Department: Electrical and Computing Engineering, Volos

Final Project: Movie Recomendation System

Vermisoglou Konstantinos 2988
kvermisoglou@uth.gr

Gkagkosis Nikolaos 3079
ngkagkosis@uth.gr

***Abstract*—Securing suggestions from reliable sources plays a pivotal role in the inherent process of human decision-making.With the internet bringing a lot of choices for consumers, sellers have a tough job of making their ads more personal. At the same time, businesses collect a ton of data on transactions, helping them understand how customers interact with their products. Recommender systems have evolved to help both buyers and sellers by using data analysis to automatically give recommendations, making decision-making easier in our digital world.This report proposes 3 different approaches using both machine learning and deep learning models in building a recommendation system.These models are then evaluated with metrics such as RMSE, MAE, Binary Cross-Entropy, fitting and testing time and finally display the top 10 suggested movies in order to check how well each model recommended movies.**

INTRODUCTION

How does TikTok figure out which video you might enjoy next? How does the Apple Store choose an app personalized for you?
In both cases, a recommendation model based on machine learning looks at how similar videos and apps are to the things you already like, and then suggests recommendations based on those similarities.
Recommendation systems, a widely utilized application of machine learning, primarily focus on predicting user preferences or ratings for various items. Their main objective is to offer suggestions based on either past user data or individual preferences.This project presents two approaches for movie recommendation system. By strategically leveraging user preferences and existing movie data, our system aims to provide personalized recommendations for films the user has not yet seen but is likely to enjoy. More specifically, users that rated a movie are represented with a unique user id. The system requires a user id as input. Once provided, it utilizes various algorithms, including Matrix Factorization, Neural Network, and Sentence Transformer, to present the top 10 recommended movies that the user has not yet seen.
Most recommendation engines can be grouped into three main subcategories, based on the method used to select and suggest products or services that cater to the individual needs of each customer.Collaborative Filtering systems analyze historical interactions alone, while Content-based Filtering systems are based on profile attributes.Hybrid techniques attempt to combine both of these designs. This project applies the first 2 methods

Literature Review

The first paper is Matrix Factorization Techniques for Recommender Systems[1] that refers to Matrix factorization techniques. They have emerged as the dominant methodology in collaborative filtering recommenders, outperforming classical nearest-neighbor techniques in accuracy. Experience with datasets like the Netflix Prize data indicates their ability to provide superior accuracy while offering a compact and memory-efficient model that is easily learnable. These techniques also exhibit the advantage of naturally integrating various crucial aspects of the data, including multiple forms of feedback, temporal dynamics, and confidence levels.

The second paper explores An Efficient Non-Negative Matrix-Factorization-Based Approach to Collaborative Filtering for Recommender Systems[2]. This work introduces a novel approach to non-negative matrix factorization (NMF) in collaborative filtering (CF), focusing on a single-element-based methodology. It replaces the standard distance loss function with sum-squared-error and uses a non-negative update process based on individual feature parameters. The proposed SNMF and RSNMF models integrate Tikhonov regularization for improved prediction accuracy while maintaining non-negativity. SNMF shows computational efficiency over WNMF, and RSNMF, combining the single-element approach and regularization, outperforms WNMF in both efficiency and accuracy.

The next paper refers to Neural Collaborative Filtering[3] that explores the application of deep neural networks in recommender systems, specifically focusing on collaborative filtering based on implicit feedback. While recent studies have utilized deep learning for recommendation, they often incorporated it to model auxiliary information. In contrast, this work introduces a general framework called NCF (Neural network-based Collaborative Filtering) that replaces traditional matrix factorization with a neural architecture capable of learning arbitrary functions from data. To enhance NCF with non-linearities, a multi-layer perceptron is proposed to learn the user-item interaction function. Empirical results on real-world datasets demonstrate significant improvements in recommendation performance compared to state-of-the-art methods, highlighting the effectiveness of using deeper layers of neural networks in collaborative filtering.

The final paper introduces a Content-based filtering for recommendation systems using multiattribute networks[4] . A novel content-based filtering (CBF) method utilizing a multiattribute network (CBF-MN) to enhance item recommendations for users. The proposed approach incorporates network analysis, measuring similarities between directly and indirectly linked items, and employs centrality and clustering techniques to capture mutual relationships and structural patterns among items. This method addresses sparsity and over-specialization issues in recommender systems, providing diverse and improved recommendations. Comparing the proposed CBF-MN with existing methods using MovieLens data reveals its superior performance in terms of accuracy and robustness. The system mitigates over-specialization problems common in traditional CBF approaches by leveraging various attributes for item characterization. The network analysis considers relationships between all items, offering a solution to the sparsity problem and ensuring diverse recommendations.
Recommendation systems play a crucial role in delivering personalized suggestions to users, employing various state-of-the-art methods. Among these, collaborative filtering (CF) stands out as one of the oldest and most effective techniques, recommending items based on the past behaviors of similar users. CF encompasses neighborhood-based CF, which relies on the ratings or purchases of similar users, and latent factor models that assume underlying factors influencing user ratings. Content-based filtering (CBF), on the other hand, recommends items based on user interests and preferences, analyzing features of previously interacted items. Hybrid recommendation systems combine CF and CBF to enhance recommendation accuracy, leveraging CF's ability to capture user preferences and CBF's proficiency in capturing item features. Additionally, neural network-based recommendation systems, a newer entrant, utilize deep learning to understand intricate relationships between users and items, capturing a broad spectrum of features and

leading to highly accurate recommendations.

DATASET AND FEATURES

The project will utilize the MovieLens 100K dataset which was generated from the MovieLens website. This dataset contains 100,000 ratings and 3,600 tag applications applied to 9,000 movies by 600 users. It contains the following files:

Files	Description
ratings.csv	This file contains all of the ratings on a scale of 0.5 to 5.0 stars that have been given to movies by users
tags.csv	Tags is typically a single word or short phrase that describes a movie. This file contains all of the tags that have been applied to movies by users
movies.csv	Movie information, such as titles, genres, and release years
links.csv	Links to the Internet Movie Database (IMDb), and The Movie Database (TMDB). This file contains identifiers that can be used to link to other sources of movie data

The following table shows the changes we made to the data for each algorithm

Algorithms	Data Preprocessing
Neural Network	Normalization of ratings in order to scale to a range [0,1]
Sentence Transformers	Removed rows without genres, Feature engineering titles and genres in order to pass it to Sentence Transformers

Exploratory Data Analysis (EDA)

Exploratory Data Analysis (EDA) is a critical step in the data analysis process that involves examining and visualizing data sets to understand their main characteristics and uncover patterns.

Ratings Distribution

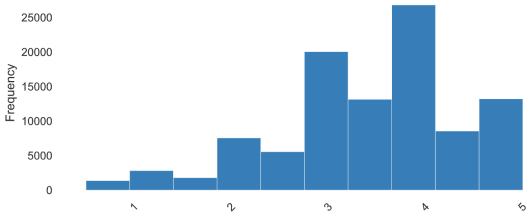

Figure 1. Mean value around 3.5 according to pandas profiling.

Most popular tags

Figure 2. Comedy netflix dark atmospheric are the common tags that users comment.

Most popular genres

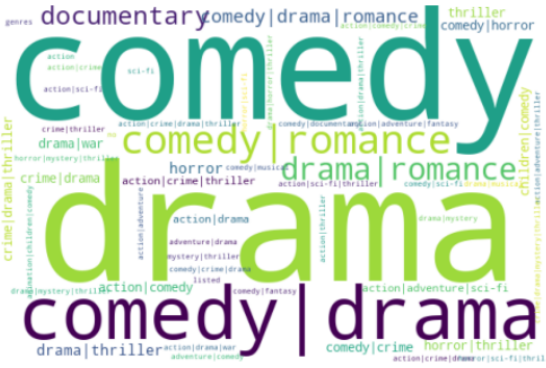

Figure 3. Drama comedy Romance are the most common genres.

Sparsity of user - movie matrix

Interactions	Users	Movies	Sparsity
100836	610	9742	98.3%

$$\text{Sparsity} = \frac{\text{Interactions}}{\text{Total Possible Interactions}}$$

$$\text{Sparsity} = 98.3\%$$

User item matrix is a sparse matrix , only 2% of the matrix is filled with ratings.

Methods

Collaborative Filtering Techniques

Matrix factorization is a collaborative filtering method used in recommender systems to predict user preferences for items. The main idea is to represent the user-item interactions as a matrix, where the rows correspond to users and the columns correspond to items. The entries in this matrix are the known ratings that users have given to items. The goal of matrix factorization is to approximate this matrix by the product of two lower-dimensional matrices, capturing the latent factors associated with users and items.
Suppose we have m users and n items, leading to an $m \times n$ matrix R where each element r_{ui} is the rating of the u-th user for the i-th item. The matrix factorization approach approximates R as:

$$R \approx P \times Q^T$$

Here:

- P is an $m \times k$ matrix representing the user factors, where k is the number of latent factors.
- Q is an $n \times k$ matrix representing the item factors.
- Q^T is the transpose of matrix Q, making the product $P \times Q^T$ an $m \times n$ matrix.

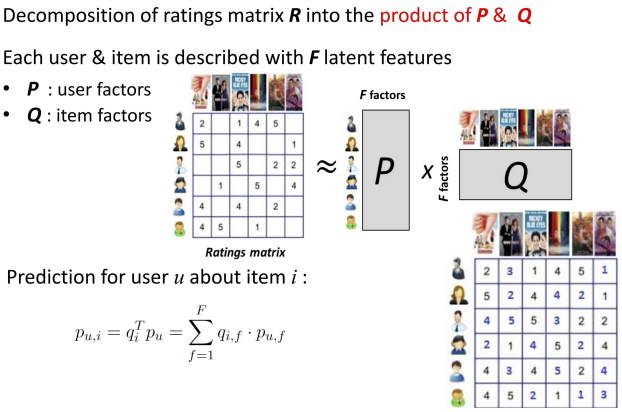

Figure 4. Matrix Factorization decomposition

Latent factors in the context of machine learning and statistics refer to variables that are not directly observed but are inferred from the observed data. These factors are ”latent” because they represent underlying structures or patterns that can explain observed relationships within the data. Here’s a more detailed explanation: In recommender systems, for example, the observed data is typically user-item interactions, such as movie ratings. Latent factor models assume that both items and users are associated with a set of factors that explain the observed preferences. These factors might capture different dimensions of the data, such as:

- Movies: genres, complexity, visual effects quality, depth of character development, etc.
- Users: preferences for certain genres, tolerance for complexity, appreciation for visual effects, etc.

Surprise Library
MF techniques were implemented using Surprise[6] which is a Python scikit library for building and analyzing recommender systems that deal with explicit rating data.
Baseline Model
A baseline serves as a point of reference

against which the performance of more advanced models is measured. By setting a starting benchmark, practitioners gain valuable insights into the efficacy of their models and the progress made over time.Normal Predictor is used as a baseline model. A random rating is predicted based on the distribution of the training set, which is assumed to be normal. The prediction $\hat{r_{ui}}$ is generated from a normal distribution $N(\hat{\mu}, \hat{\sigma^2})$ where and are estimated from the training data $|R_{train}|$ using Maximum Likelihood Estimation:

$$\hat{\mu} = \frac{1}{|R_{train}|} \sum_{r_{ui} \in R_{train}} r_{ui}$$

$$\hat{\sigma} = \sqrt{\frac{1}{|R_{\text{train}}|} \sum_{r_{ui} \in R_{\text{train}}} (r_{ui} - \hat{\mu})^2}$$

SVD

Singular Value Decomposition (SVD) is one of the most commonly used methods for matrix factorization in the context of recommendation systems. Mathematically, SVD is a technique that decomposes a matrix R into three other matrices as follows:

$$R = U \Sigma V^T$$

where

- R is an $m \times n$ matrix with m users and n items.
- U is an $m \times r$ orthogonal matrix, where r is the rank of matrix R.
- Σ (often pronounced "sigma") is an $r \times r$ diagonal matrix with non-negative real numbers on the diagonal. The elements on the diagonal are known as the singular values of R and are typically arranged in descending order. The singular values measure the importance of each latent factor.
- V^T is the transpose of an $n \times r$ orthogonal matrix V.

Applying this vanilla SVD to learn latent factors and predict ratings raises difficulties since user - item matrix needs to be full without missing ratings from the users. Previous systems depended on imputation to complete missing ratings and create a dense rating matrix. Nevertheless, imputation can be costly due to a substantial increase in data volume. Furthermore, inaccurate imputation has the potential to significantly distort the data. Modern approches of SVD and specifically surprise take account only the recorded ratings, add regularization terms to prevent overfit and learn latent factors and biases of users and movies.

A significant portion of the variability in rating values can be attributed to biases linked to either users or items. It is common to observe pronounced systematic tendencies, where certain users consistently give higher ratings than others, and certain items consistently receive higher ratings. The system tries to pinpoint the portion of these values that can be clarified by individual user or item biases (b_u and b_i respectively). A rating approximation using only biases (which is the BaselineOnly model of surprise) is:

$$\hat{r}_{ui} = b_{ui} = \mu + b_u + b_i \qquad (1)$$

So adding biases and the interaction user - movie which is the dot product of user factors p_u and item factors q_i result in the predicted rating of a user u to the movie i:

$$\hat{r}_{ui} = \mu + b_u + b_i + q_i^T \cdot p_u$$

To estimate the unknown parameters of the above equation, regularized squared error loss function is being minimized:

$$\sum_{r_{ui} \in R_{\text{train}}} (r_{ui} - \hat{r}_{ui})^2 + \lambda (b_i^2 + b_u^2 + \|q_i\|^2 + \|p_u\|^2)$$

The update of the parameters is done with

stochastic gradient descent:

$$b_u \leftarrow b_u + \gamma(e_{ui} - \lambda b_u)$$
$$b_i \leftarrow b_i + \gamma(e_{ui} - \lambda b_i)$$
$$p_u \leftarrow p_u + \gamma(e_{ui} \cdot q_i - \lambda p_u)$$
$$q_i \leftarrow q_i + \gamma(e_{ui} \cdot p_u - \lambda q_i)$$

where $e_{ui} = r_{ui} - \hat{r}_{ui}$, γ learning rate and λ regularization term.

Probabilistic Matrix Factorization

Removing biases from the SVD algorithm and you get the Probabilistic Matrix Factorization which estimates the prediction $\hat{r}_{ui}$ only with the dot product of the latent factors p_u and q_i without adding the baseline estimation from 1 and the goal is to minimize the same loss function with SVD without the biases b_i and b_u

$$\hat{r}_{ui} = q_i^T \cdot p_u$$

$$\sum_{r_{ui} \in R_{\text{train}}} (r_{ui} - \hat{r}_{ui})^2 + \lambda(\|q_i\|^2 + \|p_u\|^2)$$

The update of the parameters is done with the same way:

$$p_u \leftarrow p_u + \gamma(e_{ui} \cdot q_i - \lambda p_u)$$
$$q_i \leftarrow q_i + \gamma(e_{ui} \cdot p_u - \lambda q_i)$$

NMF

Non negative matrix factorization also factorizes the original rating matrix R into two matrices P and Q with the difference from SVD that P and Q must be non negative. The constraint of non negativity is highly significant, as it allows the features to more accurately embody real meanings, such as user interests and community tendencies making the resulting matrices easier to inspect.The dot product of the matrices P and Q is the estimated rating $\hat{r}_{ui}$:

$$\hat{r}_{ui} = q_i^T \cdot p_u$$

NMF is trying to approximate the known ratings R with the matrices P and Q.This is done by measuring the difference between them using Euclidean distance.Adding regularization terms to prevent overfit is crucial.A common selection for this job is the Frobenius norm.Hence, the loss function to be minimized is:

$$d = \|R - PQ\|^2 + \lambda_P \|P\|_F^2 + \lambda_Q \|Q\|_F^2$$
$$P, Q \geq 0$$

The optimization procedure is a regularized stochastic gradient descent with a specific choice of step size that ensures non-negativity of factors, provided that their initial values are also positive. At each step of the SGD procedure, the factors or user and item are updated as follows:

$$p_{uf} \leftarrow p_{uf} \cdot \frac{\sum_{i \in I_u} q_{if} \cdot r_{ui}}{\sum_{i \in I_u} q_{if} \cdot \hat{r}_{ui} + \lambda_u |I_u| p_{uf}}$$
$$q_{if} \leftarrow q_{if} \cdot \frac{\sum_{u \in U_i} p_{uf} \cdot r_{ui}}{\sum_{u \in U_i} p_{uf} \cdot \hat{r}_{ui} + \lambda_i |U_i| q_{if}}$$

where λ_u and λ_i are regularization parameters.

BaselineOnly

This algorithm does not capture latent factors P and Q from users and items respectively and relies only in biases for predicting a rating $\hat{r}_{ui}$ using the equation 1.For intance, if Chris wanted to rate the movie Gladiator here is how the system would respond.Suppose average rating μ is 3.5. The Gladiator is above average as a movie and is usually be rated 0.8 over the mean. Chris is a a strict user and rates the movies 0.6 lower than the mean. Hence, the prediction in movie Gladiator would be $(3.5 + 0.8 - 0.6) = 3.7$

Loss function and learning parameters are the same with SVD without the latent factors so:

$$\sum_{r_{ui} \in R_{\text{train}}} (r_{ui} - \hat{r}_{ui})^2 + \lambda(b_i^2 + b_u^2)$$

$$b_u \leftarrow b_u + \gamma(e_{ui} - \lambda b_u)$$
$$b_i \leftarrow b_i + \gamma(e_{ui} - \lambda b_i)$$

where $e_{ui} = r_{ui} - \hat{r}_{ui}$, γ learning rate and λ regularization term.

Neural Network with Keras

Keras is a deep learning API written in Python and capable of running on top of either JAX, TensorFlow, or PyTorch.But what is Deep Learning? Deep learning (DL) is a machine learning subfield that uses multiple layers for learning data representations. It involves the use of neural networks, which are computational models inspired by the structure and functioning of the human brain.Deep neural networks consist of interconnected layers of nodes (neurons), and these networks can learn complex patterns and representations by adjusting the weights and biases associated with each connection during a training process.A frequent field of DL's application is recommendation systems.
In order to understand the functioning of the model, some key points must be highlighted.

Mappers
Each user and movie ID is associated with an embedding matrix for mapping in the given context.The term "embedding" refers to a representation of discrete data in a continuous, high-dimensional space.

Dot product
The model calculates the dot product of the user vector and the movie vector, using the function tensordot, to derive a matching score, representing the predicted rating for the user-movie interaction.After that, the user and movie bias is added to this.The final match score is scaled to the [0, 1] interval via a sigmoid.

Loss Function
Binary Crossentropy is used as the loss function. Binary Crossentropy is commonly used for binary classification problems. When the number of classes M equals 2, cross-entropy can be calculated as:

$$\mathcal{J} = -(y \log(p) + (1 - y) \log(1 - p))$$

If $M > 2$ (i.e. multiclass classification), we calculate a separate loss for each class label per observation and sum the result.

$$\mathcal{J} = -\sum_{c=1}^{M} y_{o,c} \log\left(p_{o,c}\right)$$

Components	Description
M	number of classes
Log	the natural log
y	binary indicator (0 or 1) if class label c is the correct classification for observation o
p	predicted probability observation o is of class c

Optimizer
The model uses Adam optimizer. The table that includes its components and their mathematical notation are shown down below.

Components	Description
v_{dW}	the exponentially weighted average of past gradients
s_{dW}	the exponentially weighted average of past squares of gradients
$\beta 1$	hyperparameter to be tuned
$\beta 2$	hyperparameter to be tuned
$\frac{\partial \mathcal{J}}{\partial W}$	cost gradient with respect to current layer
W	the weight matrix (parameter to be updated)
a	the learning rate
ϵ	very small value to avoid dividing by zero

$$v_{dW} = \beta_1 v_{dW} + (1 - \beta_1) \frac{\partial \mathcal{J}}{\partial W}$$

$$s_{dW} = \beta_2 s_{dW} + (1 - \beta_2) \left(\frac{\partial \mathcal{J}}{\partial W}\right)^2$$

$$v_{dW}^{\text{corrected}} = \frac{v_{dW}}{1-(\beta_1)^t}$$

$$s_{dW}^{\text{corrected}} = \frac{s_{dW}}{1-(\beta_1)^t}$$

$$W = W - \alpha \frac{v_{dW}^{\text{corrected}}}{\sqrt{s_{dW}^{\text{corrected}}} + \varepsilon}$$

Content- Based Filtering

Content-based recommender system recommends items to users based on the similarity of those items.For instance, if an AppleTV user has watched many adventure movies, then recommend a movie classified in the database as having the "cowboy" genre. In a content-based system, you must construct for each item a profile, which represents important characteristics of that item. The movies file from MovieLens dataset, contains title of the movie and genres and those are the features that were used for the item profile.

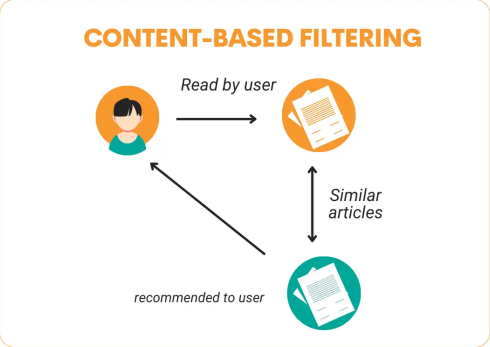

Figure 5. Intuition of content filtering

Sentence Transformer

With featuring engineering, the valuable information is extracted from these features and for each movie there is a sentence inside a list containing the title and genres of this movie. Those sentences then are transformed into vectors.There are many techniques for this purpose like CountVectorizer or TfidfVectorizer. A state of the art sentence embeddings is used, the Sbert[7] (Sentence Bidirectional Encoder Representations from Transformers) and specifically the model all-mpnet-base-v2 which is a pre trained model on more than 1 billion training pairs from different datasets.After creating those vectors it is necessary to inform the network about which sentence pairs are similar, and should be close in vector space, and which pairs are dissimilar, and should be far away in vector space.This can be done with cosine similarity.

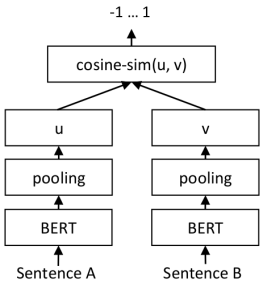

Figure 6. Siamese Network Architecture

Sentences A and B are passed through BERT (sentence embeddings models) which produces contextualized word embeddings for all input tokens in the sentences.A fixed size output dimension for vectors u and v is achieved with a pooling layer.The similarity of these embeddings u, v is computed using cosine similarity and the result is compared to the gold similarity score. This allows the network to be fine-tuned and to recognize the similarity of sentences. So the shape of our embeddings in our case will be 9742 rows and 768 columns.Each row corresponds to a movie and the fixed size 768 is from the pooling layer that corresponds to 768 features for each movie.

Experiments/Results/Discussion

In this section, the produced experiments from the above methods are evaluated and discussed.

Matrix factorization Techniques

The execution of the algorithms are based on 5-fold cross-validation procedure keeping their default parameters. The data were splitted into five parts, using four for training the algorithm and one for validating in each round. This helps us get a good overall idea of how the algorithms perform. Root Mean Square Error (RMSE) and Mean Average Error (MAE) are the error metrics in order to see how close the predictions were to the actual ratings. These measures gives a clear picture of how accurate the models are. The fitting and testing time were measured as well.

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|x_i - x|$$

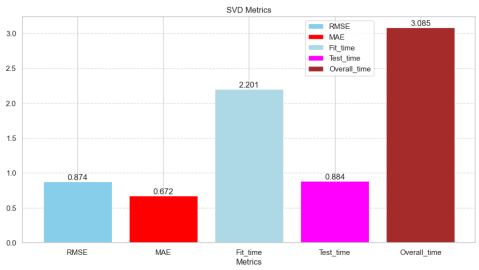

Figure 7. SVD metrics

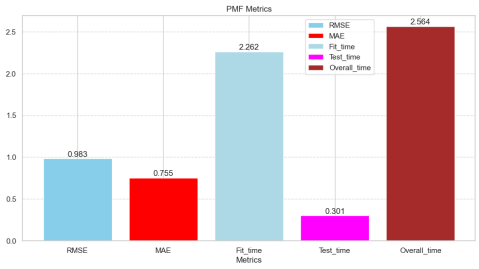

Figure 8. PMF metrics

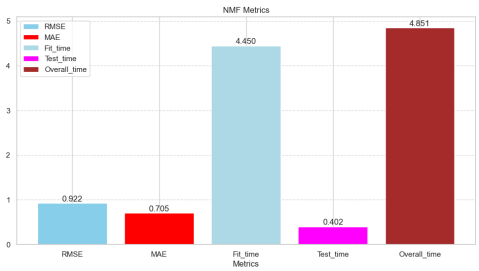

Figure 9. NMF metrics

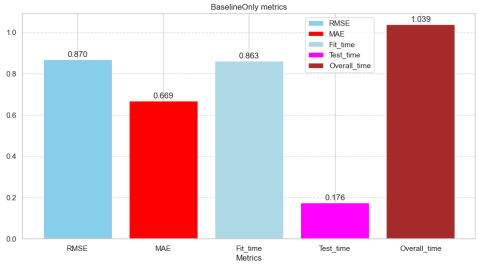

Figure 10. BaselineOnly metrics

All algorithms are performing in error metrics pretty good having below 1 point rating difference. BaselineOnly is the best overall both in time and error. Biases play a significant role in predicting the rating of a movie without using latent factors. Also

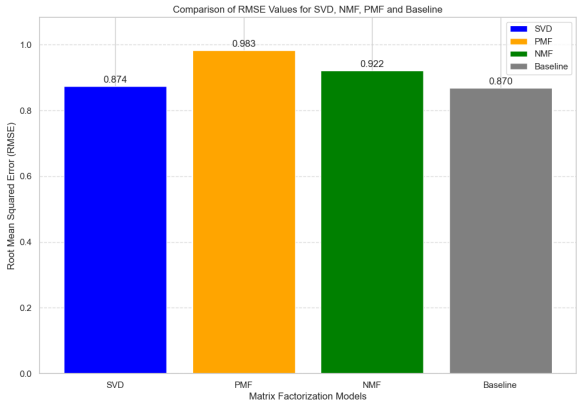

Figure 11. Comparison of RMSE values for SVD, PMF, NMF, BaselineOnly

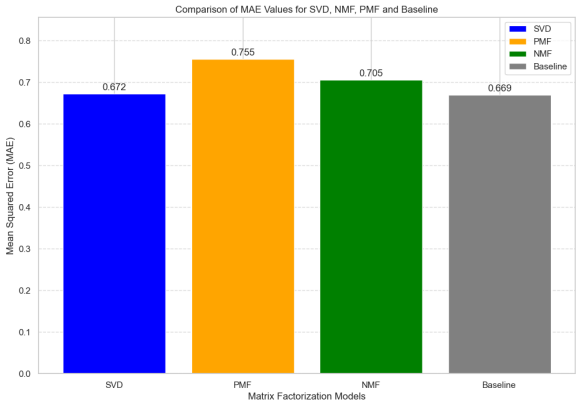

Figure 12. Comparison of MAE values for SVD, PMF, NMF, BaselineOnly

since it does not capture and calculate latent factors, it is faster than the other algorithms. SVD that is an extension of the BaselineOnly, is adding the latent factors p_u and q_i to predict the rating. From the results, it seems that this addition doesnt change almost anything in error and also slows the procedure since it tries to calculate these factors. This can be confirmed from the PMF model that is an unbiased version of SVD from the surprise. PMF's rating is based only in the factors p_u and q_i and has the worst performing.This may due to the fact that the number of factors that uses PMF as parameter is $n_{factors} = 100$ and can not capture the real factors that exists in the dataset. From the other hand, NMF also uses only p_u and q_i to estimate a rating but it is different the loss function that tries to minimize, has a constraint to be non negative p_u and q_i and uses $n_{factors} = 15$. This results in a much better performance than PMF but is slower than the others since it has constraints in calculation of p_u and q_i.

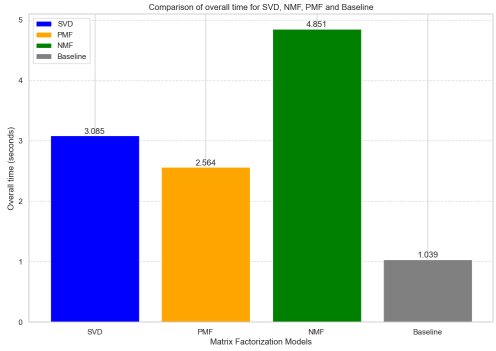

Figure 13. Comparison of time for SVD, PMF, NMF, BaselineOnly

Top 10 Movies

For the user with userId = 10 the top 10 rated and recommended movies using BaselineOnly are :

title	genres
The Intern (2015)	Comedy
First Daughter (2004)	Comedy|Romance
Skyfall (2012)	Action|Adventure|Thriller|IMAX
Dark Knight Rises, The (2012)	Action|Adventure|Crime|IMAX
Troy (2004)	Action|Adventure|Drama|War
King's Speech, The (2010)	Drama
Notebook, The (2004)	Drama|Romance
Despicable Me (2010)	Animation|Children|Comedy|Crime
Education, An (2009)	Drama|Romance
Batman Begins (2005)	Action|Crime|IMAX

Figure 14. Top 10 rated movies from userId = 10

title	genres
Dr. Strangelove or: How I Learned to Stop Worr...	Comedy|War
Shawshank Redemption, The (1994)	Crime|Drama
North by Northwest (1959)	Action|Adventure|Mystery|Romance|Thriller
Lawrence of Arabia (1962)	Adventure|Drama|War
Raiders of the Lost Ark (Indiana Jones and the...	Action|Adventure
Godfather, The (1972)	Crime|Drama
Cool Hand Luke (1967)	Drama
High Noon (1952)	Drama|Western
Ghost in the Shell (Kôkaku kidôtai) (1995)	Animation|Sci-Fi
Seventh Seal, The (Sjunde inseglet, Det) (1957)	Drama

Figure 15. Top 10 recommended movies for userId = 10 using BaselineOnly

This user watches some marvel movies including action but also watches sensitive movies. The recommendations from BaselineOnly model are also based in these genres.

Neural Network

The data were splitted in 90% for training and 10% for validation. The model was trained with batch size = 64. The binary cross entropy was used to evaluate the model's performance. The train loss is get-

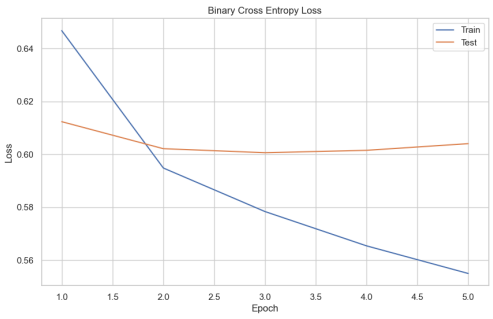

Figure 16. Binary Cross Entropy Loss

ting lower at each epoch and crosses test loss at the second epoch. Decreasing train loss indicates that the model is able to learn the patterns in the data. Test loss is also relatively low which means that the model is generalizing well to unseen data. Figure 17 shows top 10 recommendations using NN

title	genres
X-Men (2000)	Action|Adventure|Sci-Fi
Star Trek IV: The Voyage Home (1986)	Adventure|Comedy|Sci-Fi
Harry Potter and the Goblet of Fire (2005)	Adventure|Fantasy|Thriller|IMAX
Mission: Impossible (1996)	Action|Adventure|Mystery|Thriller
GoldenEye (1995)	Action|Adventure|Thriller
Matrix Revolutions, The (2003)	Action|Adventure|Sci-Fi|Thriller|IMAX
X2: X-Men United (2003)	Action|Adventure|Sci-Fi|Thriller
Star Trek: First Contact (1996)	Action|Adventure|Sci-Fi|Thriller
Shrek 2 (2004)	Adventure|Animation|Children|Comedy|Musical|Ro...
Star Wars: Episode II - Attack of the Clones (...	Action|Adventure|Sci-Fi|IMAX

Figure 17. Top 10 recommended movies for userId = 10 using NN model. The model is capturing the action and adventure factors of the user

Content

The approach and process of developing the content - based recommendation system is mentioned in the Content- Based Filtering. The idea is to recommend items to users based on the similarity of those items. The model was evaluated based on the user profile with userId = 10 which is the figure 14. The goal is to build an item profile that has as similar movies as possible with a movie from the user profile. Titles and genres were used for the item profile.

Top 10 similar movies

title	genres
Despicable Me 3 (2017)	Adventure|Animation|Children|Comedy
Despicable Me 2 (2013)	Animation|Children|Comedy|IMAX
Wallace & Gromit: The Wrong Trousers (1993)	Animation|Children|Comedy|Crime
Fantastic Mr. Fox (2009)	Adventure|Animation|Children|Comedy|Crime
Hoodwinked! (2005)	Animation|Children|Comedy
Great Mouse Detective, The (1986)	Action|Animation|Children|Crime
Beavis and Butt-Head Do America (1996)	Adventure|Animation|Comedy|Crime
Catch That Kid (2004)	Action|Adventure|Children|Comedy|Crime
Cars (2006)	Animation|Children|Comedy
Minions (2015)	Adventure|Animation|Children|Comedy

Figure 18. User 10 likes "Despicable Me" according to the figure 14. Get top 10 movies similar to "Despicable Me" that he will probably like too

"Despicable Me" is a family-friendly animated movie. Additionally, it falls under the genres of comedy and adventure, given its humorous and action-packed elements that appeal to a broad audience. The model tries to recommend movies suitable for children that share a similar theme to "Despicable Me" infusing a hint of action.Also, it makes sure to include

suggestions for the sequels of ”Despicable Me.”

Benchmarks

The summary of the results generated by Matrix Factorization algorithms and the baseline model is outlined below.

Model	RMSE	MAE	Time
Normal Predictor	1.418	1.133	0.253
SVD	0.874	0.672	3.085
PMF	0.983	0.756	2.564
NMF	0.922	0.707	4.851
BaselineOnly	0.870	0.669	1.039

while the results from Neural Network are presented here since the evaluation is different

Model	Binary Cross Entropy	Time
Neural Network	0.603	1.133

Contributions

Member 1 : Vermisoglou Konstantinos
Member 2 : Gkagkosis Nikolaos

Contributions	Members
EDA	1
MF implementation	2
MF Evaluation	1 & 2
NN	1 & 2
Content Feature Engineering	1 & 2
Content Implementation	1
Report	1 & 2

Conclusion/Future Work

The MF algorithms demonstrate good performance in error metrics, achieving a rating difference below 1 point. BaselineOnly stands out as the best, being both efficient and accurate. PMF has the worst performance. Neural Network model is being evaluated with Binary Cross Entropy achieving val_loss = 0.6 which means that, on average, the dissimilarity of actual and predicted ratings is 0.6 in terms of the normalized rating scale. Content based filtering recommends similar movies with the movie a user liked.

In future, we would approach a hybrid method combining collaborative and content based filtering in order to compare it with the methods we implemented. Also including ranking evaluations like Hit Ratio and Normalized Discounted Cumulative Gain (NDCG) would boost our confidence that relevant movies are recommended.

REFERENCES

1. Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix Factorization Techniques for Recommender Systems. 2009.
2. Xin Luo, Mengchu Zhou, Yunni Xia, and Qinsheng Zhu. An Efficient Non-Negative Matrix-Factorization-Based Approach to Collaborative Filtering for Recommender. 2014.
3. Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu and Tat-Seng Chua. Neural Collaborative Filtering. 26 Aug 2017.
4. Jieun Son, Seoung Bum Kim. Content-based filtering for recommendation systems using multiattribute networks. 2017.
5. Ruslan Salakhutdinov and Andriy Mnih. Probabilistic matrix factorization. 2008.
6. https://surprise.readthedocs.io/en/stable/
7. https://www.sbert.net
8. https://www.tensorflow.org