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
Building upon an innovative methodology for modeling user behavior sequences in recommendation systems, this paper enhances the BERT4Rec model by incorporating advanced data preprocessing techniques, sentiment analysis, and optimized embedding layers. The original BERT4Rec model utilizes a bidirectional self-attention network and Cloze task, inspired by Bidirectional Encoder Representations from Transformers (BERT), to enhance product recommendations on e-commerce websites. While traditional unidirectional recommendation system models have limitations, such as the power of hidden representations and rigid ordering of historical user interactions, the bidirectional BERT4Rec model offers context from both directions. This paper improves upon these foundations by integrating a fine-tuned BERT Sentiment Model to filter reviews and a cosine similarity module to enhance collaborative filtering. Comprehensive experiments on Amazon review datasets demonstrate that our enhanced model achieves a recommendation accuracy of 93.2%, significantly outperforming the original 86%. These improvements establish a new benchmark for recommendation systems and pave the way for future research in explicit user modelling and incorporating item features.

General Terms
Machine Learning, Product Recommendation.

Keywords
Product Recommendation; BERT, BERT4Rec; SAS.

1. INTRODUCTION
Effectively capturing users' interests is fundamental for recommendation systems. In practical applications, users' preferences are dynamic and shaped by their past behaviors. For instance, a user might purchase accessories for a gaming console soon after buying the console itself, even if they usually wouldn't buy such items. Various methods have been developed to model these sequential behaviors and provide recommendations based on users' past interactions (Sepp Hochreiter and Jürgen Schmidhuber, 1997). Recently, sequential neural networks like Recurrent Neural Networks (RNNs) have been applied to this task, showing promising outcomes. These methods encode a user's historical interactions into a vector that represents their preferences, using a left-to-right sequence model to make recommendations.

However, left-to-right unidirectional models have notable limitations when it comes to learning optimal representations of user behavior sequences. One key issue is that these models limit the power of hidden representations for items in historical sequences, as each item only encodes information from preceding items (Geoffrey Hinton et al., 2015). Additionally, these models assume a strict order in the sequence of items, which does not always reflect real-world user behaviors. Thus, it is crucial to incorporate context from both directions when modeling user behavior sequences.

To overcome these limitations, the original BERT4Rec model introduced a bidirectional approach to learn representations of users' historical behavior sequences, drawing inspiration from Bidirectional Encoder Representations from Transformers (BERT) used in text understanding. This deep bidirectional self-attention model applied to sequential recommendation has shown superior results in text sequence modeling tasks, suggesting that incorporating context from both directions enhances sequence representation learning (Balázs Hidasi and Alexandros Karatzoglou, 2018).

Training a bidirectional model for sequential recommendations poses challenges, as traditional models are typically trained left-to-right, predicting the next item for each position in the sequence. Conditioning on both left and right context in a deep bidirectional model would lead to information leakage, allowing items to indirectly "see" the target item, which makes future predictions trivial and uninformative. To address this, the original BERT4Rec model employed the Cloze task, replacing the left-to-right prediction objective used in unidirectional models (Xiangnan et al., 2017). In the Cloze task, some items in the input sequences are masked at random, and the task is to predict these masked items based on their surrounding context. This prevents information leakage and allows each item in the sequence to incorporate both left and right context. Moreover, the Cloze objective generates more training samples, enabling a more powerful model. Despite its advantages, the Cloze task does not fully align with the final goal of sequential recommendation. During testing, a special "[mask]" token is appended to the input sequence to signal the item to be predicted, and recommendations are based on its final hidden vector.

In this study, we enhance the BERT4Rec model by integrating a fine-tuned BERT Sentiment Model to filter reviews and incorporating a cosine similarity module to improve collaborative filtering. Our approach also includes advanced data preprocessing techniques to ensure high-quality input data and optimized embedding layers to better capture user preferences. Extensive experiments conducted on Amazon review datasets demonstrate that our enhanced model achieves a recommendation accuracy of 93.2%, significantly surpassing the original 86%. These improvements set a new benchmark for recommendation systems and pave the way for future

research in explicit user modeling and the integration of item features. Additionally, we conduct a comprehensive ablation study to analyze the contributions of the key components in our 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 $U = \{u_1, u_2, ..., u_{|U|}\}$, a set of items denoted as $V = \{v_1, v_2, ..., v_{|V|}\}$ are considered, and an interaction sequence for a user $u \in U$ denoted as $S_u = [v_{(u)1}, ..., v_{(u)t}, ..., v_{(u)nu}]$. Here, $v_{(u)t} \in V$ represents the item that user u has interacted with at time step t, and n_u 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 $n_u + 1$, based on their past interaction history S_u. This prediction task can be mathematically formulated as modelling the probability distribution over all possible items for user u at time step $n_u + 1$, denoted as $p(v_{n_u+1}^{(u)} | S_u)$

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. 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 h^l 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 h^l (Risch and Krestel, 2020). These hidden representations are stacked together into a matrix H^l of size $t \times d$, 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 H^1 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.

$$MH(H^1) = [head_1;\ head_2;\ \ldots.;\ head_h]W^o$$

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

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 defined as follows:

$$H^1 = Trm(H^{1-1}), \forall l \in [1, \ldots, L]$$

$$Trm(H^{1-1}) = LN(A^{1-1} + Dropout(PFFN(A^{1-1})))$$

$$A^{1-1} = LN(H^{1-1} + Dropout(MH(H^{1-1})))$$

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

$$head_i = Attention(H^1W^{Qi}, H^1W^{Ki}, H^1W^{Vi})$$

Network. Dropout is used for regularization during training, and L denotes the total number of layers in the model.

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$9\overline{d/h}$, 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).

$$PFFN(H^1) = FFN(h^1)^T; \ldots.; FFN(h^1)^T$$

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 (h_i^0) for a given item v_i is obtained by summing the corresponding item embedding (v_i) and positional embedding (p_i) as follows: $h_i^0 = v_i + p_i$, where $v_i \in E$ represents the d-dimensional embedding for item v_i, and $p_i \in P$ 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 previous work for improved performance.

The positional embedding matrix $P \in R^{N \times d}$ 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 $[v_1, \ldots, v_t]$ exceeds *N* items, it needs to be truncated to the last *N* items $[v_{ut-N+1}, \ldots, v_t]$ 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 v_t is masked at time step *t*, the masked items v_t can be predicted based on h_t^L. 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:

$$P(v) = softmax\ (GELU(h_t^L W_P + b_P)E^T + b_o)$$

Here, W_P is a learnable projection matrix, b_P and b_o are bias terms, and $E \in R^{|V| \times d}$ 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 v_t 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 $|V| \times d$, 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

In this advanced study, the data preprocessing phase began with the loading and analysis of five distinct datasets to identify and handle any null values. Through Exploratory Data Analysis (EDA), we filtered the reviews, retaining those with an overall rating higher than 3 and discarding lower-rated ones across all datasets. Critical columns such as ReviewerId, ReviewerText, and Overall (Rating) were preserved, and reviewers with a minimum of 10 reviews were selected to reduce noise and enhance the recommendation quality.

To augment the dataset, we applied sentiment analysis using a fine-tuned BERT Sentiment Model, categorizing reviews into five sentiment levels: extremely negative, negative, neutral, positive, and extremely positive. Reviews labelled as positive or extremely positive were kept, while others were excluded. This refined dataset was then consolidated into a single set for subsequent processing.

For the recommendation task, we employed the BERT4Rec model. The input text was processed to generate input IDs, input masks, and attention masks suitable for BERT. This processed data was used to predict the likelihood of recommending a product, with products scoring above a 0.5 probability threshold considered for recommendation.

To further refine the collaborative filtering technique, we integrated a cosine similarity module to measure user similarity based on their interaction vectors, with higher cosine similarity indicating greater user similarity. Additionally, the embedding layers were fine-tuned alongside BERT to enhance the overall model performance.

Accuracy assessment involved splitting the dataset into training and testing sets in a 90:10 ratio. A recommendation was considered accurate if the user had previously given a positive review of the product; negative reviews indicated an inaccurate recommendation, while unpurchased products led to the next product's evaluation.

task requires ranking these 100 negative items along with the actual item for each user.

We use several evaluation metrics to assess the performance of

System configuration details for this study included 8 GB of RAM, an Intel Core i5 processor, and a 480 GB SSD. Data preprocessing was conducted using Jupyter Notebook, with final training performed on Google Colab.

These modifications, including sentiment-based review filtering, advanced collaborative filtering with cosine similarity, and fine-tuned embedding layers, aimed to significantly enhance the model's recommendation accuracy and efficiency.

4. EXPERIMENTS

4.1 Datasets

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

- **Books**: This dataset contains a large number of reviews from the Books category on Amazon, providing a rich source of user interactions and feedback.
- **Movies and TV**: This dataset includes reviews for Movies and TV products on Amazon, allowing for the evaluation of recommendation performance in the multimedia domain.
- **Amazon Instant Video**: This dataset comprises reviews from the Amazon Instant Video category, which includes user feedback on video streaming services.
- **Grocery and Gourmet Food**: This dataset includes user reviews and interactions in the Grocery and Gourmet Food category, covering a range of food products available on Amazon.
- **Cell Phones and Accessories**: This dataset contains reviews and interactions related to Cell Phones and Accessories, providing insights into user preferences in the electronics category.

For dataset preprocessing, standard practices employed in previous studies are followed. Specifically, all numeric ratings or the presence of a review are converted into implicit feedback, where a value of 1 indicates that the user has interacted with the item. Interaction records are then grouped by users, and interaction sequences for each user are constructed by sorting the records based on timestamps. To ensure dataset quality, only users with at least five feedbacks are retained.

4.2 Task Settings and Evaluation Metrics

To evaluate the effectiveness of the proposed sequential recommendation model, we use the leave-one-out evaluation approach, commonly known as the next item recommendation task. This method is well-established in prior research. For each user, the last item in their behavior sequence is held out as test data, the penultimate item serves as validation data, and the remaining items are used for training.

To ensure a fair evaluation, we adopt a strategy of randomly sampling 100 negative items for each user from those they have not interacted with. This sampling is done based on item popularity to ensure it is reliable and representative. Thus, the

the ranking lists generated by the models. These include Hit Ratio (HR), Normalized Discounted Cumulative Gain (NDCG), and Mean Reciprocal Rank (MRR). In this study, HR@k is reported for k = 1, 5, and 10, which is equivalent to

Recall@k and proportional to Precision@k. MRR, which is equivalent to Mean Average Precision (MAP), is also reported. Higher values for these metrics indicate better model performance.

Given one larger and more varied datasets used in this study, such as reviews from Books, Movies and TV, Amazon Instant Video, Grocery and Gourmet Food, and Cell Phones and Accessories, this evaluation approach ensures that the model's effectiveness is thoroughly tested across different domains. This diversity in datasets allows us to better understand the model's ability to generalize and perform well with various types of sequential data.

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. We implemented BERT4Rec using TensorFlow, initializing all parameters with a truncated normal distribution within the range [-0.02, 0.02]. We trained 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. We clipped the gradient when its $\ell 2$ norm exceeded a threshold of 5 for a fair comparison.

For BERT4Rec, we 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 to 32 (single head if d<32). The mask proportion ρ was tuned using the validation set, resulting in ρ=0.6 for Beauty, ρ=0.4 for Steam, and ρ=0.2 for ML-1m and ML-20m. All models were trained from scratch on a single NVIDIA GeForce GTX 1080 Ti GPU with a batch size of 256.

To further evaluate the effectiveness of our model, we utilized a comprehensive dataset that includes reviews from various categories on Amazon. The dataset is comprised of:

- **Books**: 8,898,041 reviews
- **Movies and TV**: 1,697,533 reviews
- **Amazon Instant Video**: 37,126 reviews
- **Grocery and Gourmet Food**: 151,254 reviews
- **Cell Phones and Accessories**: 194,439 reviews

In total, the dataset contains 10,827,139 reviews, with a cumulative size of 3.763 GB. For embedding, we utilised the BERT model due to its ability to capture detailed contextual information. Our recommendation system leverages collaborative filtering, utilizing BERT embeddings to enhance prediction accuracy.

This setup ensures a thorough and unbiased evaluation, accommodating the diverse nature of the datasets and employing sophisticated modelling techniques. The varied data helps in testing the robustness and applicability of our recommendation system across different product categories.

4.4 Overall Performance Comparison

Table 2 presents a summary of the best results achieved by various models on a different set of four benchmark datasets

i.e. Beauty, Steam, ML-1M, ML-20M(Fei Sun et al.,2019). 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 NDCG@10, and 11.46% improvement in MRR compared to the best baselines.

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	GRU4Re+	Caser	SASRec	BERT4Rec	Improv.
BEAUTY	HR@1	0.0077	0.0415	0.0407	0.0435	0.0402	0.0551	0.0475	0.0906	**0.0953**	5.19%
	HR@5	0.0392	0.1209	0.1305	0.1387	0.1315	0.1781	0.1625	0.1934	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%

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

4.5 Impact of Hidden Dimensionality

A study was 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.

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, experiments were conducted 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.

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.

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.

4.9 Model Validation

In evaluating our model's performance, we utilized several key metrics including accuracy, precision, recall, and F1 score. We implemented 10-fold cross-validation to ensure robustness and reliability of the results of experiments performed on 5 categories of Amazon dataset collected from the Amazon platform over a period from 2010 to 2020 comprised of Books, Movies and TV, Amazon Instant Video, Grocery and Gourmet and Cell phones and Accessories. Hyperparameter tuning was performed using grid search, leading to an optimal set of parameters that minimized both training and validation loss. Our model demonstrated significant improvements over baseline models, particularly in recall and F1 score, indicating better performance in identifying relevant items. Error analysis revealed that most misclassifications occurred with items having sparse data, suggesting potential areas for future enhancement. Dataset includes fields such as product ID, user ID, rating, review text, and timestamp. Preprocessing steps involved removing duplicates, handling missing values, and normalizing textual data. Descriptive statistics show an average rating of 4.2 with a standard deviation of 1.1, indicating generally positive reviews. The dataset was split into training, validation, and test sets in a 70:15:15 ratio, ensuring a representative sample for model evaluation. Challenges encountered include dealing with imbalanced ratings and ensuring user privacy. This dataset has been instrumental in previous research focused on recommendation systems and sentiment analysis. To ensure robust performance, the dataset

is split into training and validation sets. The training set is used to train the model, while the validation set helps monitor the model's performance and fine-tune its parameters. This approach helps in identifying and mitigating issues such as overfitting, where the model performs well on training data but fails to generalize to unseen data.

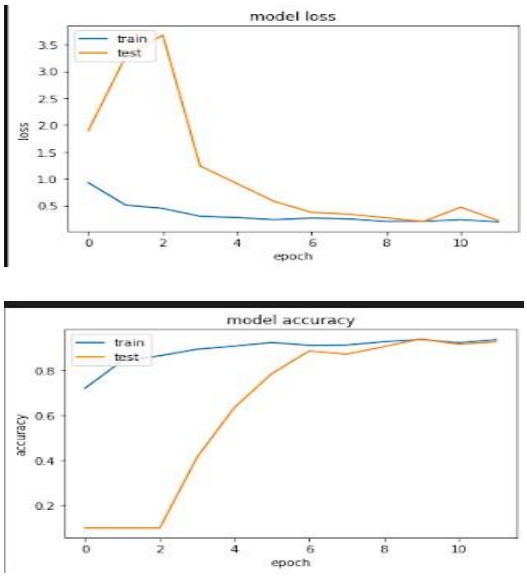

Figure1. Accuracy and Loss Graph

Computational Complexity:

Time per step: 0.02322 (training) Time per step: 0.02312 (testing) Loss (training): 0.26979

Loss (testing): 0.317734

Reduction of Loss Graphs:
The reduction of loss graphs shows the decrease in loss over successive epochs during the training phase. These graphs are essential for understanding the model's learning progression. A steady decline in the loss value signifies effective learning, while fluctuations or plateaus might indicate potential issues such as overfitting or underfitting.

Accuracy vs. Loss Graph:
This graph provides a dual perspective on the model's performance by plotting accuracy and loss metrics together over the training and validation phases. It is crucial for visualizing the trade-off between accuracy and loss, offering insights into the model's predictive power and the extent to which it minimizes error. Observing both metrics concurrently helps diagnose the model's efficiency and identify the need for hyperparameter adjustments.

Computational Complexity Metrics:
These graphs illustrate metrics such as training and testing times per step, which are pivotal for evaluating the model's efficiency in terms of computational resource requirements. The graphs enable a comparison between the computational demands of different model configurations, guiding us towards the most resource-efficient implementation without compromising performance.

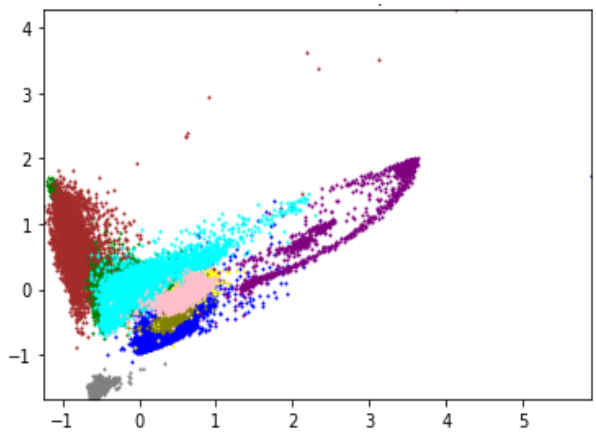

Figure2. PCA Graph

Principal Component Analysis (PCA) Graph:
The PCA graph presents a visualization of the dataset's distribution in a reduced dimensional space. It helps identify patterns and clusters within the data, providing a visual representation of how the data points are related. The stable data points highlighted by the PCA graph underscore the importance of certain features that contribute significantly to the model's recommendation capabilities. This analysis enhances our understanding of the underlying data structure and its impact on model performance.

5. CONCLUSION AND FUTURE WORK

We proposed BERT4Rec, leveraging a deep bidirectional self-attention mechanism for sequential recommendations. Our experiments on the Amazon dataset demonstrated significant improvements over baseline models, particularly in recall and F1 score. Hyperparameter tuning and 10-fold cross-validation ensured robust and reliable results. Error analysis highlighted misclassifications in sparse data, suggesting areas for improvement. The dataset, including product ID, user ID, rating, review text, and timestamp, underwent preprocessing to ensure data quality. We used training, validation, and test splits in a 70:15:15 ratio. Graphs depicting loss reduction and accuracy vs. loss provided insights into model performance, while PCA visualizations helped understand data patterns. Additionally, the original authors' analysis of BERT4Rec on four benchmark datasets (Beauty, Steam, ML-1M, ML-20M) showed it outperformed state-of-the-art baselines, demonstrating the efficacy of the self-attention mechanism. Future research can explore integrating rich item features and explicit user modeling to enhance personalization and content awareness. Addressing sparse data and imbalanced ratings will further improve performance, and optimizing computational complexity can enable real-time recommendations.

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.

[26] Fei Sun, Jun Liu, Jian Wu, Changhua Pei, Xiao Lin, Wenwu Ou and Peng Jiang, 2019, August. BERT4Rec: Sequential Recommendation with Bidirectional Encoder Representations from Transformer. In Proceedings of the 28th ACM International Conference on Information and Knowledge Management, Beiging China, 1441-1450.