Enhancing Recommendations of Items by Making Some Changes in Layers of BERT Model

Ashima Malik
Ph.D. Scholar
Computer Science and Engineering
PDM University,
Bahadurgarh, India

S. Srinivasan, PhD
HOD
Computer Science and Engineering
PDM University,
Bahadurgarh, India

Piyush Prakash, PhD
Assistant HOD, Associate Professor
Computer Science and Applications
PDM University,
Bahadurgarh, India

ABSTRACT

Introducing an innovative methodology for modeling user behavior sequences in recommendation systems, this paper proposes the use of a bidirectional self-attention network and Cloze task, drawing inspiration from Bidirectional Encoder Representations from Transformers (BERT) to enhance the recommendations of products on e-commerce websites. Traditional recommendation system models that are unidirectional have limitations, mainly in the power of hidden representations and rigid ordering of historical user interactions. Overcoming these limitations, the suggested BERT4Rec model is bidirectional, offering the context from both directions. The paper suggests utilizing the Cloze task to prevent data leakages from bidirectional conditioning. This includes masking random components within the input sequences and predicting them based on their nearby context. Comprehensive experiments are conducted, resulting in consistently better outcomes than state-of-the-art comparable options across four datasets. This exploration sets the groundwork by introducing the Cloze objective and deep bidirectional sequential modeling to the recommendation system field. Furthermore, the study is a foundation for future studies investigating explicit user modeling and incorporating item features.

General Terms

Machine Learning, Product Recommendation.

Keywords

Product Recommendation; BERT, BERT4Rec; SAS.

1. INTRODUCTION

Accurate characterization of users' interests is crucial for effective recommendation systems. In real-world applications, users' interests are dynamic and influenced by their historical behaviors. For example, a user may purchase accessories for a gaming console shortly after buying the console, even though they would not normally buy console accessories. To model such sequential dynamics in user behaviors, various methods have been proposed to make sequential recommendations based on users' historical interactions (Sepp Hochreiter and Jürgen Schmidhuber, 1997). Recently, sequential neural networks such as Recurrent Neural Networks (RNNs) have been used for sequential recommendation and have shown promising results. These methods typically encode a user's historical interactions into a vector representation of their preferences using a left-to-right sequential model and make recommendations based on this hidden representation. However, arguing that such left-to-right unidirectional models have limitations in learning optimal representations for user behavior sequences. One major limitation is that these models restrict the power of hidden representations for items in historical sequences, as each item can only encode information from previous items (Geoffrey Hinton et al., 2015). Another limitation is that unidirectional models assume a rigid order in the sequence of items, which is not always true for user behaviors in real-world applications. In fact, the choices of items in a user's historical interactions may not follow a strict order due to various external factors. Therefore, it is important to incorporate context from both directions in user behavior sequence modelling.

To address these limitations, proposing using a bidirectional model to learn representations for user's historical behavior sequences, inspired by the success of Bidirectional Encoder Representations from Transformers(BERT) in text understanding. The deep bidirectional self-attention model is applied to sequential recommendation. Bidirectional models have shown superior results in text sequence modeling tasks, indicating that incorporating context from both sides is beneficial for learning sequence representations (Balázs Hidasi and Alexandros Karatzoglou, 2018).

However, training a bidirectional model for sequential recommendation is not straightforward, as conventional sequential recommendation models are usually trained left-to-right by predicting the next item for each position in the input sequence. Jointly conditioning on both left and right context in a deep bidirectional model would cause information leakage, as it would allow each item to indirectly "see the target item", making predicting the future trivial and not learning anything useful. To overcome this problem, introducing the Cloze task as an alternative objective to replace the left-to-right prediction objective used in unidirectional models (Xiangnan et al., 2017). In the Cloze task, randomly mask some items in the input sequences and predict the masked items based on their surrounding context. This way, avoiding information leakage and allow the representation of each item in the input sequence to fuse both the left and right context in the bidirectional model. Another advantage of the Cloze objective is that it can generate more samples to train a more powerful model in multiple epochs. However, a downside of the Cloze task is that it is not consistent with the final task of sequential recommendation. To address this, during testing, appending a special token "[mask]" at the end of the input sequence to indicate the item that needs to be predicted, and then make recommendations based on its final hidden vector.

Extensive experiments are conducted on four datasets to evaluate the proposed model, and the results showed that the model consistently outperforms various state-of-the-art baselines. Additionally, conducting a thorough ablation study

to analyze the contributions of the key components in the proposed model.

2. RELATED WORK

2.1 General Recommendation

Traditional approaches to recommendation systems, such as Collaborative Filtering (CF), have been widely used to model users' preferences based on their interaction histories. One popular CF method is Matrix Factorization (MF), where users and items are projected into a shared vector space, and user-item preferences are estimated through the inner product of their vectors (Ruining He et al., 2017). Another approach is item-based neighborhood methods, which estimate a user's preference on an item by measuring its similarity with items in their interaction history using a precomputed item-to-item similarity matrix. However, in recent years, deep learning has revolutionized recommendation systems. Deep learning-based methods have since pursued two main directions. One direction focuses on improving recommendation performance by incorporating distributed item representations learned from auxiliary information such as text, images, or acoustic features into CF models. This allows for a more comprehensive understanding of items and their relevance to users. The other direction aims to replace traditional matrix factorization approaches with deep learning-based models. For instance, Neural Collaborative Filtering (NCF) utilizes Multi-Layer Perceptrons (MLP) instead of the inner product to estimate user preferences, capturing more complex user-item interactions (Petrov and Macdonald, 2022). AutoRec and CDAE, on the other hand, predict users' ratings using an Auto-encoder framework, which allows for non-linear representations of user-item preferences. These deep learning-based approaches have shown promising results in improving recommendation performance by leveraging the power of neural networks to capture intricate patterns in user-item interactions. By incorporating auxiliary information and utilizing advanced modeling techniques, deep learning-based recommendation systems have the potential to enhance recommendation accuracy and provide more personalized and relevant recommendations to users.

2.2 Sequential Recommendation

Earlier works used Markov chains (MCs) to capture sequential patterns from user historical interactions in sequential recommendation. Markov Decision Processes (MDPs) were also used to address recommendation generation as a sequential optimization problem (Sun et al., 2019). Later, Factorizing Personalized Markov Chains (FPMC) combined the power of MCs and matrix factorization (MF) to model both sequential behaviors and general interests, while high-order MCs were also adopted to consider more previous items. More recently, recurrent neural networks (RNNs) and their variants, such as Gated Recurrent Unit (GRU) and Long Short-Term Memory (LSTM), have gained popularity for modeling user behavior sequences (Nagy et al., 2021). These methods encode a user's previous records into a vector that represents their preferences, which is then used for making predictions. Several recurrent architectures and loss functions have been proposed, including session-based GRU with ranking loss (GRU4Rec), Dynamic Recurrent basket Model (DREAM), user-based GRU, attention-based GRU (NARM), and improved GRU4Rec with new loss functions (BPR-max and TOP1-max) and an improved sampling strategy.

In addition to recurrent neural networks, other deep learning models have been introduced for sequential recommendation. For example, Tang and Wang proposed a Convolutional Sequence Model (Caser) that uses both horizontal and vertical convolutional filters to learn sequential patterns. Memory Network is another approach that aims to improve sequential recommendation. STAMP, on the other hand, captures both users' general interests and current interests using a Multi-Layer Perceptron (MLP) network with attention (Ciniselli et al., 2021). These deep learning-based approaches have shown promise in modeling sequential user behaviors and capturing intricate patterns in user-item interactions over time. By considering the order of users' behaviors, these models have the potential to provide more accurate and relevant recommendations in dynamic and evolving recommendation scenarios (Lu et al., 2020). These advancements in deep learning-based sequential recommendation methods open up new possibilities for developing more effective and personalized recommendation systems that can adapt to users' changing preferences and behaviors.

2.3 Attention Mechanism

The attention mechanism has gained recognition for its potential in modeling sequential data, such as machine translation and text classification due to their effectiveness and efficiency. Recently, there has been a growing interest in leveraging the attention mechanism to enhance recommendation performance and interpretability by integrating it into the GRU model to capture user's sequential behavior. Transformer and BERT models are built entirely on multi-head self-attention and have achieved state-of-the-art results in text sequence modeling. In the field of sequential recommendation, Kang and McAuley introduced SASRec, a two-layer Transformer decoder, which captures users' sequential behaviors and achieves top-performing results on several public datasets (Mozafari et al., 2020). Although SASRec is closely related to the current work, it is a unidirectional model that uses a causal attention mask, whereas the current approach utilizes a bidirectional model and encodes users' behavior sequences with the help of the Cloze task.

3. BERT4Rec

3.1 Problem Statement

In the domain of sequential recommendation, a set of users denoted as $U = \{u1, u2, ..., u|U|\}$, a set of items denoted as $V = \{v1, v2, ..., v|V|\}$ are considered, and an interaction sequence for a user $u \in U$ denoted as $Su = [v(u)1, ..., v(u)t, ..., v(u)nu]$. Here, $v(u)t \in V$ represents the item that user u has interacted with at time step t, and nu represents the length of the interaction sequence for user u. The objective of sequential recommendation is to predict the item that user u will interact with at time step nu + 1, based on their past interaction history Su. This prediction task can be mathematically formulated as modeling the probability distribution over all possible items for user u at time step nu + 1, denoted as $p(v(u)nu + 1 \mid Su)$.

3.2 Model Architecture

A new model called BERT4Rec is introduced for sequential recommendation, which incorporates Bidirectional Encoder Representations from Transformers (BERT) into the task. BERT4Rec is constructed using the popular self-attention layer known as the Transformer layer, as shown in Figure 1b (Qiao et al., 2022). It consists of L bidirectional Transformer layers, where at each layer, the representation of each position is iteratively revised by exchanging information across all positions from the previous layer in parallel with the Transformer layer. BERT4Rec utilizes the self-attention mechanism, which allows it to directly capture dependencies between any distances. Self-attention is easy to parallelize, unlike RNN-based methods (Lin et al., 2021). BERT4Rec

captures more powerful representations of users' behavior sequences, leading to improved recommendation performance.

3.3 Transformer Layer

Hidden representations, denoted as hli can be computed for each position i at each layer in a Transformer-based model. The input sequence has a length of t, and the Transformer layer is applied iteratively to compute hli (Risch and Krestel, 2020). These hidden representations are stacked together into a matrix Hl of size t×d, where d is the hidden dimension, as the attention function is computed on all positions simultaneously during training.

The Transformer layer consists of two sub-layers: Multi-Head Self-Attention and Position-wise Feed-Forward Network.

Multi-Head Self-Attention is a key component in sequence modeling tasks, as it allows for capturing dependencies between representations regardless of their distance in the sequence. Instead of using a single attention function, a multi-head approach is adapted. In this approach, the input representations H1 are linearly projected into h subspaces using different, learnable linear projections. Then, h attention functions are applied in parallel to generate output representations, which are concatenated and projected again for further processing. This multi-head self-attention mechanism enables the model to capture complex dependencies and interactions among representations in the sequence, enhancing its ability to model sequential data effectively.

$$MH(H1) = [head1;\ head2;\ \ldots.;\ headh]Wo$$

$$Headi = Attention(H1WQi, H1WKi, H1WVi)$$

The Attention function used is the Scaled Dot-Product Attention, which computes a softmax over the dot product of the query (Q) and key (K) matrices, scaled by a temperature parameter pd/h, and then multiplied with the value (V) matrix. The SoftMax operation produces a probability distribution that indicates the importance of different positions in the input sequence.

The Position-wise Feed-Forward Network is employed to introduce nonlinearity and interactions between dimensions in the outputs of the self-attention sub-layer (Shi and Lin, 2019). This network operates independently and uniformly at each position in the sequence. It comprises of two affine transformations, with a Gaussian Error Linear Unit (GELU) activation function applied in between. This allows the model to incorporate nonlinearity and capture complex interactions between different dimensions in the input sequence, enhancing the expressive power of the model. The Position-wise Feed-Forward Network is applied to the outputs of the self-attention sub-layer, enabling the model to capture more nuanced and nonlinear patterns in the data, which is important for achieving higher performance in various sequence modeling tasks (Wang et al., 2019).

$$PFFN(H1) = FFN(h11)T; \ldots.; FFN(h1t)TT$$

$$FFN(x) = GELU(xW(1) + b(1))W(2) + b(2)$$

The GELU activation function is a smoother alternative to the standard Rectified Linear Unit (ReLU) activation function, commonly used in OpenAI GPT and BERT.

To capture item-item interactions effectively in the user behavior sequence, self-attention mechanisms are utilized. However, to learn more intricate item transition patterns, it is beneficial to stack multiple self-attention layers (Sun et al., 2019). Nevertheless, as the network goes deeper, it becomes more challenging to train. To mitigate this, residual connections are employed around each of the two sub-layers with layer normalization. Additionally, dropout is applied to the output of each sub-layer before normalization. This means that the output of each sub-layer is normalized using the layer normalization function (LN) after adding the input (x) to the output of the dropout function applied to the sub-layer output (Nozza et al., 2020). Layer normalization is used to normalize the inputs across all the hidden units in the same layer, stabilizing and accelerating network training. By incorporating residual connections, layer normalization, and dropout, the model can effectively mitigate the challenges associated with deep network architectures, allowing for better training and improved performance in capturing complex patterns in the data.

In summary, the hidden representations of each layer in BERT4Rec are refined as follows:

$$H1 = Trm(H1 - 1), \forall \in [1, \ldots, L]$$

$$Trm(H1 - 1) = LN(A1 - 1 + Dropout(PFFN(A1 - 1)))$$

$$A1 - 1 = LN(H1 - 1 + Dropout(MH(H1 - 1)))$$

where Trm denotes the Transformer layer, LN denotes the layer normalization function, MH denotes the Multi-Head Self-Attention, and PFFN denotes the Position-wise Feed-Forward Network. Dropout is used for regularization during training, and L denotes the total number of layers in the model.

3.4 Embedding Layer

As mentioned previously, the Transformer layer (Trm) lacks awareness of the input sequence order due to the absence of recurrence or convolutional modules. To incorporate sequential information, Positional Embeddings are injected into the input item embeddings at the bottom of the Transformer layer stacks. The input representation (h0i) for a given item vi is obtained by summing the corresponding item embedding (vi) and positional embedding (pi) as follows: h0i = vi + pi, where vi ∈ E represents the d-dimensional embedding for item vi, and pi ∈ P represents the d-dimensional positional embedding for the position index i (Tsai et al., 2019). In this study, learnable positional embeddings are used instead of fixed sinusoid embeddings from a previous work for improved performance.

The positional embedding matrix P ∈ R N × d enables the model to identify the portion of the input it is processing. However, it also imposes a limitation on the maximum sentence length (N) that the model can handle. Thus, if the input sequence [v1, . . . , vt] exceeds N items, it needs to be truncated to the last N items [vu t−N +1, . . . , vt] where t > N. This truncation ensures that the model operates within the maximum sentence length N, allowing for effective utilization of positional embeddings to capture the sequential information in the input sequence.

3.5 Output Layer

As mentioned previously, the Transformer layer (Trm) lacks awareness of the input sequence The final output H_L for all items in the input sequence is obtained after passing through L layers that exchange information hierarchically across all positions in the previous layer. If the item vt is masked at time step t, the masked items vt can be predicted based on h_{Lt}. This prediction is generated using a two-layer feed-forward network

with GELU activation in between, which produces an output distribution over target items using the softmax function:

$$P(v) = softmax\ (GELU(h_{Lt} W_P + B_P)E^T + b_o)$$

Here, W_P is a learnable projection matrix, b_P and b_o are bias terms, and E ∈ R |V |×d is the embedding matrix for the item set V. To mitigate overfitting and reduce model size, a shared item embedding matrix is utilized in both the input and output layers. The output distribution P(v) represents the probabilities of the masked items vt being each possible target item in the item set V. The GELU activation function is used to introduce non-linearity in the network, allowing it to capture complex patterns in the data (Rogers et al., 2021). The projection matrix W_P and bias terms b_P and b_O are learned during the training process, allowing the model to adapt and optimize its predictions based on the input data. The embedding matrix E is a matrix of size |V|×d, where |V| represents the size of the item set V and d represents the embedding dimension. The embedding matrix is used to represent the items in a continuous vector space, allowing the model to capture semantic relationships between items. By sharing the embedding matrix in the input and output layers, the model can benefit from the shared information, improving generalization and reducing the risk of overfitting. tabs, and so on.

3.6 Model Learning

To efficiently train the proposed model, a new objective called Cloze task (also known as "Masked Language Model") is applied to sequential recommendation. Cloze task is a test that involves a portion of language with some words removed, and the participant is asked to fill in the missing words (Lu et al., 2020). In this case, for each training step, a random proportion ρ of all items in the input sequence is masked by replacing them with a special token "[mask]", and then the original ids of the masked items are predicted solely based on their left and right context. The final hidden vectors corresponding to "[mask]" are then fed into an output softmax over the item set, similar to conventional sequential recommendation. The loss for each masked input S′u is defined as the negative log-likelihood of the masked targets, which is calculated based on the probability of the true item for the masked item.

An additional advantage of using Cloze task is that it can generate more samples to train the model (Mozafari et al., 2020). In conventional sequential predictions, each sequence of length n produces n unique samples for training, however, with BERT4Rec, n*k samples can be obtained (if k items are randomly masked) in multiple epochs, allowing for training a more powerful bidirectional representation model.

To address the mismatch between the training and the final sequential recommendation task introduced by Cloze task, the special token "[mask]" is appended to the end of the user's behavior sequence during testing, and the next item is predicted based on the final hidden representation of this token. Additionally, during training, samples that only mask the last item in the input sequences are produced, similar to fine-tuning for sequential recommendation, which can further improve the recommendation performances (Akhtyamova, 2020).

For example: The hidden vectors corresponding to the special token "[mask]" are used to generate an output softmax over the item set, like conventional sequential recommendation. The loss for each masked input sequence S ′ u is computed as the negative log-likelihood of the true targets for the masked items, denoted as v ∗ m, in the sequence S m u, which contains randomly masked items.

One advantage of using the Cloze task is that it can generate more training samples for the model (Ciniselli et al., 2021). Assuming a sequence of length n, conventional sequential predictions produce n unique samples for training, while BERT4Rec with Cloze task can generate n*k samples (where k is the number of randomly masked items) in multiple epochs. This enables training a more powerful bidirectional representation model that captures richer contextual information from both left and right contexts, leading to improved recommendation performance.

This approach can be seen as a form of fine-tuning for sequential recommendation, and it has the potential to further enhance the recommendation performance by better aligning the training objective with the ultimate prediction goal of the model.

Table 1. Datasets Statistics

Datasets	#users	#items	#actions	Avg. length	Density
Beauty	40,226	54,542	0.35m	8.8	0.02%
Steam	281,428	13,044	3.5m	12.4	0.10%
ML-1m	6040	3416	1.0m	163.5	4.79%
ML-20m	138,493	26,744	20m	144.4	0.54%

4. EXPERIMENTS

4.1 Datasets

Assessing the performance of the proposed model on four real-world datasets that cover diverse domains and levels of sparsity. These datasets include:

- Amazon Beauty: This dataset comprises product review data obtained from Amazon.com, which was originally crawled by McAuley et al. [34]. The data is divided into separate datasets based on top-level product categories, and in the evaluation, focus on the "Beauty" category.
- Steam: This dataset is collected from Steam, a large online video game distribution platform, as curated by Kang and McAuley [22].
- MovieLens: This is a widely used benchmark dataset for evaluating recommendation algorithms. Utilize two established versions of MovieLens, namely MovieLens 1m (ML1m) and MovieLens 20m (ML-20m).

For dataset preprocessing, standard practices are followed employed in previous studies. Specifically, convert all numeric ratings or the presence of a review into implicit feedback, where a value of 1 indicates that the user has interacted with the item. Then group the interaction records by users and construct interaction sequences for each user by sorting the records based on timestamps. To ensure dataset quality, adopt the common practice of retaining only users with at least five feedbacks. Table 1 provides an overview of the statistics of the processed datasets.

4.2 Task Settings and Evaluation Metrics

To assess the effectiveness of the sequential recommendation models, utilizing the widely used leave-one-out evaluation approach, also known as the next item recommendation task, which has been employed in previous studies. For each user, withhold the last item in their behavior sequence as the test data, designate the item just before the last as the validation set, and use the remaining items for training.

To ensure a fair and consistent evaluation, adopting the common strategy used in where randomly sample 100 negative items for each user from the items they have not interacted with. To ensure the sampling is reliable and representative, then sample these negative items based on their popularity. Consequently, the task involves ranking these 100 negative items along with the ground truth item for each user.

Then utilize a variety of evaluation metrics to assess the performance of the ranking lists generated by the models. These metrics include Hit Ratio (HR), Normalized Discounted Cumulative Gain (NDCG), and Mean Reciprocal Rank (MRR). HR@k, which is equivalent to Recall@k and proportional to Precision@k, is reported with k = 1, 5, and 10 in this study. Additionally, MRR is equivalent to Mean Average Precision (MAP). Higher values for these metrics indicate better performance.

4.3 Baselines and Implementation Details

To evaluate the effectiveness of the method, comparing it with several baseline methods commonly used in the field of sequential recommendation. These baselines include:

- POP: This is a simple baseline that ranks items based on their popularity, determined by the number of interactions.
- BPR-MF: This baseline optimizes matrix factorization using implicit feedback and a pairwise ranking loss.
- NCF: This baseline models user-item interactions using a Multi-Layer Perceptron (MLP) instead of the inner product used in matrix factorization.
- FPMC: This baseline combines matrix factorization with first-order Markov Chains (MCs) to capture users' general taste as well as their sequential behaviors.
- GRU4Rec: This baseline uses a Gated Recurrent Unit (GRU) with a ranking-based loss to model user sequences for session-based recommendation.
- GRU4Rec+: This is an improved version of GRU4Rec that incorporates a new class of loss functions and sampling strategy.
- Caser: This baseline employs a Convolutional Neural Network (CNN) in both horizontal and vertical ways to model high-order MCs for sequential recommendation.
- SASRec: This baseline uses a left-to-right Transformer language model to capture users' sequential behaviors and has shown state-of-the-art performance in sequential recommendation.

For some of the baselines (NCF, GRU4Rec, GRU4Rec+, Caser, SASRec), the code is used provided by the corresponding authors. For BPR-MF and FPMC, are implemented using TensorFlow. Considering common hyperparameter settings such as hidden dimension size, $\ell 2$ regularizer, dropout rate, etc., and tuned them on the validation sets. It has been reporting the results of each baseline under its optimal hyperparameter settings. Implementing BERT4Rec with TensorFlow, initializing all parameters using a truncated normal distribution with a range of [-0.02, 0.02]. Although provided training the models using the Adam optimizer with a learning rate of 1e-4, $\beta 1 = 0.9$, $\beta 2 = 0.999$, $\ell 2$ weight decay of 0.01, and linear decay of the learning rate. Clipping the gradient when its $\ell 2$ norm exceeded a threshold of 5 for fair comparison. It set the layer number L = 2 and head number h = 2, using the same maximum sequence length as in previous works (N = 200 for ML-1m and ML-20m, N = 50 for Beauty and Steam datasets). The dimensionality of each head was empirically set as 32 (single head if $d < 32$). The mask proportion ρ is tuned and using the validation set, resulting in $\rho = 0.6$ for Beauty, $\rho = 0.4$ for Steam, and $\rho = 0.2$ for ML-1m and ML-20m. All the models were trained from scratch on a single NVIDIA GeForce GTX 1080 Ti GPU with a batch size of 256.

4.4 Overall Performance Comparison

Table 2 presents a summary of the best results achieved by various models on four benchmark datasets. The last column shows the performance improvement of BERT4Rec compared to the best baseline. NDCG@1 results are omitted as they are equal to HR@1 in the experiments.

The non-personalized POP method performs the worst on all datasets, as it does not consider users' personalized preferences based on their historical records. Among all the baseline methods, sequential methods such as FPMC and GRU4Rec+ consistently outperform non-sequential methods like BPR-MF and NCF on all datasets. This indicates that considering sequential information is beneficial for improving recommendation system performance.

Among the sequential recommendation baselines, Caser performs better than FPMC on all datasets, especially on the dense dataset ML-1m, suggesting that modeling high-order MCs (Markov Chains) is beneficial for sequential recommendation. However, Caser tends to perform worse than GRU4Rec+ and SASRec, especially on sparse datasets, possibly due to the small order L used in high-order MCs, which do not scale well. Furthermore, SASRec performs significantly better than GRU4Rec and GRU4Rec+, indicating that the self-attention mechanism is a more powerful tool for sequential recommendation.

Based on the results, it is evident that BERT4Rec performs the best among all methods on all four datasets, outperforming the strongest baselines. On average, BERT4Rec achieves 7.24% improvement in HR@10, 11.03% improvement in

Table 2. Performance comparison of methods for next-item prediction. Bold indicates best, underlined indicates second best, with statistically significant improvements over baselines (p < 0.01)

Datasets	Metric	POP	BPR-MF	NCF	FPMC	GRU4REC	GRU4Rec+	Caser	SASRec	BERT4Rec	Improv.
BEAUTY	HR@1	0.0077	0.0415	0.0407	0.0435	0.0402	0.0551	0.0475	<u>0.0906</u>	**0.0953**	5.19%
	HR@5	0.0392	0.1209	0.1305	0.1387	0.1315	0.1781	0.1625	<u>0.1934</u>	**0.2207**	14.12%

	HR@10	0.0762	0.1992	0.2142	0.2401	0.2343	0.2654	0.2590	0.2653	**0.3025**	14.02%
	NDCG@5	0.0230	0.0814	0.0855	0.0902	0.0812	0.1172	0.1050	0.1436	**0.1599**	11.35%
	NDCG@10	0.0349	0.1064	0.1124	0.1211	0.1074	0.1453	0.1360	0.1633	**0.1862**	14.02%
	MRR	0.0437	0.1006	0.1043	0.1056	0.1023	0.1299	0.1205	0.1536	**0.1701**	10.74%
STEAM	HR@1	0.0159	0.0314	0.0246	0.0358	0.0574	0.0812	0.0495	0.0885	**0.0957**	8.14%
	HR@5	0.0805	0.1177	0.1203	0.1517	0.2171	0.2391	0.1766	0.2559	**0.2710**	5.90%
	HR@10	0.1389	0.1993	0.2169	0.2551	0.3313	0.3594	0.2870	0.3783	**0.4013**	6.08%
	NDCG@5	0.0477	0.0744	0.0717	0.0945	0.1370	0.1613	0.1131	0.1727	**0.1842**	6.66%
	NDCG@10	0.0665	0.1005	0.1026	0.1283	0.1802	0.2053	0.1484	0.2147	**0.2261**	5.31%
	MRR	0.0669	0.0942	0.0932	0.1139	0.1420	0.1757	0.1305	0.1874	**0.1949**	4.00%
ML-1M	HR@1	0.0141	0.0914	0.0397	0.1386	0.1583	0.2092	0.2194	0.2351	**0.2863**	21.78%
	HR@5	0.0715	0.2866	0.1932	0.4297	0.4673	0.5103	0.5353	0.5434	**0.5876**	**8.13%**
	HR@10	0.1358	0.4301	0.3477	0.5946	0.6207	0.6351	0.6692	0.6629	**0.6970**	4.15%
	NDCG@5	0.0416	0.1903	0.1146	0.2885	0.3196	0.3705	0.3832	0.3980	**0.4454**	11.91%
	NDCG@10	0.0621	0.2365	0.1640	0.3439	0.3627	0.4064	0.4268	0.4368	**0.4818**	10.32%
	MRR	0.0627	0.2009	0.1358	0.2891	0.3041	0.3462	0.3648	0.3790	**0.4254**	12.24%
ML-20m	HR@1	0.0221	0.0553	0.0231	0.1079	0.1459	0.2021	0.1232	0.2544	**0.3440**	35.22%
	HR@5	0.0805	0.2128	0.1358	0.3601	0.4657	0.5118	0.3804	0.5727	**0.6323**	10.41%
	HR@10	0.1378	0.3538	0.2922	0.5201	0.5844	0.6524	0.5427	0.7136	**0.7473**	4.72%
	NDCG@5	0.0511	0.1332	0.0771	0.2239	0.3090	0.3630	0.2538	0.4208	**0.4967**	18.04%
	NDCG@10	0.0695	0.1786	0.1271	0.2895	0.3637	0.4087	0.3062	0.4665	**0.5340**	14.47%
	MRR	0.0709	0.1503	0.1072	0.2273	0.2967	0.3476	0.2529	0.4026	0.4785	18.85%

NDCG@10, and 11.46% improvement in MRR compared to the best baselines.

Question: Are the improvements in performance attributed to the bidirectional self-attention model or the Cloze objective in BERT4Rec?

To investigate the effects of the bidirectional self-attention model and the Cloze objective in BERT4Rec, we conducted experiments where the Cloze task only masked one item at a time, isolating the effects of these two factors. In comparison to SASRec, the BERT4Rec (with 1 mask) predicts the target item by conditioning on both left and right context. The results,

reported in Table 3 for Beauty and ML-1m with d = 256 due to space limitations, show that BERT4Rec with 1 mask outperforms SASRec on all evaluation metrics, highlighting the importance of bidirectional representations in sequential recommendation. Additionally, the last two rows of the table indicate that the Cloze objective also contributes to improved performance.

Table 3. Analysis of bidirectional and Cloze models with dimensionality d = 256

MODEL	BEAUTY			ML-1m		
	HR@10	NDCG@10	MRR	HR@10	NDCG@10	MRR
SASRec	0.2653	0.1633	0.153	0.6629	0.4368	0.379
BERT4Rec(1 mask)	0.294	0.1769	0.161	0.6869	0.4696	0.412
BERT4Rec	0.3025	0.1862	0.17	0.697	0.4818	0.4254

Question 2: What are the reasons and mechanisms behind the superior performance of bidirectional models compared to unidirectional models?

To address this question, the aim is to uncover significant patterns by visualizing the average attention weights of the last 10 items during the test on the Beauty dataset, as shown in Figure 1. Four representatives are provided attention heat-maps from different layers and heads, taking into consideration the limitation of space for reporting.

Several observations have been made from the results of our analysis. Firstly, we noticed that attention varies across different heads, with head 1 in layer 1 tending to focus on items on the left side, while head 2 prefers items on the right side. Secondly, attention also varies across different layers, with layer 2 tending to focus more on recent items, as it is directly connected to the output layer and recent items play a crucial role in predicting the future. Additionally, it has been observed that some heads tend to attend on the [mask] token, possibly indicating a way for self-attention to propagate sequence-level state to the item level. Most importantly, unlike unidirectional models that can only attend to items on the left side, BERT4Rec can attend to items on both sides, which suggests that bidirectional modeling is essential and beneficial for user behavior sequence modeling.

In further studies, the impact of hyperparameters is examined such as hidden dimensionality (d), mask proportion (ρ), and maximum sequence length (N) on the model's performance. It will analyze one hyperparameter at a time while keeping the remaining hyperparameters at their optimal settings. Due to space limitations, it will only report NDCG@10 and HR@10 for these follow-up experiments.

4.5 Impact of Hidden Dimensionality

A study is conducted to investigate how the hidden dimensionality (d) affects the recommendation performance of neural sequential methods. Firstly, it has been noticed that the performance of each model tends to converge as the dimensionality increases. However, a larger hidden dimensionality does not necessarily result in better model performance, particularly on sparse datasets such as Beauty and Steam, this phenomenon may be attributed to overfitting. Furthermore, it has been observed that Caser exhibited unstable performance on four datasets, which could limit its usefulness. On the other hand, self-attention-based methods such as SASRec and BERT4Rec consistently achieved superior performance on all datasets.

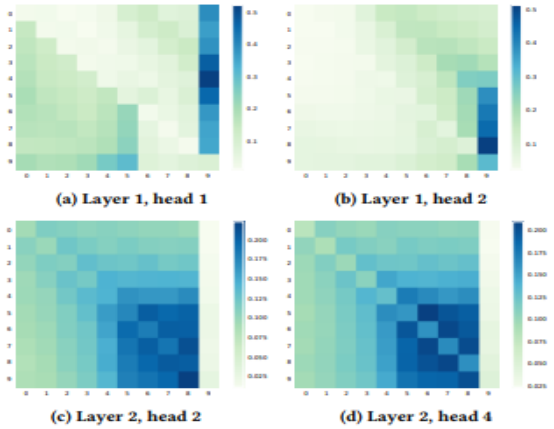

Figure1. Visual representations of average attention weights on Beauty dataset, with the last position denoted as "[mask]" (viewed best in color)

4.6 Impact of Mask Proportion ρ

The proportion of masked items (denoted as ρ) during model training is a crucial factor that directly impacts the loss function. It is important to strike a balance with ρ, as using an excessively small value may not provide enough information for the model to learn effectively, while using an overly large value could make training difficult due to the need to predict too many items based on limited context. To investigate this, conducting experiments to evaluate the effect of varying ρ on recommendation performance across different datasets. The results reveal a general pattern, where performance decreases as ρ increases beyond 0.6 in all datasets. Notably, the performances of ρ = 0.2 consistently outperform those of ρ = 0.1 in all datasets, confirming the earlier claim. Additionally, it has been observed that the optimal ρ value is highly dependent on the sequence length of the dataset. For datasets with short sequences (e.g., Beauty and Steam), the best performances are achieved with ρ = 0.6 (Beauty) and ρ = 0.4 (Steam), whereas datasets with long sequences (e.g., ML-1m and ML-20m) tend to perform better with a smaller ρ value of 0.2. This is reasonable because in datasets with longer sequences, a larger ρ would result in a higher number of items that need to be predicted. For instance, with ρ = 0.6, 98=⌊163.5×0.6⌋ items on average per sequence are required to be predicted for ML-1m, whereas for Beauty, it would be only 5=⌊8.8×0.6⌋ items. The former would be more challenging for model training.

Table 4. The performance results of different maximum lengths (N) on the model's performance

		10	20	30	40	50
BEAUTY	#samples/s	5504	3256	2284	1776	1441
	HR@10	0.3006	0.3061	0.3057	0.3054	0.3047
	NDCG@10	0.1826	0.1875	0.1837	0.1833	1832

		10	50	100	200	400

ML-1m	**#samples/s**	14255	8890	5711	2918	1213
	HR@10	0.6788	0.6854	0.6947	0.6955	0.6898
	NDCG@10	0.4631	0.4743	0.4758	0.4759	0.4715

Table 5. Ablation analysis of NDCG@10 on four datasets, with bold indicating improved performance and ↓ indicating a drop of more than 10% compared to the default version

Architecture	Dataset			
	Beauty	**Steam**	**ML-1m**	**ML-20m**
	0.1832	0.2241	0.4759	0.4513
w/Ope	0.1741	0.2060	0.2155↓	0.2867↓
w/Opffn	0.1803	0.2137	0.4544	0.4296
w/o LN	0.1642↓	0.2058	0.4334	0.4186
w/o RC	0.1619↓	0.2193	0.4643	0.4483
w/o Dropout	0.1658	0.2185	0.4553	0.4471
1 layer (L=1)	0.1782	0.2122	0.4412	0.4238
3 layer(L=3)	**0.1859**	**0.2262**	**0.4864**	**0.4661**
4 layers (L= 4)	**0.1834**	**0.2279**	**0.4898**	**0.4732**
1 head (h=1)	**0.1853**	0.2187	0.4568	0.4402
4 head (h=4)	0.1830	**0.2245**	**0.4770**	**0.4520**
8 heads (h=8)	0.1823	**0.2248**	0.4743	**0.4550**

4.7 Impact of Maximum Sequence Length N

It has also been examined the impact of the maximum sequence length (denoted as N) on the recommendation performance and efficiency of the model. Table 4 presents the results of recommendation performances and training speed with different N values on the Beauty and ML-1m datasets.

It has been found that the optimal N value is also closely tied to the average sequence length of the dataset. For example, Beauty dataset performs best with a smaller N value of 20, while ML-1m dataset achieves optimal performance with N = 200. This suggests that user behavior in short sequence datasets is influenced by more recent items, while in long sequence datasets, less recent items play a role. It should be noted that the model does not consistently benefit from a larger N, as a larger N can introduce more noise along with extra information. However, the given model remains stable as N increases, indicating that it can effectively attend to informative items from noisy historical records.

One scalability concern of BERT4Rec is its computational complexity per layer, which is O(n^2d), where n is the sequence length and d is the hidden dimension. Fortunately, the results in Table 4 demonstrate that the self-attention layer can be effectively parallelized using GPUs, mitigating this concern.

4.8 Ablation Study

Finally, it has been conducted ablation experiments on several key components of BERT4Rec to gain a better understanding of their impacts. These components include positional embedding (PE), position-wise feed-forward network (PFFN), layer normalization (LN), residual connection (RC), dropout, the number of self-attention layers (L), and the number of heads in multi-head attention (h). Table 5 presents the results of the default version (L=2, h=2) and its eleven variants on all four datasets, with a dimensionality of d=64, while keeping other hyperparameters at their optimal settings.

5. CONCLUSION AND FUTURE WORK

Proposing BERT4Rec, a deep bidirectional sequential model for sequential recommendation, leveraging the success of deep bidirectional self-attention architecture in language understanding. The given model incorporates a Cloze task for masked item prediction using both left and right context during training. Through extensive experiments on four real-world datasets, demonstrating that BERT4Rec outperforms state-of-the-art baselines in sequential recommendation tasks. Looking forward, there are exciting avenues for future research that can further enhance the efficacy of BERT4Rec. One promising direction involves exploration of integration of rich item features, such as category and price, to provide a more comprehensive understanding of items and their relationships. Additionally, there is a need to delve deeper into explicit user modelling, particularly in scenarios where users exhibit multi-session behavior. Incorporating a user component into the model architecture could contribute to a more content-aware and personalized sequential recommendation system.

6. REFERENCES

[1] Akhtyamova, L., 2020, April. Named entity recognition in Spanish biomedical literature: Short review and BERT model. In 2020 26th Conference of Open Innovations Association (FRUCT) (pp. 1-7). IEEE. https://arrow.tudublin.ie/cgi/viewcontent.cgi?article=1016&context=ittscicon

[2] Balázs Hidasi and Alexandros Karatzoglou. 2018. Recurrent Neural Networks with Top-k Gains for Session-based Recommendations. In Proceedings of CIKM. ACM, New York, NY, USA, 843–852.

[3] Ciniselli, M., Cooper, N., Pascarella, L., Poshyvanyk, D., Di Penta, M. and Bavota, G., 2021, May. An empirical study on the usage of BERT models for code completion. In 2021 IEEE/ACM 18th International Conference on Mining Software Repositories (MSR) (pp. 108-119). IEEE. https://arxiv.org/pdf/2103.07115

[4] F. Maxwell Harper and Joseph A. Konstan. 2015. The MovieLens Datasets: History and Context. ACM Trans. Interact. Intell. Syst. 5, 4, Article 19 (Dec. 2015), 19 pages.

[5] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network. In Deep Learning and Representation Learning Workshop.

[6] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep Residual Learning for Image Recognition. In Proceedings of CVPR. 770–778.

[7] Lee, J.S. and Hsiang, J., 2019. Patentbert: Patent classification with fine-tuning a pre-trained bert model. arXiv preprint arXiv:1906.02124. https://arxiv.org/pdf/1906.02124

[8] Lin, J., Liu, Y., Zeng, Q., Jiang, M. and Cleland-Huang, J., 2021, May. Traceability transformed: Generating more accurate links with pre-trained bert models. In 2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE) (pp. 324-335). IEEE. https://arxiv.org/pdf/2102.04411

[9] Lu, W., Jiao, J. and Zhang, R., 2020, October. Twinbert: Distilling knowledge to twin-structured compressed bert models for large-scale retrieval. In Proceedings of the 29th ACM International Conference on Information & Knowledge Management (pp. 2645-2652). https://arxiv.org/pdf/2002.06275

[10] Mozafari, M., Farahbakhsh, R. and Crespi, N., 2020. Hate speech detection and racial bias mitigation in social media based on BERT model. PloS one, 15(8), p.e0237861. https://doi.org/10.1371/journal.pone.0237861

[11] Nagy, A., Bial, B. and Ács, J., 2021. Automatic punctuation restoration with BERT models. arXiv preprint arXiv:2101.07343. https://arxiv.org/pdf/2101.07343

[12] Nozza, D., Bianchi, F. and Hovy, D., 2020. What the [mask]? making sense of language-specific BERT models. arXiv preprint arXiv:2003.02912. https://arxiv.org/pdf/2003.02912

[13] Petrov, A. and Macdonald, C., 2022, September. A Systematic Review and Replicability Study of BERT4Rec for Sequential Recommendation. In Proceedings of the 16th ACM Conference on Recommender Systems (pp. 436-447). https://arxiv.org/pdf/2207.07483

[14] Qiao, Y., Zhu, X. and Gong, H., 2022. BERT-Kcr: prediction of lysine crotonylation sites by a transfer learning method with pre-trained BERT models. Bioinformatics, 38(3), pp.648-654. http://structpred.life.tsinghua.edu.cn/pdf/10.1093_bioinformatics_btab712.pdf

[15] Risch, J. and Krestel, R., 2020, May. Bagging BERT models for robust aggression identification. In Proceedings of the Second Workshop on Trolling, Aggression and Cyberbullying (pp. 55-61). https://aclanthology.org/2020.trac-1.9.pdf

[16] Rogers, A., Kovaleva, O. and Rumshisky, A., 2021. A primer in BERTology: What we know about how BERT works. Transactions of the Association for Computational Linguistics, 8, pp.842-866. https://direct.mit.edu/tacl/article-pdf/doi/10.1162/tacl_a_00349/1923281/tacl_a_00349.pdf

[17] Ruining He and Julian McAuley. 2016. Fusing Similarity Models with Markov Chains for Sparse Sequential Recommendation. In Proceedings of ICDM. 191–200.

[18] Ruining He, Wang-Cheng Kang, and Julian McAuley. 2017. Translation-based Recommendation. In Proceedings of RecSys. ACM, New York, NY, USA, 161–169.

[19] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. Neural Computation 9, 8 (Nov. 1997), 1735–1780.

[20] Shi, P. and Lin, J., 2019. Simple bert models for relation extraction and semantic role labeling. arXiv preprint arXiv:1904.05255. https://arxiv.org/pdf/1904.05255

[21] Sun, F., Liu, J., Wu, J., Pei, C., Lin, X., Ou, W. and Jiang, P., 2019, November. 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). https://arxiv.org/pdf/1904.06690.pdf%EF%BC%89

[22] Sun, S., Cheng, Y., Gan, Z. and Liu, J., 2019. Patient knowledge distillation for bert model compression. arXiv preprint arXiv:1908.09355. https://arxiv.org/pdf/1908.09355

[23] Tsai, H., Riesa, J., Johnson, M., Arivazhagan, N., Li, X. and Archer, A., 2019. Small and practical BERT models for sequence labeling. arXiv preprint arXiv:1909.00100. https://arxiv.org/pdf/1909.00100

[24] Wang, Z., Ng, P., Ma, X., Nallapati, R. and Xiang, B., 2019. Multi-passage bert: A globally normalized bert model for open-domain question answering. arXiv preprint arXiv:1908.08167. https://arxiv.org/pdf/1908.08167.pdf)

[25] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural Collaborative Filtering. In Proceedings of WWW. 173–182.