**Movie Recommender System**

Introduction to AI

Instructor: Dr. Hala El Zayed

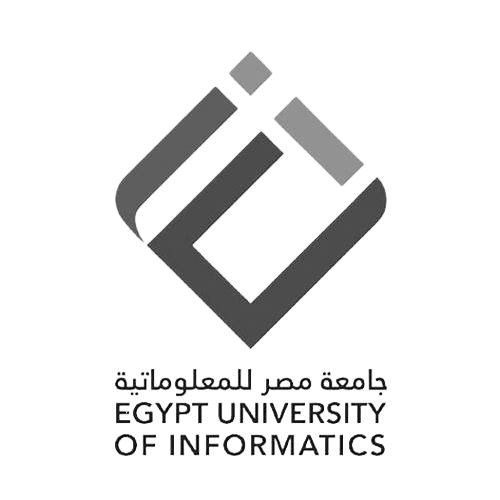
Academic year: 2024-2025

Submitted by:

Mourad ElShorbagy, 22-101026

Amr Edris, 22-101097

Youssef Haytham, 22-101229



COMPUTER

ENGINEERING

Table of Contents

[Problem 3](#_Toc199079693)

[Approach 3](#_Toc199079694)

[Data Preprocessing: 3](#_Toc199079695)

[Steps Performed: 3](#_Toc199079696)

[Collaborative Filtering (Neural Collaborative Filtering - NeuMF): 4](#_Toc199079697)

[Training Strategy Overview 5](#_Toc199079698)

[Evaluation Objective 5](#_Toc199079699)

[Final Prediction Pipeline 5](#_Toc199079700)

[Why Collaborative Filtering 5](#_Toc199079701)

[Architecture (NeuMF): 5](#_Toc199079702)

[General Matrix Factorization (GMF): 5](#_Toc199079703)

[Final GMF Model: 6](#_Toc199079704)

[Multilayer Perceptron (MLP): 6](#_Toc199079705)

[Final MLP Model: 7](#_Toc199079706)

[Neural Matrix Factorization (NeuMF): 7](#_Toc199079707)

[Steps of the NeuMF model 8](#_Toc199079708)

[Final Model: 9](#_Toc199079709)

[Strengths of NeuMF 9](#_Toc199079710)

[Weaknesses of NeuMF 9](#_Toc199079711)

[Results 10](#_Toc199079712)

[Challenges 10](#_Toc199079713)

[References 12](#_Toc199079714)

# Problem

In today’s digital age, the entertainment landscape has drastically evolved, with streaming platforms offering users access to an immense and ever-growing library of movies and TV shows. While this abundance of content provides unparalleled choice, it also creates a paradox of choice—users often feel overwhelmed by the sheer volume of available titles. With thousands of movies spanning numerous genres, languages, and styles, deciding what to watch can become a time-consuming and frustrating process. Many users find themselves aimlessly scrolling through endless lists, unsure of what will truly capture their interest or meet their current mood. This phenomenon, often referred to as decision fatigue, can lead to dissatisfaction, decreased platform engagement, and in some cases, users abandoning the service altogether.

Moreover, the diversity of user preferences adds another layer of complexity. Everyone’s taste in movies is unique, influenced by factors such as genre affinity, favorite actors or directors, past viewing history, and even current trends. Generic or one-size-fits-all recommendation approaches fail to cater to these nuances, resulting in irrelevant suggestions that do little to enhance the viewing experience.

A personalized movie recommender system offers a practical solution to this challenge by automatically analyzing users’ past behaviors, preferences, and movie attributes to generate tailored recommendations. By intelligently filtering through the massive content pool, such systems can present users with options that align closely with their interests, reducing browsing time and improving satisfaction. Beyond improving the user experience, personalized recommendations have a significant impact on platform engagement metrics, such as watch time, user retention, and overall satisfaction. Streaming services that implement effective recommendation engines gain a competitive edge by keeping their users more engaged and fostering loyalty in an increasingly crowded market.

# Approach

## **Data Preprocessing:**

The dataset used was **movielens-1m-dataset** by Oded Golden, sourced from Kaggle. The file used: was ratings.dat

### Steps Performed:

1. **Dataset Download and Extraction**
   * Used the Kaggle API to download and extract the required files.
   * Only ratings.dat was extracted for use.
2. **Cleaning the data**
   * Dealt with NaN whether it is by removing the whole row for missing critical info (id) or placing 0 (budget, revenue).
   * Centered the ratings colum by subtractin the mean (new mean = 0)
3. **Extracting correct posters url**
   * Using tmdb API, we were able to extract the correct poster url with the help of the given tmdb ID. (extracted only the first 5000 movies)
4. **Preprocessing:**
   * This converts real userId and movieId values (which can be arbitrary integers) into **consecutive integer indices** starting from 0 for the embedded layers.
   * Counted how many unique users and movies there are after encoding.
   * Centered the rating by subtracting it with the mean.
   * Split the Data into a training and a test set

## Collaborative Filtering (Neural Collaborative Filtering - NeuMF):

To model user preferences based solely on historical user-item interactions, we implemented a **Neural Collaborative Filtering (NeuMF)** approach. This model combines the strengths of two other models:

* **Generalized Matrix Factorization (GMF)**:  
  Captures linear user–item interactions using element-wise multiplication of user and movie embeddings.
* **Multi-Layer Perceptron (MLP)**:  
  Captures complex, non-linear relationships through dense layers applied to the concatenated user and movie embeddings.

Our hybrid system integrates these approaches, balancing content attributes with user behavior patterns to provide personalized and diverse recommendations.

## **Training Strategy Overview**

The GMF and MLP branches were first trained independently using their respective best hyperparameters found via Keras Tuner (Hyperband). Once optimized, their learned embeddings were merged and fine-tuned end-to-end in the NeuMF model.

## Evaluation Objective

The models were trained to minimize Mean Squared Error (MSE) between predicted and actual (centered) ratings, ensuring better alignment with real user preferences.

## **Final Prediction Pipeline**

For each user, the model predicted scores for unseen movies. The top-N movies with the highest predicted ratings were selected as personalized recommendations.

## **Why Collaborative Filtering**

Collaborative filtering was chosen over purely content-based methods because it leverages collective user behavior patterns, enabling the system to recommend even items with sparse metadata as long as interaction data is available.

# Architecture (NeuMF):

## General Matrix Factorization (GMF):

To start off, a breakthrough came with matrix factorization (MF), where it represents each user and item by a low-dimensional latent vector, and predicts a user’s preferences as the dot product of their vectors. In other words, it projects users and items into a shared “latent space” so that their interactions is captured by the angle between their vectors. For example, in a movie recommender, dimensions of the vector might implicitly capture factors like “action vs. romance” or “mainstream vs. niche”. The score for a user–item pair is then computed as a simple inner product of these vectors. Matrix factorization proved powerful and is widely used, but it is fundamentally *linear* (only dot products), which may miss complex patterns in the data.

**Generalized Matrix Factorization (GMF)** builds on the idea of MF by casting it in a neural-network framework. In GMF, we still learn a latent embedding for each user and item, but instead of a plain dot product we take the **element-wise product** of the two vectors and feed it to an output neuron. If pu is the user vector and qi is the item vector, GMF computes:

This interaction vector is passed to a trainable output layer to produce a predicted score. If the output weights are fixed to one and no activation function is applied (identity), GMF reduces to traditional MF. However, by using trainable weights and a customizable activation function, GMF gains additional flexibility. In our case, we used a **linear activation function** to predict real-valued ratings in the 1–5 range. This makes GMF better suited for explicit feedback datasets while still maintaining a simple one-layer neural architecture that captures user-item interactions effectively.

### Final GMF Model:

A screenshot of a computer

AI-generated content may be incorrect.

## Multilayer Perceptron (MLP):

MLP Models for recommendation take a more expressive approach. Here we still learn separate embeddings for users and items, but instead of multiplying them, we **concatenate** them into one long vector and feed it into a feed forward neural network with several hidden layers. This allows the model to learn arbitrary nonlinear interactions between every component of the user and item vectors. In practice, an MLP branch might stack a few dense layers (using ReLU activations) to transform the concatenated embedding into a final score. This neural network can, in principle, capture very complex patterns (e.g. combinations of latent factors) that a simple dot product cannot. However, it also introduces many more parameters and can be prone to overfitting if the data is sparse or the model is too large.

### Final MLP Model:

A screenshot of a computer

AI-generated content may be incorrect.

## Neural Matrix Factorization (NeuMF):

NeuMF combines the strengths of GMF and MLP. The idea is to run *two* parallel subnetworks for each user–item pair:

**GMF branch:** Look up a user embedding **p** and an item embedding **q** and compute their element-wise product **x** = **p** ⊙ **q**, and use a linear layer to score this vector.

**MLP branch:** Look up a separate user embedding **u** and item embedding **v**; concatenate them [u; v] and feed through several dense layers (with activations like ReLU) to learn complex interactions.

**Combine:** The outputs of the GMF and MLP branches (just before their final output layers) are **concatenated** into one vector, which is then passed through a final output layer (linear activation function) with the addition of a bias vector.

**Output:** The final prediction should look something like this:

*ŷui* = **h***T* [ **pu** ⊙**q***i*; *ϕ*( [**u***u*; **v***i*] )])

where **ϕ** denotes the MLP Dense Layer output, and **h** is the weight vector of the final output layer.

## Steps of the Final NeuMF model

1. **Embedding lookup:** For a given user–item pair (u, i), fetch a user vector **p**u and item vector **q**i for the GMF branch, and *separate* user vector **u**u and item vector **v**i for the MLP branch. (These are all learned parameters.)
2. **GMF Interaction Vector:** Compute element-wise product **x** = **p**u ⊙ **q**i. This vector **x** is then fed through a linear output layer to give a GMF score component
3. **MLP Interaction Vector:** Form the vector **z** = [**u**u; **v**i] by concatenation. Pass **z** through several hidden layers (each with ReLU activation). The final hidden layer produces a vector ***ϕ*** that encodes complex user–item features.
4. **Fusion:** Concatenate the output of the GMF branch (often just **x** itself) with the final MLP hidden vector ***ϕ***. Feed this combined vector through a final dense layer to compute the predicted preference . This final layer has a weight vector **h** that linearly combines the two branches’ information.

### Final Model:

A screenshot of a computer program

AI-generated content may be incorrect.

## Strengths of NeuMF

By combining GMF and MLP, NeuMF can model both linear and complex non-linear interactions between users and items . The GMF part retains the efficiency and generalization of classic MF, while the MLP part adds expressiveness to capture intricate patterns. In practice, this ensemble often yields stronger recommendation accuracy than MF or deep nets alone

## Weaknesses of NeuMF

The added flexibility comes at a cost. NeuMF has many more parameters (two embedding layers plus a deep net), so it requires more data and careful tuning to avoid overfitting. In environments with very sparse data, NeuMF can overfit more easily than simpler models. Like other collaborative methods, it still faces the cold-start problem (new users/itemswithout history) and relies solely on interaction data . Finally, as a neural model, NeuMF is more complex and less interpretable than classic MF or neighborhood models

# Results

The Recommender system deliverd decent results across multiple dimensions of performance.

The hybrid neural collaborative filtering model goes beyond surface-level features by learning latent representations of both users and movies. This allows the system to capture complex and subtle user preferences, resulting in more accurate rating predictions. Throughout the training process, the model shows steady convergence, with a continuous decrease in validation errors, however it seemed to slightly overfit.

We evaluated the model using several metrics, including **MSE**, and **MAE**:

|  |  |
| --- | --- |
| Metric | Value |
| MSE | 0.7756 |
| MAE | 0.6716 |

The Trivial case (our baseline) which is guessing the mean for all items results in the following:

|  |  |
| --- | --- |
| Metric | Value |
| MSE | 1.2538 (Same as Variance) |
| MAE | 0.9360 |

Some sample Predictions can be seen below:

A screenshot of a computer code

AI-generated content may be incorrect.

These values while not the greatest are a decent start for any recommender system. If integrated with a content-based recommender system, the resulting hybrid model may yield great results.

# Challenges

Developing a high-performing movie recommender system presents several challenges:

* **Data Sparsity:**  
  User rating data is inherently sparse, with many users rating only a handful of movies. This limits collaborative filtering efficacy and necessitates strong content-based and hybrid methods.
* **Cold Start Problem:**  
  New users or movies with insufficient interaction data challenge model accuracy. Our collaborative based filtering model especially suffers from this problem
* **Overfitting:**

The Model was extremely prone to overfitting. Dropout Layers, Batch Normalization, and Regularization techniques were deployed to try and minimize this issue as much as possible.

# References

<https://www.algolia.com/blog/ai/the-anatomy-of-high-performance-recommender-systems-part-iv>

<https://ar5iv.labs.arxiv.org/html/1708.05031>

<https://pmc.ncbi.nlm.nih.gov/articles/PMC10973760/>

<https://ar5iv.org/pdf/1708.05031>

<http://d2l.ai/chapter_recommender-systems/neumf.html>

F. Maxwell Harper and Joseph A. Konstan. 2015. The MovieLens Datasets: History

and Context. ACM Transactions on Interactive Intelligent Systems (TiiS) 5, 4,

Article 19 (December 2015), 19 pages. DOI=http://dx.doi.org/10.1145/2827872