matrix Factorization, and Neural Collaborative Filtering.

2 Literature Review

Collaborative filtering (CF) is an RS technique that generates entity preferences from the preferences of the other entities in the group. These entities can be either users (recommend items used by similar users)[1], or items (recommend users for the items based on users of similar items).[2,3] CF can generally be grouped into three categories: latent factor models, memory or neighborhood-based approaches, and hybrid models.[4] Latent factor models, such as Matrix Factorization[5, 6] and

SVD[7], project users and items into a shared embedding space and use their latent features to find correlations between the entities and have been very successful. Neighborhood based approaches, such as clustering were used at Amazon[8], to form recommendations by identifying groups of similar users or items based on previous interaction history. Hybrid approaches, such as SVD++[9] combine both approaches, often making the best of the local correlations discovered by the former and the global correlations discovered by the latter.

Deep learning (DL) has seen exponential growth over the past decade, and is currently considered the state-of-the-art in machine learning, with wide ranging applications in computer vision, natural language processing and other domains.[10, 11, 12] DL techniques have also been applied to recommendation systems, with all of the current benchmarks being dominated by methods such as graph representation learning[13] and variational autoencoders[14] and even exploration of LLMs such as chatGPT for recommendation[15].

3 Dataset

3.1 Description

The MovieLens dataset[17] contains records of movie ratings by the users of the MovieLens website and is maintained by the GroupLens organization, which is a research group at the University of Minnesota that maintains and provides access to the different MovieLens datasets. The dataset was chosen due to its popularity as a benchmark for testing new recommendation algorithms and has been extensively peer-reviewed.

The data set contains explicit user-item interactions in the form of ratings (from 1 to 5, inclusive) and contains user demographics (age, occupation, etc.) and movie features (genre, year of release, etc.), in a mix of numerical and categorical attributes.

The dataset comes in various sizes, but we use the one with 100,000 samples in our work, due to its smaller size being more convenient to run tests on.

3.2 Preprocessing

The stock dataset comes as a set of text files containing user-item-rating tuples, user-feature tuples and item-feature tuples. We treated the dataset differently between analysis and modeling, and thus followed different preprocessing steps for both. The difference arose from our decision to factor in explicit ratings and features in our data analyses, but perform the modeling on implicit (binary) interactions, where if a user rated a movie, we considered it as the user having watched it, otherwise, the user hadn't watched the movie.

During analysis we kept all features of the dataset, using Min-Max scaling to scale numerical attributes to [0, 1], and One-Hot encoding categorical features.

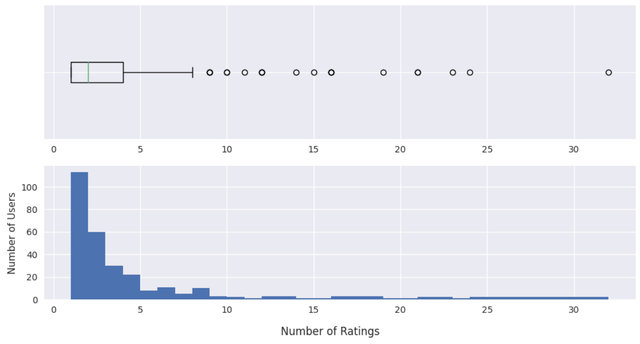
For modeling the data, we first discarded all features but the ratings, since we were not testing content-based filtering algorithms. At this point, every explicit rating in our dataset is an implicit positive sample for our training set. To produce the implicit negative samples for each user, we now sample randomly from items that they haven't rated. This is done by first creating a mapping from each user to the set of movies that they have rated. Now, for every movie rated by a user, a randomly sampled movie that is absent from the user's rated set is added to the training set.

For evaluation using the HR@K and NDCG@K metrics (refer to section 5 for more details), we created lists of 100 negative samples, and 1 positive sample that was held out of the training set for that user.

Finally, when creating the training, validation and test splits, we made sure that users who had rated a single item were part of the training set to avoid the cold-start problem[21], i.e. there may exist movies that have too few user ratings to be recommendable, and there may exist users who have rated too few movies to be adequately profiled. Additionally, items that had too few ratings or too many ratings were dropped during training.

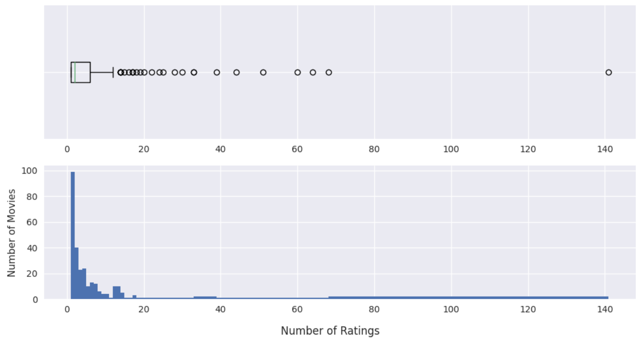
3.3 Data Analyses

Our goal was to find patterns in the data that we could exploit to get better results and to confirm if our model is performing correctly. The patterns can be used to ascertain that our model's predictions fit the observations gleaned from the data. This is especially helpful in the case of unlabeled samples.

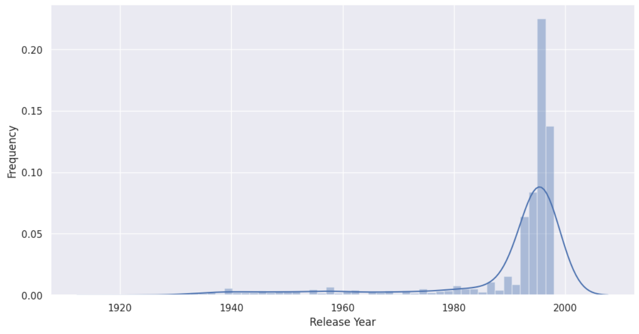


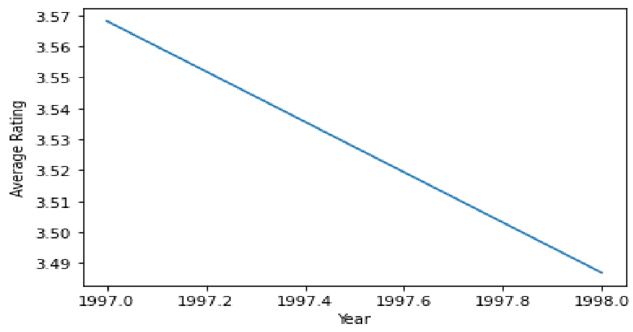
We started by performing a simple frequency analysis over various features in the dataset. In the figure above we can see the distribution of ratings over users. The distribution is left-skewed, indicating that while many users rate only a handful of movies, a smaller subset of users are highly active, contributing ratings for hundreds of films. This suggests a difference in user behavior, with casual raters and a few avid movie watchers who rate extensively.

We also found the distribution of ratings over movies, as shown in the figure below. This too is a left-skewed graph, and the fact that many movies have a low number of ratings might suggest that either the dataset has a lot of niche films or that many users tend to rate only the movies they are familiar with, passing over lesser-known films.

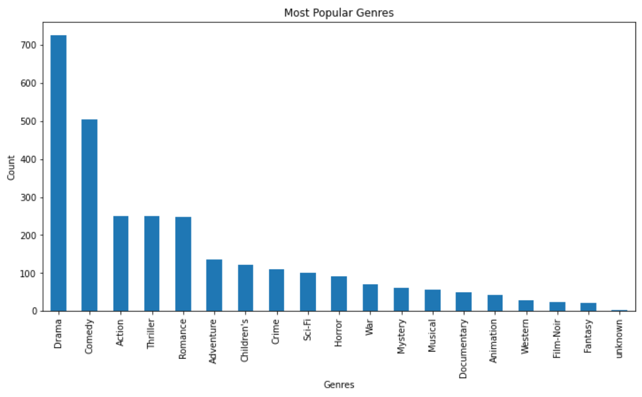


Both of the above observations indicate that our model may be likely to suffer from the cold-start problem, since the user-item interaction matrix is extremely sparse, and there are many users and movies with only a few assigned ratings.

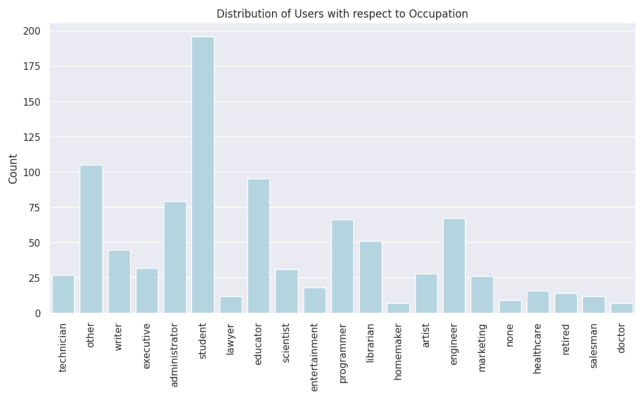




Following this we plotted a correlation matrix between all the features in the dataset. The matrix is too large to fit here and has been provided at the end of the document. An interesting correlation was discovered between the release dates of movies and their ratings. We can see clearly in the graphs above that as the number of movies released each year increases the average score goes down. This seems like an obvious inference considering that there are likely to be only a few good movies among a growing number of productions. Additionally, it is likely that users selectively go back to give high ratings to the few old movies that left a lasting impression.



Going back to frequency analysis, we found *Drama* to be the most popular genre, followed by *Comedy* and *Action*. Surprisingly, based on correlation scores, we can see that the most frequent age group in our data (students, aged ~20 years), actually correlates negatively with the *Drama* genre of movies, rating it poorly compared to their counterparts.



The graph above reveals the distribution of users across various occupational categories. It is interesting to note that *students* emerged as having the greatest predilection to rate movies compared to other professional spheres. This could be attributed to students' tendency to consume a greater quantity of media and furthermore, a greater likelihood of rating the films they view. It is also noteworthy that occupations such as homemakers and doctors are less common, as indicated by the figure.

From the correlation matrix, we also see some correlation between genres and genders of the users, with women seeming to prefer *Romance* movies over men, and men seeming to prefer *Sci-Fi* over women.

Another notable correlation between features can be found between the genres themselves, where certain genres are likely to co-occur frequently, for example, *Animation* movies are likely to also be genre classified into *Children's* and *Musical*. Similarly, *Mystery* and *Thriller* genres also seem to co-occur.

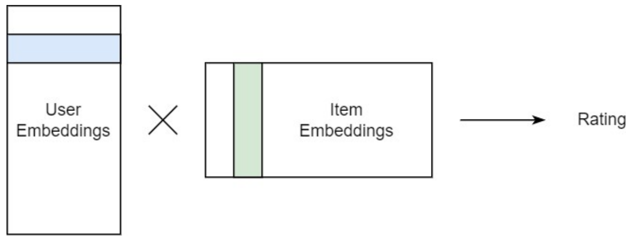
Finally, we would like to bring attention to some likely specious and perhaps even damaging biases seen in our data. For example, the gender and occupation correlation scores seem to strongly imply that a *male* user is more likely to be a *technician, programmer* or *engineer*, whereas a *female* user is more likely to have the *healthcare* and *homemaker occupations.* We might want to get rid of these demographic labels to avoid subjecting our model to these biases, however it may pose a significant challenge to do so without impacting recommendation quality.

4 Method

We compared two algorithms - Matrix Factorization (MF) which is a popular classical method, and Neural Collaborative Filtering (NCF) which is a newer method employing neural networks. Both methods are essentially regression models that rate items on a scale of 0 to 1.

4.1 Models

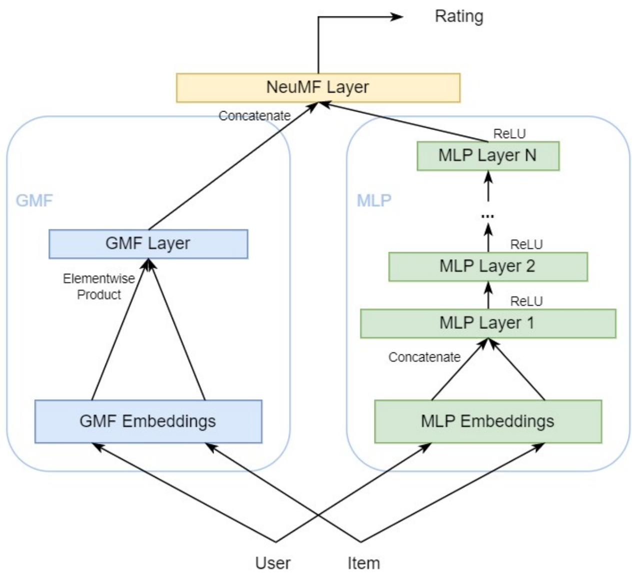
4.1.1 Matrix Factorization (MF)



Let vector u_i represent user i, and vector e_j represent item j. The rating r_{ij} given by the user to the item can now be calculated as the inner product of the two vectors, as shown below;

$$r_{ij} = u_i^T . e_i$$

4.1.2 Neural Collaborative Filtering (NCF)



The CNF method (He et al., 2017) [18] extends MF, by adding a multi-layer perceptron (MLP), and passes a combined output of the Generalized MF (GMF) and MLP subsystems through the final Neural MF (NeuMF) layer to produce the rating. This can be seen in the figure above.

Each of the two subsystems have their own embedding matrices, and thus their own vector representations for each user and item. Let $u_{(GMF, i)}$ and $u_{(MLP, i)}$ be the vectors for user i from the GMF and MLP embedding matrices respectively. Let $e_{(GMF, j)}$ and $e_{(MLP, j)}$ be the vectors for item j from the GMF and MLP embedding matrices respectively. Thus, outputs of the two modules can be described as shown below;

$\mathrm{GMF}(u_i, e_j) = h_{GMF}(u_{(GMF, i)} \square e_{(GMF, j)})$

$\mathrm{MLP}(u_i, e_j) = a_{ReLU}(h_{(MLP, X)}(\ldots$
$a_{ReLU}(h_{(MLP, 2)}($
$a_{ReLU}(h_{(MLP, 1)}(u_{(GMF, i)} \oplus e_{(GMF, j)}))$
$))$
$\ldots))$

Where □ is elementwise multiplication of vectors, h_{GMF} is a dense layer (GMF layer in the diagram), $\oplus$ is concatenation of vectors, $h_{(MLP, L)}$ is the Lth dense layer (MLP layers in the diagram) and a_{ReLU} is the ReLU activation function which induces non-linearity in the model.

Thus, the final output of the combined NCF model is,

$$r_{ij} = a_{Sigmoid}(h_{NCF}(\mathrm{GMF}(u_i, e_j) \oplus \mathrm{MLP}(u_i, e_j)))$$

where h_{NCF} is a dense layer (NeuMF layer in the diagram) and $a_{Sigmoid}$ is the Sigmoid activation function to project output into the [0, 1] range.

4.2 Objective Functions

The source literature on MF details the use of Root Mean Squared Error (RMSE) to find the error between the predicted rating and the true rating. The source literature on NCF details the use of Binary Cross Entropy (BCE) to do the same. In addition to these two, we decided to break away from the literature and also implemented the Bayesian Personalized Ranking (BPR) loss for our models.

The three functions all treat the recommendation problem differently - RMSE treats the problem as one of regression, trying to reduce the model's error compared to the true value directly. BCE treats the problem as one of binary classification, treating the implicit rating as a probabilistic variable in the range [0, 1]. Finally BPR treats the problem as one of finding contrast, encouraging the model to rate negative samples as low as possible and positive samples as high as possible, and thus maximize the difference between the two.

4.2.1 Root Mean Squared Error (RMSE)

$$RMSE = \sqrt{1/N \times \sum_{i=1}^{N} (y_i - r_i)^2}$$

where N is the number of samples, y_i is the true value for the ith sample, r_i is the predicted value for the ith sample.

4.2.2 Binary Cross Entropy (BCE)

$$BCE = -1/N \sum_{i=1}^{N} [y_i \times log(r_i) + (1 - y_i) \times log(1 - r_i)]$$

where N is the number of samples, y_i is the true value (0 or 1) for the ith sample, r_i is the predicted probability for the ith sample.

4.2.3 Bayesian Personalized Ranking (BPR)

$$BPR = \sum_{(u,i,j)\in D_s} [log\, \sigma(d_{uij}) - \lambda_\theta \cdot ||\theta||^2]$$

$$d_{uij} = r_{ui} - r_{uj}$$

where D_s is the training set of triplets (u, i, j) such that the user u prefers item i (positive item with true rating 1) over item j (negative item with rating 0), d_{uij} is the difference between the ratings for the two items as predicted by the model. σ is the sigmoid function. λ_θ is a regularization parameter and $||\theta||^2$ represents the squared L2 norm of the model parameters.

4.3 Optimization

As prescribed by the reference literature, we used Stochastic Gradient Descent (SGD) with Nesterov momentum[22] to optimize models when using RMSE, and used Adam to optimize models using BCE. We also SGD to optimize the BPR based models. We will not be going into the details of these optimization methods, as that would be going out of the scope of this paper.

5 Evaluation

For evaluation of the two methods, we employ two commonly used leave-one-out metrics - Normalized Discounted Cumulative Gain (NDCG) [19] and Hit Ratio (HR)[20]. We randomly hold out one watched item, along with N unwatched items, and ask the model to rank them by rating. Intuitively, HR@K measures if the positive item ranked within the top K rated items, whereas NDCG@K also penalizes the lower rank of the positive item within those K items (the equations for the same can be found in the citations).

NDCG is calculated by dividing the discounted cumulative gain (DCG) of the ranked list by the DCG of the ideal ranked list, which is the list with the relevant items ranked in the most optimal order. NDCG ranges from 0 to 1, with higher values indicating better performance. The equation for the same is given below;

$$NDCG@K = 1/Z \sum_{i=1}^{K} [(2^{rel_i} - 1) / log_2(i + 1)$$

where K is the number of top-ranked items considered, rel_i is the relevance score of the item at the position i, Z is a normalization constant to ensure the NDCG is between 0 and 1.

Hit Ratio can be calculated as the fraction of users for which the system ranks the positive item within the top K, out of all the users for which the predictions were made, as shown below;

$$HR@K = \frac{Number\ of\ hits\ in\ top\ K\ recommendations}{Total\ Number\ of\ test\ cases}$$

We validated our results by using train, evaluation, and test splits of the dataset. Since the dataset is well established as a recommendation systems benchmark, we didn't expect any issues with our sample size. To maintain consistency with prior research utilizing the same dataset, we planned on keeping the outliers during testing, although we did remove the outliers during training to see if that improves our performance (there was no significant impact).

6 Experimentation

We ran multiple experiments combining model architectures and the losses. The code for all experiments has been open sourced and is publicly available.[1]

[1] *Link to repository: https://github.com/IamAdiSri/csci_5502_project*

6.1 Dataset

The Movielens dataset is available in multiple sizes, including 100k, 1M, 1B and more. We wrote parsers and preprocessors for the 100k and 1M sized datasets (these are available in the repository linked in the footnote), however, due to time and compute constraints we were only able to experiment with the 100k dataset.

After preprocessing the 100k samples, we found 943 unique users and 1682 unique items in the dataset. We generated 943 test samples (one per user) and generated 198,114 samples for training with RMSE and BCE losses, and we generated 100,000 samples for training with BPR loss.

6.2 Model Initialization and Parameters

We used the same parameters as described in the source paper for all our models. We used embeddings of size 40 for the MF model, size 16 for GMF model, and size 32 for the MLP model. The MLP model also used three hidden layers of sizes 32, 16, and 8 respectively.

The MF model weights were initialized with Gaussian sampling in the range (-0.05, 0.05) and the NCF model and its submodules' weights were initialized with Gaussian sampling in the range (-0.01, 0.01), as specified in the source papers.

6.3 Hyperparameter Tuning

For our architecture parameters such as number of layers, layer sizes and embedding sizes, we stuck with the hyperparameters described in the source papers. However, the optimizer hyperparameters such as learning rate, momentum and weight decay rate described in the papers did not work for us, and we had to use grid search to tune these ourselves.

Ideally, we would have replaced the grid search with Bayesian optimization (better performance and higher efficiency) to find all of the relevant hyperparameters (we are only searching for a small subset of our parameter space), but we were limited by the time and compute costs of running such prolonged experiments (200 experiments took around an 1.5 hours, however when considering every possible hyperparameter we reached approximately of 10^6 experiments).

The final parameters for each of the models can be found in the *<model_name>_modeling.ipynb* notebooks available in the repository.

6.4 Pretraining

The source literature for the NCF model recommends first pretraining the GMF and MLP models independently with BCE loss and then using the pretrained weights in the NCF model which is further finetuned. We tried a version with pretraining and one without.

7 Results and Analyses

	HR			NDCG		
	@1	**@5**	**@10**	**@1**	**@5**	**@10**
MF-RMSE	0.31	0.65	0.80	0.31	0.49	0.54
NCF-BCE	0.23	0.62	0.77	0.30	0.47	0.52
MF-BPR	0.34	**0.73**	**0.87**	0.34	**0.55**	**0.59**
NCF-BPR	**0.38**	0.38	0.38	**0.38**	0.38	0.38

The findings for each model along with the loss function used are given in the table above. We performed a comprehensive study comparing the performance of classical matrix factorization (MF) with neural matrix factorization (NCF) in recommendation systems, with evaluation over multiple metrics and training using multiple objectives. We found the results to be particularly insightful, and have described our observations below.

7.1 MF vs. NCF

As we can see, between the MF-RMSE and NCF-BCE which were built without change from their respective source material, the classical matrix factorization (MF-RMSE) outperforms neural matrix factorization (NCF-BCE), although the improvement was rather small. Furthermore, using BPR to train the two models leads us to the best performing MF-BPR model with the highest HR@5, HR@10, NDCG@5 and NDCG@10 scores. Overall, we can say that the classical method wins out over the newer one.

7.2 RMSE/BCE vs. BPR

The results were particularly interesting in that they demonstrate how impactful the loss functions were to the model performance. Models using RMSE and BCE displayed comparable levels of performance. However, there was a significant change in scores between the non-BPR models and those employing BPR. In fact, all of the top scores occur when we use BPR as the loss function and we can clearly see that it works better than the rest.

8 Conclusions

It is interesting to see that the classical matrix factorization model exhibited a significant advantage over its neural counterpart. This was an unexpected outcome, considering that there is a difference of 9 years between the publication of MF (2008) and NCF (2017), and that the prevailing perception is that of neural models generally surpassing classical models in handling complex data patterns. The disparity in performance suggests that non-neural models are still relevant, and can still outperform state-of-the-art neural models. Several factors could have possibly contributed to this outcome:

1. Dataset size disparity: The neural model we adapted was originally trained on Movielens 1M (containing ~1 million samples), whereas the classical MF model was trained on a dataset of 100k samples. In our study, both models were trained on the same 100,000 samples dataset. This discrepancy in training data volume could have disadvantaged the neural model, which may perform better with larger datasets.

2. Lack of content based features: The results also suggest that the efficacy of neural versus classical models can be context-specific, and that when the remaining features (user's age, movie's genre etc.) are added to the input, the neural model ends up performing better, as deep learning is a generally better solution for extracting learnable abstractions from high dimensional data.

3. Model complexity and optimization: Neural models, with their inherent complexity, are more sensitive to tuning when trained on smaller datasets, like the 100k sample size we used. Classical models, being simpler, are often more robust to the same in such scenarios.

5 Milestones

Our roadmap consisted of the following milestones;

1. **Data preprocessing and analysis:** We performed an in-depth exploration of the dataset, calculated basic statistics, visualized data distributions, and identified potential outliers or anomalies. This helped us gain insights into user behavior and

movie characteristics. The next step involved data preprocessing to transform the data for building recommendation models. We expected to finish this milestone by mid-October. **[Completed]**

2. **Model implementation and testing:** We implemented the recommendation models - Matrix Factorization and Neural Collaborative Filtering using suitable libraries or frameworks. The next step involved testing the model where we split the dataset into training and testing sets to evaluate the initial model performance. Next, we assessed the recommendation accuracy by utilizing common evaluation metrics like NDGC and HR. Finally, we fine-tuned the models' hyperparameters to optimize their performance. We intended to finish this milestone by mid-November. **[Completed]**

3. **Improving model performance:** We tried to modify our model to see if we can improve performance. We ended up using grid search to tune our hyperparameters and employing BPR loss. We intended to finish this milestone by the end of November. **[Completed]**

4. **Graph neural network:** As part of our feedback we were suggested to have an additional stretch goal, for which we planned to implement the current state-of-the-art Graph-based Hybrid Recommendation System [13], which utilized both implicit and explicit features. We were unable to meet this goal because running grid searches on the present architecture itself took a lot of time and we would have been unable to tune the graph neural network anyways because of its larger memory, compute and time footprint. **[Stretch Goal, Dropped]**

6 Challenges

In our work till now, we've faced the following challenges;

1. **Dealing with bias:** As explained in section 3.3, our data shows a number of misleading biases. There are a few approaches we can employ to get rid of these biases, for example through stratified sampling or completely dropping certain features, however it is likely that any method we employ will have a negative impact on the recommendation performance.

2. **Preprocessing:** Training, testing and evaluating recommendation models is a complex task, since the objective function for the model, i.e. pointwise loss, may not make for as interpretable an evaluation metric as NDCG or HR. Unfortunately the two of them require considerably different preprocessing to be carried out, while also tracking outliers to mitigate as much of the cold-start problem as possible.

3. **The cold-start problem:** While we haven't started modeling our data just yet, from our data analyses in section 3.3, we can see that our user-item interaction matrix is sparse and there are a lot of users and items with very few ratings, making them more difficult to generate recommendations for. This will be an ongoing problem to solve as we proceed with modeling.

4. **Discrepancies between reference papers:** The source paper for MF tested the model on Movielens 100k dataset and only provided the RMSE and precision scores for the model during evaluation. On the other hand, the source paper for NCF only tested the model on the Movielens 1M dataset and only provided the HR and NDCG scores for the model. Thus, we had no way to compare the models when starting the project.

5. **Missing reference details:** While we had source papers to refer to, the papers often missed key details from their implementation (optimizer hyperparameters were a big point of grief) that made it harder to duplicate their results.

6. **Lack of reference code:** We wanted to make a unified system where all the code was written in a single framework (Pytorch) and could be easily tested with the same dataset and training loops. We did succeed, but it was a challenge to do so because we could not find the source implementation for MF, and the NCF model was implemented in Keras. Keras due to its nature of being a high-level ML API hid quite a few details of the implementation as well, which we had to manually figure out and add to our Pytorch code.

7 Future Work

In the future, we aim to refine our approach by incorporating insights from the following strategies:

1. **Dataset expansion**: Evaluate models on the MovieLens 1M dataset to assess scalability and performance on larger datasets.

2. **Comprehensive grid search**: Extend parameter tuning efforts beyond optimizers to include model parameters, enhancing overall model performance.

3. **Content-based feature integration**: Investigate the incorporation of content-based features into the recommendation pipeline, potentially improving the accuracy of predictions.

4. **Content-based and hybrid models**: Explore the implementation of content-based and hybrid models to enhance the recommendation system's ability to capture diverse user preferences and content characteristics.

REFERENCES

[1] Joseph A. Konstan, Bradley N. Miller, David Maltz, Jonathan L. Herlocker, Lee R. Gordon, and John Riedl. 1997. GroupLens: applying collaborative filtering to Usenet news. Commun. ACM 40, 3 (March 1997), 77–87. https://doi.org/10.1145/245108.245126

[2] Xue, Feng, et al. "Deep item-based collaborative filtering for top-n recommendation." *ACM Transactions on Information Systems (TOIS)* 37.3 (2019): 1-25.

[3] Gao, Min, Zhongfu Wu, and Feng Jiang. "Userrank for item-based collaborative filtering recommendation." *Information Processing Letters* 111.9 (2011): 440-446.

[4] Roy, D., Dutta, M. A systematic review and research perspective on recommender systems. *J Big Data* **9**, 59 (2022). https://doi.org/10.1186/s40537-022-00592-5

[5] Billsus, Daniel, and Michael J. Pazzani. "Learning collaborative information filters." *Icml*. Vol. 98. 1998.

[6] Gábor Takács, István Pilászy, Bottyán Németh, and Domonkos Tikk. 2008. Matrix factorization and neighbor based algorithms for the netflix prize problem. In Proceedings of the 2008 ACM conference on Recommender systems (RecSys '08). Association for Computing Machinery, New York, NY, USA, 267–274. https://doi.org/10.1145/1454008.1454049

[7] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. 2001. Item-based collaborative filtering recommendation algorithms. In Proceedings of the 10th international conference on World Wide Web (WWW '01). Association for Computing Machinery, New York, NY, USA, 285–295. https://doi.org/10.1145/371920.372071

[8] G. Linden, B. Smith and J. York, "Amazon.com recommendations: item-to-item collaborative filtering," in *IEEE Internet Computing*, vol. 7, no. 1, pp. 76-80, Jan.-Feb. 2003, doi: 10.1109/MIC.2003.1167344.

[9] Yehuda Koren. 2008. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In SIGKDD.

[10] Chai, Junyi, et al. "Deep learning in computer vision: A critical review of emerging techniques and application scenarios." *Machine Learning with Applications* 6 (2021): 100134.

[11] Lopez, Marc Moreno, and Jugal Kalita. "Deep Learning applied to NLP." *arXiv preprint arXiv:1703.03091* (2017).

[12] Wang, Fei, Lawrence Peter Casalino, and Dhruv Khullar. "Deep learning in medicine—promise, progress, and challenges." *JAMA internal medicine* 179.3 (2019): 293-294.

[13] Zamanzadeh Darban, Zahra, and Mohammad Hadi Valipour. "GHRS: Graph-Based Hybrid Recommendation System with Application to Movie Recommendation." Expert Systems with Applications, vol. 200, Aug. 2022, p. 116850. Crossref, https://doi.org/10.1016/j.eswa.2022.116850.

[14] Kim, Daeryong, and Bongwon Suh. "Enhancing VAEs for Collaborative Filtering." Proceedings of the 13th ACM Conference on Recommender Systems, Sept. 2019. Crossref, https://doi.org/10.1145/3298689.3347015.

[15] Gao, Yunfan, et al. "Chat-rec: Towards interactive and explainable llms-augmented recommender system." *arXiv preprint arXiv:2303.14524* (2023).

[16] Jia Rongfei, Jin Maozhong, & Liu Chao. (2010). A new clustering method for collaborative filtering. 2010 International Conference on Networking and Information Technology. doi:10.1109/icnit.2010.5508465

[17] F. Maxwell Harper and Joseph A. Konstan. 2015. The MovieLens Datasets: History and Context. ACM Transactions on Interactive Intelligent Systems (TiiS) 5, 4: 19:1–19:19. https://doi.org/10.1145/2827872

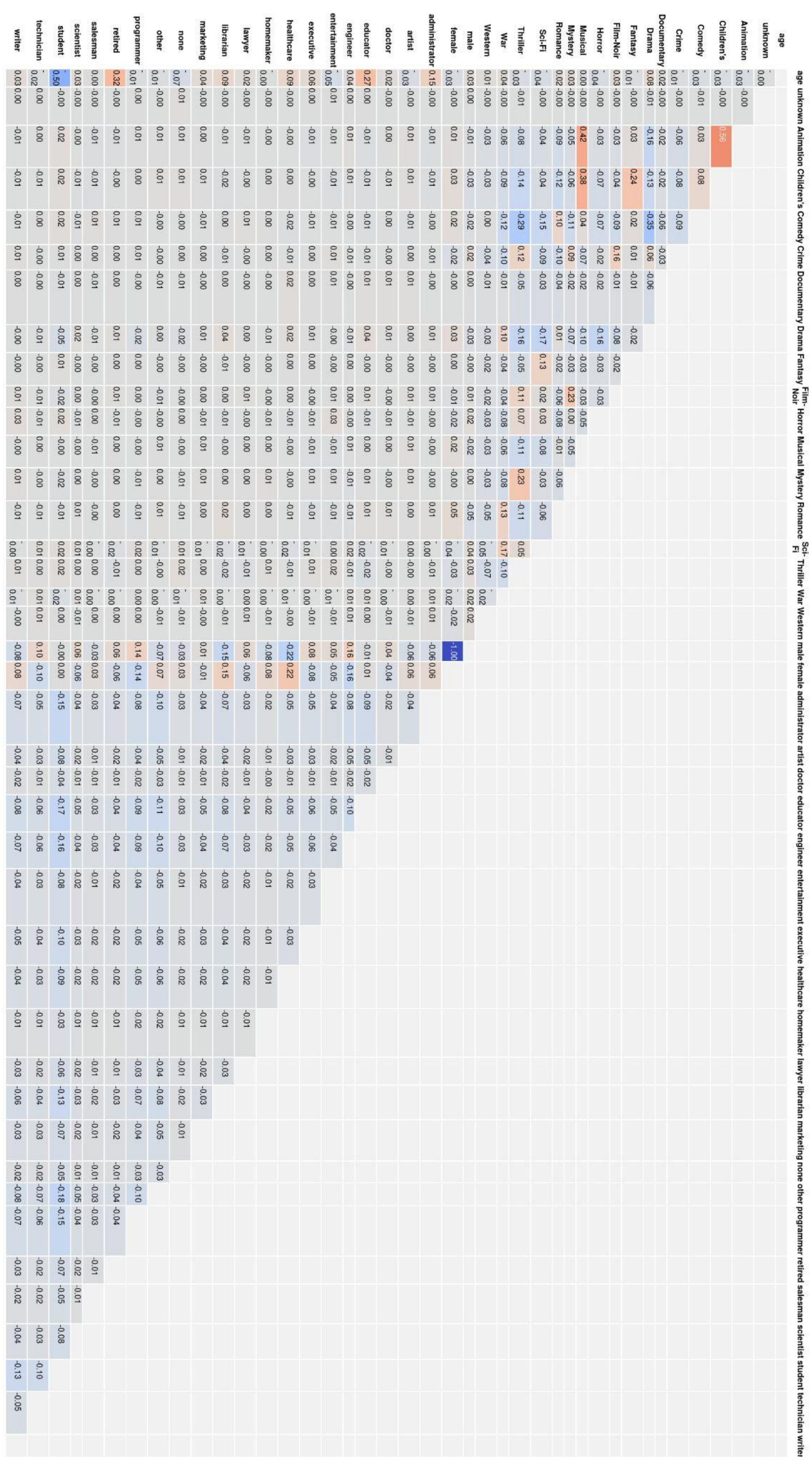
[18] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural Collaborative Filtering. In Proceedings of the 26th International Conference on World Wide Web (WWW '17). International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, CHE, 173–182. https://doi.org/10.1145/3038912.3052569

[19] https://towardsdatascience.com/demystifying-ndcg-bee3be58cfe0

[20] https://towardsdatascience.com/ranking-evaluation-metrics-for-recommender-systems-263d0a66ef54

[21] https://en.wikipedia.org/wiki/Cold_start_(recommender_systems)

[22] Yurii Nesterov. Introductory lectures on convex optimization: A basic course, volume 87. Springer Science & Business Media, 2013.


age
unknown
Animation
Children's
Comedy
Crime
Documentary
Drama
Fantasy
Film-Noir
Horror
Musical
Mystery
Romance
Sci-Fi
Thriller
War
Western
male
female
administrator
artist
doctor
educator
engineer
entertainment
executive
healthcare
homemaker
lawyer
librarian
marketing
none
other
programmer
retired
salesman
scientist
student
technician
writer