Enhance Recommender Systems with Hybrid BERT and Matrix Factorization Embeddings

Thanh Nam Tran [1]

Abstract

Recommender systems (RC) have played a critical role in enhancing user experiences across digital platforms, from social media to e-commerce and streaming services. In recent years, transformer-based models like BERT have proved to perform well in sequence-aware recommendation tasks. However, these models often suffer from the cold-start problem due to a lack of generalized prior knowledge. In this project, we investigate a hybrid approach that integrates Matrix Factorization (MF) into a BERT-based transformer model. Specifically, we first capture collaborative filtering signals using MF and then feed the resulting item embeddings into the BERT model as initialization. We compare two models — a BERT-only baseline and the proposed hybrid model — to evaluate whether incorporating MF improves recommendation performance.

1. Description and motivation

1.1. Introduction

Nowadays, to maximize a user's time using an online service is to maximize revenue from online customers. Any browsing customer is a potential buyer in e-commerce websites. Continuously personalized content keeps users engage in social media platforms such as TikTok. In 2024, TikTok generated an estimated 23 billion revenue in 2024, a 42.8% in 2024 (Business of Apps, 2024). This shows how the importance of having a good recommender system can directly affect revenue.

1.2. Motivation

In the paper BERT4Rec: Sequential Recommendation with Bidirectional Encoder Representations from Transformers (Sun et al., 2019), we are introduced to a new Recommendation System model shown to outperform several existing baselines. Despite its strong performance, BERT4Rec has limitations. According to the authors, it tends to underperform when trained on small or sparse datasets. In this project, we aim to address these limitations by cooperating a BERT4Rec-like model with the Matrix Factorization (MF) technique. By leveraging MF to provide pre-learned item embeddings, we hope to improve training efficiency and enhance performance in data-scarce scenarios.

1.3. Dataset

In this project, we will be using the a version of the "MovieLens" dataset (Harper & Konstan, 2015). This is a popular benchmark dataset for evaluating recommendation algorithms. It has been cited in more than 6000 research papers. We select the small MovieLens consists of 100,000 ratings and 3,600 tag applications applied to 9,000 movies by 600 users and was Last updated 9/2018.

We intend to use the smaller subset to replicate the cold start issue. We will evaluate the hybrid model versus BERT4Rec to see if the hybrid approach helps cold-start and sparsity problems typically encountered in real-world recommendation settings.

2. Related Work

2.1. Deep Learning based Recommender System

Our idea of combining the Matrix Factorization technique with BERT4Rec come from The survey Deep Learning based Recommender System: A Survey and New Perspectives (Zhang et al., 2019). The paper highlights the growing trend of combining deep learning with traditional collaborative filtering methods.

2.2. BERT4Rec

Our work is heavily inspired by these 3 reference. First is the original BERT4Rec: Sequential Recommendation with Bidirectional Encoder Representations from Transformers (Sun et al., 2019). Their public code repository of the model can be found on their Github (Sun).

In summary, BERT4Rec is inspired by Masked Language Modeling (MLM) from the original BERT in NLP (Sun

[*]Equal contribution [1]University of Maryland, Baltimore County. Correspondence to: Thanh Nam Tran <ttran19@umbc.edu>.

et al., 2019). Instead of predicting a missing word in a sentence, BERT4Rec predicts a masked item in a user's sequence of interactions.

Let's say a user watched the following movies (in order):

`[Spider-Man, The Matrix, Titanic, Toy Story, The Godfather]`

BERT4Rec might randomly mask one or more movies in this sequence:

`[Spider-Man, [MASK], Titanic, [MASK], The Godfather]`

Then it learns to predict:

- **The Matrix** from the context `[Spider-Man, Titanic, The Godfather]`
- **Toy Story** from the same context

Secondly, we learned how to conduct our setup from the paper *"A Systematic Review and Replicability Study of BERT4Rec for Sequential Recommendation"*, Petrov and Macdonald (Petrov & Macdonald, 2022c). They highlight inconsistencies in prior experimental practices which includes evaluation metrics such as Recall@10 and NDCG@10.

Finally, in a complementary study, *"Effective and Efficient Training for Sequential Recommendation using Recency Sampling"*, the same authors (Petrov & Macdonald, 2022b) propose a training strategy that prioritizes recent interactions during sampling, arguing that such a method improves both efficiency and performance in sequential recommender systems. Their open-source repository can be found at `bert4rec_repro` (Petrov & Macdonald, 2022a).

The evaluation data is shown below

Dataset	Uniformly Sampled 100 Negatives	Popularity Sampled 100 Negatives	Unsampled
Movielens-1M	0.8039	0.6975	0.2821
Movielens-20M	0.9453	0.7409	0.2886

Table 1. Recall@10 (HIT@10) for BERT4Rec

Dataset	Uniformly Sampled 100 Negatives	Popularity Sampled 100 Negatives	Unsampled
Movielens-1M	0.6008	0.4751	0.1516
Movielens-20M	0.7827	0.5259	0.1732

Table 2. NDCG@10 for BERT4Rec

2.3. Matrix factorization

About the Matrix factorization (MF) technique, it is a widely used collaborative filtering technique that effectively models global user-item interactions using low-rank latent representations. In An Introduction to Matrix Factorization and Factorization Machines in Recommendation Systems and Beyond by Yuefeng Zhang (Zhang, 2022) provides a comprehensive foundation for understanding the strengths and scalability of MF in large-scale recommendation systems.

An example of how MF work:

Table 3. A small MovieLens rating matrix (— denotes missing values).

User / Movie	Toy Story (ID=1)	Jumanji (ID=2)	Grumpier Old Men (ID=3)
User 1	4	5	2
User 2	5	3	—
User 3	—	2	4

Table 4. Learned user embeddings $U \in \mathbb{R}^{3\times 2}$.

User	$\mathbf{u}_{:,1}$	$\mathbf{u}_{:,2}$
User 1	0.68	0.12
User 2	0.75	0.20
User 3	0.30	0.55

Table 5. Learned item embeddings $V \in \mathbb{R}^{3\times 2}$.

Movie	$\mathbf{v}_{:,1}$	$\mathbf{v}_{:,2}$
Toy Story	0.80	0.10
Jumanji	0.70	−0.20
Grumpier Old Men	0.10	0.90

The original ratings $\hat{R}$ are approximated by the dot-product:

$$\hat{r}_{ui} \approx \langle \mathbf{u}_u, \mathbf{v}_i \rangle,$$

Our theory is that by pertaining and temporarily freeze item embeddings with matrix factorization, MiniBERT4Rec doesn't have to learn everything from scratch. It would jumps straight into sequence learning with MF results, which speeds up training and gives a boost to items that barely have any ratings.

3. Implementation

3.1. Matrix Factorization Collaborative Filtering

Matrix Factorization (MF) models user–item interactions by learning two embedding tables in a shared latent space (Zhang, 2022; Koren et al., 2009). In code, we define

$$\mathbf{U} \in \mathbb{R}^{(m+1)\times k}, \quad \mathbf{V} \in \mathbb{R}^{(n+1)\times k}$$

via two PyTorch `nn.Embedding` layers of size $(m+1)\times k$ and $(n+1)\times k$, where index 0 is reserved for padding, and k (the embedding dimension) defaults to 32.

For a batch of user–item pairs $\{(u_j, i_j)\}$, the model's forward pass simply computes

$$\hat{r}_{u_j i_j} = \langle \mathbf{u}_{u_j}, \mathbf{v}_{i_j} \rangle = \sum_{d=1}^{k} U_{u_j,d} \, V_{i_j,d}. \quad (1)$$

We train by minimizing the mean squared error between observed ratings $r_{u_j i_j}$ and predictions $\hat{r}_{u_j i_j}$. Concretely, using PyTorch's `nn.MSELoss` and the Adam optimizer, we minimize

$$\mathcal{L} = \frac{1}{|\mathcal{K}|} \sum_{(u,i)\in\mathcal{K}} \left(r_{ui} - \hat{r}_{ui}\right)^2, \quad (2)$$

where $\mathcal{K}$ is the set of all observed ratings in the training split.

After training, the learned embeddings $\mathbf{V}$ can be extracted and fed into the MiniBERT4Rec model.

3.2. BERT-Like Sequential Recommendation Model

Our base model MiniBERT4Rec is a simplified transformer for sequential recommendation.

Input & Embedding Initialization Instead of using a complicated set of input including special [CLS] token, sequence of the original items list and mask token, positional embedding, our model simply use item vectors only. In the hybrid variants, the vectors has added weight as result from MF model.

Masking & Prediction

For mask technique, during training we hide one random position per sequence. But, when evaluating and recording the loss, we only look at the last position's output (that final time step) to predict the "next" item. Whereas, the original BERT4Rec mask more than one and randomly during both process.

Positional Encoding

Original BERT4Rec adds sinusoidal (or learned) positional embeddings so the model knows each item's absolute position in the sequence. Our MiniBERT4Rec don't use any positional encoding.

Transformer Encoder Configuration

BERT4Rec typically uses Transformer encoder of 2-4 layers of GELU activations and feed-forward sublayers. Our MiniBERT4Rec uses PyTorch's default `TransformerEncoderLayer` (PyTorch Core Team, 2024c) with ReLU activation.

Output Head & Weight

In the original, the output projection is weight-tied to the input embedding matrix. Our MiniBERT4Rec uses a separate linear head (`nn.Linear`) (PyTorch Core Team, 2024b) with its own weights.

Optimization & Regularization

Both the original BERT4Rec and our MiniBERT4Rec training often uses AdamW optimizer(PyTorch Core Team, 2024a).

As the result, while being inspired from the original BERT4Rec model, we have simplified our MiniBERT4Rec architecture significantly to reduce complexity and improve training efficiency due to our computational resources.

4. Research Questions

This project investigates the effectiveness of combining collaborative filtering and sequence-based modeling for movie recommendation. The following research questions guide our work:

1. **RQ1:** Does initializing a BERT-based sequential recommendation model with matrix factorization embeddings improve top-k recommendation performance (Hit@k and NDCG@k) in data-scarce settings?

2. **RQ2:** Can hybridizing collaborative filtering signals with transformer-based sequence models mitigate the cold-start and sparsity issues commonly encountered in deep recommender systems?

3. **RQ3:** How does the training efficiency (in terms of convergence speed and loss behavior) of a hybrid MF+BERT model compare to a BERT-only model trained from scratch?

5. Training and Evaluation Results

With the implementation, we start our model training and evaluation. Both MiniBERT4Rec models is run with the configuration of 20 epochs. In the hybrid approach, the MF is run with 30 epochs.

5.1. Only MiniBERT4Rec Model

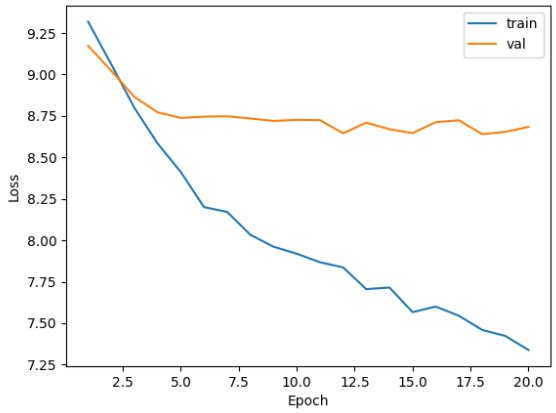

Figure 1. Architecture of our BERT-based sequential recommendation model (MiniBERT4Rec).

From the figure, we can see that the train loss drops steadily from 9.3 to 7.3, but the loss value plateaus around 8.6 after epoch 5. That gap suggests the model is starting to overfit the small MovieLens split.

For evaluation metrics, we record Hit@10 is 1.6% and NDCG@10 is 0.8%. The metric values are quite low, but they are expected.

5.2. Hybrid Model

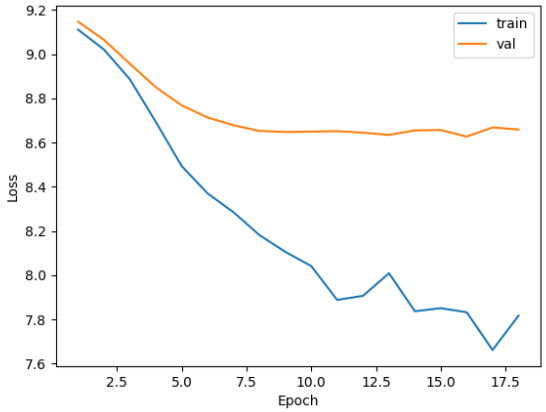

Figure 2. Overview of the hybrid model combining BERT and matrix factorization embeddings.

Compare to the previous figure, we can see that the Hybrid model has faster convergence and slightly lower loss. In addition, the Hybrid loss value plateaus a bit lower—around 8.64 by epoch 7—and stays under the pure-BERT curve throughout.

For the evaluation metrics, the Hit@10 jumps from 1.6% to 2.8% which is a 70% improvement. NDCG@10 is now doubled from 0.8% to 1.36%.

5.3. Recommendation Function

We then use our model to recommend movie for the User ID 1. User 1 saw 231 movies. Our MiniBERT4Rec models masked the last movie Crumb (1994) [Documentary]. Here are the 5 movies that our models think the user will want to watch next.

Model	Movie	Genre(s)	Score
BERT-only	The Shawshank Redemption (1994)	Crime, Drama	0.0058
	The Firm (1993)	Drama, Thriller	0.0053
	The Prestige (2006)	Drama, Mystery, Sci-Fi, Thriller	0.0035
	Beauty and the Beast (1991)	Animation, Children, Fantasy, Musical, Romance, IMAX	0.0033
	Maverick (1994)	Adventure, Comedy, Western	0.0031
Hybrid	Schindler's List (1993)	Drama, War	0.0041
	The Silence of the Lambs (1991)	Crime, Horror, Thriller	0.0040
	Raiders of the Lost Ark (1981)	Action, Adventure	0.0038
	Saving Private Ryan (1998)	Action, Drama, War	0.0034
	Forrest Gump (1994)	Comedy, Drama, Romance, War	0.0034

Table 6. Top-5 recommendations for User 1

While one example might not fully support the performance of both model since there are many users in our dataset. We can somewhat see that both BERT-only and the hybrid struggle to recommend "Crumb (1994)". BERT-only model yields more genre diversity while the hybrid model leans on global popularity.

Another observation we can see is that BERT-only picks include a crime drama (Shawshank Redemption), a thriller (The Firm), a mystery/Sci-Fi (Prestige), a children's animation (Beauty and the Beast) and even a western (Maverick). These movies reflect a sequence closer the user's recent history. ON the other hands, the Hybrid model suggests popular titles such as Schindler's List, Silence of the Lambs, Raiders of the Lost Ark, etc., showing the collaborative-filtering aspect that MF technique brought.

6. Conclusion

1. **RQ1:** Does initializing a BERT-based sequential recommendation model with matrix factorization embeddings improve

top-k recommendation performance (Hit@k and NDCG@k) in data-scarce settings?

Answer: On the Movieslen small dataset, adding MF embeddings to MiniBERT4Rec raised Hit@10 from approximately 1.6% to 2.8% and NDCG@10 from approximately 0.8% to 1.36%.

2. **RQ2:** Can hybridizing collaborative filtering signals with transformer-based sequence models mitigate the cold-start and sparsity issues encountered in the original BERT4Rec model?

   Answer: Only partially. We found out that the hybrid's recommendations skew heavily toward popular movies (e.g., *Schindler's List*) but still fail to predict masked item. There is only 2.8% of the time, the recommender system successfully included the correct item in the top 10 recommended items.

3. **RQ3:** How does the training efficiency (in terms of convergence speed and loss behavior) of a hybrid MF+BERT model compare to a BERT-only model trained from scratch?

   Answer: The hybrid model performs clearly better. Its validation loss dips below the pure-BERT curve early (e.g., 8.64 vs. 8.68), and remains lower throughout training. This indicates faster convergence and marginally improved final performance.

While the hybrid model performs better, we can't conclude which model is a better Recommendation System. In real life scenario, one might argue that the MiniBERT4Rec-Only is better since the recommended movies are more similar to the user history movie watched list. Conversely, some might argue that a user might want popular recommended result that MF brought into the hybrid model.

7. Limitations

While our hybrid model demonstrates improved ranking performance over the BERT-only baseline, several limitations remain:

- **BERT4Rec Simplification** During our simplification process, our MiniBERT4Rec might not have the same quality as the original BERT4Rec. Thus, it's arguably affect the quality of this investigation.
- **Only 1 Dataset Used** We only used the MovieLens 100K dataset to support our theory. However, it would be better to perform our model on larger dataset to provide the ground truth comparison.
- **Computational Cost:** As mentioned, we did attempt our study using the 32-M movie dataset which consists of 32 million ratings and two million tag applications applied to 87,585 movies by 200,948 users and was collected in 10/2023 and was released 05/2024 (Harper & Konstan, 2015). However, due to our resources the operations were taking too long.

7.1. Future Work

We plan to experiment with different masking strategies (e.g., span masking) and positional encoding methods to improve sequential learning. In addition, we can also attempt to train our models with larger dataset mentioned in the limitation section. In the long term, we can integrate the user rating information may further enhance recommendation accuracy and interpretability. Finally, we hope to explore options to deploy our models to real-world systems or web-applications.

References

Business of Apps. Tiktok key statistics. https://www.businessofapps.com/data/tik-tok-statistics/, 2024. Accessed: 2025-03-26.

Harper, F. M. and Konstan, J. A. The movielens datasets: History and context. *ACM Transactions on Interactive Intelligent Systems*, 5(4):19:1–19:19, 2015. doi: 10.1145/2827872. URL https://dl.acm.org/doi/10.1145/2827872.

Koren, Y., Bell, R., and Volinsky, C. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, 2009.

Petrov, A. and Macdonald, C. Bert4rec reproducibility repository. https://github.com/asash/bert4rec_repro, 2022a. Accessed: 2025-05-17.

Petrov, A. and Macdonald, C. Effective and efficient training for sequential recommendation using recency sampling. In *Proceedings of the 16th ACM Conference on Recommender Systems (RecSys '22)*, pp. 313–323. ACM, 2022b. doi: 10.1145/3523227.3546785. URL https://doi.org/10.1145/3523227.3546785.

Petrov, A. and Macdonald, C. A systematic review and replicability study of bert4rec for sequential recommendation. In *Proceedings of the 16th ACM Conference on Recommender Systems (RecSys '22)*, pp. 324–334. ACM, 2022c. doi: 10.1145/3523227.3548487. URL https://doi.org/10.1145/3523227.3548487.

PyTorch Core Team. torch.optim.AdamW — pytorch documentation. https://pytorch.org/docs/stable/generated/torch.optim.AdamW.html, 2024a. Accessed: 2025-05-18.

PyTorch Core Team. torch.nn.Linear — pytorch documentation. https://pytorch.org/docs/stable/generated/torch.nn.Linear.html, 2024b. Accessed: 2025-05-18.

PyTorch Core Team. torch.nn.TransformerEncoderLayer — pytorch documentation. https://pytorch.org/docs/stable/generated/torch.nn.TransformerEncoderLayer.html, 2024c. Accessed: 2025-05-18.

Sun, F. Bert4rec: Sequential recommendation with bidirectional encoder representations from transformer. https://github.com/FeiSun/BERT4Rec. Accessed: 2025-05-17.

Sun, F., Liu, J., Wu, J., Pei, C., Lin, X., Ou, W., and Jiang, P. Bert4rec: Sequential recommendation with bidirectional encoder representations from transformer. In *Proceedings of the 28th ACM international conference on information and knowledge management*, pp. 1441–1450. ACM, 2019.

Zhang, S., Yao, L., Sun, A., and Tay, Y. Deep learning based recommender system: A survey and new perspectives. *ACM Computing Surveys*, 52(1):1–38, 2019. doi: 10.1145/3285029.

Zhang, Y. An introduction to matrix factorization and factorization machines in recommendation systems and beyond. *arXiv preprint arXiv:2203.11026*, 2022. URL https://arxiv.org/abs/2203.11026.