Aditya suman
Registration No : 229310225
Long Term Internship

PROJECT

Movie Recommendation System using Transformer-Based Deep Learning Model

Introduction

A movie recommendation system using the MovieLens 1M dataset. The goal of the project is to predict how much a user would like a movie based on their past watching history and ratings. Recommendation systems like this are widely used by platforms such as Netflix, Amazon Prime, and YouTube to personalize user experiences. Our model learns the patterns in user preferences and movie features to suggest movies that users are likely to enjoy.

Dataset Used

The dataset used in this project is called MovieLens 1M, which is a popular dataset for testing recommendation systems. It has about one million ratings given by over six thousand users to more than three thousand movies. The dataset has three main files.

The users file contains information about each user such as their age group, gender, and occupation. The movies file includes the movie title and its genres, such as Action, Comedy, or Drama. The ratings file contains the ratings that users gave to different movies along with the time when they rated them. This data helps the model learn what type of movies each user prefers

Data Preprocessing

Before using the data, it needed to be cleaned and organized properly. User IDs and movie IDs were changed into simple text formats like “user_1” or “movie_1” so that they could be easily processed. The movie genres were separated into individual categories such as Action, Comedy, Drama, and others, and each movie was marked with 1 or 0 depending on whether it belonged to that genre. The ratings were also converted into numerical values that the model could understand.

The data was then grouped by user so that all movies rated by the same user were together. This helped create a sequence of movies watched by each user, which is useful for understanding their watching behavior over time.

Sequence Creation

To help the model learn better, sequences of movies were created for every user. Instead of looking at single ratings, the system looks at a short sequence of movies a user has watched and their ratings. For example, if a user watched five movies, the first four are used as input and the fifth one is treated as the movie to predict. The sequence length was set to 4, and the step size was 2, which means the model moves forward two steps after each sequence.

After this, the dataset was split into training data = 85% and testing data=15%. The training set is used to teach the model, while the test set is used to check how well the model performs on unseen data.

Feature Encoding

All the information about users and movies was converted into numbers so the computer could process it. This is called feature encoding. Each user and movie got its own numerical ID. Special layers called embeddings were used to convert these IDs into dense vectors that capture hidden patterns.

For example, two movies that belong to similar genres might get similar embedding values. In the same way, users with similar preferences may also have embeddings that are close to each other. The genres of movies were also included as extra information so that the model could understand what kind of movies were being rated.

Model Architecture

The main model used in this project is a Transformer-based deep learning model built using Keras with JAX backend. Transformers are advanced models that can handle sequence data very well. They are commonly used in text processing tasks, but here they are used to understand sequences of movies.

The model takes a sequence of movies and ratings as input and tries to predict what rating a user might give to the next movie. It uses a Multi-Head Attention layer which helps the model focus on the most important movies in a user's watching history. Positional embeddings are also used so that the model can understand the order in which movies were watched.

After this, the data goes through several Dense (fully connected) layers that combine all the learned features. The output layer gives a single value which is the predicted rating for the next movie. The model was trained using the Adagrad optimizer with a learning rate of 0.01. The Mean Squared Error (MSE) was used as the loss function, and Mean Absolute Error (MAE) was used as the performance metric.

Model Training

The model was trained for two epochs, meaning it went through the entire training dataset twice to learn from it. During training, it adjusted its internal parameters to minimize the difference between the predicted and actual ratings. After training, the model was tested on the test dataset, which it had not seen before. The test results showed how well the model could predict ratings for new movies. The final performance was measured using MAE, where a smaller value means more accurate predictions.

Results and Discussion

The results showed that the Transformer-based model was able to learn user preferences and predict ratings fairly well. Even though the model was trained for only a small number of epochs, it captured important relationships between users and movies. The use of the Transformer architecture helped it understand the order and relationships between the movies watched by a user.
With more training time, tuning of parameters, and by including more user and movie features, the performance of the model could be improved further. This experiment proved that deep learning models can provide better recommendations compared to traditional techniques like simple collaborative filtering.

Future Work

There are several ways to improve this project in the future. The model can be trained for more epochs to increase accuracy. More user information, such as location or time of viewing, could also be included to make predictions more

personal. Another improvement would be to combine this method with other approaches like collaborative filtering to make a **hybrid recommendation system**. Finally, the model can be deployed as a web or mobile application where users can get personalized movie recommendations directly.