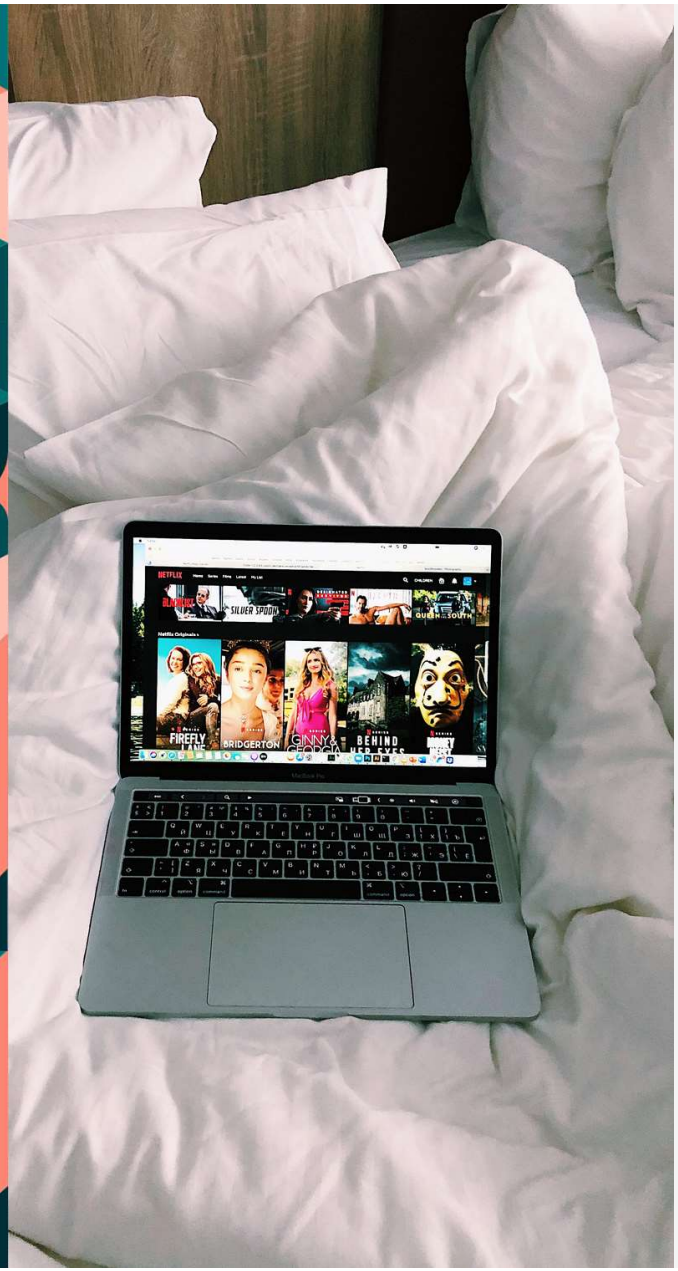


Advanced Artificial Intelligence, Machine Learning and Deep Learning Final Project

Group 2
Ozgur Gumus
Ghazaleh Monsef Shemordeh
Paulette Masengesho

18/06/2024

Assignment 1

Recommender System from Scratch

Neural Matrix Factorization:

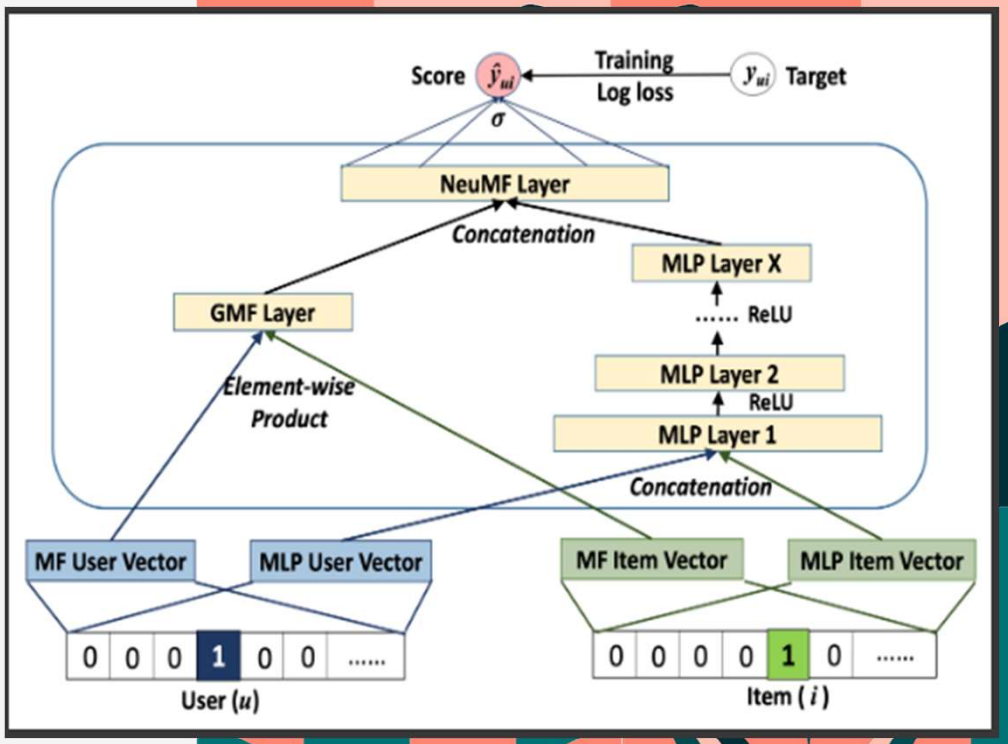
- Combines Generalized Matrix Factorization (GMF) and Multi-Layer Perceptron (MLP) to enhance recommendation systems.

Enhanced Interaction Modelling:

- GMF captures simple, linear interactions.
- MLP captures complex, non-linear interactions.
- Combining both models captures a wider range of user-item interactions.

Improved Recommendations:

- Provides more accurate and personalized recommendations.
- Leverages strengths of both GMF and MLP for robust performance.



Movielens Dataset

We use the MovieLens 100K dataset, which has 100,000 ratings from 1000 users on 1700 movies.
The ratings are given to us in form of <userID, itemID, rating, timestamp> tuples. Each user has a minimum of 20 ratings.

- **User ID:** Unique identifier for each user.
- **Item ID:** Unique identifier for each item.
- **Interaction Value:** Could be a rating, click, or purchase indicator.
- **Timestamp:** When the interaction took place, useful for temporal analysis.

User ID	Item ID	Rating	Timestamp
1	101	5	2023-01-15
2	102	3	2023-01-16
1	103	4	2023-01-17
3	101	2	2023-01-18

How do we manage the data

Convert Ratings: Transform ratings to an implicit scenario (_reindex()):

- Positive interactions = 1
- All other interactions = 0

Split Data:

Splits the data into training and testing sets using _leave_one_out() method, ensuring each user has one interaction in the test set.

Sampling Strategy:

•Positive samples: Existing records in the dataset. get_train_instance() and get_test_instance()

•Negative samples: Assume all missing user-item interactions are negative. (_negative_sampling())

•For each positive interaction, sample 4 random negative interactions.

•Ensures a balance of positive and negative samples for training the classifier.

Evaluation Metrics

Hit Ratio (HR@10):

•**Definition:** Measures if the correct item is among the top 10 recommended items.
•**Purpose:** Indicates how often the actual item is in the top recommendations.

$$HR@10 = \frac{\text{Number of hits}}{\text{Total number of users}}$$

Normalized Discounted Cumulative Gain (NDCG@10):

•**Definition:** Measures the ranking quality of the top 10 recommendations.
•**Purpose:** Takes into account the position of the correct item in the recommended list.

$$\text{DCG@10} = \sum_{i=1}^{10} \frac{rel_i}{\log_2(i+1)}$$

$$\text{NDCG@10} = \frac{\text{DCG@10}}{\text{IDCG@10}}$$

•reli is the Relevance score of the item at position i (1 if relevant, 0 if not).
•IDCG is the ideal DCG, which is the maximum possible DCG.

GMF and MLP

Generalized Matrix Factorization (GMF):

- **Definition:** An extension of matrix factorization that learns linear interactions between users and items.
- **Purpose:** Captures simple, linear interactions.

Structure:

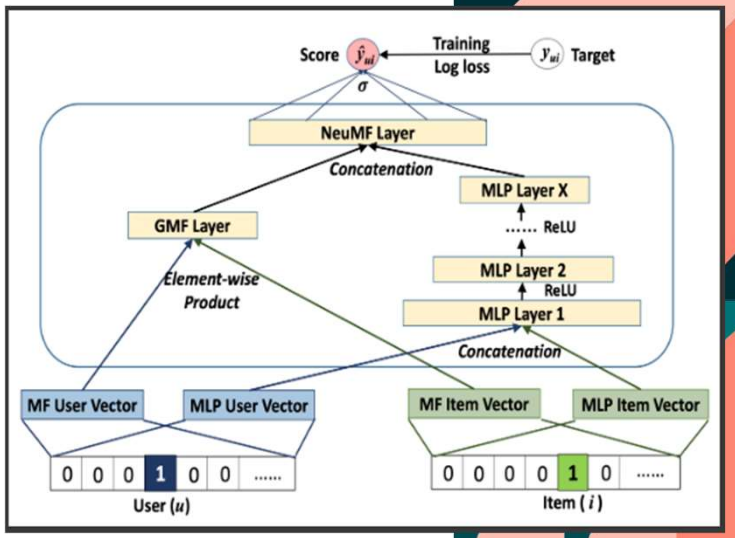
- **User and Item Embeddings:** Represent users and items in a latent feature space.
- **Element-wise Product:** Computes the interaction between user and item embeddings.
- **Output Layer:** Produces a prediction score.

Combining MLP and GMF: Neural Matrix Factorization (NMF)

Structure:

- **Shared Embedding Layer:** Common user and item embeddings for both models.
- **Separate Paths:** GMF and MLP process the embeddings separately.
- **Concatenation and Prediction:** Combine outputs from GMF and MLP for final prediction.

Combining MLP's flexibility and GMF's efficiency, NMF provides a robust method for modeling user-item interactions.

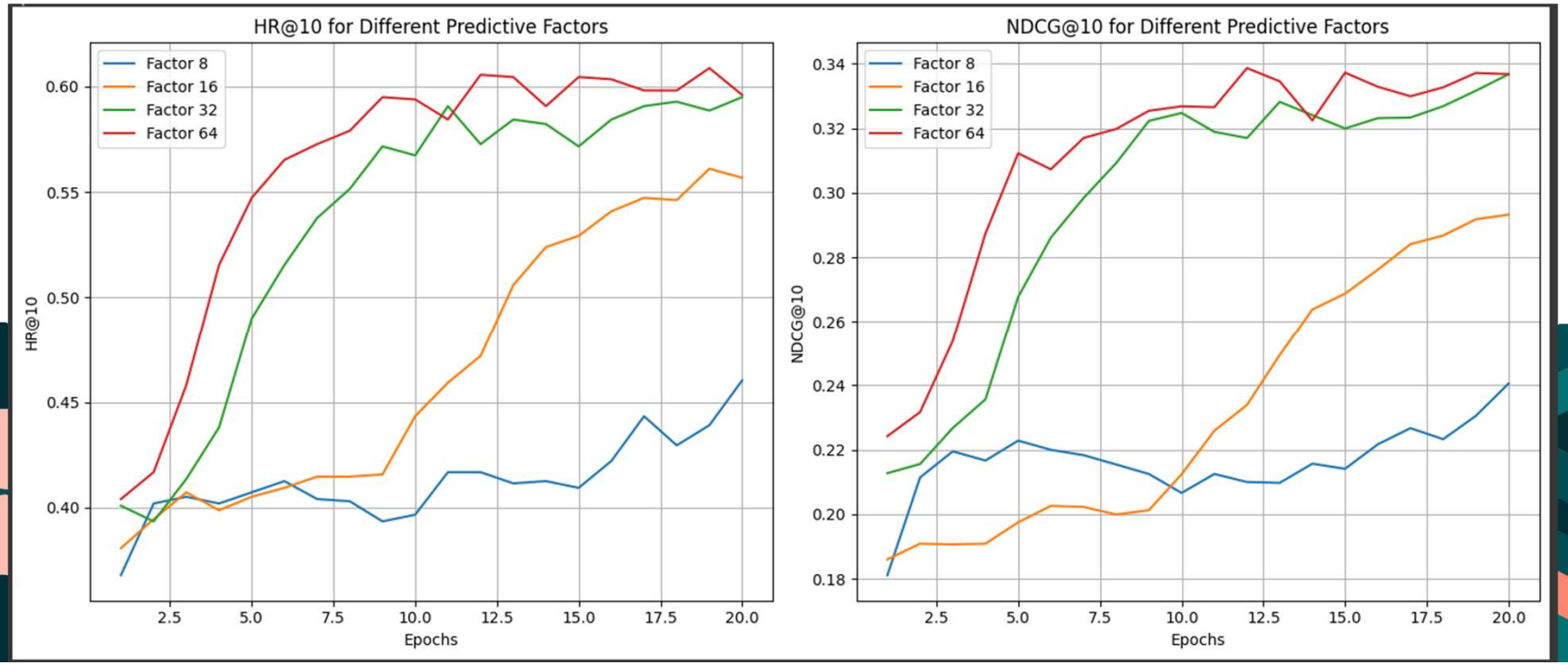


Results

We check the with respect to the number of predictive factors [8, 16, 32, 64] for NCF that we built

HR Measures how often the correct item is predicted and NDCG measures the ranking quality of the top 10 recommendations.

- **Factor 64 (Red):** Best performance, highest HR@10 and NDCG@10.
- **Factor 32 (Green):** Good performance, second highest.
- **Factor 16 (Orange):** Moderate improvement.
- **Factor 8 (Blue):** Slowest improvement, lowest overall.



Assignment 2

Explainable AI

- The primary aim of using explainable AI in recommender systems is to create a more transparent, trustworthy, and user-friendly system in human-understandable terms.
- In this assignment, we added additional information to our model, such as movie titles, release date, genre of the movie, user's age, gender, and profession, to understand how these factors affect our model's ability to recommend movies accurately.

Objectives

- Train a deep neural network on the enhanced dataset using PyTorch.
- Use Captum to analyze and interpret which features were most important and how the network reached its predictions.

Movielens Dataset

In Assignment 1, the dataset focused solely on user-item interactions, including user IDs, item IDs, ratings, and timestamps.

Assignment 2 expanded this by adding movie details (title, release date, genres, description link) and user demographics (age, gender, profession). These additional features aim to improve model accuracy and enable more personalized movie recommendations.

	user_id	item_id	rating	timestamp	age	gender	occupation	zip_code	title	release_date	...	Fantasy	Film-Noir	Horror	Musical	Mystery	Romance	Sci-Fi	Thriller	War	Western
0	196	242	3	881250949	49	M	writer	55105	Kolya (1996)	24-Jan-1997	...	0	0	0	0	0	0	0	0	0	0
1	305	242	5	886307828	23	M	programmer	94086	Kolya (1996)	24-Jan-1997	...	0	0	0	0	0	0	0	0	0	0
2	6	242	4	883268170	42	M	executive	98101	Kolya (1996)	24-Jan-1997	...	0	0	0	0	0	0	0	0	0	0
3	234	242	4	891033261	60	M	retired	94702	Kolya (1996)	24-Jan-1997	...	0	0	0	0	0	0	0	0	0	0
4	63	242	3	875747190	31	M	marketing	75240	Kolya (1996)	24-Jan-1997	...	0	0	0	0	0	0	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
99995	863	1679	3	889289491	17	M	student	60089	B. Monkey (1998)	06-Feb-1998	...	0	0	0	0	0	1	0	1	0	0
99996	863	1678	1	889289570	17	M	student	60089	Mat' i syn (1997)	06-Feb-1998	...	0	0	0	0	0	0	0	0	0	0
99997	863	1680	2	889289570	17	M	student	60089	Sliding Doors (1998)	01-Jan-1998	...	0	0	0	0	0	1	0	0	0	0
99998	896	1681	3	887160722	28	M	writer	91505	You So Crazy (1994)	01-Jan-1994	...	0	0	0	0	0	0	0	0	0	0
99999	916	1682	3	880845755	27	M	engineer	N2L5N	Scream of Stone (Schrei aus Stein) (1991)	08-Mar-1996	...	0	0	0	0	0	0	0	0	0	0

100000 rows x 31 columns

NEW MLP MODEL

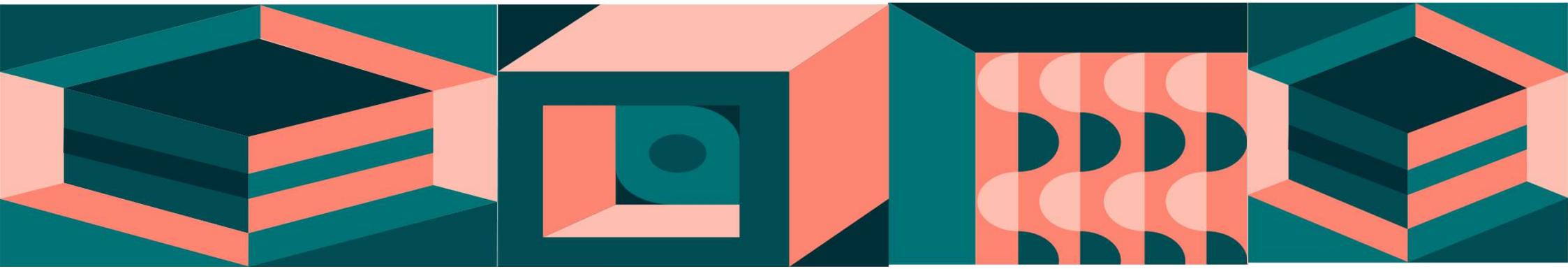
The model enhances data handling by including user and item features in addition to their indices. The dataset was prepared by merging user, item, and ratings data, encoding categorical variables, extracting relevant features, and splitting the dataset into features (x) and target variable (y). This involved one-hot encoding for categorical data and extracting the year of release for date information. The MLP model was trained for several epochs, with performance evaluated after each epoch using hit rate and NDCG at the top 10 predictions to assess recommendation quality.

The best results were: hr@10: 1.0, ndcg@10: 0.9312606617215708

```
Epoch 1/20, Loss: 1.7713558927774429, Test Loss: 1.1319229205131531, HR@10: 0.9, NDCG@10: 0.7945888761465525
Epoch 2/20, Loss: 1.3258313459873199, Test Loss: 1.126025127696991, HR@10: 0.9, NDCG@10: 0.8562709565844046
Epoch 3/20, Loss: 1.2231384754657746, Test Loss: 1.1211804275512696, HR@10: 0.7, NDCG@10: 0.7696657879931038
Epoch 4/20, Loss: 1.1728879509806633, Test Loss: 1.1123889560222626, HR@10: 1.0, NDCG@10: 0.9312606617215708
Epoch 5/20, Loss: 1.1459223960876466, Test Loss: 1.1139882318973542, HR@10: 0.9, NDCG@10: 0.8693706609392435
Epoch 6/20, Loss: 1.1354971200704576, Test Loss: 1.1144884311676024, HR@10: 0.8, NDCG@10: 0.8889833109441271
Epoch 7/20, Loss: 1.1244791563749312, Test Loss: 1.1053902613639832, HR@10: 0.9, NDCG@10: 0.8488399080003002
Epoch 8/20, Loss: 1.1150532513022422, Test Loss: 1.109481538915634, HR@10: 0.8, NDCG@10: 0.8379914071441864
Epoch 9/20, Loss: 1.111498880469799, Test Loss: 1.10291031126976, HR@10: 0.8, NDCG@10: 0.8489872673472417
Epoch 10/20, Loss: 1.1062166270017624, Test Loss: 1.0984316017627715, HR@10: 0.8, NDCG@10: 0.8215338413257087
Epoch 11/20, Loss: 1.1062370218276978, Test Loss: 1.0996685044765473, HR@10: 0.9, NDCG@10: 0.9119632934807679
Epoch 12/20, Loss: 1.1042817186832428, Test Loss: 1.091455059146881, HR@10: 0.8, NDCG@10: 0.8480729454022794
Epoch 13/20, Loss: 1.1033594709277152, Test Loss: 1.0906037564754487, HR@10: 0.7, NDCG@10: 0.8862202595073044
Epoch 14/20, Loss: 1.1002715816259385, Test Loss: 1.1060885109901428, HR@10: 0.8, NDCG@10: 0.8779060464241195
Epoch 15/20, Loss: 1.0956347774982453, Test Loss: 1.0844011430263518, HR@10: 0.8, NDCG@10: 0.8718776403837345
Epoch 16/20, Loss: 1.095231030535698, Test Loss: 1.087475179052353, HR@10: 0.7, NDCG@10: 0.8547016575084689
Epoch 17/20, Loss: 1.0940866896033288, Test Loss: 1.0893110891819, HR@10: 0.8, NDCG@10: 0.9233463394127297
Epoch 18/20, Loss: 1.0920590510725976, Test Loss: 1.0903705729961395, HR@10: 0.8, NDCG@10: 0.8794025876245247
Epoch 19/20, Loss: 1.0912723059177398, Test Loss: 1.088985029411316, HR@10: 0.7, NDCG@10: 0.857921083681506
Epoch 20/20, Loss: 1.0905516947031022, Test Loss: 1.0822072010040282, HR@10: 0.8, NDCG@10: 0.8624505029608284
```

Comparison between the 2 models

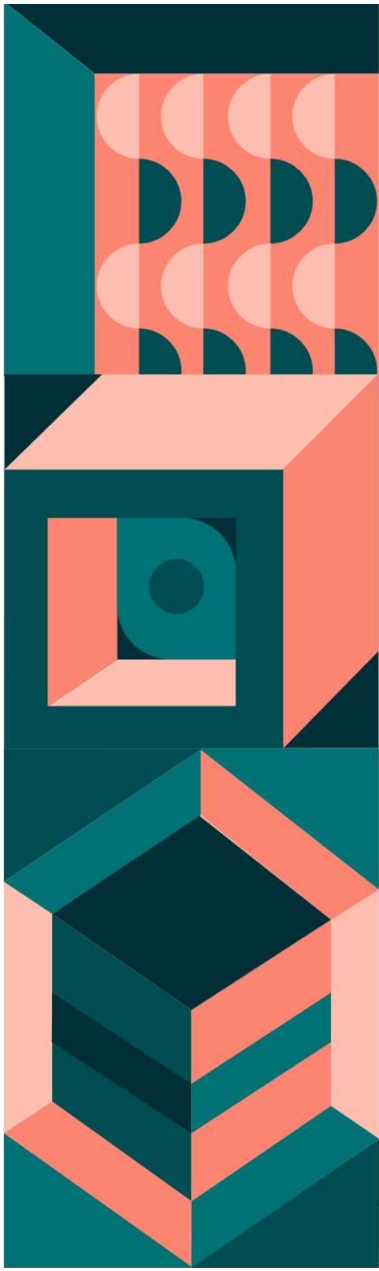
These two recommendation systems take different approaches. The first prioritizes collaborative filtering with embedding for implicit feedback scenarios, like users clicking on items without explicit ratings. The second model goes beyond collaborative filtering and predicts ratings directly. It uses a wider range of data, including user demographics and item details, making it suitable for situations with richer contextual information available.



Adding LIME

LIME is used to explain the predictions made by the trained model. LIME requires a function that takes movie and user information and uses the trained model to predict a rating.

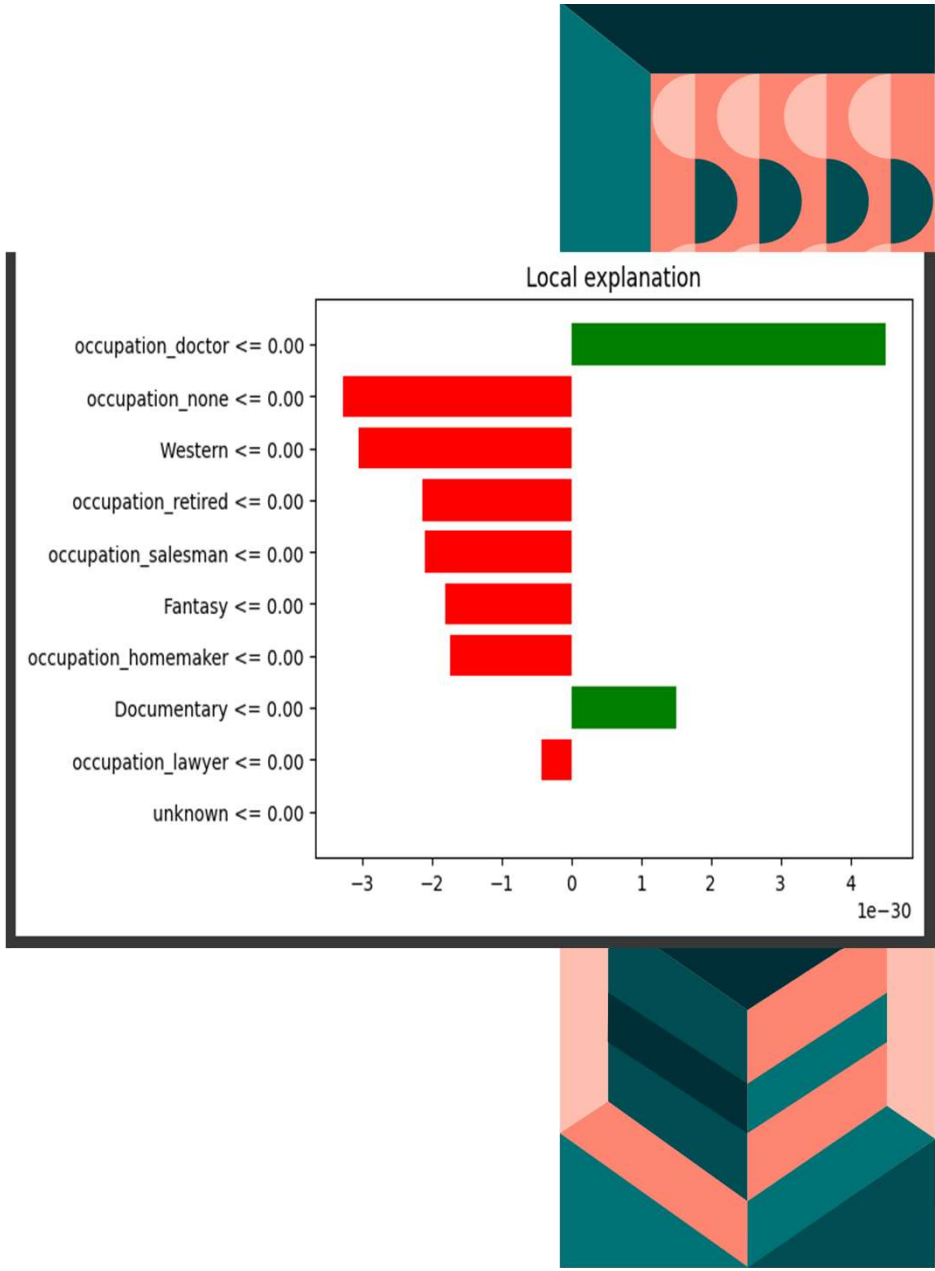
To build a movie rating prediction model using LIME, we first prepare the data by loading user and movie information, converting text features to numbers, and potentially creating new features. This data is then split into training and testing sets for the model to learn from and be evaluated on. Next, a multi-layer perceptron neural network is trained on the data, with its performance monitored on the unseen testing data to ensure it generalizes well to new movies and users.



LIME needs information about the features used by the model to make explanations. Once set up, it can analyze a particular data point from the testing set. LIME then explains the model's prediction for this chosen point, identifying the most influential features.

In essence, LIME helps you understand how a model makes decisions for individual predictions by highlighting the most important features.

LIME can identify the top features that most influenced the model's rating prediction. The results shows that occupation_doctor and documentary impacted positivery the model.

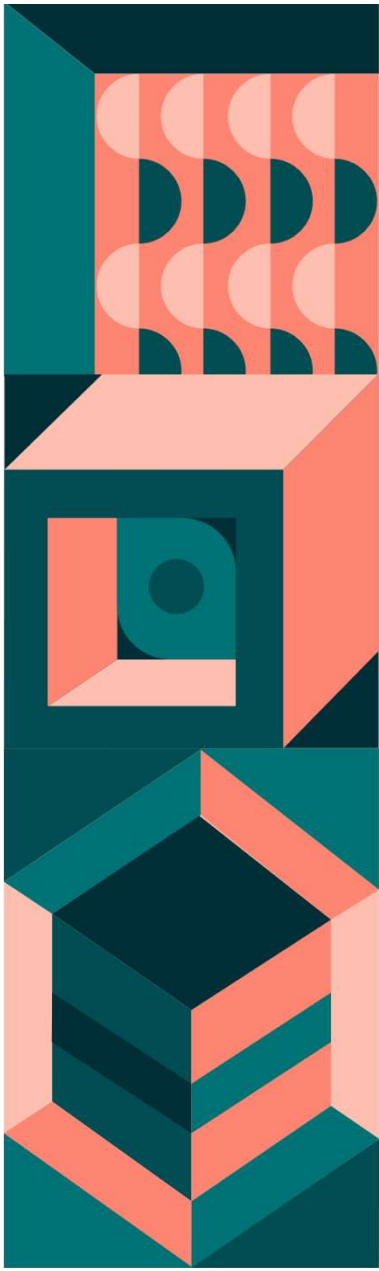


Implementing SHAP

SHAP was used in the code to provide interpretable explanations of the model's predictions, ensuring fair attribution of feature importance and aiding in model debugging and trust-building.

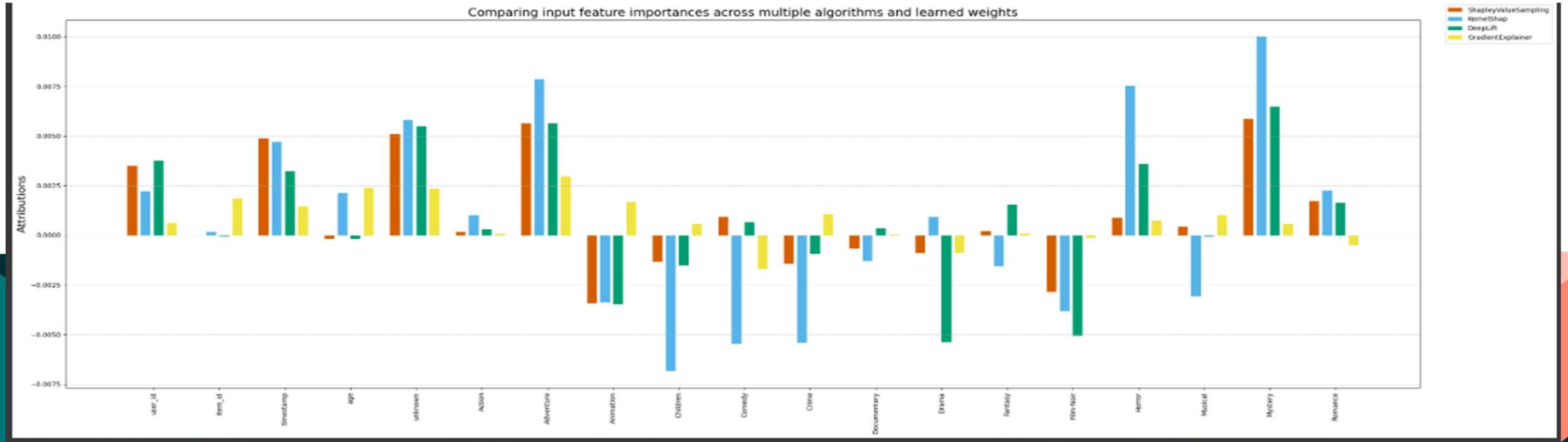
Libraries used:

- !Pip install shap: used for explaining the predictions of machine learning models. It calculates feature importances to understand how different features contribute to a model's output.
- !Pip install captum: it provides various algorithms for explaining model predictions, including gradient-based methods and shap-based methods.
- Pip install --upgrade captum: upgrading ensures you have access to the newest features and bug fixes.



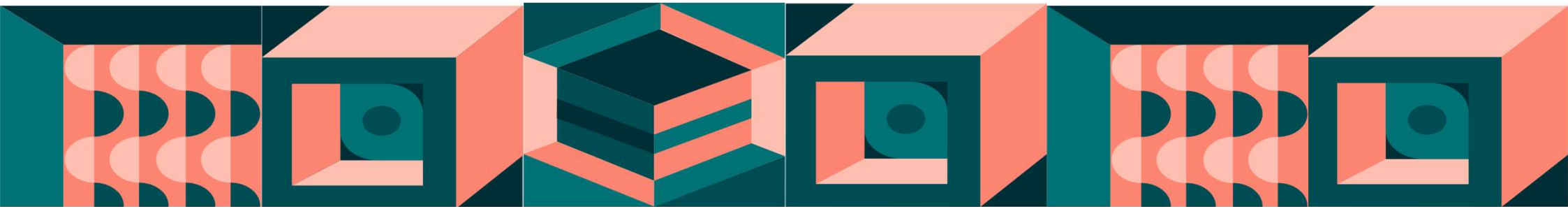
Results

The graph compares feature importance across multiple explainability algorithms for a simplified multi-layer perceptron (MLP) model. Positive values indicate a positive contribution to the prediction, while negative values indicate a negative contribution. Some features, like item_id and adventure, have consistent positive attributions across all methods, indicating their robust importance in the model's predictions. Other features show variations in importance between the methods, highlighting areas that need further investigation.



SHAP vs LIME

SHAP and LIME are two distinct explainability methods. SHAP uses Shapley values from game theory to provide deep insights into model behavior and feature interactions, offering stable and consistent explanations across data points and models. However, it can be computationally expensive for complex models or large datasets. LIME creates simpler models that mimic complex model behavior around specific data points, offering easy-to-understand local explanations for individual predictions. LIME is generally faster and useful for quick explanations or debugging specific outcomes, though its explanations can vary and may lack consistency. Choosing between SHAP and LIME depends on whether in-depth understanding (SHAP) or quick, localized explanations with limited computational resources (LIME) are needed.



Assignment 3

Representation Learning for Movie on Graph

- The primary goal is to improve movie recommendation systems using advanced graph representation learning techniques.
- By understanding the relationships between movies through user interactions, we can provide more accurate and relevant recommendations.

Graph Representation Learning:

- It involves representing data in the form of a graph where entities (movies) are nodes, and relationships (user ratings) are edges.
- This method captures complex interactions that traditional methods might miss.

Visualization and Analysis:

- Visualizing embeddings helps in understanding how well the model has captured the relationships between movies.
- It provides a tangible way to see clusters of similar movies and evaluate the model's performance.

Movies Graph

Smaller dataset is used to with respect to the 100k dataset. The used dataset is the ml-latest-small.

	movieId	title	genres
0	movie_1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy
1	movie_2	Jumanji (1995)	Adventure|Children|Fantasy
2	movie_3	Grumpier Old Men (1995)	Comedy|Romance
3	movie_4	Waiting to Exhale (1995)	Comedy|Drama|Romance
4	movie_5	Father of the Bride Part II (1995)	Comedy

	userId	movieId	rating	timestamp
0	1	movie_1	4.0	964982703
1	1	movie_3	4.0	964981247
2	1	movie_6	4.0	964982224
3	1	movie_47	5.0	964983815
4	1	movie_50	5.0	964982931

Random Walks and Training Pairs

Our purpose is to define functions to build random walks and generate training pairs for the Skip-Gram model.

Random Walks:

- Generates sequences of nodes (movies) by traversing the graph.
- Start from a node and move to the next node based on edge weights.
- Higher weight edges have a higher probability of being chosen.

Sample Generation and DataLoader

- Objective is to generate training samples for the Skip-Gram model.
- Create a DataLoader to manage the training data efficiently

Generating Samples:

- For each walk, create (target, context) pairs within a specified window size.
- Labels: 1 for real context pairs, 0 for negative samples.
- Weights to balance positive and negative samples.

Creating DataLoader:

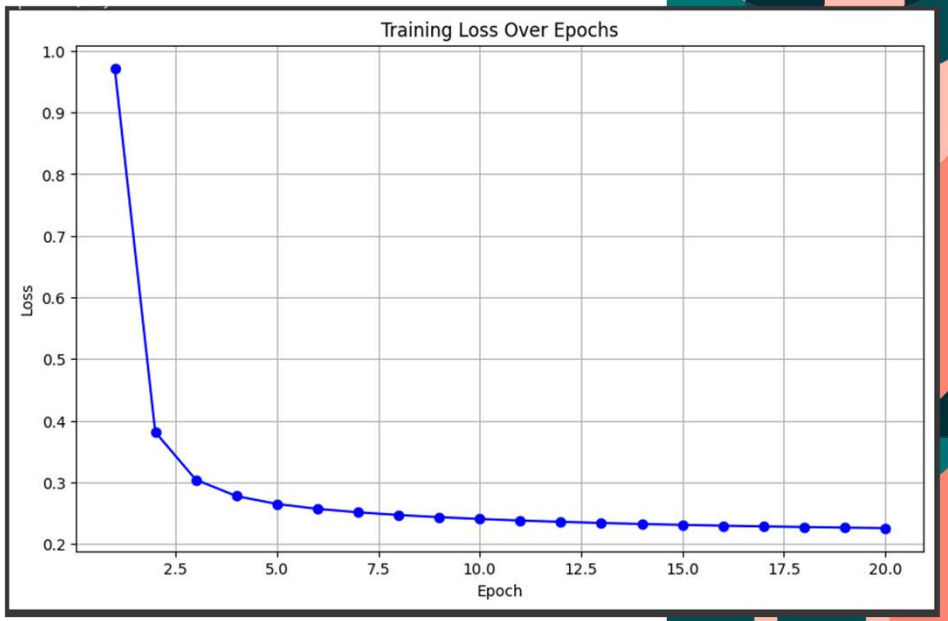
- Purpose is to efficiently batch and shuffle data during training.
- PyTorch's DataLoader is used to manage the dataset.
- Ensures batches are created and fed into the model during each epoch.

Skip-Gram Model with Negative Sampling:

- The Skip-Gram model aims to predict the context words (movies) given a target word (movie).
- Negative sampling is used to efficiently train the model by updating only a small number of weights per iteration.

Word2Vec

- The Word2Vec model is designed to learn vector representations (embeddings) of words or, in this case, movies.
- It consists of two embedding layers: one for target words (movies) and one for context words (movies).



Training Results:

- The model starts with a high loss, indicating initial poor performance.
- Over 20 epochs, the loss decreases significantly, showing improvement.
- The plot shows the training loss decreasing and stabilizing, indicating effective learning and convergence.

Results

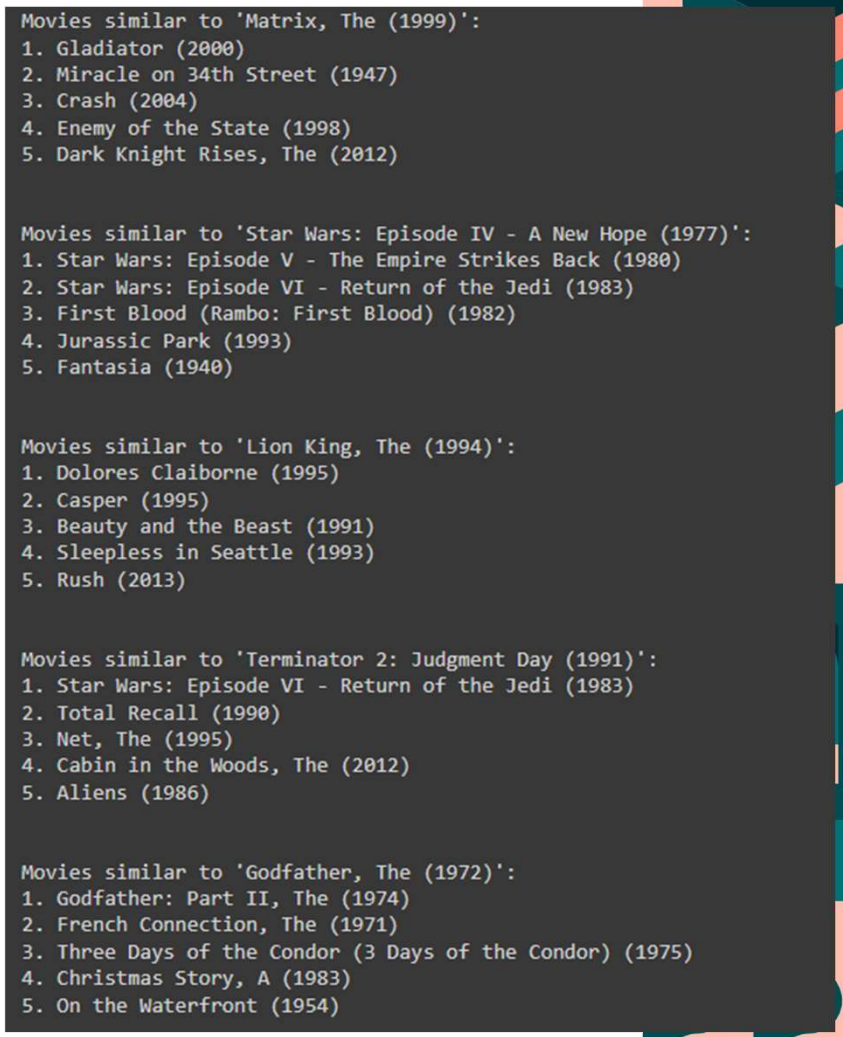
Similar to Star Wars: Episode IV: Star Wars: Episode V - The Empire Strikes Back (1980) and Star Wars: Episode VI - Return of the Jedi (1983) are direct sequels, making them natural recommendations. Also, First Blood (1982) and Jurassic Park (1993), while being action-packed like Star Wars, differ in setting and themes.

Similar to Lion King: Beauty and the Beast (1991) and Casper (1995) are fitting choices as they are family-friendly and involve themes of love and personal growth. Also, Total Recall (1990) and Aliens (1986) are excellent matches due to their science fiction and action elements, similar to "Terminator 2".

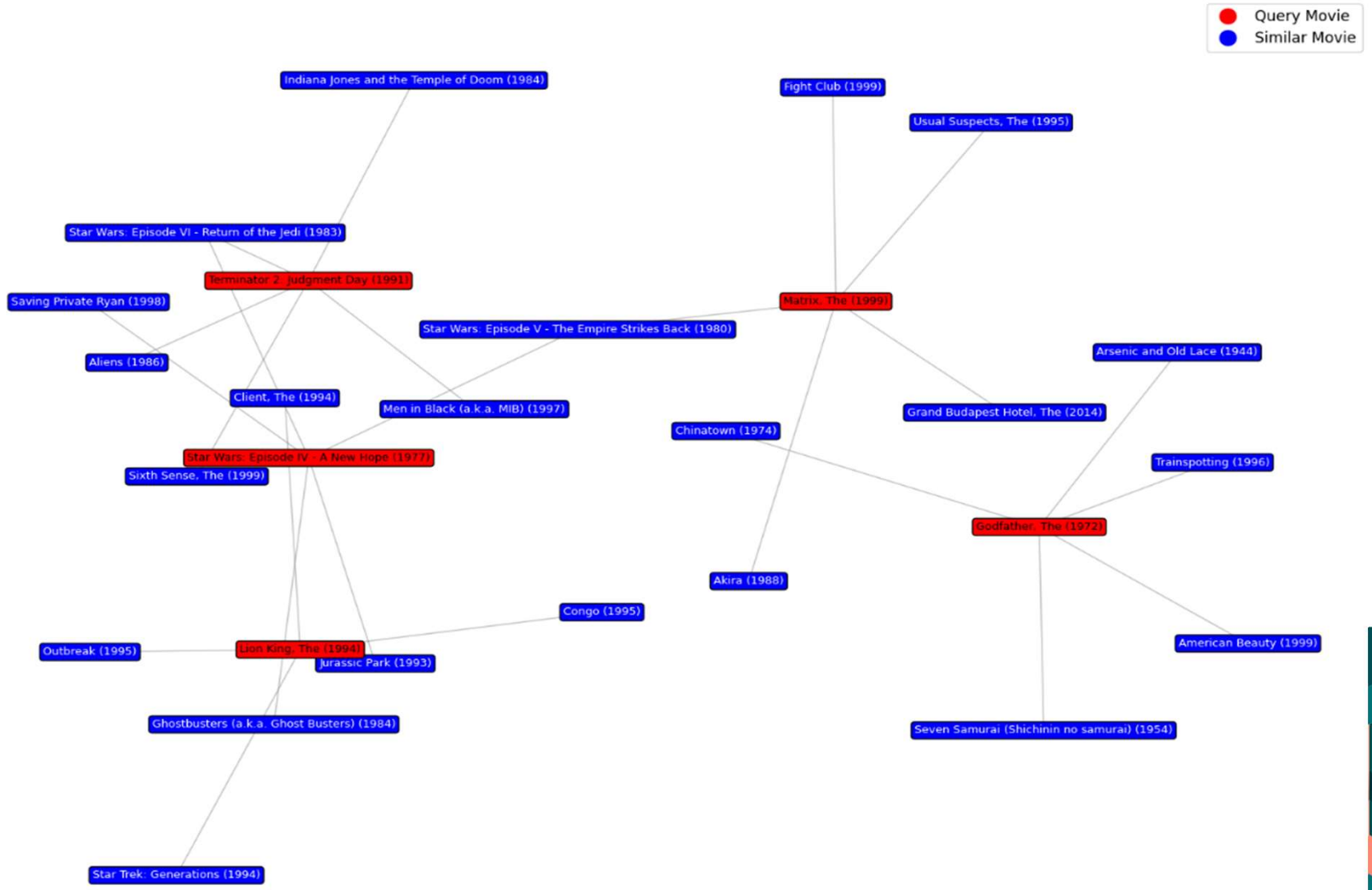
Similar to Godfather: Godfather: Part II, The (1974) is a direct sequel, making it a perfect match. French Connection, The (1971) and On the Waterfront (1954) are strong matches due to their crime and drama themes.

Similar to Matrix: Gladiator (2000) and Dark Knight Rises, The (2012) are action-packed and epic movies, which share a similar intense and dramatic tone with "The Matrix".

General Comments *Overall Accuracy: *The recommendations are generally accurate, especially for movies within the same franchise or those with similar genres and themes. However, there are some notable outliers that do not fit the expected genre or thematic elements of the query movies.



Final Graph

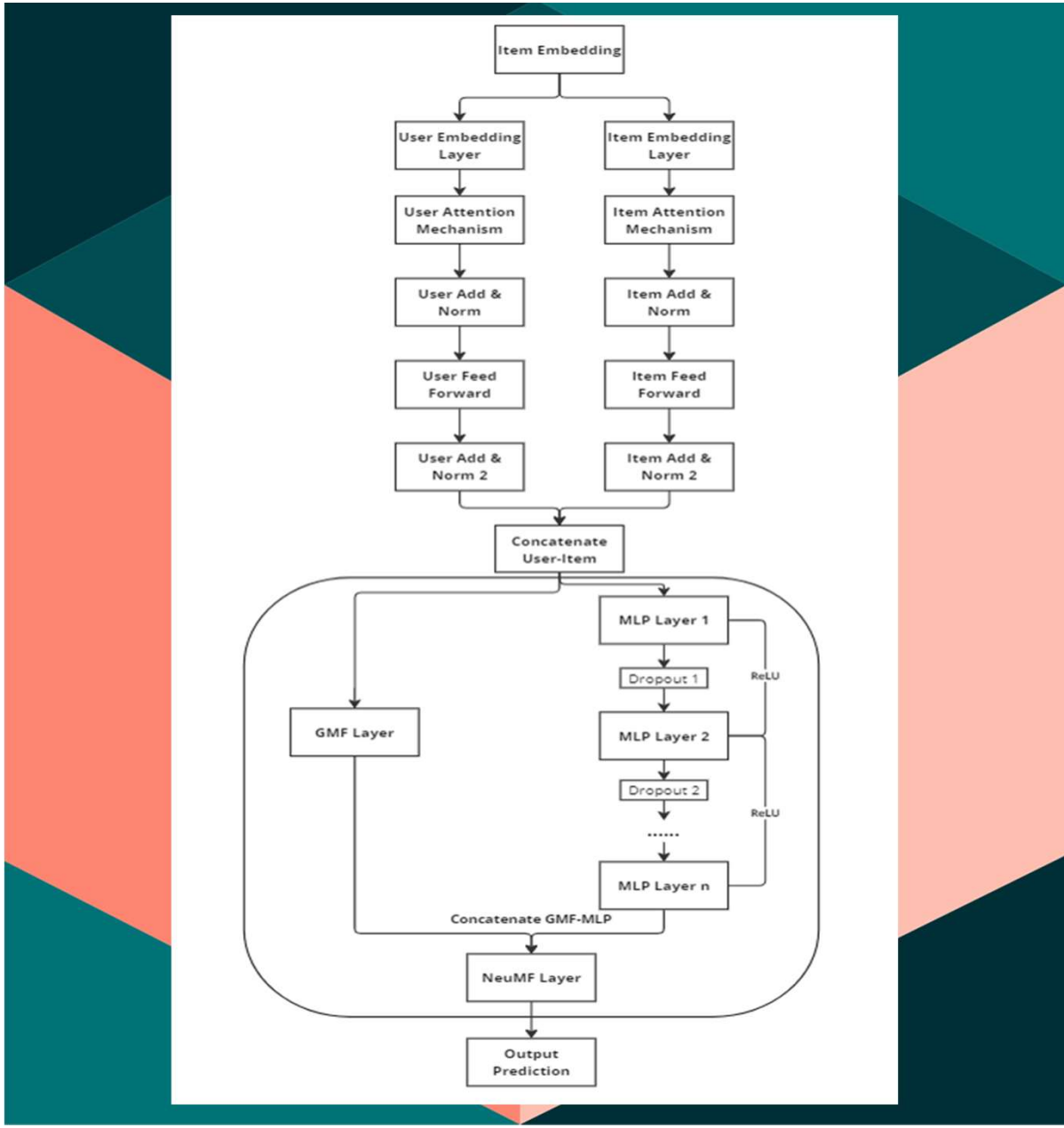

Query Movie
Similar Movie
Indiana Jones and the Temple of Doom (1984)
Fight Club (1999)
Usual Suspects, The (1995)
Star Wars: Episode VI - Return of the Jedi (1983)
Terminator 2: Judgment Day (1991)
Saving Private Ryan (1998)
Matrix, The (1999)
Star Wars: Episode V - The Empire Strikes Back (1980)
Arsenic and Old Lace (1944)
Aliens (1986)
Client, The (1994)
Men in Black (a.k.a. MIB) (1997)
Grand Budapest Hotel, The (2014)
Chinatown (1974)
Star Wars: Episode IV - A New Hope (1977)
Trainspotting (1996)
Sixth Sense, The (1999)
Godfather, The (1972)
Akira (1988)
Congo (1995)
American Beauty (1999)
Outbreak (1995)
Lion King, The (1994)
Jurassic Park (1993)
Ghostbusters (a.k.a. Ghost Busters) (1984)
Seven Samurai (Shichinin no samurai) (1954)
Star Trek: Generations (1994)

Assignment 4

Attention-based Recommender System

- Attention mechanisms improve our recommender system's performance by dynamically weighting the most relevant features in user and item embeddings. This focus allows the model to capture subtle patterns and complex interactions, leading to more accurate predictions.

- By adapting to individual user-item pairs, attention enhances personalization, ensuring each recommendation is tailored to specific preferences, ultimately providing a more precise and effective recommendation system.

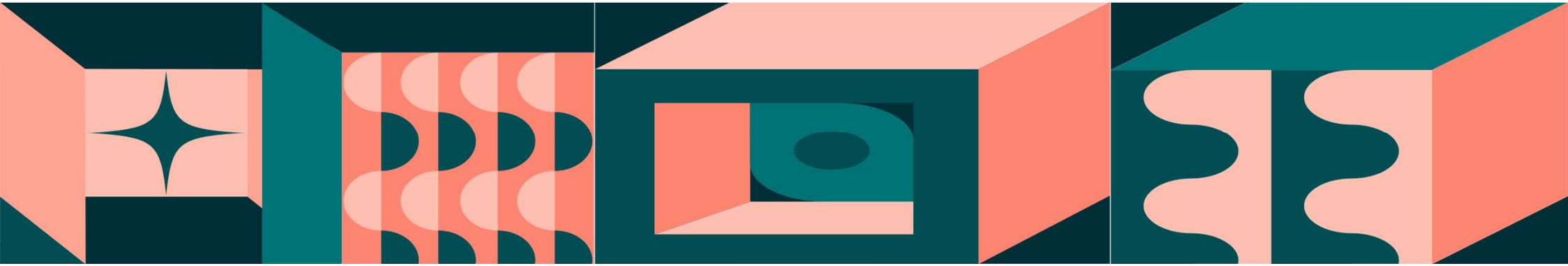


Attention classes

Linear Layers: Linear layers are used to project the input embeddings into query, key, and value vectors.

Computing Attention Weights: The attention weights are computed as the scaled dot product of the query and key vectors, followed by a softmax operation to ensure the weights sum to one.

Applying Attention Weights: The attention weights are then used to compute a weighted sum of the value vectors, which represents the focused information.



NCFWithAttention Class

Using Binary Cross-Entropy with Logits Loss and the Adam optimizer, we trained the model across multiple epochs. During training, the model calculated predictions and attention weights, updated parameters, and recorded training and validation losses**.**

Attention Layers for Users and Items: Two separate attention layers were introduced for user and item embeddings. These layers learn to focus on different parts of the user and item embeddings.

Applying Attention: In the forward pass, attention is applied to the user and item embeddings. The attention mechanism computes a weighted sum of the embeddings, where the weights are determined by the relevance of each part of the embeddings.

Concatenation: The attention-weighted user and item embeddings are concatenated to form a combined vector.

MLP Layers: The concatenated vector is then passed through multiple MLP layers, which further process the information to make the final prediction.

Output Layer: The processed vector is passed through a linear layer to produce the predicted rating or interaction probability.

Improve the performance of recommendations by leveraging the attention mechanism

Attention-based version of the recommender system, shows better performance metrics (HR and NDCG) compared to the original NeuMF model. The improvement can be attributed to the enhanced capability of the attention mechanism to capture more relevant interactions between users and items.

Why the Attention-Based Model Performs Better Attention Mechanism: The attention mechanism allows the model to focus on the most relevant parts of the input data, effectively weighing the importance of different user-item interactions. This helps in capturing more nuanced relationships.

Enhanced Representation: By focusing on the relevant interactions, the model generates more informative user and item embeddings, leading to better predictions.

Adaptive Weights: Unlike static weights in traditional neural networks, the attention mechanism dynamically adjusts the weights of the interactions based on their relevance. This adaptability helps the model generalize better to different user preferences and item characteristics.

Learning Important Features: The attention mechanism helps the model to learn and emphasize important features while ignoring the less important ones, leading to improved generalization on the test set and consequently better performance metrics.

Results

- The two graphs illustrate the neural collaborative filtering (NCF) model's performance over 20 epochs. The first graph, "HR and NDCG over Epochs," shows Hit Rate (HR@10) and Normalized Discounted Cumulative Gain (NDCG@10). HR@10 increases from 0.45 to 0.65 within 7 epochs, while NDCG@10 rises from 0.40 to 0.55, indicating rapid improvement in recommendation accuracy and quality.
- The second graph, "Training and Validation Loss over Epochs," tracks the model's learning process. Training loss decreases steadily from 0.35, showing effective learning. Validation loss starts at 0.15, decreases slightly, and then stabilizes, suggesting optimal performance without overfitting.