Personalized Video Recommendation System Final Report

Zhouquan Jin | Fanshi Meng | Xinyue Tao | Ruiqi Xie | Wuyuqing Yang

Introduction/Background

Personalized video recommendations are crucial for streaming platforms, enhancing user experience by suggesting tailored content. Recommendation systems have evolved with various methods improving accuracy and scalability. Traditional methods, such as matrix factorization introduced by Koren (2009) revolutionized accuracy in recommendation engines [1]. More recently, to handle dynamic user preferences, Quadrana et al. (2018) explored sequence-aware recommendation systems, proposing models like Recurrent Neural Networks (RNNs) [2]. He et al. (2017) introduced Neural Collaborative Filtering (NCF), which integrates deep learning with collaborative filtering [3]. This project aims to develop personalized video recommendation systems using machine learning methods.

The dataset used is the MovieLens 1M and MovieLens 32M Dataset, consisting of 1 million ratings from 6000 users on 4000 movies. Released 2/2003.

Dataset Link: MovieLens Dataset

Problem Definition

With the advancement of science and technology, online streaming media platforms are experiencing rapid growth, leading to an increasing number of such platforms emerging in the public's view. However, this also brings attention to a prevalent issue faced by streaming users today - content overload. When confronted with a vast array of movie and video resources, users often struggle to find content that aligns with their interests and preferences, significantly impacting their experience and reducing user engagement. Given that traditional recommendation systems fail to adequately consider individual differences and personalized needs of users, the video content suggested by these platforms lacks the appeal and relevance required for user satisfaction. Therefore, our project aims to develop and evaluate a personalized video recommendation system based on the MovieLens dataset. By analyzing users' viewing history, rating preferences, and application tags, we aim to generate tailored recommendations that cater specifically to their interests. Through this personalized recommendation strategy, we aspire to efficiently assist users in discovering new videos while enhancing overall user satisfaction and retention on the platform.

Methods

Data Preprocessing Method: Negative Sampling

Why Negative Sampling?

To ensure the effectiveness and efficiency of training the NCF model, we use negative sampling as the data preprocessing method. Since we are using NCF as the model, when handling the implicit feedback data, negative sampling is essential because such data typically only contain positive interactions instead of negative interactions. Additionally, in NCF, we need to use HR (Hit Ratio) as a metric, so it is necessary for us to use negative sampling to preprocess the data.

Implementation of Negative Sampling:

- **Positive Interactions**: We first extract interactions between users and movie ratings from the original dataset.
- **Negative Sampling Strategy**: For each user, we randomly select a fixed number of negative samples from items they have not interacted with, ensuring that the dataset includes items in which the user has shown no interest.
- **Combining Positive and Negative Samples**: We combine the positive and negative samples to form the training dataset. Positive samples are labeled as 1, and negative samples are labeled as 0 for model training and evaluation.

Method 1: NCF (Supervised Learning)

To implement a personalized video recommendation system, we choose to use the Neural Collaborative Filtering (NCF) model. By integrating deep learning with collaborative filtering techniques, the NCF model enhances the performance of traditional matrix factorization models through nonlinear modeling capabilities. This model not only captures the complex interactions between users and items, but also provides greater flexibility to adapt to various data characteristics. In this section, we are going to discuss why we chose to use the NCF model and the details about how to implement the NCF model.

Why NCF?

Why Collaborative Filtering (CF)?

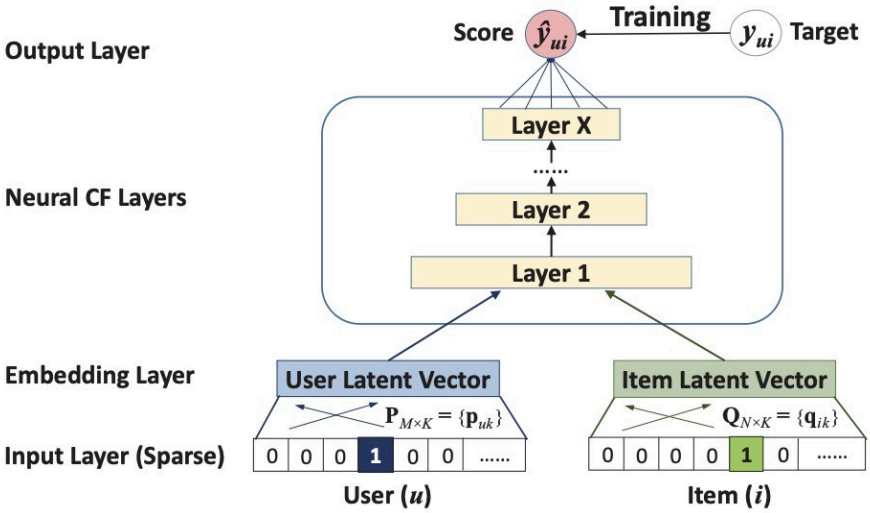
The implementation of a movie recommendation system can be achieved by methods such as collaborative filtering and matrix factorization. Collaborative Filtering (CF) generates movie recommendations by analyzing interaction data between users and user ratings for movies. The core idea of CF is to recommend content based on the similarities between users or between items.

CF algorithms rely heavily on user-movie rating interaction data to provide recommendations, which makes this approach highly adaptable. When implementing a movie recommendation system, it does not require additional detailed content information about the movies, significantly reducing data dependency and enhancing the algorithm's applicability.

Moreover, CF algorithms recommend movies based on analyzing user preferences for certain types of movies, whether they are liked or disliked. This approach emphasizes building a user profile, resulting in more personalized recommendations that align better with user interests. Additionally, CF algorithms have dynamic adaptability; as user interests change, the recommendations can be updated in real-time to ensure that the system continually provides personalized movie recommendations.

Why Neural Collaborative Filtering (NCF)?

Traditional collaborative filtering models are mainly divided into user-based collaborative filtering and item-based collaborative filtering. These methods achieve recommendations either by finding users with similar interests or by analyzing the similarity between movies. However, these models rely on linear computations, which can be inadequate for handling complex interactions. Neural Collaborative Filtering (NCF) combines the strengths of collaborative filtering and neural networks by using neural network modeling. On the basis of linear relationships (such as Generalized Matrix Factorization, GMF), it can capture more complex nonlinear relationships (such as those modeled by Multi-Layer Perceptron, MLP). This combination allows NCF to more accurately model the relationship between user preferences and movie ratings, thereby enhancing the performance and effectiveness of recommendation systems.



Neural Collaborative Filtering Framework [2]

The process of NCF

Four-Layer Structure of the Model

According to [2], the NCF model can be divided into the following four layers:

- **Input Layer**: In this layer, it receives the IDs of users and items and converts them into embedding vectors.
- **Embedding Layer**: In this layer, it maps the one-hot encoded user and item IDs into a low-dimensional dense vector space to capture the latent features of users and items.

- **Hidden Layer (GMF and MLP)**: In this layer, it is used for modeling the linear and non-linear interactions between users and the rating movies.
- **Output Layer**: In this layer, it combines the outputs of GMF and MLP and generates the final prediction score through a fully connected layer.

Optimization Function

According to [2], the loss function is defined as:

$$L = -\sum_{(u,i)\in Y\cup Y^-} [y_{ui}\log(\hat{y}_{ui}) + (1-y_{ui})\log(1-\hat{y}_{ui})]$$

Where:

- Y denotes the set of observed interactions (positive samples), and Y^- represents the set of negative samples.
- y_{ui} is the actual interaction value:
  - $y_{ui} = 1$ indicates an interaction.
  - $y_{ui} = 0$ indicates no interaction.
- $\hat{y}_{ui}$ is the predicted relevance score of item i for user u.

Summary of GMF and MLP

GMF and MLP are both used in the hidden layers of the NCF model, responsible for modeling linear and non-linear relationships respectively.

GMF (Generalized Matrix Factorization)

GMF is the linear component in the NCF framework, and it is used to capture the linear relationships between users and items. GMF computes the interaction between the user embedding vector (p_u) and the item embedding vector (q_i) through element-wise product:

$$\phi_{\text{GMF}}(p_u, q_i) = p_u \odot q_i$$

MLP (Multi-Layer Perceptron)

MLP is the part of the NCF framework used to model non-linear relationships by learning complex interactions between users and items through a multi-layer neural network. The MLP model is defined as follows:

$$z_1 = \phi_1(p_u, q_i) = [p_u q_i]^T$$

$$\phi_2(z_1) = a_2(W_2^T z_1 + b_2)$$

$$\vdots$$

$$\phi_L(z_{L-1}) = a_L(W_L^T z_{L-1} + b_L)$$

$$\hat{y}_{ui} = \sigma(h^T \phi_L(z_{L-1}))$$

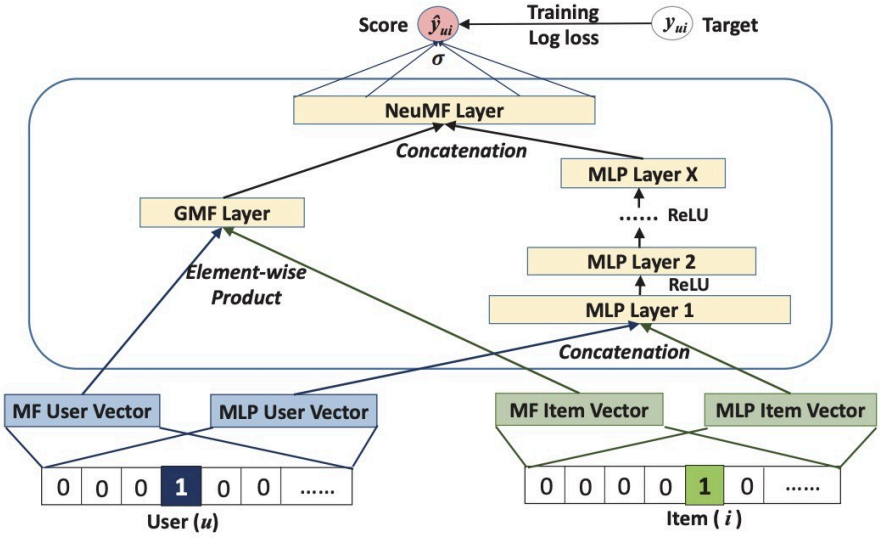
Where:

- W_x: Weight matrix in the xth layer.
- b_x: Bias vector in the xth layer.
- a_x: Activation function for the xth layer.

From NCF to NeuMF

According to [2], to fuse GMF and MLP, the straightforward idea is to let GMF and MLP share the same embedding layer, and then combine the outputs of their interaction functions.

However, sharing embeddings of GMF and MLP might limit the performance of the fused model. Hence, [2] provides another kind of way to fuse GMF and MLP, which allows GMF and MLP to learn separate embeddings, and combine the two models by concatenating their last hidden layer. This model is called “NeuMF”. This model combines the linearity of MF and non-linearity of DNNs for modeling user–movie latent structures.



Neural matrix factorization model [2]

Method 2: PCA and K-Means Clustering (Unsupervised Learning)

After completing the NCFmodel, we found a significant problem on the model: the cold-start problem. When new users join the system, there is no prior interaction information available (e.g., rating records or viewing history), making it impossible to use collaborative filtering algorithms like NCF to provide effective recommendations. This issue limits the applicability of the NCF model in scenarios involving new users.

To address this challenge, we turned to unsupervised learning methods and decided to employ a clustering approach to classify movies based on their content features. Specifically, we performed clustering analysis on all available movies, grouping similar movies together based on their content. For each cluster, we selected the top two movies with the highest average ratings and recommended them to users. This approach not only provides initial recommendations for new users but also allows us to collect users' rating data, which can serve as foundational information for future collaborative filtering methods.

This content-based recommendation strategy offers a simple and efficient solution for cold-start scenarios. It enhances the system's flexibility and lays the groundwork for more personalized recommendations in the future.

Why Unsupervised Learning?

Unsupervised learning methods, such as clustering, have significant advantages when grouping items based on inherent feature similarities. Unlike collaborative filtering, which relies on user interaction data, unsupervised methods are solely based on the characteristics of the items themselves. This makes them particularly useful when user interaction data is sparse or unavailable. While collaborative filtering can provide more personalized movie recommendations, it often becomes ineffective when lacking historical user data.

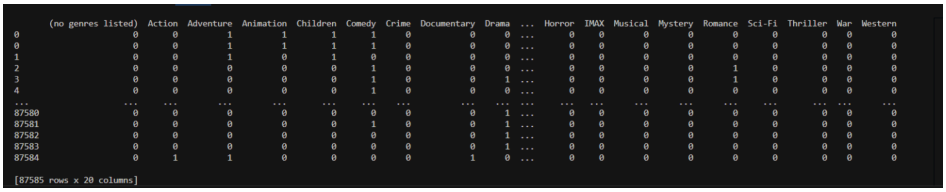
To make our movie recommendation system more complete, we introduced clustering and PCA methods. To be more specific, we used the genre tags of movies as the standard for grouping similar movies. First, we applied PCA to extract the principal components of movie genres, and then grouped movies with similar content into the same cluster. From each cluster, we selected the top-rated movies to recommend to new users. This approach not only addresses the cold-start problem but also makes the movie recommendation system more robust, laying a solid foundation for subsequent personalized recommendations based on collaborative filtering.

Why PCA?

The primary reason for choosing PCA is the high dimensionality and sparsity of the feature matrix constructed from movie genres. Each movie typically has only two or three associated genre tags, but the dataset lists a total of twenty movie genres. This results in a highly sparse feature matrix, and directly applying clustering algorithms (such as K-Means) on such a matrix often yields suboptimal results because the algorithm struggles to compute meaningful similarities between movies effectively.

To address this issue, we used Principal Component Analysis (PCA) to reduce the dimensionality of the genre-based feature matrix. PCA extracts the most significant principal components from the data, enabling us to represent each movie in a lower-dimensional space with dense and compact features. These transformed features retain the most critical information about the movie genres while discarding

redundant or irrelevant details. This not only significantly improves the performance of clustering algorithms but also enhances the interpretability of the clustering results.

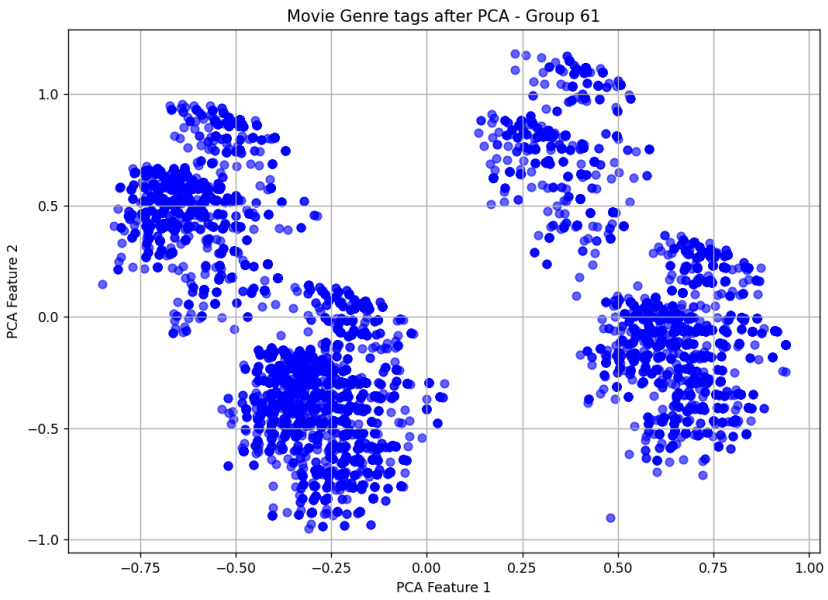


	(no genres listed)	Action	Adventure	Animation	Children	Comedy	Crime	Documentary	Drama	...	Horror	IMAX	Musical	Mystery	Romance	Sci-Fi	Thriller	War	Western
0	0	0	1	1	1	1	0	0	0	...	0	0	0	0	0	0	0	0	0
0	0	0	1	1	1	1	0	0	0	...	0	0	0	0	0	0	0	0	0
1	0	0	1	0	1	0	0	0	0	...	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	1	0	0	0	...	0	0	0	0	0	1	0	0	0
3	0	0	0	0	0	0	1	0	0	1	...	0	0	0	0	0	1	0	0	0
4	0	0	0	0	0	1	0	0	0	...	0	0	0	0	0	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
87580	0	0	0	0	0	0	0	0	1	...	0	0	0	0	0	0	0	0	0
87581	0	0	0	0	0	1	0	0	1	...	0	0	0	0	0	0	0	0	0
87582	0	0	0	0	0	0	0	0	1	...	0	0	0	0	0	0	0	0	0
87583	0	0	0	0	0	0	0	0	1	...	0	0	0	0	0	0	0	0	0
87584	0	1	1	0	0	0	0	1	0	...	0	0	0	0	0	0	0	0	0

[87585 rows x 20 columns]

After encoding, there are a total of twenty genre tags

Additionally, by reducing the dimensions to two, we are able to visualize the clustering results in a two-dimensional space, providing an intuitive representation that helps analyze and validate the effectiveness of the clustering. The introduction of PCA is a crucial step in this method, as it greatly enhances the accuracy and efficiency of the recommendation system.



The distribution of movies after PCA dimensionality reduction

Clustering using K-Means

After performing PCA to reduce the dimensionality of the movie genre features, we applied the K-Means clustering algorithm to group movies with similar content into clusters. K-Means is a simple yet efficient clustering method that works by minimizing the intra-cluster variance, making it suitable for the low-dimensional, compact data produced by PCA. After clustering, we could select the top two highest-rated movies from each cluster for recommendation. The clustering and the final recommended movies are placed in the Result part.

Method 3: Advanced NCF (Supervised Learning)

Knowing about the NCF models, we wonder what would happen if we change the architecture of NCF. The third learning is a supervised method with adjustment on the architecture of original NCF. Specifically, we added an attention mechanism and a hierarchical fusion layer, and we increased the number and size of the layers in MLP.

Attention mechanism

Since we are combining the results of GMF and MLP to generate the results of NeuMF, we can adjust the attention weights for GMF and MLP to focus more on the crucial features while learning. The weighted outputs from GMF and MLP would be combined to form a fusion vector for the next step.

In particular, we first compute the attention weights for both models with Dense layers, which learns the specific weights appropriate for the output of GMF and MLP. At the same time, we use a softmax activation to ensure the normalization, facilitating computation in the next steps. Then, we use the element-wise multiplication to apply attention weights to the outputs of GMF and MLP with Multiply(). This allows the model to selectively amplify or suppress the output features on their significance in learning. Last, we combine the weighted outputs of GMF and MLP into a fusion_vector, instead of merely predict_vector in the original model. In this way, we are able to accomplish the weighted attention distribution of outputs from two models [5].

Why Attention Mechanism?

Not all features are crucial when training. For example, since we are evaluating the recommendation system with ratings, the timestamp would be less significant while this still occurs in the dataset. Therefore, we are eager to find a solution to balance the performance. Attention mechanism would be the answer. It helps to specify the most crucial components in the specific model and boost its influence to make better performance.

Particularly, GMF is to handle the linear relationship between user-item interactions; therefore, attention layer would help GMF to focus more on the simple user-item interactions, such as direct relationship of rating and the movieID for each userID. On the other hand, MLP is responsible to non-linear relationship between users and movies. In this case, attention layer would aid the MLP model to focus more on the complex relationships, or potential features, such as the underlying clusters among the movies. With attention mechanism, the model is able to allocate computational capacity where it matters most in a dynamic way.

Hierarchical Fusion

After having the more weighted fusion_vector, we want to add more layers of the output to refine the fusion, rather than simply concatenate it as the original model. Hierarchical fusion would be an optimal solution for that. Specifically, we pass the fusion_vector into a series of fully connected layers with dropout to further refine the combined representation [6].

The fusion_vector is fed in to Dense() layer with linear transformations to apply a learned weight matrix to the input, and ReLU activation would be introduced to add non-linearity to improve its performance on complex relationship. After each Dense() layer, a dropout layer with 0.3 would be applied to regularize its results. After that, the processed output would be passed to final prediction layer, which predicts the probability that a user will interact with an item.

Why Hierarchical Fusion?

Applying hierarchical fusion rather than simply merging the outputs of two models have several benefits. First, the hierarchical structure ensures that the model learns progressively complex patterns as the data flows through the layers. Each layer builds upon the outputs of the previous one. In addition, a seamless combination between GMF and MLP can be ensured in this way with increasing learning capacity. Deeper layers provide additional learnable parameters, which allows the model to capture more patterns when combining the results, or patterns that can not be fully captured by GMF or MLP alone.

Although we are not using multiple dataset, one of the benefits of hierarchical fusion is its flexibility among different dataset. It enables the model to adapt to the specific patterns of the dataset by learning higher-level representations. Therefore, in the next step if we want to test its generalizability, hierarchical fusion can help to maintain a relatively promising level of performance among different sizes or qualities of datasets.

Change Layer Architecture

Besides adjusting NeuMF, which integrates output of GMF and MLP, we adjust the MLP itself, as it is the main component of deep learning model. We changed the architecture of MLP by make deeper and larger layers of it. Particularly, we changed the layer from [64, 32, 16, 8] to [128, 64, 32, 16, 8].

We add one layer to the original architecture, and adjust its size accordingly. The original model contains four layers with 64 neurons in the first layer, 32 neurons in the second, and so on. Now, we have five layers and the first layer contains 128 neurons, the second 64 neurons, and so on.

Why Change Layer Architecture?

This might be the most intuitive adjustment as we know the different effects neural network generates by different layers. Particularly, increasing the size of the layer would help MLP to capture more features while learning complex patterns. As more neurons passed in in each layer, more fine-grained information of interactions between user and items can be passed in to give better performances. In addition, larger layer would facilitate learning for datasets with a large number of users and items, or diverse outcomes.

Similarly, increasing the depth of neural network by adding more layers can help to capture more complex patterns through a hierarchical way to process feature interactions. Lower layers focus on learning simple, direct relationship between user and items. Middle layers elevate the simple features to more abstract patterns, such as potential user preferences, or latent movie features. Higher layers deal with high-level features that represents non-linear patterns. Therefore, increasing depth of neural network ensures the model to contain higher layers which helps capture higher order relationships between users and items. This may particularly be useful in complicated dataset.

Results and Discussion

Quantitative Metrics

Hit Ratio (HR) and Normalized Discounted Cumulative Gain (NDCG) are two widely used metrics in evaluating recommendation systems, especially for top-K recommendations.

- **Hit Ratio (HR):** Hit Ratio is a straightforward metric that measures the accuracy of the recommendation system in including the relevant (or ground truth) item within the top-K recommended items for a user. A hit occurs if the ground truth item appears in the top-K list, indicating the model's success in recommending relevant content. Hit Ratio is calculated as:

$$\text{Hit Ratio} = \frac{\text{Number of Hits}}{\text{Total Number of Users}}$$

This metric provides a binary assessment for each recommendation, where each hit contributes positively to the ratio. A higher Hit Ratio reflects better recommendation accuracy in terms of recall for the relevant items.

- **Normalized Discounted Cumulative Gain (NDCG):** NDCG is a position-sensitive metric that evaluates the quality of ranked lists, taking into account both the presence of relevant items and their positions in the recommendation list. It is based on the concept of Discounted Cumulative Gain (DCG), which accumulates gains (relevance scores) from top to bottom of the list, applying a logarithmic discount factor to penalize lower-ranked positions. The formula for DCG at rank K is:

$$\text{DCG} = \sum_{i=1}^{K} \frac{2^{\text{rel}_i} - 1}{\log_2(i+1)}$$

where rel_i is the relevance score of the item at position i, rel equals 1 if the item at position i is the ground truth item, and 0 otherwise. To normalize DCG, IDCG (Ideal DCG) is computed based on the best possible ranking, and NDCG is then calculated as:

$$\text{NDCG} = \frac{\text{DCG}}{\text{IDCG}}$$

NDCG ranges from 0 to 1, where a value closer to 1 indicates that the relevant items are ranked highly, improving the recommendation quality from a ranking perspective.

Silhouette Score, Calinski-Harabasz Index and Davies-Bouldin Index are three useful metrics to evaluate clustering models. These metrics provide insights into clustering quality, balancing cohesion and separation.

- **Silhouette Score:** The Silhouette Score measures how similar a data point is to its assigned cluster compared to other clusters. It ranges from -1 to 1:

+1: Indicates the point is well-matched to its cluster and far from others.

0: Suggests the point lies on the decision boundary between clusters.

-1: Implies the point might belong to the wrong cluster.

It is computed as s=(b-a)/max(a,b), where aa is the average intra-cluster distance, and bb is the average distance to the nearest cluster. Higher values indicate better-defined clusters.

- **Calinski-Harabasz Index:** Also called the Variance Ratio Criterion, this index evaluates the ratio of between-cluster dispersion to within-cluster dispersion. It is defined as:

$$CH = \frac{\text{trace}(S_B)/(k-1)}{\text{trace}(S_W)/(n-k)}$$

where S_B is the between-cluster scatter matrix, S_W is the within-cluster scatter matrix, k is the number of clusters, and n is the total number of samples. A higher score indicates better cluster separation.

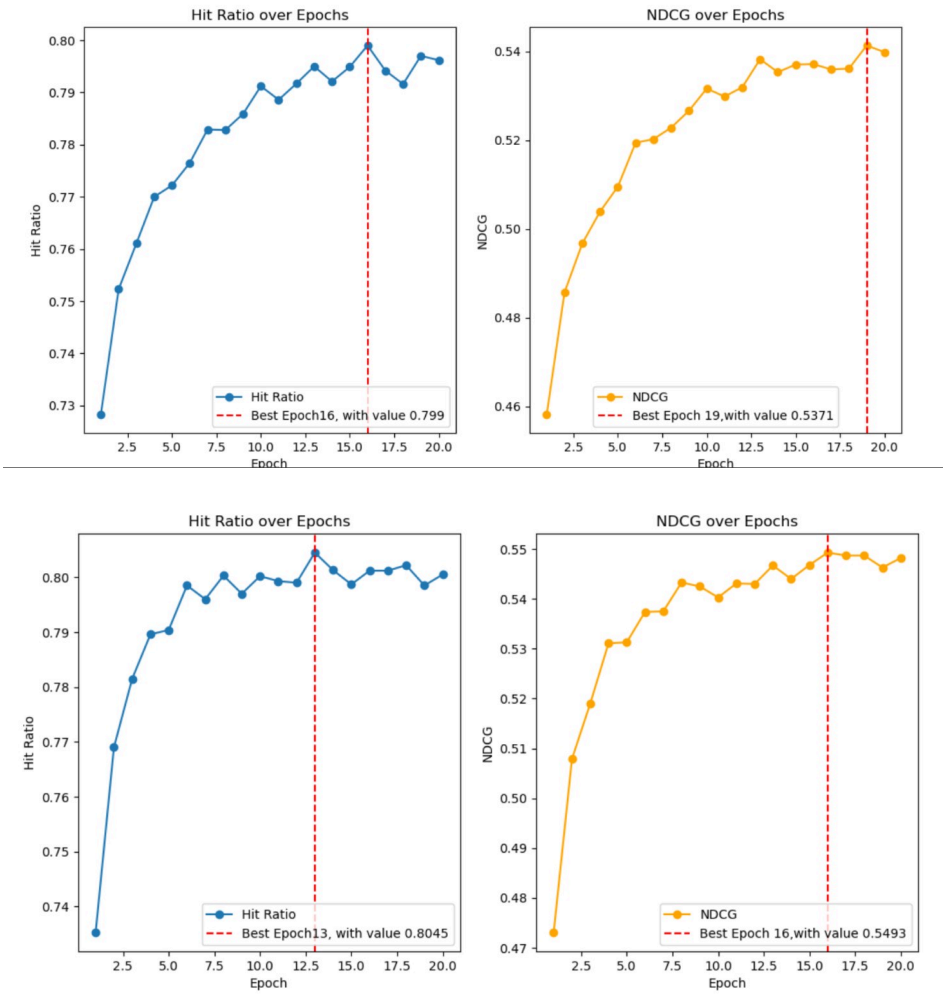
- **Davies-Bouldin Index:** This index measures the average similarity ratio of each cluster with its most similar cluster. A lower value indicates better clustering. It is defined as:

$$DB = \frac{1}{k}\sum_{i=1}^{k} \max\left(\frac{s_i + s_j}{d_{ij}}\right)$$

where s_i is the average distance of points in cluster i to its centroid, and d_{ij} is the distance between centroids of clusters i and j.

Method 1

We plotted two images to represent the convergence speed and evaluation parameter changes after adding a dropout layer to the model. The two values converge at a relatively reasonable value in the end, indicating the results are valid. We can see that the best Hit Ratio remains 0.799 and NDCG is up to 0.5371 for the original model. After applying the Dropout layer, the HR and NDCG performs a little better with the best value of 0.8045 and 0.5493. While there is not a significant increase of the value, we can observe that the accuracy converges faster than the original model, indicating its functionality of regularization.



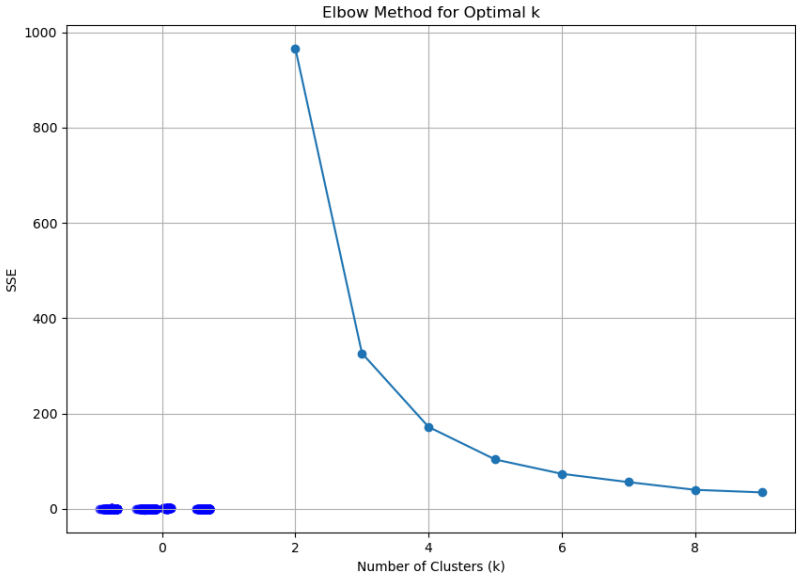
The Hit Ratio and NDCG of two models. Above one is original model and the below one is our updated model

Dropout Layer is a regularization technique used in neural networks to prevent overfitting by randomly "dropping out" a proportion of units in a layer during each training iteration. By adding a dropout layer in our model, we can find that not only the Hit Ratio increases but the converging speed becomes faster.

However, the limited improvement of Dropout layer may be due to several reasons. First, our model may not be overfitting, since in the data preprocessing stage, we randomly select data points from the dataset to generate training and testing data. In this case, Dropout may not be that effective. In addition, the dropout rate we chose may not be optimal. Later we may try to adjust this value to perform better or use other methods, such as batch normalization to elevate the HR and NDCG.

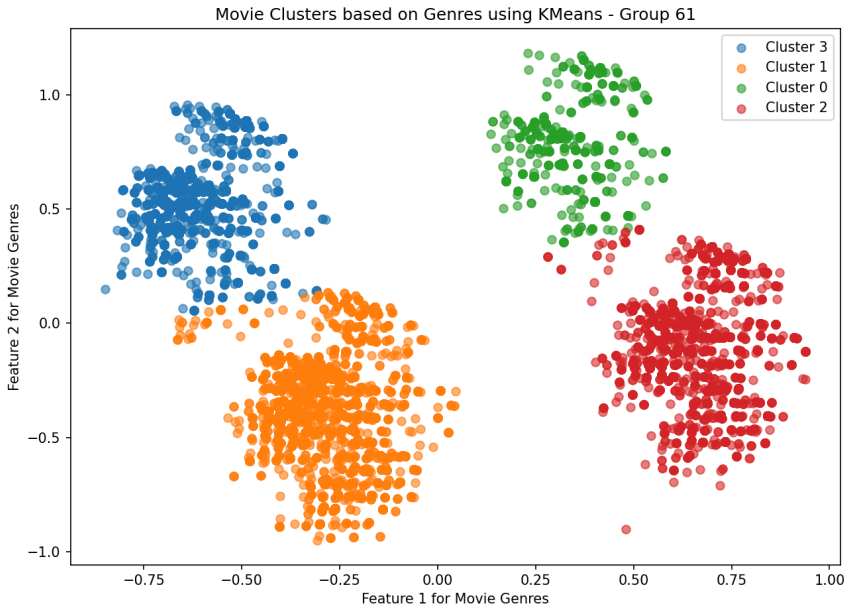
Method 2

Firstly,we apply the Elbow Method to determine the number of clusters we would apply. Finally, we choose to use 4 clusters in the method.



The relation between SSE and Number of Clusters

After applying PCA methods in our dataset, we get the following result, along with these evaluation metrics marked in the plot.



Clustering with K-Means

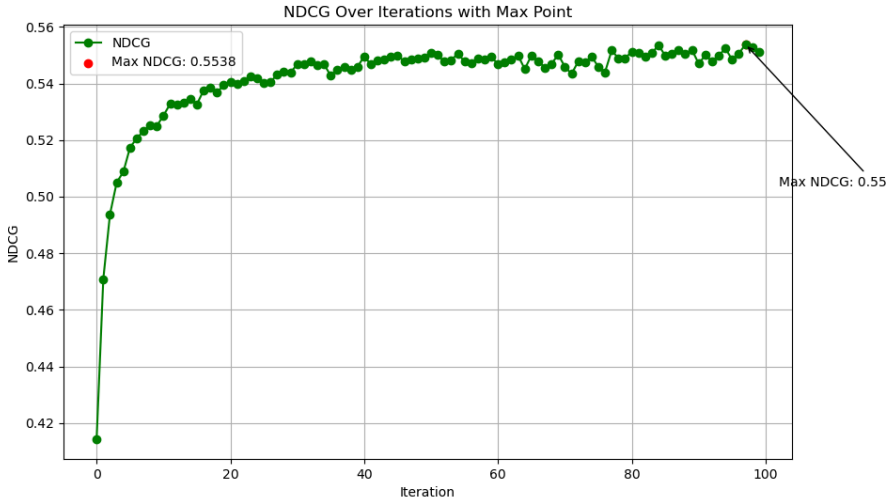
Finally, we get recommendations based on clustering results. Here are the eight movies we recommend based on clustering:

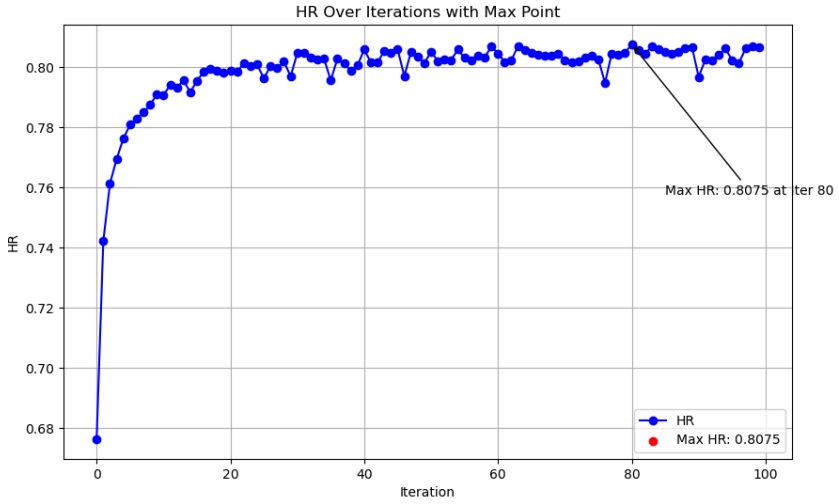
Recommended Movies (Top 2 per Cluster)

	No.	Title	Average rating	Cluster
0	1	Loaded (2014)	5.0000	0
1	2	Girls' Club (1936)	5.0000	0
2	3	As Seen Through These Eyes (2008)	5.0000	1
3	4	Always a Bridesmaid (2000)	5.0000	1
4	5	Deadly Delicious (Shuang Shi Ji) (2008)	5.0000	2
5	6	alaskaLand (2013)	5.0000	2
6	7	Parasites, Les (1999)	5.0000	3
7	8	The Two Firefighters (1968)	5.0000	3

Method 3

After applying hierarchical fusion and rebuilding the NCF layer architecture, we plot the figure indicating the relationship between Hit Ratio and NDCG versus iteration. We find that the modified model demonstrates significantly improved performance, with both Hit Ratio and NDCG showing consistent upward trends as the number of iterations increases. This suggests that the new architecture effectively captures deeper interactions and hierarchical information, leading to better model fitting and enhanced recommendation accuracy.





Comparison and Next Steps

Kmeans is an unsupervised clustering algorithm that groups data points into clusters to maximize intra-cluster similarity and minimize inter-cluster similarity. It is used for grouping users or videos (e.g., dividing users into different interest groups.) and suitable for initial analysis, especially in cold-start or sparse data scenarios. NCF, on the other hand, is a deep learning-based collaborative filtering algorithm that learns user-item interaction features through neural networks, apply in Large-scale user behavior data recommendation and scenario when data is abundant and complex nonlinear relationships need modeling.

K-means Clustering algorithm and Neural Collaborative FIltering algorithm both have their strengths and disadvantages. To be specific, K-means is simple, efficient, and easy to implement. We have provided recommendation results easily without providing information to new users to get recommendations. In other words, K-means can address cold starts by providing general recommendations based on group characteristics.

However, K-means has limited effectiveness for video recommendation problems when handling nonlinear user-item interactions. In this case, NCF can build high-accuracy, personalized video recommendation systems with sufficient user behavior data and computational resources.

In addition, we found out even we adjust the architecture of NCF in the third model, the HR and NDCG did not improve significantly. We suggest that might be due to the size of the dataset, since all strategies we utilized are all actually more effective in larger datasets.

Therefore, in the next step, we may try training and testing the advanced NCF model on a larger and more diverse dataset to make more generalized conclusion. Since deep learning models are often considered

black boxes, it can be challenging to explain why a specific optimization did not work. Considering this, the next step is to study deep learning concepts, gain practical experience, and strive to make effective optimizations to the model.

References

[1] Qilong Ba, Xiaoyong Li, and Zhongying Bai, “Clustering collaborative filtering recommendation system based on SVD algorithm,” 2013 IEEE 4th International Conference on Software Engineering and Service Science, May 2013. doi:10.1109/icsess.2013.6615466

[2] X. He et al., “Neural collaborative filtering,” Proceedings of the 26th International Conference on World Wide Web, pp. 173–182, Apr. 2017. doi:10.1145/3038912.3052569

[3] M. Quadrana, P. Cremonesi, and D. Jannach, “Sequence-aware Recommender Systems,” Proceedings of the 26th Conference on User Modeling, Adaptation and Personalization, Jul. 2018. doi:10.1145/3209219.3209270

[4] Recommendation System https://www.nvidia.com/en-us/glossary/recommendation-system/

[5] Z. Li, Q. Liu, J. He, Q. Liu, and G. Sun, “An improved recommendation algorithm based on attention mechanism,” 2023 International Conference on Advances in Electrical Engineering and Computer Applications (AEECA), pp. 632–637, Aug. 2023. doi:10.1109/aeeca59734.2023.00118 [6] Y. Zhou, J. Guo, H. Sun, B. Song, and F. R. Yu, “Attention-guided multi-step fusion: A hierarchical fusion network for multimodal recommendation,” Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 1816–1820, Jul. 2023. doi:10.1145/3539618.3591950

Gantt Chart

Phase 1 - Proposal							
Week6	2024-09-23	2024-09-29	Determine the Topic	Find the data set			
Week7	2024-10-01	2024-10-07	Proposal Due at Oct 4	Peer Evaluation	Proposal	Presentation File	Video Recording
Phase 2 - Midpoint Report							
Week8	2024-10-09	2024-10-15					
Week9	2024-10-17	2024-10-23					
Week10	2024-10-25	2024-10-31	Group Meeting separate task	Review Proposal's Feedback			
Week11	2024-11-02	2024-11-08	Report Due at Nov 8	Peer Evaluation	Organize Report		
Phase 3 - Final Project							
Week12	2024-11-10	2024-11-16					
Week13	2024-11-18	2024-11-24	Review Proposal's Feedback	Group Meeting seprate task			
Week14	2024-11-26	2024-12-02	Code Implecation	Report	Presentation File	Video Recording	Peer Evaluation

Member Contributions

Member	Final Contributions
Zhouquan Jin	Code implementation of PCA and K-means clustering, Report method part of PCA and K-means clustering, Review the report and final check
Fanshi Meng	Result section in report, Review the report and final check
Xinyue Tao	Introduction and Background section in report, Review the report and final check, Presentation doc, GANTT chart, Contribution table, Video presentati
Ruiqi Xie	Code implementation of Advanced NCF, Report method part of Advanced NCF, Review the report and final check
Wuyuqing Yang	Video presentation, Review the report and final check, Page built, Github repository update