Movie Recommendation System: A Hybrid Approach

Bandaru Bhavyanath
B180399EC

Chevuru Sai Jaswanth
B180668EC

Eruguri Jeshwanth
B180556EC

Jarapala Mohith Pamar
B1801027EC

Under the co-guidance of
Dr. Vinay Joseph
Assistant Professor,
ECE Department, NITC.

Under the co-guidance of
Dr. Nujoom Sageer Karat
Faculty Member,
ECE Department, NITC.

***Abstract*—Natural language processing is crucial in today's progressing world because it helps overcome linguistic ambiguity and gives helpful quantitative structure to data for several downstream applications, like speech recognition and text analytics. Natural language processing has made great progress recently using the BERT framework. Recommendation System is another significant area that is very popular and useful for people to make better and faster-automated decisions. Surveys show we spend more than 100 days of our lives choosing what to watch. A Movie Recommendation System can suggest a set of movies to users based on their preferences or the popularities of the movies. To improve the quality of a movie recommendation system, we are developing a hybrid approach by combining collaborative filtering using Matrix Factorization [1] and content-based filtering using BERT4Rec [2] which is an extension of the BERT framework for recommendation systems. Several studies indicate that the hybrid approach can provide more accurate recommendations. We plan to utilize the MovieLens dataset, which is a representative and popular real-world benchmark in recommendation field. We will evaluate the performance of our novel movie recommendation system.**

***Index Terms*—Recommendation System, Collaborative filtering, Content based filtering, BERT4Rec.**

I. Introduction

Natural Language Processing (NLP) is an area of artificial intelligence that deals with natural language interaction between computers and humans. NLP isn't a new subject, but it's progressing quickly thanks to a growing interest in human-machine communication, as well as the availability of massive data, powerful computation, and improved algorithms. The development of BERT changed the NLP paradigm significantly. BERT4Rec [2] is the first to utilize the BERT framework for sequential recommendation tasks, achieving state-of-the-art performances. Hybrid recommender systems aim to model users' profiles for personalized recommendations. They also help users find the movies of their choice based on the movie experience of other users efficiently and effectively without wasting much time browsing.

II. Problem Statement

Our problem is to design and assess a novel movie recommendation system that utilizes a hybrid approach to build the recommender system and applies the BERT framework, which is typically used in NLP settings, to recommendation systems. The hybrid approach enables us to investigate the performance of methods like BERT4Rec combined with collaborative filtering and avoid disadvantages of pure approaches such as the lack of information about the domain dependencies in collaborative filtering and people's preferences in content-based systems.

III. Our approach

To design our Hybrid Recommendation system, firstly, we process the relevant information present in the dataset by sending it through a model based on Matrix Factorization [1] for collaborative filtering. The immediate output is a list of movies recommended to a user based on historical interactions of the similar users. This output is then fed to the sequential model based on BERT4Rec [2] to achieve content-based filtering. The resulting output is a list of movies predicted by our model recommended to the user. Refer Figure 1 for the abstract architecture of our model.
To assess our movie recommendation system, the accuracy of the predictions made by our model is evaluated using the rec-metrics library.

A. Tools

1) Packages:: We plan to use NumPy, Pandas, PyTorch to develop the model in Jupyter Notebook and Google Colab.

2) Dataset:: MovieLens is a popular benchmark dataset for evaluating recommendation algorithms. This dataset contains user's ratings of movies, as well as movie genre tags. We plan to use ML-1m variant.

3) Metrics:: Evaluation of our model using the rec-metrics library, which includes RMSE, Mean Average Precision, etc.

IV. Collaborative Filtering

Collaborative filtering (CF) recommender systems create predictions based on user-item relationships, with no further

knowledge about the users or objects required. Neighborhood-based CF and latent factor models are two typical ways for constructing CF recommender systems.
The user-item rating matrix is used in latent factor models for movie recommendations to try to profile both users and items using a number of latent variables. A feature in the context of movies can be a genre, target age range, or even something utterly unintelligible. Matrix factorization is an excellent latent factor approach for recommender systems.

A. *Matrix Factorization*

Matrix Factorization (MF) [1] is a method for calculating a system's latent factor model based on user-item interaction. Users on one axis and items on the other, these user-item interactions can be represented as a matrix. Even on relatively sparse matrices, Matrix Factorization has been shown to provide very strong predictions.
By factoring the rating matrix r into a product of two latent component matrices, u for users and m for movies, the matrix factorization approach decreases the dimensions of the rating matrix r. Refer equation in the Figure 2 for depiction of latent component matrices.

$$r = um^T \tag{1}$$

f is the number of features extracted, n the number of users and i number of movies
Each row u_n is a vector of features for a user n and each row m_i is a vector of features for an item i. The product of these vectors creates an estimate of the original rating.

$$r_{ni} = u_n m_i^T \tag{2}$$

B. *Matrix Factorization with bias*

To improve the predictions use a bias for movie i, called b_i, bias for user u, called b_u, and global rating average μ to model the rating r_{ui}.

$$r_{ni} = b_i + b_n + u_n m_i^T \tag{3}$$

In addition to learning the bias independently, This can be solved using a stochastic gradient descent approach.

Algorithm:: The algorithm using stochastic gradient descent was implemented in the following steps:

1) Initialize matrices u and m of size users $\times K$ and Movies $\times K$ with random values from a uniform distribution over [0,0.05] where K stands for the number of features that will be extracted.
2) Iterate over all the observed ratings in training dataset.
   a) Calculate a predicted rating.
   b) Calculate error for the predicted rating to that of actual rating using Mean Square Error.
   c) Update u and m according to error with the help of Adam Optimizer.
3) Recommend the top movies by calculating model learned ratings by taking dot product of user and item embeddings.

V. Content Based Filtering

Content-based filtering leverages item similarity to suggest items that are related to what a user enjoys. Content-based filtering generates suggestions by matching keywords and attributes assigned to database items with user profiles.

A. *Transformer*

Transformer is a novel architecture that adopts the self-attention mechanism. RNNs, LSTMs and GRUs still need more-prominent attention mechanism to address the long-range dependencies and also sequential computation prevents parallelization. Transformers solved these issues and also paved way to modern state of recent pre-trained models like BERT. The other difference between RNN models and transformers is that RNNs are feedback neural networks which require temporal dependency on weights and the transformers exhibit feed-forward neural network behavior. Refer Figure 3 for the transformer architecture.

B. *BERT*

Bi-directional Encoder Representations from Transformers (BERT) was first introduced by researchers at Google AI team. BERT is stack of certain number of transformer encoder layers, mainly pre-trained on masked language modelling(MLM) and next sentence prediction(NSP). Fine-tuning BERT can solve tasks like Neural Machine Translation, Question Answering, Sentiment Analysis, Text Summarization, etc.

C. *BERT4Rec*

BERT4Rec is a sequential recommendation model that uses deep bidirectional self-attention to model user behavior sequences. Modeling consumer's changeable preferences based on their past behaviors is difficult yet necessary for recommendation systems. For creating suggestions, previous systems used sequential NN's to encode users historical interactions from left to right into hidden representations. Such unidirectional models are sub-optimal due to flaws such as:

(a) Unidirectional architectures limit the power of hidden representation in user's behavior sequences.
(b) They often assume a rigidly ordered sequence, which is not always feasible.

Refer Figure 4 for BERT4Rec architecture.

VI. Evaluation

A. *Matrix Factorization*

1) *Training:*

- Split the Movie-Lens 1m dataset into training, validation and test sets in the ratio 70:15:15 with help of scikit-learn library.
- Initialize the user matrix and item matrix uniformly in between [0,0.05] with tensors, using PyTorch module.
- Calculate the predicted ratings by multiplying user matrix with the transpose of the item matrix.
- Mean square error is calculated by the help of original ratings and predicted ratings and use Adam-optimizer to update the weights with the help of PyTorch module.

2) Loss function: The loss function(L) adopted in the model is the Mean Square Error:

$$L = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y_i})^2 \tag{4}$$

3) Validation: The Mean Squared Error was the major way for evaluating the prediction system's accuracy. The most recent Movie-Lens dataset was used, which included 1,000,209 anonymous ratings of approximately 3,900 movies made by 6,040 Movie-Lens users. The data for the ratings was split as 70:15:15 between training, validation and testing. The training and validation losses vs epochs is plotted, refer Figure 5.

4) Test: Along with calculating the training and validation losses, we have set aside 15% of our dataset to test our model. We have used MSE to calculate the cost of the test data based on our model predictions.

5) Cosine Similarity: For testing the recommendations predicted by our model for a particular user, we have used cosine similarity. We have taken the most recently highest rated movie of that particular user from the training dataset and found the cosine similarity score between that movie and his corresponding predictions.

B. BERT4Rec

1) Training: We train the BERT4Rec model using Masked Language modelling (MLM), we send the user historical movies and mask them with a certain probability, and the model predicts the masked movies based solely on its bidirectional context and then calculate the loss by comparing with the hidden representations.

2) Loss function: We define the loss for each masked input as the negative log-likelihood of the masked targets.

$$\mathcal{L} = \frac{1}{|\mathcal{S}_u^m|}\sum_{v_m \in \mathcal{S}_u^m} -\log P\left(v_m = v_m^* \mid \mathcal{S}_u'\right) \tag{5}$$

3) Testing: Append the mask token to the end of user's behavior sequence, and then predict the next item based on the final hidden representation of this token. During training, we additionally create samples that solely mask the last item in the input sequences. It works in the same way as fine-tuning does for sequential recommendations, and it can help to improve recommendation efficiency.

VII. Results

On the ML-1m dataset, we used our two versions of matrix factorization, one with and one without item/user bias. Two smaller feature matrices, one for (users $\times$ latent factors) and one for (latent factors $\times$ movies), were found as a result of the matrix factorization using gradient descent. The matrix multiplication of the two feature matrices will produce ratings approximate to the original rating matrix.
The Mean Square Error, refer Figure 6, after testing the model is 0.756. Calculating the cosine similarity score between user's most recently highest rated movie and his recommendations shows that the predictions given by the model are relevant to the user. Refer Figure 7 for list of movies sorted by ratings and timestamps. Refer Figure 8 cosine similarity scores of recommendations.
As a result, we were able to get recommendations of the most relevant movies to a user based on similar movie interests of other users.

VIII. Conclusion

- Achieved results for collaborative filtering part in the proposed hybrid recommendation system.
- The Masked Language Modeling is used by BERT4Rec to anticipate the masked item in each sequence.
- The embedding layer, transformer encoder, and projection layer are the three main components of BERT4Rec.
- The MLM also helps us to get product representations utilizing the bidirectional context.

IX. Future Work

(i) Implementation and evaluation of BERT4Rec.
(ii) Design of hybrid combination.
(iii) Evaluating the performance of our hybrid model.

References

[1] Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," *Computer*, vol. 42, no. 8, pp. 30–37, 2009.
[2] F. Sun, J. Liu, J. Wu, C. Pei, X. Lin, W. Ou, and P. Jiang, "Bert4rec: Sequential recommendation with bidirectional encoder representations from transformer," in *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, ser. CIKM '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 1441–1450. [Online]. Available: https://doi.org/10.1145/3357384.3357895

Appendix

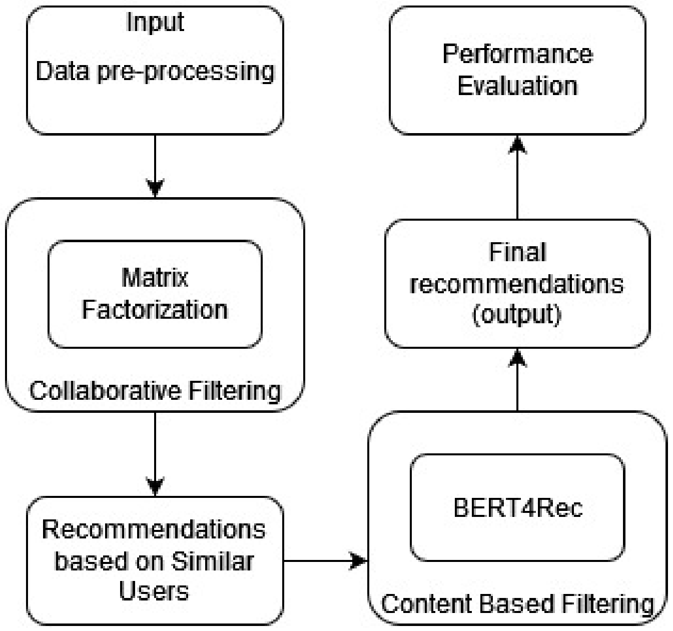


Fig. 1. Abstract Architecture of the model

$$\begin{bmatrix} r_{11} & . & . & . & r_{1i} \\ . & . & & & . \\ . & & . & & . \\ . & & & . & . \\ r_{n1} & . & . & . & r_{ni} \end{bmatrix} = \begin{bmatrix} u_1 \\ u_2 \\ . \\ . \\ . \\ u_n \end{bmatrix} \times \begin{bmatrix} m_1 & m_2 & . & . & . & m_i \end{bmatrix}$$

$$r = um^T$$

Fig. 2. Latent component matrices

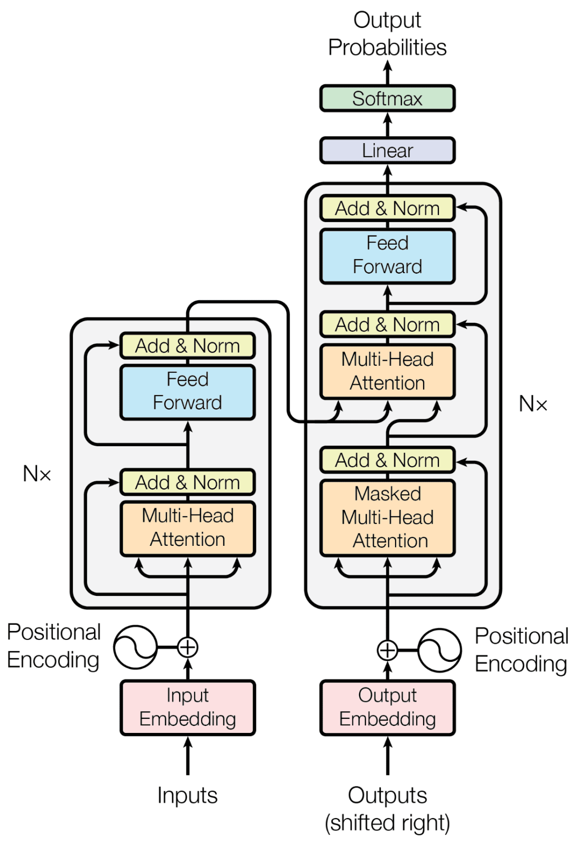


Fig. 3. Transformer Architecture (Source)

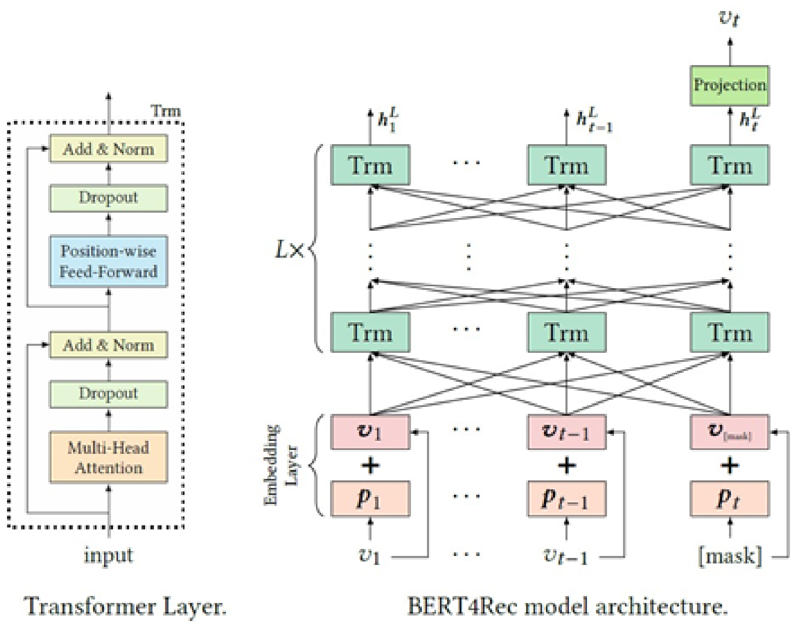


Fig. 4. BERT4Rec Architecture [2]

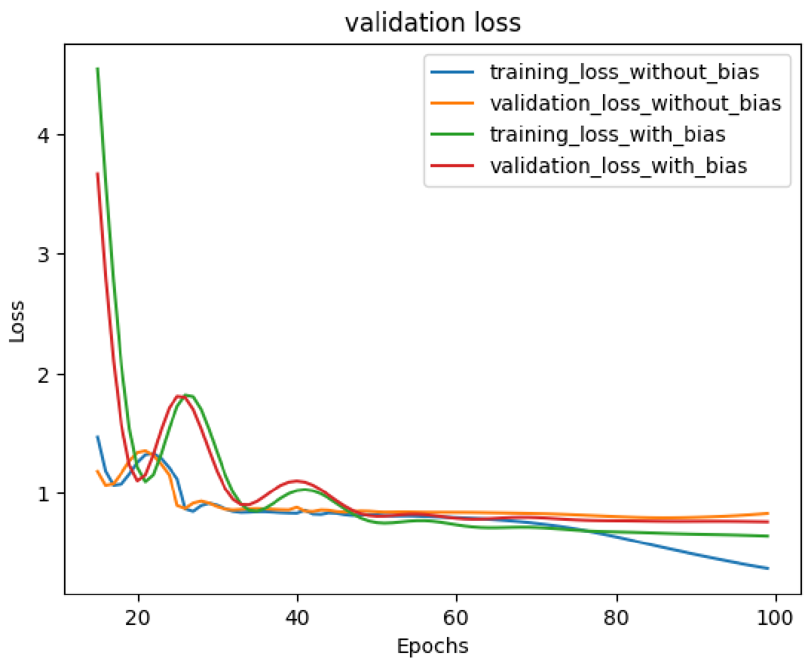
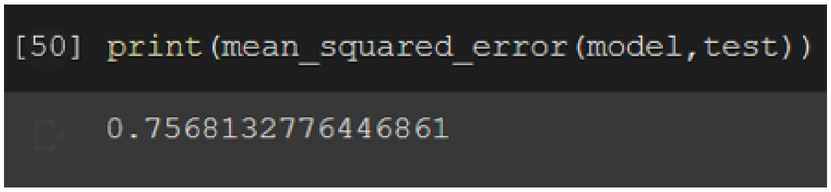


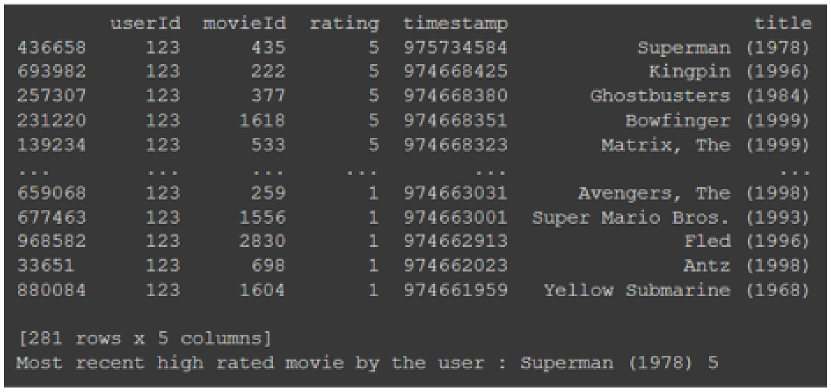
Fig. 5. Training and Validation Losses vs Epochs



```
[50] print(mean_squared_error(model,test))

     0.7568132776446861
```

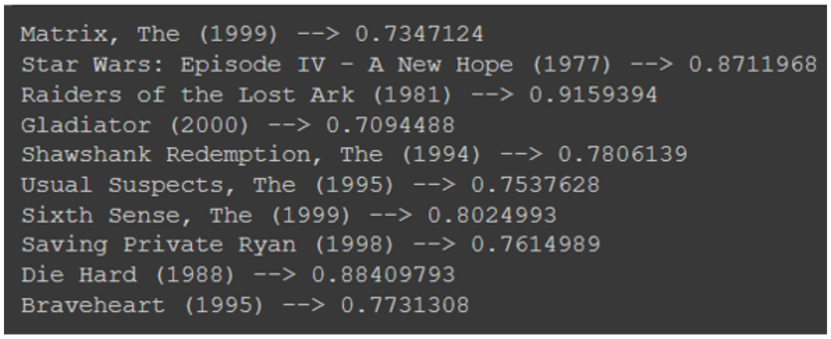
Fig. 6. Test dataset evaluation



```
        userId  movieId  rating  timestamp                        title
436658     123      435       5  975734584              Superman (1978)
693982     123      222       5  974668425               Kingpin (1996)
257307     123      377       5  974668380          Ghostbusters (1984)
231220     123     1618       5  974668351             Bowfinger (1999)
139234     123      533       5  974668323           Matrix, The (1999)
...        ...      ...     ...        ...                          ...
659068     123      259       1  974663031         Avengers, The (1998)
677463     123     1556       1  974663001     Super Mario Bros. (1993)
968582     123     2830       1  974662913                 Fled (1996)
33651      123      698       1  974662023                 Antz (1998)
880084     123     1604       1  974661959      Yellow Submarine (1968)

[281 rows x 5 columns]
Most recent high rated movie by the user : Superman (1978) 5
```

Fig. 7. Most recent highest rated movie by the user



```
Matrix, The (1999) --> 0.7347124
Star Wars: Episode IV - A New Hope (1977) --> 0.8711968
Raiders of the Lost Ark (1981) --> 0.9159394
Gladiator (2000) --> 0.7094488
Shawshank Redemption, The (1994) --> 0.7806139
Usual Suspects, The (1995) --> 0.7537628
Sixth Sense, The (1999) --> 0.8024993
Saving Private Ryan (1998) --> 0.7614989
Die Hard (1988) --> 0.88409793
Braveheart (1995) --> 0.7731308
```

Fig. 8. Cosine similarity for the recommendations