PROJECT 99: TRANSFORMER-BASED MOVIE RECOMMENDATIONS

CRESCENT XIONG [ZIHANX3@SEAS], CHUAN LI [CHUANLI1@SEAS], JINZE WANG [WANGJZ@SEAS],

ABSTRACT. Modeling users' dynamic preferences from their historical behaviors is challenging and crucial for recommendation systems. Previous methods employ sequential neural networks to encode users' historical interactions from left to right into hidden representations for making recommendations with known limitations such as information loss over long distances. To address them, BERT4Rec[14] is proposed, which employs the deep bidirectional self-attention to model user behavior sequences. BERT4Rec has proved its effectiveness in many datasets and outperforms various state-of-the-art sequential models consistently. Despite that, several directions remain to be explored. One of them is to incorporate rich item features into BERT4Rec instead of just modeling features. Our team attempts to incorporate movie genre information to BERT4Rec. Besides that, we also introduce user component into the model for explicit user modeling.

1. INTRODUCTION

Precisely characterizing users' interests is essential to Recommender Systems (RS) and has many real-world applications. This also becomes a challenging problem as the users' interests are dynamic and always evolving, but somewhat influenced by their historical behaviors. Thus, scientists often treat this as a sequential problem.

To model such sequential dynamics in user behaviors, various methods have been proposed [7, 11, 13]. They predict the successive item that a user is likely to interact with given the user's past interactions. The shared paradigm among these models is to encode a user's historical interactions into a vector using a left-to-right sequential model and make recommendations based on this hidden representation.

Sequential RS emerged from traditional RS to address the limitations of static user-item interaction analysis. Unlike conventional content-based and collaborative filtering methods, sequential RS consider the temporal order of user actions, recognizing that recent interactions often provide more relevant insights into current preferences. This approach is crucial in dynamic environments like online media or e-commerce, where user interests can quickly evolve. Some of the sequential recommendation architecture is shown in Figure A1.

Despite their effectiveness, such unidirectional models are not sufficient to learn optimal representations for user behavior sequences due to two major drawbacks. The first one is that the architecture of these models restrict the power of hidden representations, where each item can only encode the information from previous item. The other one is that such models assume a rigidly ordered sequence over data which is not always true for real-world user behaviors [9, 17].

To address these limitations, BERT4Rec was proposed [14]. Inspired by the success of BERT [2], this model applies the deep bidirectional self-attention model to sequential recommendation. BERT4Rec leverages the transformer architecture to analyze user-item interactions bidirectionally, providing a more comprehensive understanding of user behavior sequences. This method allows the model to capture complex, context-dependent patterns in user preferences, leading to more accurate and personalized recommendations. The BERT4Rec model structure is shown in Figure A1.

1.1. Contributions.

The original implementation of BERT4Rec was upon Tensorflow frameworks. Thanks to [10], we were able to make changes based on existing PyTorch implementation of the model. The author, besides the original BERT4Rec model, also added additional sampling techniques, which greatly improved the model performance. An existing hyperparameter setting is also provided in the GitHub repository. What we contribute mainly in this project include the following:

(1) Adjusting hyperparameters based on exploratory data analysis (EDA), experiments, and original paper settings. Specifically, we use mask probability of 0.2 as recommended by BERT4Rec paper [14], changing the implicit ranking threshold ($4 \rightarrow 3$), and confirming the thresholds in pruning, including the minimum ratings that users posted and the minimum ratings movies received.
(2) Besides conventional user behavior and positional information, we also incorporate user information in modeling, which is also a recommended exploratory direction by authors in [14].
(3) We includes the development of a novel genre embedding strategy. This method averages the embedding vectors of individual genres associated with a movie to form a unified genre embedding. This representation captures the multi-genre nature of films in a high-dimensional space, enhancing the movie's feature representation. For movies without genre information or those that are masked, we default to a zero vector, ensuring consistency in the embedding space.

(4) We compare a popularity-based model, default BERT baseline implemented in PyTorch with random sampling techniques, new BERT4Rec model with user embeddings, and new Bert4Rec with genre embeddings, and evaluate them in four different metrics, including Precision@k, Recall@k, Normalized Discounted Cumulative Gain, and Mean Reciprocal Rank.

2. Related Work

2.1. **Sequential Recommendation.** Early works on sequential recommendation usually capture sequential patterns from user historical interactions using Markov chains (MCs). Recently, RNN and its variants, Gated Recurrent Unit (GRU) [1] and Long Short-Term Memory(LSTM) [8], are becoming more and more popular for modeling user behavior sequences [3, 6, 7]. The basic idea of these methods is to encode user's previous records into a vector with various recurrent architectures and loss functions, including session-based GRU with ranking loss (GRU4Rec) [7], user-based GRU [3], and improved GRU4Rec with new loss function and an improved sampling strategy [6].

2.2. **Attention Mechanism.** Attention mechanism has shown promising potential in modeling sequential data. Recently, some works try to employ the attention mechanism to improve recommendation performances and interpretability [12]. For example, Li et al. [12] incorporate an attention mechanism into GRU to capture both the user's sequential behavior and main purpose in session-based recommendation.

The works mentioned above basically treat attention mechanism as an additional component to the original models. In contrast, Transformer [16] and BERT [5] are built solely on multi-head self-attention and achieve state-of-the-art results on text sequence modeling. Recently, there is a rising enthusiasm for applying purely attention-based neural networks to model sequential data for their effectiveness and efficiency.

For sequential recommendation, Kang and McAuley [11] introduce a two-layer Transformer decoder (i.e., Transformer language model) called SASRec to capture user's sequential behaviors and achieve state-of-the-art results on several public datasets. SASRec is closely related to our work. However, it is still a unidirectional model using a casual attention mask, while we use a bidirectional model to encode users' behavior sequences with the help of Cloze task.

3. Data

In this project, we use the MovieLens 20M dataset [4], provided by the GroupLens Research Project at the University of Minnesota, which is a widely recognized and extensively used dataset in the field of recommendation systems. Comprising 20 million ratings and 465,000 tag applications, the dataset spans 27,000 movies rated by 138,000 users. The ratings, ranging from 0.5 to 5 stars, were collected through the MovieLens website during the period from January 09, 1995, to March 31, 2015.

This rich dataset is instrumental for building and testing collaborative filtering algorithms, which are key in recommendation systems. Its large scale and diverse user base make it an ideal benchmark for evaluating the effectiveness of various recommendation strategies. The dataset not only includes user ratings but also provides movie metadata, such as genres and timestamps of ratings, offering a comprehensive set of features for in-depth analysis and modeling.

4. Approach

4.1. **Baseline.** To begin with, we use the simplest baseline that ranks items according to their popularity judged by the number of interactions. We recommend the most popular movies, excluding those that the user has seen before, to the user. And the popularity is measured by the number of users who have rated that movie.

4.2. **Bert4Rec.** The objective of our model is to output the most possible movies the user would like to watch based its previous behaviors. Let u be the user and v_i^u be the i^{th} movie that user u watched in chronological order. Then, the objective can be formalized as modeling probability p described in Equation 1.

$$p(v_{n+1}^u | v_1^u, v_2^u, \ldots, v_n^u) \tag{1}$$

The detailed mathematics of multi-headed attention layer is included in the original paper of BERT4Rec [14]. Here we only demonstrate the differences we make to the model.

4.2.1. *User Embeddings.* The original BERT4Rec model encodes the input in the following way. For user u, item v_i and its input representation h_i^0, there's nothing related with user information according to Equation 2.

$$h_i^0 = v_i + p_i \tag{2}$$

Now, we add a new embedding representing the users collected in parsing the input data. Let U be the embedding for all users and j be the index of user u, the input representation in Equation 3 can now contain encoded information from not only item, position but also user itself.

$$h_i^0 = v_i + p_i + U_j \tag{3}$$

4.2.2. *Genre Embeddings.* In our recommendation model, alongside user-specific embeddings, we incorporate movie genre information to enrich the feature representation. Each genre is uniquely labeled and mapped to an embedding vector in a high-dimensional space.

The genre embedding for a movie with multiple genres is computed by averaging the embeddings of its individual genres. This is formalized as:

$$\mathbf{g}_m = \frac{1}{N_m} \sum_{i=1}^{N_m} \mathbf{e}_{g_i}$$

where $\mathbf{g}_m$ represents the genre embedding for movie m, N_m is the number of genres associated with movie m, and $\mathbf{e}_{g_i}$ is the embedding vector for the i-th genre of movie m.

In cases where the movie token is a mask or an unknown entity, the genre embedding defaults to a zero vector:

$$\mathbf{g}_m = \mathbf{0} \quad \text{(for masked or unknown movies)}$$

The final embedding for a movie is obtained by adding the genre embedding to token embedding and position encoding. This approach combines user interaction data with content-based features, enhancing the model's capability to discern and predict user preferences more accurately.

4.3. **Evaluation Metrics.** After research, we decided to use 4 metrics to evaluate our model performance.

(1) Precision@k in Equation 4 measures the relevance of our model recommendations.

$$\text{Precision@}k = \frac{\#\text{of relevant recommended items@}k}{\#\text{of recommended items@}k} \tag{4}$$

(2) Recall@k in Equation 5 measures how many relevant items appear in our recommendation out of all relevant items.

$$\text{Recall@}k = \frac{\#\text{of relevant recommended items@}k}{\#\text{of relevant items@}k} \tag{5}$$

(3) Normalized Discounted Cumulative Gain (NDCG)

(a) Calculate the Discounted Cumulative Gain at k (DCG@k):

$$DCG@k = \sum_{i=1}^{k} \frac{2^{rel_i} - 1}{\log_2(i+1)}$$

(b) Calculate the Ideal DCG at k (IDCG@k), which is the DCG@k for a perfect ranking.

(c) Normalize the DCG@k by the IDCG@k to obtain NDCG@k:

$$NDCG@k = \frac{DCG@k}{IDCG@k} \tag{6}$$

A NDCG@k score of 1 indicates the ideal ranking, while a score below 1 indicates the degree of improvement needed.

(4) Mean Reciprocal Rank (MRR)

The Mean Reciprocal Rank is defined as the average of the reciprocal ranks of the first correct recommendation or relevant item:

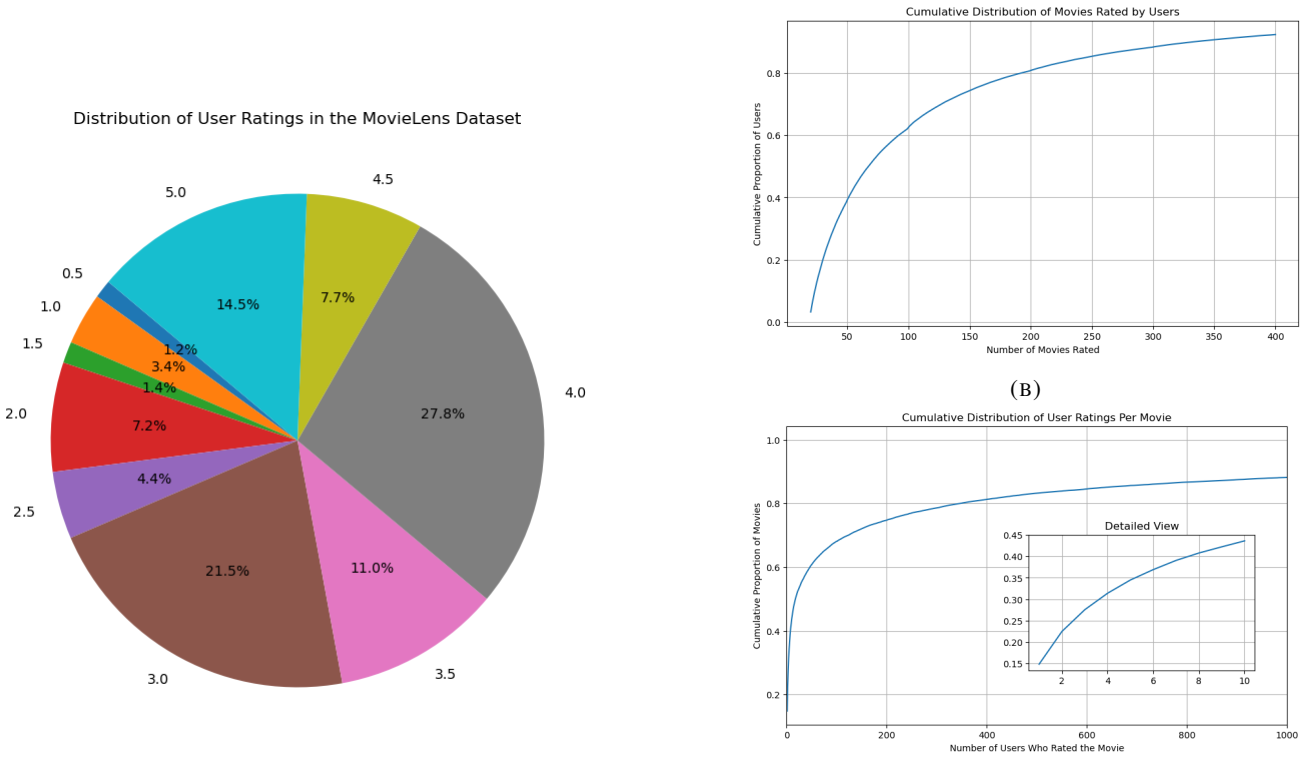
$$MRR = \frac{1}{|Q|} \sum_{i=1}^{|Q|} \frac{1}{\text{rank}_i} \tag{7}$$

where $|Q|$ is the number of queries, and rank_i is the position of the first relevant item for the i-th query in the ranked list of recommendations.

5. Experimental Results

5.1. **EDA and Data Preprocessing.** There are several steps involved in preprocessing based on EDA results:

(1) In our assessment of the models that provide sequential recommendations, we employ a widely recognized method known as the leave-one-out evaluation, particularly for predicting the subsequent item recommendation. Specifically, we reserve the final action in each user's sequence as the test set, the penultimate action as the validation set, and the earlier actions for the training set. To ensure an evaluation that is both straightforward and equitable, we pair each item in the test set, which serves as the ground truth, with 100 negative items that the user has not previously engaged with, aligning with established practices in the literature. These negative items are selected based on their popularity to



(A) (B) (C)

FIGURE 1. Analysis of MovieLens 20M dataset

guarantee that our sampling is both reliable and reflective of general user trends. The primary objective in this context is to rank the negative items alongside the ground truth for every individual user [14].

(2) In our exploratory data analysis (EDA) of the MovieLens 20M dataset, we focused on transforming explicit ratings into implicit feedback to train a recommendation system. To do this, we needed to decide on a rating threshold above which ratings would be considered positive indications of user preference.

The provided pie chart of rating distribution (Figure 1a) showed a substantial volume of ratings at 3.0 and above. We tested two thresholds: 3.0 and 4.0. The threshold of 3.0 yielded better results (Table 1) in our final recommendation model. This improvement is likely due to the larger training set obtained by including ratings of 3.0 and above as positive feedback, thus capturing a wider array of user preferences and leading to a more robust model.

TABLE 1. Comparison of Performance Metrics for Minimum Ratings of 3 and 4

	Recall@10	NDCG@10	Recall@5	NDCG@5	Recall@1	NDCG@1	MRR
MinRating=3	0.9565	0.7995	0.9042	0.7823	0.6328	0.6328	0.7509
MinRating=4	0.9532	0.7828	0.8976	0.7646	0.6024	0.6024	0.7300

(3) In the process of preparing dataset for our recommendation system, a decision was made to exclude user interactions from individuals who rated fewer than five movies. This filtering criterion was applied based on the rationale that users with fewer than five ratings do not provide sufficient information to establish reliable preference patterns. Such sparse data can introduce noise into the recommendation algorithm, leading to a degradation in model accuracy. By setting this threshold, we aimed to improve the robustness of the model by training it on users with a more substantial interaction history, thus enhancing the predictive quality of the system. Fortunately, the shortest rating sequence among all users is around 20 (Figure 1b).

(4) we opt to retain movies that had received at least one rating. This decision is informed by the observation that approximately 25% of movies in the MovieLens 20M dataset have a limited number of ratings (Figure 1c). Excluding these movies would significantly reduce the diversity and size of our training set, which could, in turn, limit the recommendation system's ability to learn nuanced user preferences across a broad spectrum of films. By preserving

movies with minimal ratings, we ensure that our model is exposed to a wider array of content, thereby enhancing its capacity to generalize and potentially discover underlying patterns that would otherwise be lost. This approach also mirrors real-world scenarios where recommendation systems often have to contend with long-tail distributions of item interactions.

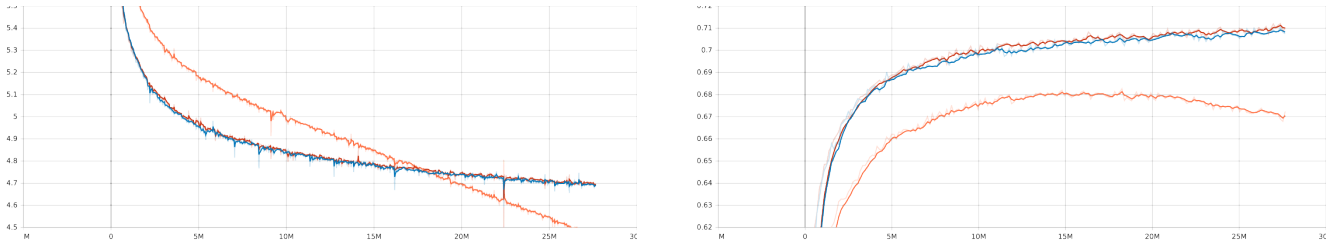
TABLE 2. Comparison of Performance Metrics across Different Models

	Recall@10	NDCG@10	Recall@5	NDCG@5	Recall@1	NDCG@1	MRR
Popularity Baseline	0.0654	0.2339	0.0388	0.2553	0.0011	0.2895	0.0201
Bert4Rec	0.9565	0.7995	0.9042	0.7823	0.6328	0.6328	0.7509
Bert4Rec w/ User Embedding	0.9535	0.7874	0.8983	0.7693	0.6114	0.6114	0.7359
Bert4Rec w/ Genre Embedding	**0.9571**	**0.8010**	**0.9063**	**0.7843**	**0.6346**	**0.6346**	**0.7525**

5.2. **Baselines.** As shown in Table 2, the non-personalized POP baseline gives the worst performance.

5.3. **Bert4Rec and the Two Variants.** Based on the hyperparameters concluded in EDA, we compare the performances across the default Bert4Rec model and the two variants. We intend to use precision@k and recall@k altogether to ensure that our model is not only outputing relevant items but also including most of them in the first k ranks. Furthermore, NDCG takes the ranking into consideration, and MRR measures the effectiveness of our model in predicting the next (one) item.

Besides, we find that the current implementation of Bert4Rec with random sampling [15] from [10], which results in great improvements, measured in the four metrics we list above in Table 2. Meanwhile, we also observe potential overfitting within the model due to the fixed seed in the process of negative sampling. According to Figure 2a and Figure 2b, we can observe a clear overfitting trend ocurring for Bert4Rec with User Embedding. Nevertheless, based on the current implementation, we are still able to draw the conclusion that including rich features like genre information of items could improve the model to give better results. On the other hand, adding user embedding needs careful fine-tuning and requires some deeper dive to incorporate this feature so that the model can properly take advantage of it.



(A) Training Loss in Number of Weight Changes

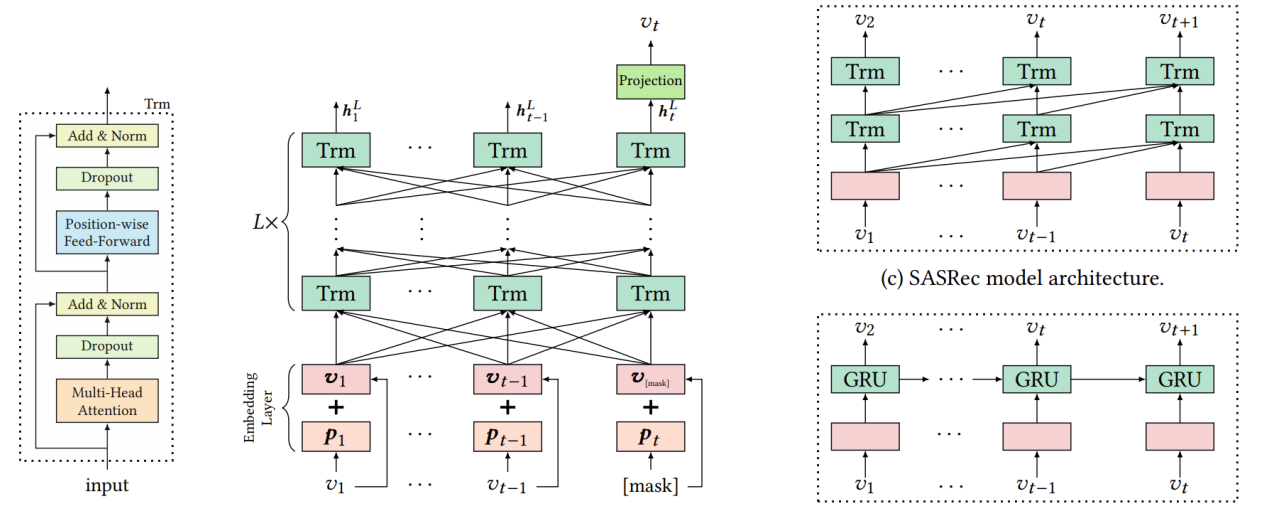
(B) Validation Recall@1 in Number of Weight Changes

FIGURE 2. Performance Overview on Test Set for BERT4Rec (Blue), Bert4Rec w. User Embedding (Orange), Bert4Rec w. Genre Embedding (Red), Excluding Outliers

6. DISCUSSION

In this project, we compare across four different models, the popularity-based model, Bert4Rec (with random sampling), Bert4Rec incorporating user embedding, and Bert4Rec with genre embedding. The latter two are inspired from the Bert4Rec authors' suggestions [14]. From our experiments, simple popularity-based model which doesn't utilize item/position/user information doesn't generalize well to every user. On the other hand, complex models like Bert4Rec is able to capture such information and trend and so provide good-enough predictions over the next items. Moreover, we find that benefiting from rich features like genre included in the training stage, the model can even generalize even better. Nevertheless, adding more features also require more work in fine-tuning; as indicated in our experiment, adding user embedding directly without much fine-tuning would actually hurt the model performance slightly.

If we have more time, we would also like to test the model with both embeddings incorporated and also test this model on other datasets to see if there's any differences.



(a) Transformer Layer. (b) BERT4Rec model architecture. (c) SASRec model architecture. (d) RNN based sequential recommendation methods.

FIGURE A1. Differences in sequential recommendation model architectures. BERT4Rec learns a bidirectional model via Cloze task, while SASRec and RNN based methods are all left-to-right unidirectional model which predict next item sequentially.[14]

APPENDIX A. FIGURES

REFERENCES

[1] Kyunghyun Cho et al. *Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation*. 2014. arXiv: 1406.1078 [cs.CL].

[2] Jacob Devlin et al. "Bert: Pre-training of deep bidirectional transformers for language understanding". In: *arXiv preprint arXiv:1810.04805* (2018).

[3] Tim Donkers, Benedikt Loepp, and Jürgen Ziegler. "Sequential User-Based Recurrent Neural Network Recommendations". In: *Proceedings of the Eleventh ACM Conference on Recommender Systems*. RecSys '17. Como, Italy: Association for Computing Machinery, 2017, pp. 152–160. ISBN: 9781450346528. DOI: 10.1145/3109859.3109877. URL: https://doi.org/10.1145/3109859.3109877.

[4] F. Maxwell Harper and Joseph A. Konstan. "The MovieLens Datasets: History and Context". In: *ACM Trans. Interact. Intell. Syst.* 5.4 (Dec. 2015). ISSN: 2160-6455. DOI: 10.1145/2827872. URL: https://doi.org/10.1145/2827872.

[5] Ruining He, Wang-Cheng Kang, and Julian McAuley. "Translation-based Recommendation". In: *Proceedings of the Eleventh ACM Conference on Recommender Systems*. RecSys '17. ACM, Aug. 2017. DOI: 10.1145/3109859.3109882. URL: http://dx.doi.org/10.1145/3109859.3109882.

[6] Balázs Hidasi and Alexandros Karatzoglou. "Recurrent Neural Networks with Top-k Gains for Session-based Recommendations". In: *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*. CIKM '18. ACM, Oct. 2018. DOI: 10.1145/3269206.3271761. URL: http://dx.doi.org/10.1145/3269206.3271761.

[7] Balázs Hidasi et al. *Session-based Recommendations with Recurrent Neural Networks*. 2016. arXiv: 1511.06939 [cs.LG].

[8] Sepp Hochreiter and Jürgen Schmidhuber. "Long Short-Term Memory". In: *Neural Comput.* 9.8 (Nov. 1997), pp. 1735–1780. ISSN: 0899-7667. DOI: 10.1162/neco.1997.9.8.1735. URL: https://doi.org/10.1162/neco.1997.9.8.1735.

[9] Liang Hu et al. "Diversifying Personalized Recommendation with User-session Context." In: *IJCAI*. 2017, pp. 1858–1864.

[10] SungMinCho Jaywonchuang. *BERT4Rec-VAE-Pytorch*. Apr. 2020. URL: https://github.com/jaywonchung/BERT4Rec-VAE-Pytorch.

[11] Wang-Cheng Kang and Julian McAuley. *Self-Attentive Sequential Recommendation*. 2018. arXiv: 1808.09781 [cs.IR].

[12] Jing Li et al. *Neural Attentive Session-based Recommendation*. 2017. arXiv: 1711.04725 [cs.IR].

[13] Steffen Rendle, Christoph Freudenthaler, and Lars Schmidt-Thieme. "Factorizing personalized markov chains for next-basket recommendation". In: *Proceedings of the 19th international conference on World wide web*. 2010, pp. 811–820.

[14] Fei Sun et al. *BERT4Rec: Sequential Recommendation with Bidirectional Encoder Representations from Transformer*. 2019. arXiv: 1904.06690 [cs.IR].

[15] Zhu Sun et al. "Are We Evaluating Rigorously? Benchmarking Recommendation for Reproducible Evaluation and Fair Comparison". In: *Proceedings of the 14th ACM Conference on Recommender Systems*. RecSys '20. Virtual Event, Brazil: Association for Computing Machinery, 2020, pp. 23–32. ISBN: 9781450375832. DOI: 10.1145/3383313.3412489. URL: https://doi.org/10.1145/3383313.3412489.

[16] Ashish Vaswani et al. *Attention Is All You Need*. 2023. arXiv: 1706.03762 [cs.CL].

[17] Shoujin Wang et al. "Attention-based transactional context embedding for next-item recommendation". In: *Proceedings of the AAAI conference on artificial intelligence*. Vol. 32. 1. 2018.