ST312 Group Project

Comparative Analysis of Collaborative Filtering Algorithms on MovieLens 100K Dataset

Word count: 4898

Candidate ID: 24155, 24588

Date: 25 April 2024

Abstract

People make decisions when using the internet, like choosing what movies to watch, websites to visit, what information to select from search results, or which products to buy online. Our applied statistics report focuses on recommending interesting movies to users. The huge amount of available movies makes it hard for users to find what they need. Recommender systems with implicit or explicit data have been created to help with this problem by helping users in making decisions and finding relevant data, while this report focuses on explicit data. These systems predict what users will rate. In this report, we compared different ways of using matrix factorisations and transformers to see how well they could predict what movies users would like, when there's a lot of data and many users, and sometimes when there's a cold start problem too. This report compares two matrix factorisation methods with three transformer implementations that help users choose interesting movies and how well they perform. The report only includes collaborative filtering methods to find similarities between users. The advantages and disadvantages of each approach, limitations and suggestions for future research are included.

1. Introduction

This project aims to find the best recommendation system. The project's research question can be divided into three parts. We firstly assess how well traditional methods perform, secondly we compare whether less intuitive techniques such as transformers outperform basic factorisation methods. Thirdly, we investigate if any improvements can be made to the transformers architecture.

The report uses 100k movielens dataset, which contains 100,836 ratings from 610 users on 9,742 movies on a scale of 0.5-5 stars in half-star increments. Our dataset is a relatively new dataset compared to the more commonly used 1M dataset. It spans from 1996 to 2018, which allows analysis of rating patterns over a long time. Larger datasets allow evaluation on more realistic data sizes, but our report only uses 100k ratings dataset due to its more recent cutoff year and our computation resources limit with a single GPU.

This report firstly provides an overview of past findings in this field, then we describe our dataset, and compare five different models after preprocessing our data, before making a conclusion. Specifically matrix factorisation with stochastic gradient descent and three different Transformer-models are implemented for this report. Matrix factorisation is further divided into two parts, stochastic gradient descent with and without bias terms. Three Transformer-based models are employed. These three transformer models have contained different embeddings from the information of user-item id, rating and movie genres for positional encoding to predict rating, capturing more personalised patterns based on users behaviour in the real dataset.

Our models predict continuous ratings unseen in the training dataset, rather than treating it as a multiclass classification problem. We looked at how well the systems worked when they predicted the most recent two ratings by a user sorted by timestamp.

2. Literature Review

Recommender systems can be categorised into three main categories based on the type of input data and methods used: content-based, collaborative filtering, and knowledge-based (Jannach et al., 2010; Kuzelewska, 2014). Content-based recommender systems use attribute vectors of items created from textual descriptions. The similarity between items is used to recommend new items to users (Jannach et al., 2010).

Collaborative filtering recommender systems find similar users or items from archives of user behaviour data, which is movie ratings in our project's dataset. It can be further divided into memory-based, which calculates recommendations directly from the data, and model-based, which builds a model of the data to generate recommendations (Kuzelewska, 2014; Sarwar et al., n.d.).

Knowledge-based recommender systems take a different approach by using explicit knowledge about items, such as movie genres and user ages, rather than past behaviour data. This approach is particularly suitable for recommending infrequently purchased items such as cameras (Jannach et al., 2010).

Hybrid approaches are a newer way to combine multiple methods and overcome the limitations of individual ones. Hybrid systems can address the cold-start problem of pure collaborative filtering systems, which occurs when no data is available for new users or items (Kużelewska, 2014). Clustering algorithms into collaborative filtering recommenders can also improve scalability (Sarwar et al., 2002; Kużelewska, 2014). The hybrid system demonstrates performance comparable to memory-based collaborative filtering while being more computationally efficient.

For matrix factorisation methods (B.Thorat et al., 2015), Stochastic Gradient Descent (SGD) has been widely applied to optimise the matrix factorisation process in recommender systems due to its efficiency in handling large datasets. SGD updates the model parameters incrementally. Traditional models such as simple SVD and memory-based collaborative filtering struggle with sparse data and high dimensionality (Ricci et al., 2011). Recent studies incorporate regularisation terms to prevent overfitting, improving the robustness of the predictions in sparse datasets (Hidasi & Karatzoglou, 2018).

For self-attention architectures, they can capture relationships between users and items (Geng et al., 2021). A medium blog post (Hughes, 2022) demonstrated that transformers, combined with ratings timestamp improves the accuracy of movie ratings prediction. Transformers have improved the capability of recommender systems to process and predict complex user-item interactions efficiently. Transformers adapt to different datasets without significant architectural changes. For instance, incorporating basic identifiers such as movie IDs and user IDs along with positional encodings allows the model to capture temporal relationships in user behaviour, which is important for understanding preferences (Kang et al., 2018). Past user interactions as a sequence can also be added to improve future rating prediction by recognizing patterns in historical data. Such sequential modelling mimics the human cognitive process of relying on memory, thereby refining the predictions (Wu et al., 2020) . Lastly, the addition of item features introduces a rich layer of item-specific information. This helps the transformer in tailoring recommendations that resonate more closely with individual tastes and preferences (Jin et al., 2022).

However, the Hamburger paper (Geng et al., 2021) suggests that simple matrix factorisation techniques may still provide a computationally lighter alternative to model global dependencies in user-item interactions.

3. Dataset, Methods

3.1 Data

MovieLens (*The MovieLens Datasets: History and Context: ACM Transactions on Interactive Intelligent Systems: Vol 5, No 4*, n.d.) is a research-based movie recommendation system developed by the GroupLens Research Lab at the University of Minnesota. This platform allows users to rate movies and provides recommendations for other films that users might enjoy based on these ratings. MovieLens offers several datasets that typically include user ratings, tagging data, and metadata about the movies. We have selected the most recent dataset which involves activities such as 5-star ratings and free-text tagging performed by 610 randomly selected users who rated at least 20 movies each. Users contributed to a total of 100,836 ratings and 3,683 tag applications across 9,742 movies. It spans from March 29, 1996, to September 24, 2018. The dataset does not include any demographic information and features about the users, unlike the older and widely researched 1M movielens dataset.

3.2 Forming and Definition

The dataset has four files: links.csv, movies.csv, ratings.csv, and tags.csv. These files contain links to IMDb, detailed movie information, user ratings, and tagging data respectively. Detailed information in the **Table 1** below:

Table 1:			
File name	Column name	Data types	Description

ratings	userId	integer	user identifier in the dataset
	movieId	integer	movie identifier in the dataset
	rating	float	movie rating given by the user, range (0.5-5)
	timestamp	integer	the time of rating, recorded in UNIX time format
tags	userId	integer	user identifier in the dataset
	movieId	integer	movie identifier in the dataset
	tag	string	user-defined description of the movie
	timestamp	integer	the time of rating, recorded in UNIX time format
movies	movieId	integer	movie identifier in the dataset
	title	string	movie name and released time
	genres	string	movie genres
links	movieId	integer	movie identifier in the dataset
	imdbId	integer	Identifier for movies used by imdb
	tmdbId	integer	Identifier for movies used by tmdb

Table 1 offers a basic understanding of our dataset. User and movie IDs serve as primary keys, enabling merging on these keys. Ratings and movie genres, as the most valuable additional information, will subsequently be embedded in the model.

The median number of ratings per user is 70. It is a skewed distribution with a relatively small number of highly active users contributing a large portion of the ratings. On the movie side, the data has sparsity patterns. The maximum number of ratings for a single movie is 329, while the minimum is just 1 rating. The median number of ratings per movie is only three. Most movies have very few ratings. 19 genres of movie are involved in the dataset and movies can have more than one genre.

3.3 Methods

We explore the dataset using histograms, line graphs, and cloud plots to understand the data.

Matrix factorisation and Transformers are used for predicting movie ratings comparison. Matrix factorisation is used for dimensionality reduction and finding the latent matrices in recommendation systems. Transformer as a deep learning technique has a self-attention mechanism, which finds patterns within user-item interactions and takes into account their temporal sequence. Transformers will generate 3 results of predictive accuracy since we created three different transformer-models that include different embeddings (rating records and movie genres).

Python packages we used in the project are pytorch, numpy, pandas, statsmodels, wordcloud, matplotlib, and seaborn.

4. Exploratory Data Analysis

This section of the report has eight plots to provide a preliminary overview of our dataset.

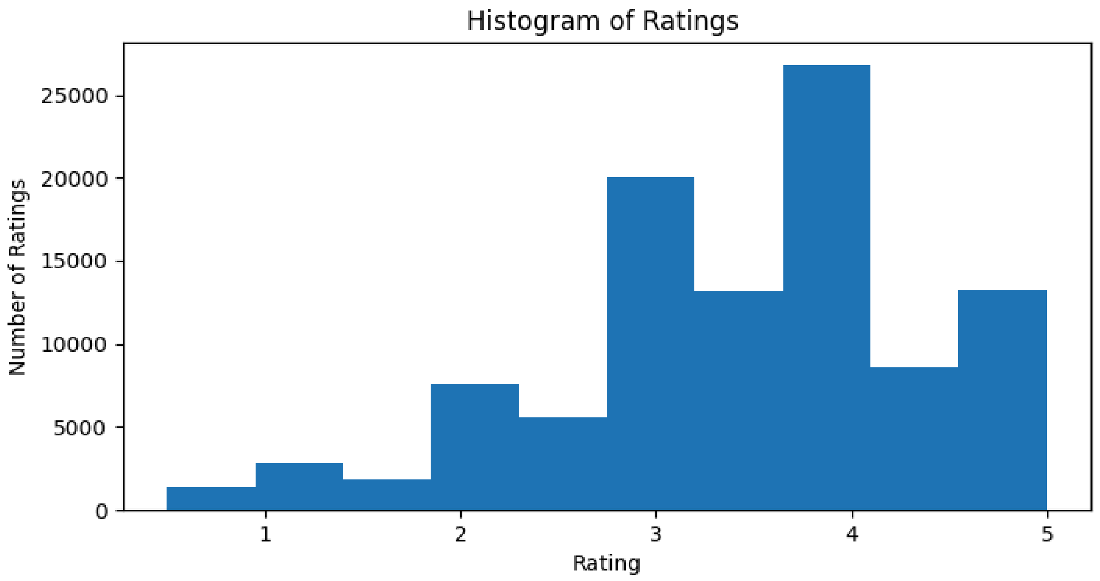


Figure 4.1

Figure 4.1 shows the distribution of movie ratings given by users. The histogram is a leftward skewed distribution with a high peak at 4 star ratings, followed by 3 star ratings. We find an interesting pattern that users are less likely to give half ratings compared to whole point ratings. Inferring from the plot, users also rate movies more positively overall rather than one or two starts.

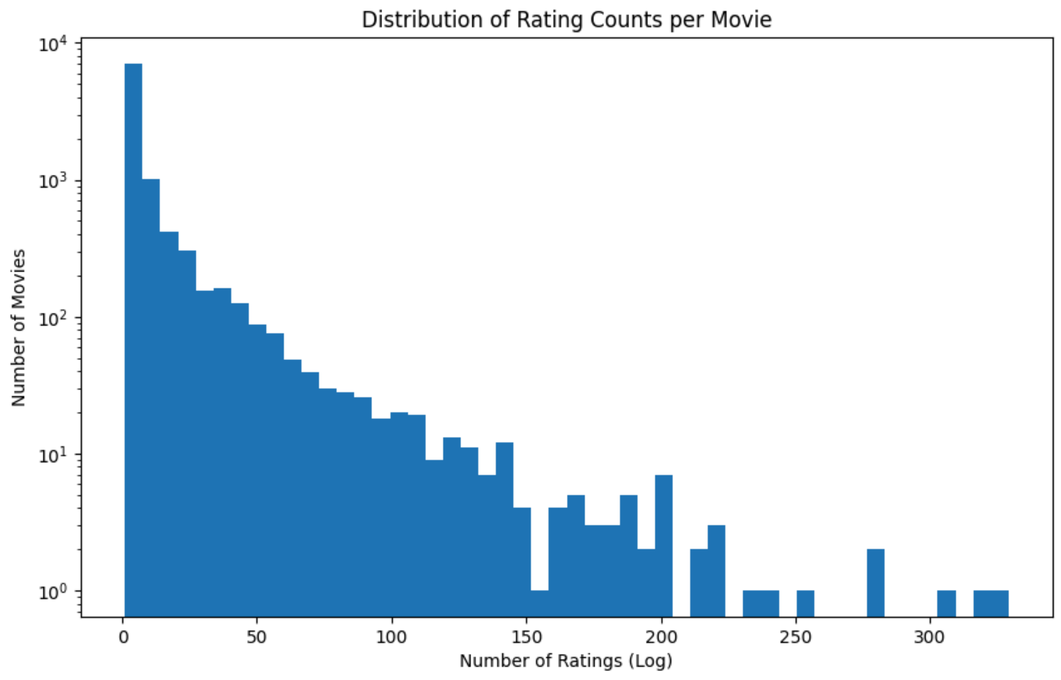


Figure 4.2

Figure 4.2 shows most movies have a low number of ratings (less than 100), and few movies have a large number of ratings (over 200). Rightward skewed distribution shows that a small number of popular movies account for almost all of the ratings.

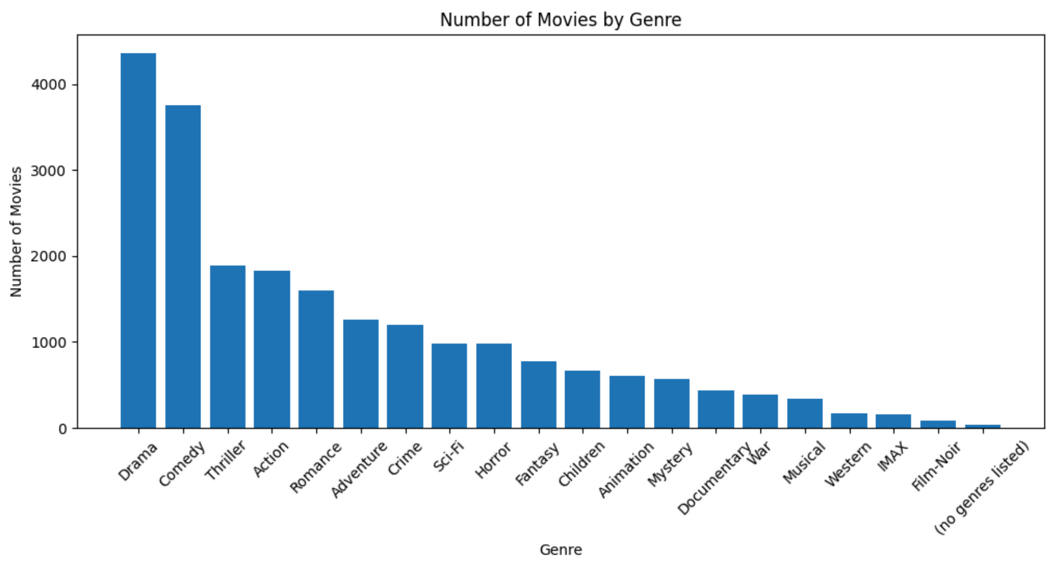


Figure 4.3

Figure 4.3 shows the number of movies in the dataset for each genre. the "Drama" genre has a lot of movies, followed by "Comedy" and "Thriller". Genres like "Documentary", "War", "Musical", and "(no genres listed)" have relatively fewer movies. Most movies have a low number of ratings (less than 100), while very few movies have a large number of ratings (over 200).

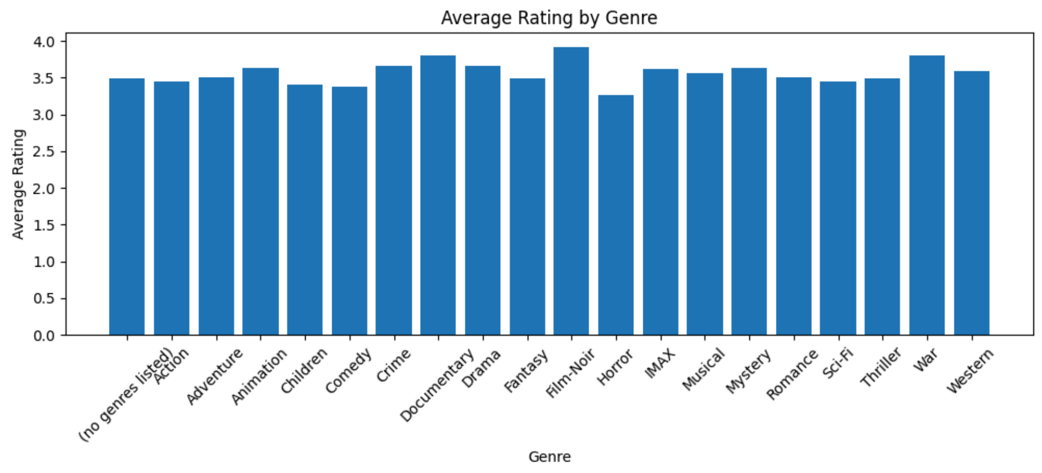


Figure 4.4

Figure 4.4 shows that genre information themselves are not effective predictors of movie quality. Genres like "Western", "Film-Noir", "War", and "Documentary" have slightly higher average ratings compared to genres like "Animation", "Children's", and "Comedy".

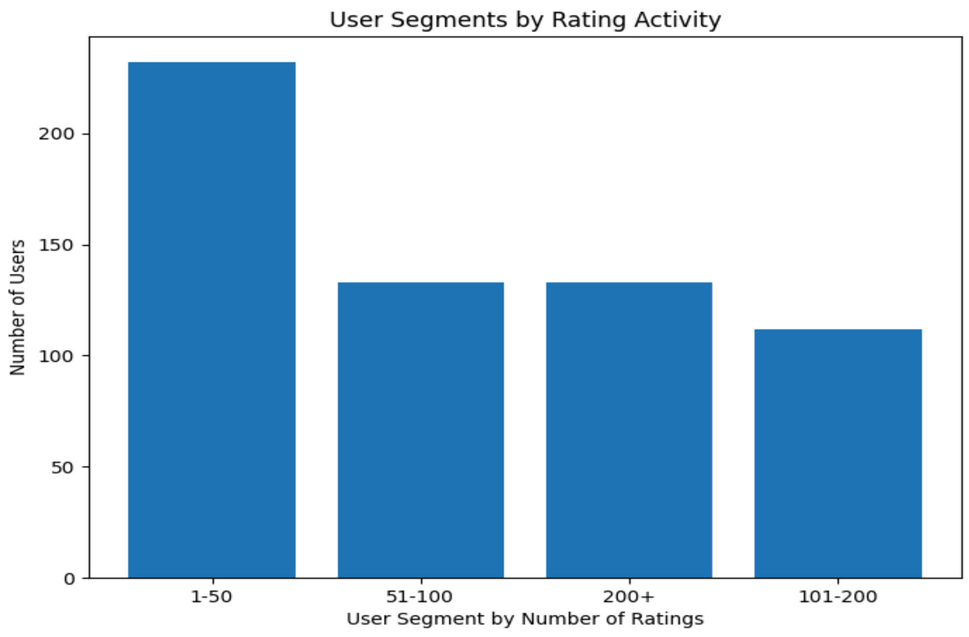


Figure 4.5

Figure 4.5 bar chart categorises users into four segments based on their number of ratings. Most users are in the "1-50" segment, with fewer users in the "51-100", "200+", and "101-200" segments. Users with limited data and the cold-start problem for them might be an issue during the subsequent matrix factorisation step.

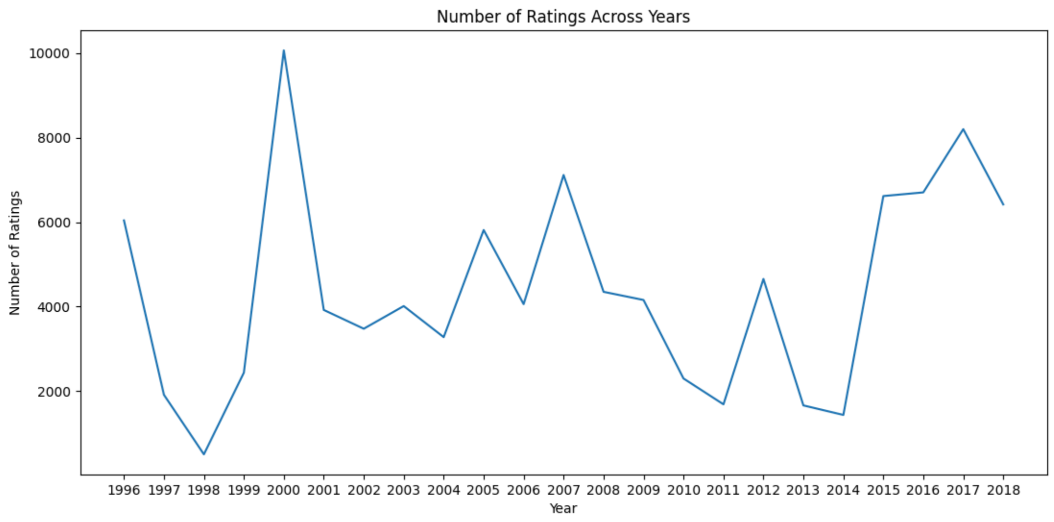


Figure 4.6

Figure 4.6 is a line chart, it has the number of ratings provided by users across different years from 1996 to 2018. Year 2000 saw a peak in rating activity, possibly related to the release of popular movies or an overhaul of movielens website.

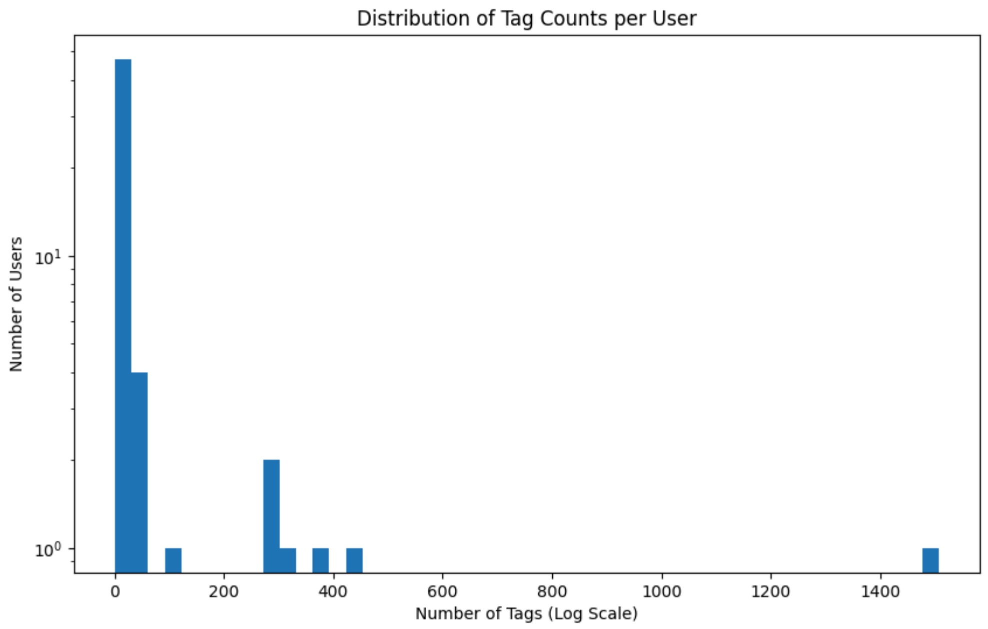


Figure 4.7

Compared to the rating distributions, the seventh tag count distribution is more skewed even with a log scale on the x axis. Few users created tags in the first case, and most of these users created a small number of tags (less than 200). Very few users wrote a large number of tags (over 1000). Our group Tags data looks less useful for predicting movie ratings in a movie recommendation system because the data is very incomplete and sparse.

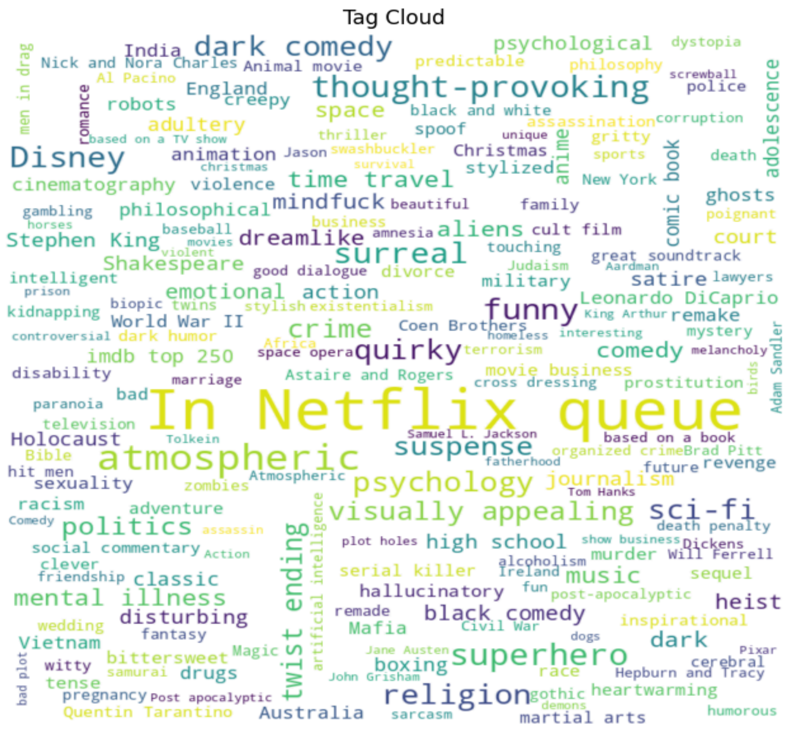


Figure 4.8

Finally, the tag cloud shows the most frequently written tags by users to label movies. The most popular tag is the user's intention to watch the movie on Netflix. Popular tags include genres like "comedy", "drama", emotional descriptors like "dark", "psychological", and other descriptive words like "thought-provoking", "quirky", and "atmospheric".

5. Models

5.1 Data Preprocessing

Our group mainly uses two files, one is user ratings and the other is movie details for our model building. The ratings and movies datasets are initially merged on the movie identifier. The combined DataFrame has the rating information and the corresponding movie titles instead of only movie IDs by the ratings file.

For our train-test split, we get the last two ratings for each user by timestamp value. We then selected the last two ratings for each user. Another function is then used to mark these last two ratings as the validation set. The original DataFrame is then divided into two separate DataFrames: one for training, containing all ratings except the last two per user, and another for validation, comprising only the last two ratings from each user. Train dataset has **99616** ratings, while the validation dataset has **1220** ratings.

We created two lookup tables to encode user and movie identifiers as input features for the models. They map the unique user identifiers and movie titles to numeric indices starting from one.

5.2 Baseline Model

Inferring whether a model's performance is due to luck or actual architecture is difficult without a baseline model. A median rating (3.5) is initially calculated from the training set for all rows in the validation set. The performance of this baseline model is then assessed using three common metrics: Mean Absolute Error (MAE) is 0.897, Mean Squared Error (MSE) is 1.197, and Root Mean Squared Error (RMSE) is 1.094. These are benchmarks if all predictions are 3.5.

5.3 Matrix Factorisation

5.3.1 Stochastic Gradient Descent Matrix Factorisation

Matrix factorisation decomposes the sparse user-item ratings matrix into lower dimensional user and item matrices. Traditional Singular Value Decomposition does not work for our dataset because the dataset has missing values. The predicted rating for the last 2 ratings by a user is calculated as the dot product between the corresponding user and item vectors: $\hat{r}_{ui} = \mathbf{x}_u^T \mathbf{y}_i$. In the formula, $\hat{r}_{ui}$ is the predicted rating for user u on item i, $\mathbf{x}_u$ is the user latent vector, and $\mathbf{y}_i$ is the item latent vector. However, this basic formulation does not account for the fact that some users may tend to give higher ratings on average while some items may receive higher ratings overall. To address this, bias terms can be introduced for each user and item: $\hat{r}_{ui} = \mu + b_u + b_i + \mathbf{x}_u^T \mathbf{y}_i$ (Zhang et al., 2012). In this formula, μ is the global average rating, b_u is the user bias term, and b_i is the item bias term. These bias terms are the deviations from the global average for each specific user and item.

The model parameters (user/item vectors and biases) are learned by minimising the regularised squared error loss function using Stochastic Gradient Descent (SGD):

$$L = \sum_{r_{ui} \in R_{train}} (r_{ui} - \hat{r}_{ui})^2 + \lambda \left(b_u^2 + b_i^2 + ||\mathbf{x}_u||^2 + ||\mathbf{y}_i||^2 \right)$$

where R_{train} is the set of known ratings in the training set and λ is the regularisation parameter to prevent overfitting.

At each SGD iteration, the parameters are updated based on the gradient of the loss with respect to each parameter. The update rules for a user bias term and user latent vector are:

$b_u \leftarrow b_u - \alpha\left(2(r_{ui} - \hat{r}_{ui}) + \lambda b_u\right)$

$\mathbf{x}u \leftarrow \mathbf{x}u - \alpha\left(2(rui - \hat{r}ui)\mathbf{y}_i + \lambda \mathbf{x}_u\right)$ where α is the learning rate.

The key hyperparameters in two SGD models are number of latent factors is 120, learning rate is set at 0.02, number of epochs is 50, and batch size is 512, regularisation is set at a small number of 0.1 for user/item biases and vectors because overfitting is not a significant issue after trial and errors with the regularisation term.

The models were trained on all ratings excluding the last 2 ratings per use. Last two ratings were held out for validation as shown in the data preprocessing section of the report. The loss function was Mean Squared Error (MSE) and evaluation metrics also included Mean Absolute Error (MAE).

In this part, we aim to compare With vs Without Bias performance for our matrix factorisation models. After 50 epoch (Appendix 2), the model with bias terms clearly outperformed the one without bias (Appendix 1), The model with bias has a best validation MSE of 2.046 and a best validation MAE of 1.123, while the model without bias has a best validation MSE of 2.310, and a best validation MAE of 1.186. The bias model achieves a 11% lower MSE and 5% lower MAE compared to the model without biases. The learning curves also show the bias model's training and validation losses decrease faster and to a lower value than the regular model. Both models' training and learning loss decrease smoothly with each epoch, while the validation loss witnesses more fluctuations. The train-test loss gap also widened during training.

The importance of user and item biases is shown by this performance gap. By modelling the tendency for some users to rate higher/lower than others and some items to receive higher/lower ratings on average, the biassed model can better capture patterns in the data and make more accurate predictions. Intuitively, the biases provide a personalised "baseline" for each user-item pair on top of which the matrix factorisation can uncover more subtle preference patterns.

We have shown that extending matrix factorisation with user and item biases is an effective technique to significantly improve rating prediction accuracy, as evidenced by the MSE reduction on this dataset. The bias terms help account for systematic user and item effects, allowing SGD to learn better latent representations. Matrix factorisation models are significantly better than our baseline model above.

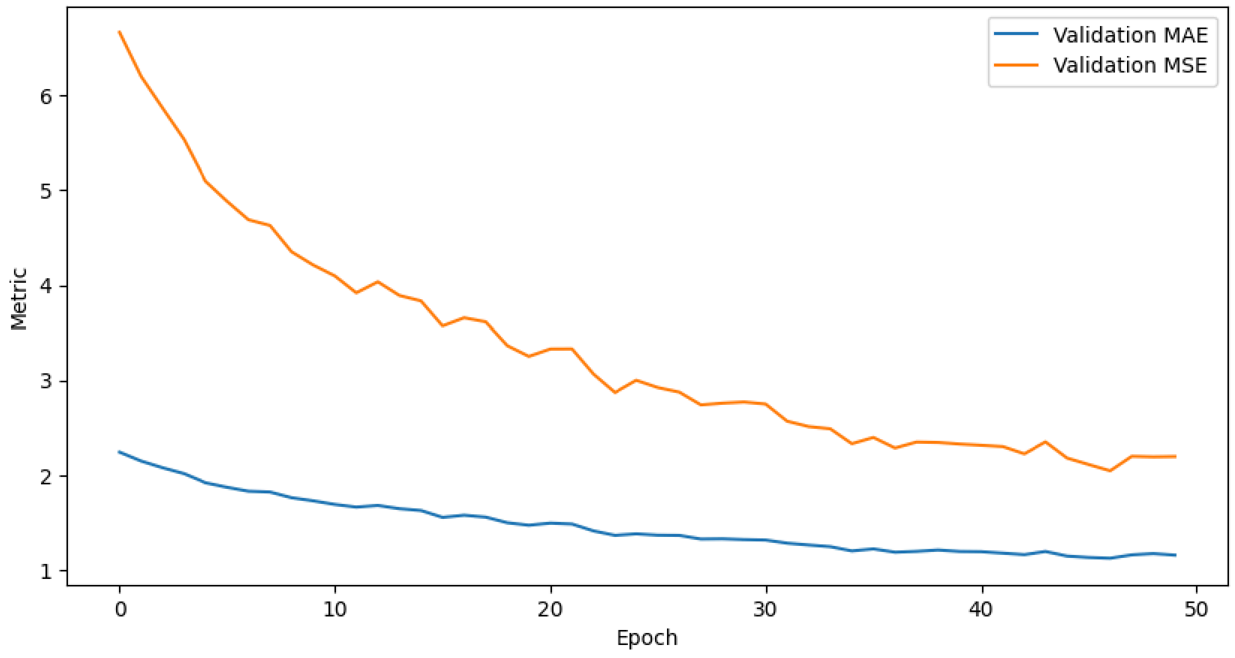


Figure 5.1: Validation MAE and MSE for SGD Factorisation with Bias

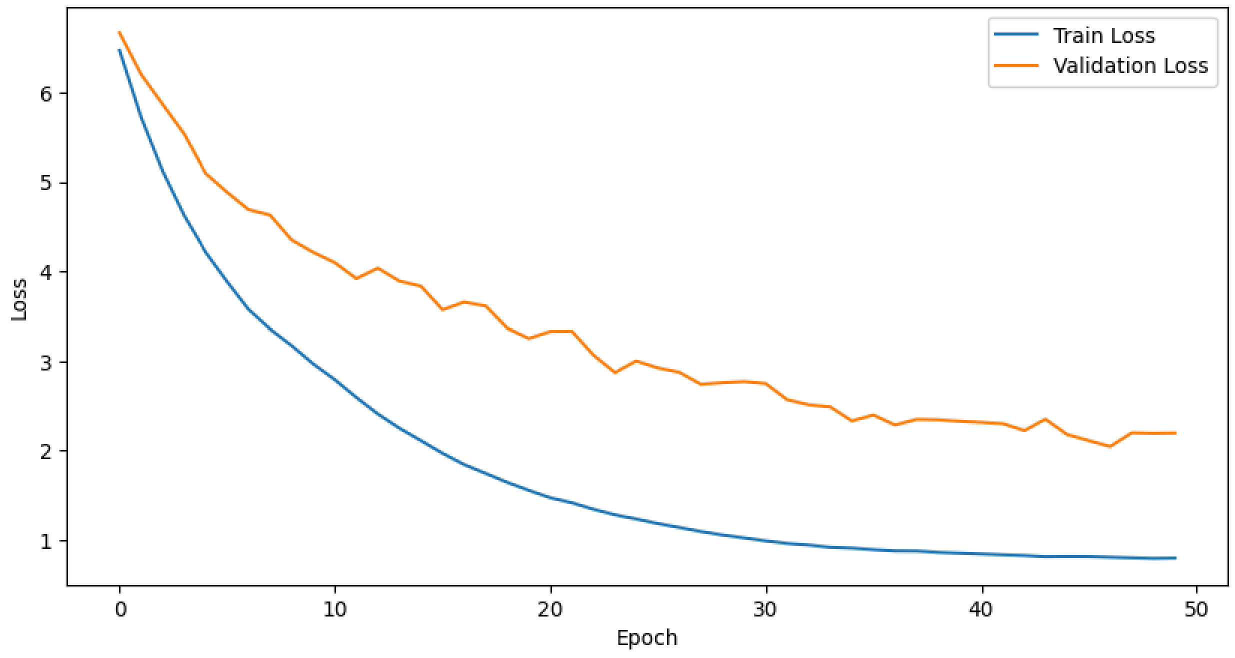


Figure 5.2: Training Loss and Validation Loss (MSE) for SGD Factorisation with Bias

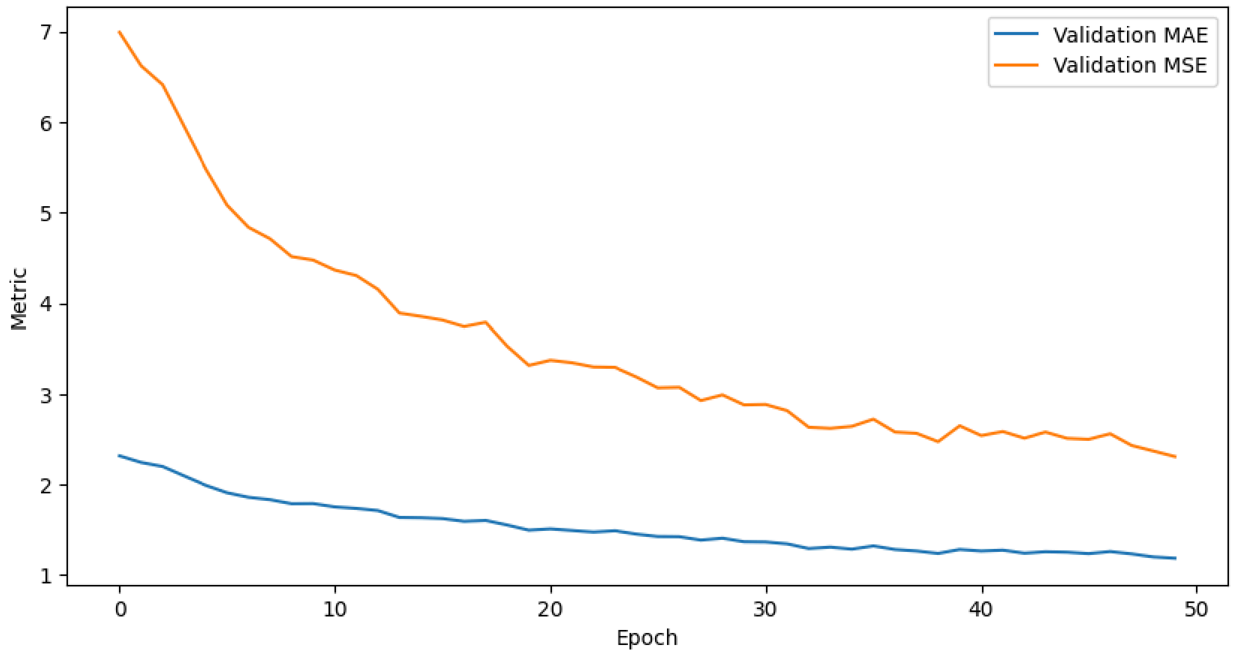


Figure 5.3: Validation MAE and MSE for SGD Factorisation without Bias

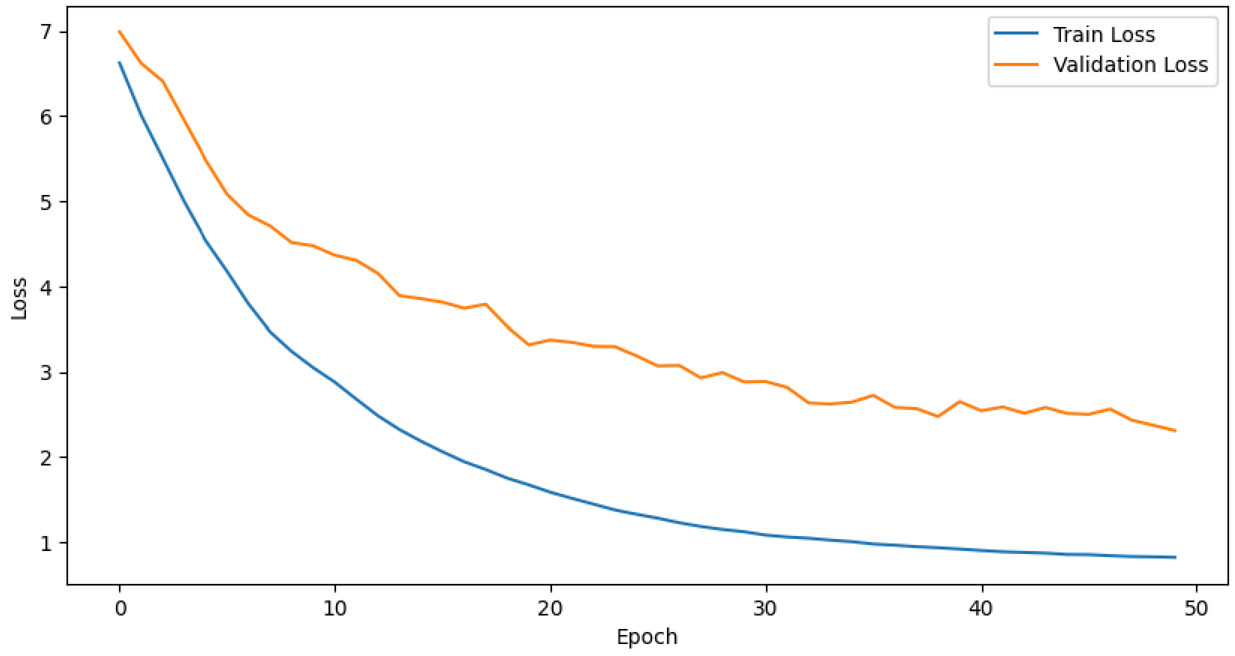


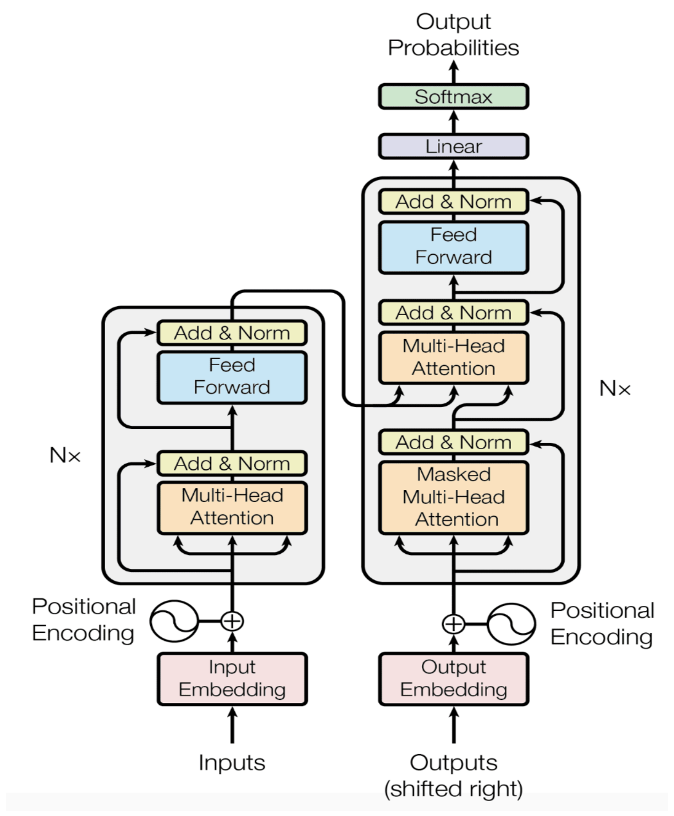
Figure 5.4: Training Loss and Validation Loss (MSE) for SGD Factorisation without Bias

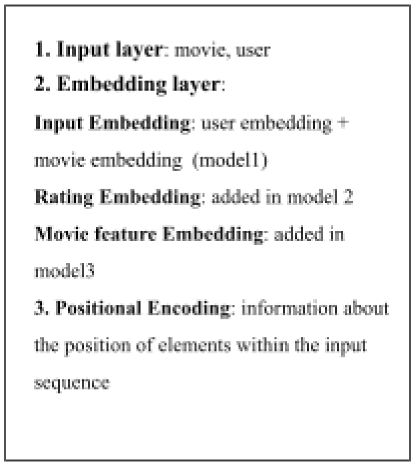
5.4 Transformers

5.4.1 Justification of Transformer

This project then turns to the transformer model, known for its self-attention mechanism that can adeptly process sequences of data. The hypothesis supporting this choice is that the sequential order of movie ratings — a digital footprint of a user's viewing journey — could yield critical insights into their preferences. Transformers, with their ability to handle sequential embeddings, might view the temporal progression of user interactions. This is probably significant when predicting movie ratings, as the order may reflect evolving tastes or serial consumption patterns, such as watching a sequence of related films. By encoding this sequential data, we anticipate that the transformer model will uncover potential behavioural patterns that can enhance recommendations.

5.4.2 Transformer Architecture, Embeddings




1. Input layer: movie, user

2. Embedding layer:

Input Embedding: user embedding + movie embedding (model1)

Rating Embedding: added in model 2

Movie feature Embedding: added in model3

3. Positional Encoding: information about the position of elements within the input sequence

The structure of the Transformer architecture taken from 'Attention Is All Need(Vaswani et al., 2017)'

The diagram shows the basic architecture of the transformer. Transformers work by processing data through a series of self-attention mechanisms, which enable the model to consider the entire input sequence and determine the importance of each part in relation to the rest. This is enriched by positional encodings, which give the model information about the order of items in the sequence.

In the context of movie recommendation systems, transformers can utilise these techniques to analyse sequences of user-item interaction. They can discern complex patterns in viewing habits and understand how the context of previous movie ratings influences preferences of future movies. This is why transformers can be adapted for recommendation systems: they are excellent at handling sequential data and can capture both user-specific preferences and broader watching trends.

From the diagram, transformers have a component named 'Encoder' ,which can process the input sequence (represented by the left block in the diagram). Each encoder layer within the transformers contains two sub-layers: a multi-head self-attention mechanism and a position-wise feed-forward network (FFN). The encoder layer makes use of positional encoding, allowing the model to focus on the full contents of the input sequence at each position. When handling a movie recommendation system, positional encodings enable the model to recognize viewing sequences, such as a user who might watch action movies followed by science fiction, and this pattern can help the model predict the next movie that the user might be interested in.

Overall, the encoder portion allows us to encode additional context into the learned embeddings for each movie, and then using a fully connected neural network to make the rating predictions. So we created three transformer-models which contains different learned embeddings:

Figure 5.5:

Models	Movie Embedding (MovieId)	User Embedding (UserId)	Rating Embedding (Movie rating history)	Movie Genre Embedding (movie genres)
Model 1	✓	✓		
Model 2	✓	✓	✓	
Model 3	✓	✓	✓	✓

5.4.3 Three Transformer-models Results

Train dataset has 99616 ratings, while the validation dataset has 1220 ratings. Last two ratings were held out for validation as shown in the data preprocessing section of the report. The loss function was Mean Squared Error (MSE) and evaluation metrics also included Mean Absolute

Error (MAE). The major parameters in model building are learning rate (0.002), number of epochs (15) and batch size (512). Three models have the same parameters. Early stopping rule is applied in the model building to get the best results and save computational resources. Furthermore, we set the length of input sequence to 10 which allows the Transformer to capture patterns and dependencies within this fixed subset of a user's interactions, which can then be used to predict their preferences or the ratings they might give to other movies.

The first Transformer model1, depicted in **Figure 5.6**, presents a dichotomy in its training and validation performance over seven epochs. The training loss demonstrates a consistent decrease, reflecting the model's capacity to learn effectively from the data. Conversely, the validation loss experiences an initial decline at epoch 2, stabilising thereafter, which could signal the model's limitations in generalising to unseen data. Additionally, the right graph portrays a gradual reduction in both validation MAE and MSE, with the MAE reaching a low of 0.75 and the MSE bottoming out around 0.94 at epoch 2. These figures suggest that while the model refines its predictions with training, the plateau in validation metrics may indicate a potential overfit, warranting further strategies to enhance its predictive robustness. So we have the model2 which has rating history as an embedding to positional encoding process.

Figure 5.6:

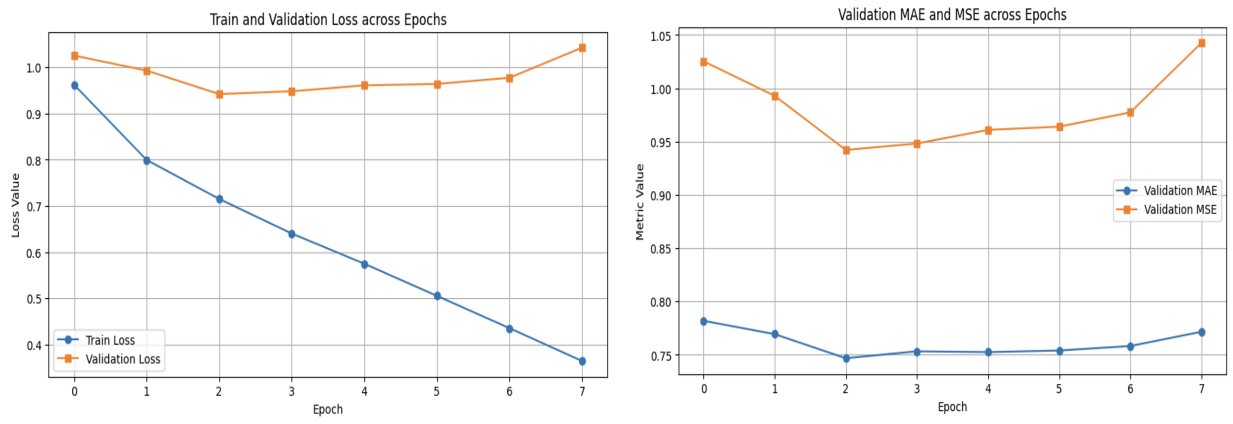


Figure 5.7 reveals the performance dynamics of our Transformer model2. Over seven epochs, the training loss steadily diminishes, indicating effective learning from the training data with a drop from approximately 1.0 to about 0.4. However, the validation loss demonstrates initial improvement, reaching a lowest point at epoch 2 before plateauing and increase, which suggests the model's reduced adaptability to new data. This plateau might point to the onset of overfitting. Correspondingly, the validation MAE and MSE exhibit minimal fluctuation post-epoch 2, with the MAE stabilising near 0.73 and the MSE vibrating slightly above 0.89. These metrics imply that while the model’s learning is initially robust, its capacity for generalisation requires review. Future iterations may benefit from strategies to boost the model performance on unseen data.

Compared to model1, there is an improvement in model2 accuracy because MSE increases from 0.94 to 0.89.

Figure 5.7:

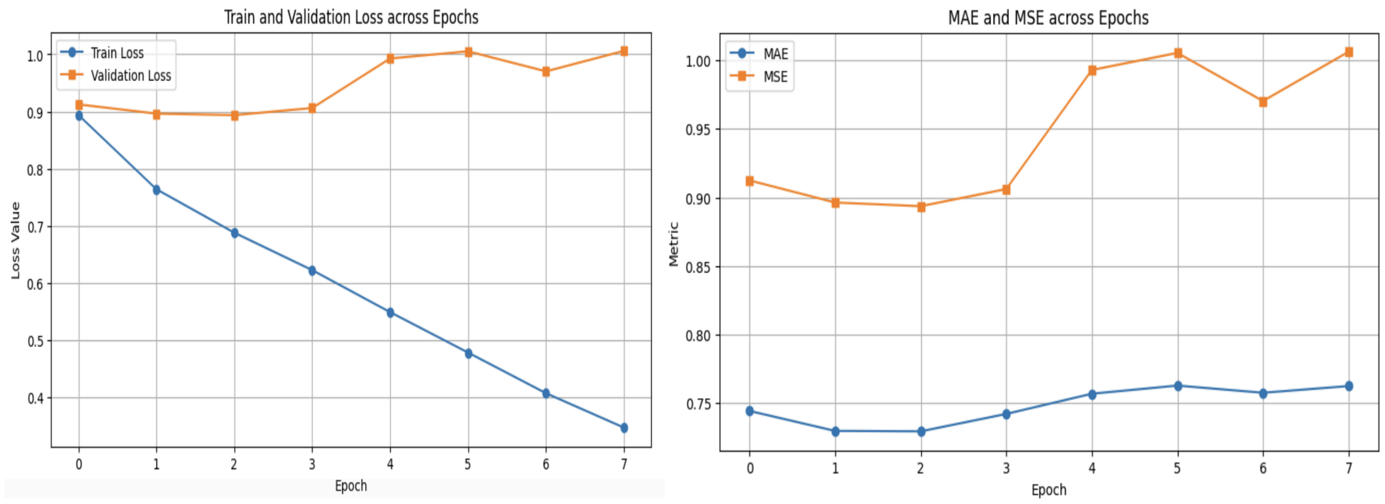
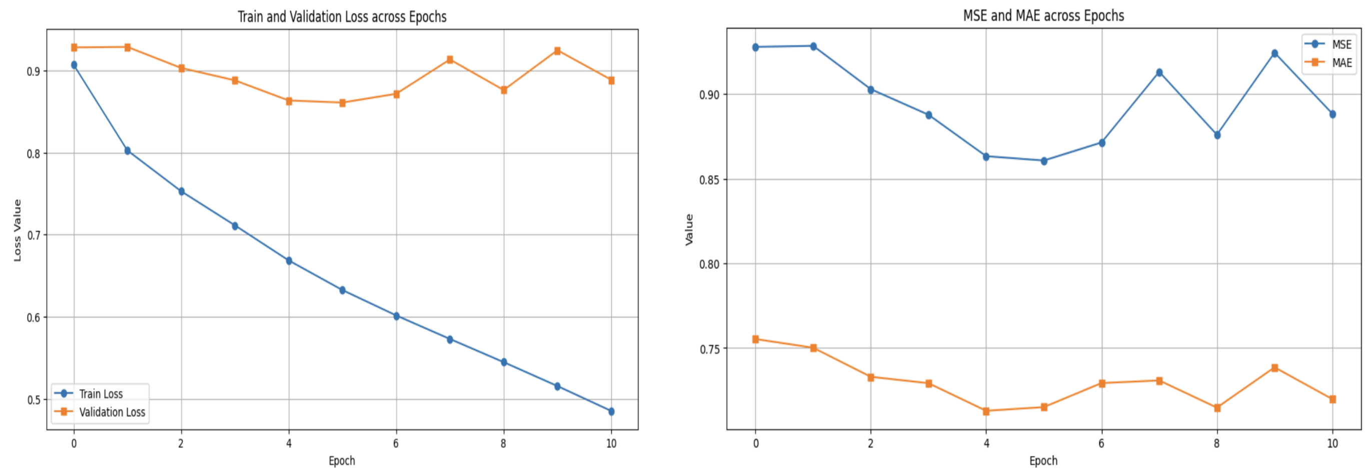


Figure 5.8 shows the training and validation loss, alongside the Mean Absolute Error (MAE) and Mean Squared Error (MSE) for a Transformer model3 over ten epochs. The training loss declines smoothly, suggesting that the model is learning well from the training data. However, the validation loss starts to fluctuate after a drop at epoch 5 , which could mean the model isn't improving on new data after the initial learning phase. The right graph shows the MAE remains fairly constant, indicating consistent prediction accuracy, while the MSE has some large fluctuation after MSE reaches the lowest point(0.86), which might point to variability in the model's performance across different epochs. The MSE decrease (from 0.89 to 0.86) proves the ‘movie genre’ is beneficial for recommendation system design in terms of MSE measurement.

Figure 5.8:



According to the three results, model3 presents the strongest prediction power in movie rating, proving the importance and effectiveness of sequential information in building a movie recommender and especially movie category is beneficial information for movie recommendation systems.

6. Limitations

Regarding the matrix factorisation models. Firstly, the method's linearity assumption assumes that user preferences and item features can be simplified into linear combinations of hidden factors, whose latent factor has a size of 120 in our case. This simplification makes the model easier to handle, but it does not capture the complex and non-linear ways people actually make choices. This can make the recommendations less effective. How user tastes or item popularity change over time is not treated. Our factorisation models treat older ratings the same as newer ones. Human Beings preferences usually change with time, so this assumption may not be appropriate. Adding a way to factor in these changes over time could make the recommendations more accurate.

Another limitation is that the model only uses explicit ratings like star ratings. However, indirect feedback such as clicks, views, and purchases can also provide valuable clues about what users might like. These data types are unavailable with movielens itself, so the link file may be used in the future to join the data with other sources. Also the data is usually very sparse. Most potential user-item interactions are not recorded, making it hard to learn good user and item profiles. Techniques that fill in missing data, group similar items or users, or combine different recommendation methods might help solve this problem. Mean Squared Error (MSE) and Mean Absolute Error (MAE), might not fully match the real goals of the system, such as recommending items that users will truly enjoy. Using other metrics such as novelty measures may provide a more accurate assessment of how well the system is doing. Otherwise, if the recommendation system only recommends movies that are overall popular and highly rated to the general public, it does not achieve the goal of a personalised recommendation system due to the overlap.

Cold-start problem is another concern not properly addressed. New users or items that don't have much rating data join the system. The model finds it hard to learn about these new users or items, often leading to poor recommendations. This is a common issue with methods that rely on past data to predict future likes and dislikes. Hybrid systems can be used to take in movie genres and explicit user preferences as additional data sources.

Transformers in movie recommendation systems face several key challenges. One major issue is managing long-term dependencies. These models often have trouble recognizing connections between widely spaced events in a user's history of movie interactions, potentially overlooking important contextual information. Moreover, overfitting is a common issue due to the model's high parameter count. Interpretability is another concern, the complex nature of Transformer architectures makes it difficult to understand the decision-making process behind their recommendations. Finally, computational complexity is a practical limitation, as the training and deployment of Transformers require substantial computational resources, which can be a challenge for application of Transformers in real-time scenarios.

7. Conclusion

This project aimed to explore methods for recommending movies by utilising two approaches; matrix factorization and Transformer based models. These techniques were tested on the MovieLens 100K dataset, which consists of ratings given by 610 users for 9,742 movies.

The matrix factorisation technique, utilising Stochastic Gradient Descent, extracts user and movie data into a series of latent factors, simplifying the high-dimensional rating data into an easier form. This method hypotheses that a smaller, latent feature space can effectively represent user preferences and movie characteristics. On the other hand, Transformers apply a self-attention mechanism, relying on the order in rating history. This approach is based on the statistical principle that patterns over time may hold significant predictive power, with the sequential order potentially providing deeper insights into a user's preferences.

The comparison between the two methods showed that Transformer-based models have relatively higher predictive accuracy than matrix factorisation models. To be specific, the model using matrix factorisation with bias performed better due to lower Mean Squared Error and Mean Absolute Error on validation data, highlighting the importance of recognizing user and item-specific biases. On the other hand, the third Transformer-model excelled among the other models since it contained learned embedding 'genre', which was reflected by lowest MSE. These outcomes suggest that for sequential interaction data and the accuracy of predictions, Transformers may hold an advantage over matrix factorisation techniques.

In conclusion, our findings revealed that while matrix factorisation models are robust and computationally less intensive approach, Transformers provide an advantage in capturing the dynamic order of user-item interaction. The inclusion of bias terms in matrix factorisation and the integration of sequential data in Transformers perform to be able to increase predictive accuracy. These insights contribute to more complex and personalised recommender systems, indicating the advantage of advanced techniques in addressing real recommendation challenges. Future research may delve into hybrid models that integrate the strengths of both methodologies, potentially improving in understanding user preferences.

8. Bibliography

Kang, W. C., & McAuley, J. (2018, November). Self-attentive sequential recommendation. In *2018 IEEE international conference on data mining (ICDM)* (pp. 197-206). IEEE. https://doi.org/10.1109/ICDM.2018.00035

B.Thorat, P., M. Goudar, R., & Barve, S. (2015). Survey on Collaborative Filtering, Content-based Filtering and Hybrid Recommendation System. *International Journal of Computer Applications*, *110*(4), 31–36. https://doi.org/10.5120/19308-0760

Comparing matrix factorization with transformers for MovieLens recommendations using PyTorch-accelerated | by Chris Hughes | Data Science at Microsoft | Medium. (n.d.). Retrieved 24 April 2024, from https://medium.com/data-science-at-microsoft/comparing-matrix-factorization-with-transformers-for-movielens-recommendations-using-8e3cd3ec8bd8

Geng, Z., Guo, M.-H., Chen, H., Li, X., Wei, K., & Lin, Z. (2021). *Is Attention Better Than Matrix Decomposition?* (arXiv:2109.04553). arXiv. https://doi.org/10.48550/arXiv.2109.04553

Hidasi, B., & Karatzoglou, A. (2018). Recurrent Neural Networks with Top-k Gains for Session-based Recommendations. *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, 843–852. https://doi.org/10.1145/3269206.3271761

Jannach, D., Zanker, M., Felfernig, A., & Friedrich, G. (2010). *Recommender Systems: An Introduction*. Cambridge University Press.

Kuzelewska, U. (2014). Clustering Algorithms in Hybrid Recommender System on MovieLens

Data. *Studies in Logic, Grammar and Rhetoric*, *37*(1), 125–139. https://doi.org/10.2478/slgr-2014-0021

Ricci, F., Rokach, L., & Shapira, B. (2011). Introduction to Recommender Systems Handbook. In F. Ricci, L. Rokach, B. Shapira, & P. B. Kantor (Eds.), *Recommender Systems Handbook* (pp. 1–35). Springer US. https://doi.org/10.1007/978-0-387-85820-3_1

Sarwar, B. M., Karypis, G., Konstan, J., & Riedl, J. (n.d.). *Recommender Systems for Large-scale E-Commerce: Scalable Neighborhood Formation Using Clustering*.

The MovieLens Datasets: History and Context: ACM Transactions on Interactive Intelligent Systems: Vol 5, No 4. (n.d.). Retrieved 26 April 2024, from https://dl.acm.org/doi/abs/10.1145/2827872?casa_token=ciGlKG_9pYYAAAAA:t2mSD2soMNwf3bmf4KCJA4A2tJVyWJA5eWYi6DQNM2w6g_ebk3m5uGUAEnDNsjKHqJ0i-p9i6Urbkg

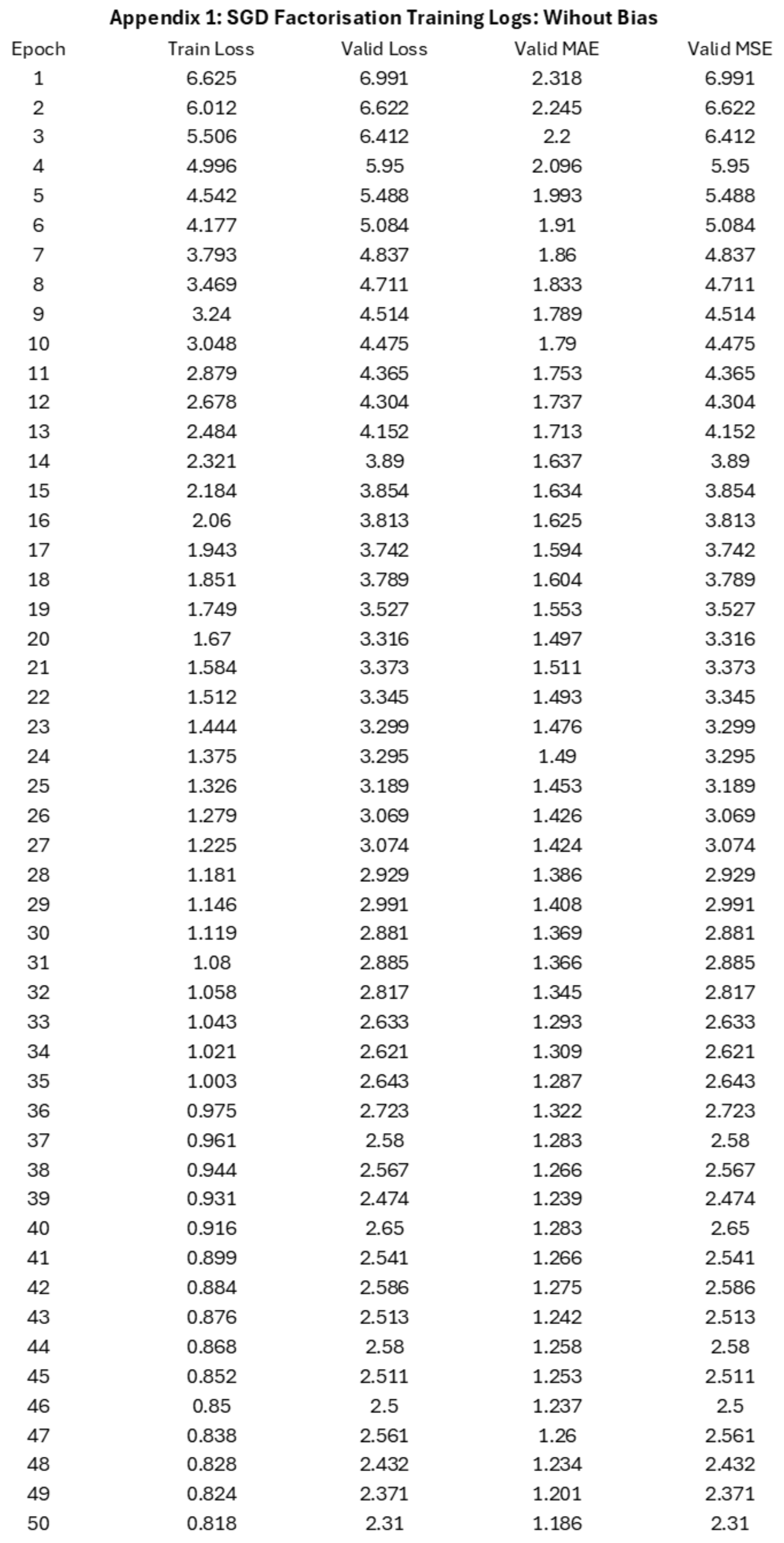
Zhang, Z., Zhao, K., & Zha, H. (2012). Inducible regularization for low-rank matrix factorizations for collaborative filtering. *Neurocomputing*, *97*, 52–62. https://doi.org/10.1016/j.neucom.2012.05.010

Wu, L., Li, S., Hsieh, C. J., & Sharpnack, J. (2020, September). SSE-PT: Sequential recommendation via personalized transformer. In *Proceedings of the 14th ACM conference on recommender systems* (pp. 328-337).

Jin, X., Shen, J., Miao, T., & Wu, Y. (2022, September). Transformer Recommendation Machine for Rate Prediction. In *Proceedings of the 2022 5th International Conference on Machine Learning and Machine Intelligence* (pp. 41-47).https://doi.org/10.1145/3568199.3568205

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, *30*.

9. Appendix



Appendix 1: SGD Factorisation Training Logs: Wihout Bias

Epoch	Train Loss	Valid Loss	Valid MAE	Valid MSE
1	6.625	6.991	2.318	6.991
2	6.012	6.622	2.245	6.622
3	5.506	6.412	2.2	6.412
4	4.996	5.95	2.096	5.95
5	4.542	5.488	1.993	5.488
6	4.177	5.084	1.91	5.084
7	3.793	4.837	1.86	4.837
8	3.469	4.711	1.833	4.711
9	3.24	4.514	1.789	4.514
10	3.048	4.475	1.79	4.475
11	2.879	4.365	1.753	4.365
12	2.678	4.304	1.737	4.304
13	2.484	4.152	1.713	4.152
14	2.321	3.89	1.637	3.89
15	2.184	3.854	1.634	3.854
16	2.06	3.813	1.625	3.813
17	1.943	3.742	1.594	3.742
18	1.851	3.789	1.604	3.789
19	1.749	3.527	1.553	3.527
20	1.67	3.316	1.497	3.316
21	1.584	3.373	1.511	3.373
22	1.512	3.345	1.493	3.345
23	1.444	3.299	1.476	3.299
24	1.375	3.295	1.49	3.295
25	1.326	3.189	1.453	3.189
26	1.279	3.069	1.426	3.069
27	1.225	3.074	1.424	3.074
28	1.181	2.929	1.386	2.929
29	1.146	2.991	1.408	2.991
30	1.119	2.881	1.369	2.881
31	1.08	2.885	1.366	2.885
32	1.058	2.817	1.345	2.817
33	1.043	2.633	1.293	2.633
34	1.021	2.621	1.309	2.621
35	1.003	2.643	1.287	2.643
36	0.975	2.723	1.322	2.723
37	0.961	2.58	1.283	2.58
38	0.944	2.567	1.266	2.567
39	0.931	2.474	1.239	2.474
40	0.916	2.65	1.283	2.65
41	0.899	2.541	1.266	2.541
42	0.884	2.586	1.275	2.586
43	0.876	2.513	1.242	2.513
44	0.868	2.58	1.258	2.58
45	0.852	2.511	1.253	2.511
46	0.85	2.5	1.237	2.5
47	0.838	2.561	1.26	2.561
48	0.828	2.432	1.234	2.432
49	0.824	2.371	1.201	2.371
50	0.818	2.31	1.186	2.31

Appendix 2: SGD Factorisation Training Logs: With Bias

Epoch	Train Loss	Valid Loss	Valid MAE	Valid MSE
1	6.47	6.666	2.242	6.666
2	5.722	6.202	2.149	6.202
3	5.119	5.866	2.078	5.866
4	4.623	5.535	2.015	5.535
5	4.217	5.097	1.92	5.097
6	3.887	4.885	1.872	4.885
7	3.575	4.689	1.83	4.689
8	3.356	4.629	1.822	4.629
9	3.173	4.351	1.762	4.351
10	2.969	4.212	1.729	4.212
11	2.793	4.097	1.691	4.097
12	2.595	3.918	1.663	3.918
13	2.409	4.034	1.68	4.034
14	2.25	3.89	1.646	3.89
15	2.111	3.833	1.627	3.833
16	1.969	3.57	1.554	3.57
17	1.844	3.656	1.577	3.656
18	1.744	3.613	1.557	3.613
19	1.644	3.365	1.498	3.365
20	1.555	3.252	1.471	3.252
21	1.472	3.33	1.494	3.33
22	1.416	3.331	1.484	3.331
23	1.343	3.066	1.411	3.066
24	1.281	2.871	1.364	2.871
25	1.235	3.001	1.381	3.001
26	1.184	2.924	1.366	2.924
27	1.14	2.876	1.364	2.876
28	1.095	2.741	1.326	2.741
29	1.056	2.759	1.328	2.759
30	1.023	2.772	1.32	2.772
31	0.99	2.75	1.315	2.75
32	0.962	2.569	1.282	2.569
33	0.943	2.512	1.263	2.512
34	0.919	2.489	1.245	2.489
35	0.909	2.332	1.2	2.332
36	0.892	2.398	1.222	2.398
37	0.878	2.286	1.187	2.286
38	0.876	2.349	1.196	2.349
39	0.862	2.345	1.21	2.345
40	0.853	2.328	1.194	2.328
41	0.844	2.315	1.192	2.315
42	0.834	2.301	1.177	2.301
43	0.827	2.224	1.161	2.224
44	0.813	2.351	1.195	2.351
45	0.816	2.18	1.146	2.18
46	0.814	2.112	1.131	2.112
47	0.807	2.046	1.123	2.046
48	0.8	2.198	1.159	2.198
49	0.795	2.193	1.172	2.193
50	0.798	2.196	1.156	2.196

Acknowledgement:

The use of generative AI tools

Our group acknowledges that we used GPT-4 (OpenAI, https://openai.com/gpt-4) and Meta-Llama-3-70B (https://huggingface.co/meta-llama/Meta-Llama-3-70B) to check grammar of our initial draft, to proofread our final draft, check contradictory points in our literature review part and to produce the intext citation and the bibliography in APA format. These tools also provided explanations on SGD factorisation and attention-mechanisms. In addition, our group used GPT-4 to edit plot margins for matplotlib for EDA section and debug indices misalignment for data preprocessing before factorisation and debug pytorch training loop.