Received April 23, 2022, accepted May 17, 2022, date of publication May 23, 2022, date of current version June 1, 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3177610

HybridBERT4Rec: A Hybrid (Content-Based Filtering and Collaborative Filtering) Recommender System Based on BERT

CHANAPA CHANNARONG[1], **CHAWISA PAOSIRIKUL**[1], **SARANYA MANEEROJ**[1], **AND ATSUHIRO TAKASU**[2]

[1]Department of Mathematics and Computer Science, Faculty of Science, Chulalongkorn University, Bangkok 10330, Thailand
[2]National Institute of Informatics, The Graduate University for Advanced Studies, SOKENDAI, Tokyo 101-8430, Japan

Corresponding author: Saranya Maneeroj (saranya.m@chula.ac.th)

ABSTRACT Because a user's behavior depends mainly on the user's current interests, which can change over time, the sequential recommendation approach has become more prevalent in recommender systems. Many methods have been proposed that aim to model sequential recommendations. One of these is BERT4Rec, which applies the bidirectional-encoder-representations-from-transformers (BERT) technique to model user behavior sequences by considering the target user's historical data, i.e., a content-based filtering (CBF) approach. Despite BERT4Rec's effectiveness, we argue that considering only this historical data is insufficient to provide the most accurate recommendation. We believe that if BERT were to consider other users' interactions in its analysis, it would increase the model accuracy. Therefore, we propose a new method called HybridBERT4Rec, which applies BERT to both CBF and collaborative filtering (CF). For CBF, we want to extract the characteristics of the target user's interactions with purchased items. (We implement this in the same way as in BERT4Rec, with our model generating a target user profile.) For CF, we want to find neighboring users who are similar to the target user. Here, we extract the target item's characteristics using all other users who rated the target item as a second input to BERT. This generates a target item profile. After obtaining both profiles, we use them to predict a rating score. We experimented with three datasets, finding that our model was more accurate than the original BERT4Rec.

INDEX TERMS Recommender system, sequential recommendation, hybrid recommendation, BERT.

I. INTRODUCTION

Nowadays, recommender systems play an essential role in our daily lives via many websites and applications such as YouTube, Facebook, Netflix, and Amazon, and they help users make decisions according to their preferences.

Most recommendations are based on traditional methods such as collaborative filtering (CF), which recommends items to users based on their historical preferences, content-based filtering (CBF) [1], which recommends items to users based on item characteristics and user preferences, or hybrid filtering, which combines CF and CBF to obtain richer recommendations. These approaches assume that all interactions between users and items in the users' historical interactions are equally important and they therefore employ all available historical data in calculating user preferences.

The associate editor coordinating the review of this manuscript and approving it for publication was Fabrizio Messina.

However, modeling user preferences in a static way like traditional methods is insufficient in practice since it can only represent the users' general preferences. Moreover, users' preferences do change over time. For example, as shown in Figure 1, Alice usually watches romantic movies, but for the latest movie, she chose ''Spider-Man: No Way Home'' because she had previously watched ''Spider-Man: Far From Home,'' rather than romantic movies as usual. Therefore, the *sequencing* of users' historical interactions is now becoming widespread in recommender systems to represent the

FIGURE 1. ***Sequencing*** **of users' historical interactions.**

current and recent preferences of a user for more accurate recommendations.

Many methods have been proposed recently for modeling sequential recommendations based on users' historical interactions and most involve recurrent neural networks (RNNs) [2]. For example, the dynamic recurrent basket model (DREAM) [3] was introduced to model the "next basket" recommendation, which involves learning a representation of a user's interests at different times and capturing global sequential features among the users' baskets over time. An input example of this model is a "basket sequence," with each basket containing a variety of items. To learn a dynamic representation of a user, DREAM generates the latent vector representation of a basket by aggregating the representation vectors of items in the basket, using a latent vector representation as the input layer of a recurrent architecture. Finally, the model calculates the user's scores for all items over time to indicate if the user is more likely to purchase the corresponding item.

The multiview RNN (MV-RNN) model [4] was introduced to alleviate the item "cold-start" problem, which uses a representation of items with multiview features as an input. These features contain both indirectly observable (latent) features and directly observable (images and text description) features, which can alleviate the item cold-start problem. There are three approaches to combining multiview features, namely feature concatenation, feature fusion, and multimodal marginalized denoising autoencoding. [26] MV-RNN then applies the recurrent structure to capture a user's interests and build a user representation. Finally, the model is trained using a Bayesian-personalized-ranking framework and the back-propagation-through-time algorithm to calculate the difference in user preferences at every time step. The limitations of these RNN-based models include gradient-vanishing and exploding problems and that RNN only trains left to right.

Other previous work [5] on sequential recommendations adopted the deep bidirectional self-attention (BERT) model [6] for predicting the next item with which users would be likely to interact. Given the historical interaction and using applied-masked-language modeling [7] randomly on the items, the masked items are predicted from the characteristics of the surrounding items. BERT-based models use only the historical data for a target user without considering other users' interaction with the target item (the CBF approach). Despite these models' effectiveness, we argue that considering only historical data is insufficient to provide the most accurate recommendations. Accuracy would be improved if BERT were extended to include a CF-based element.

In addition to the CBF approach used in previous models, we introduce user-based CF by considering other users in the system who have interacted with the target item. Instead of simply attending to the target users' *item sequences*, we are also interested in *user sequence* ratings for the target item. Therefore, we apply attention mechanisms in BERT by supplying another input, namely the user sequence for the target item. The user sequence comprises all those users who have interacted with the target item. After constructing the input as an item sequence (the CBF part) and a user sequence (the CF part), our proposed system **HybridBERT4Rec** generates both the target user profile and the target item profile. The target user profile comprises the similarity of every pair of consecutive *items* in the historical sequence of the target user. (Consequently, it provides information about whether the next item is likely to be the target item.) The target item profile comprises the similarity of every pair of consecutive *users*, indicating who the target user's "neighbors" are likely to be. In the prediction state, we utilize the obtained target user profile and target item profile to calculate the target user's rating score with respect to the target item.

The contributions of our paper are as follows.

- We propose **HybridBERT4Rec**, a hybrid (CBF and CF) recommender system that extends the original BERT-based models.
- We show empirically that the proposed model outperforms a traditional model (BERT4Rec [5]).

II. RELATED WORK

In this section, we briefly review several studies related to our model, including Hybrid recommendation, neural CF and neural CBF.

A. HYBRID RECOMMENDATION

A hybrid recommendation system is a type of recommendation system that combines two or more recommendation strategies in various ways to make use of their complementing benefits. Traditional hybrid recommendations are a combination of collaborative filtering (CF), which recommends items to users based on the preferences of other similar users or neighbors, and content-based filtering (CBF), which recommends items to users based on similar item characteristics with which users interacted. Different hybrid recommendation systems have emerged in several fields. For example, [8] proposes a hybrid approach that combines movie ratings with textual information, which are tags and genres of movies to perform, movie recommendations; or [9], introduces a hybrid approach for learning material recommendations by using learner profiles (users), learning materials (items), and good learner ratings. However, in traditional hybrid approaches, they cannot capture latent user preferences, hence neural-based approaches are introduced. Aside from that, most neural-based recommendation techniques are either CBF-based or CF-based. Both CF and CBF are not incorporated in neural recommendations.

B. NEURAL CF

Most research on recommender systems has adopted a CF approach to recommending items based on users' historical preferences. One popular CF approach is matrix factorization (MF) [10], which generates a prediction rating by factoring the user–item rating matrix and representing the user and item as a vector of latent features. The inner product between the user and item latent vectors is then calculated to learn the user's preference for the item.

Another approach uses neural CF (NCF) [11], which aims to model user preferences by replacing the inner product with a multilayer perceptron (MLP). The rating of user u for item i ($\widehat{r}_{u,i}$) is then calculated as

$$\widehat{r}_{u,i} = f(U^T v_u, I^T v_i | \, U, I, \theta_f) \tag{1}$$

where $U \in \mathbb{R}^{M \times K}$ and $I \in \mathbb{R}^{N \times K}$ denote the K latent factor matrices for the users and items, respectively. v_u and v_i are vectors with "one-hot" encoding for identifying a user and an item, respectively. Function f is a multilayer neural network (see (2)) and θ_f is the model parameter for function f

$$\begin{aligned} f(U^T v_u, I^T v_i | \, U, I, \theta_f) \\ = \phi_{out}(\phi_Z(\ldots \phi_2(\phi_1(U^T v_u, I^T v_i) \ldots)) \end{aligned} \tag{2}$$

where ϕ_{out} and ϕ_Z are the mapping functions for the output layer and the Z-th NCF layer, respectively. The model parameters are learned by minimizing the square loss function, which is represented as

$$L = \sum w_{u,i}(r_{u,i} - \widehat{r}_{u,i})^2 \tag{3}$$

where $r_{u,i}$ is the true rating score of user u for item i and $w_{u,i}$ is the weight of training instances (u, i).

In general, the CF approach recommends the items for target users by considering other-user information. Therefore, it does not capture the relationship between items in a user's historical sequence. However, some research on recommender systems aims to solve this problem. [27]

C. NEURAL CBF

To alleviate this capture problem, that is, the relation between items in the users' historical sequences, the CBF approach aims to learn the relations between items by considering the content that describes the items. Many neural approaches have been introduced and applied to content-based scenarios. One example involves a convolutional neural network (CNN) [12] that generates local features and a pooling layer for concise representation; this approach uses an MLP in a nonlinear activation function and back-propagation for training. Another example uses an RNN [2] to learn sequential data.

This paper focuses on sequential data and we now discuss its two main associated concepts, sequential recommendation and BERT4Rec.

1) SEQUENTIAL RECOMMENDATION

As described above, As described above, most recommender systems consider the entire history of user interactions, which limits these recommender systems' accuracy. This is because the current behavior of the user depends on the user's current interests. Therefore, sequential recommendation, which considers the order of a user's historical interactions, was proposed and has become popular. Most sequential recommendation systems are based on RNNs and their variants, long short-term memory (LSTM) [13], or gated recurrent units (GRUs) [14].

The essential idea of these methods is to generate a representation by encoding users' historical records into vectors in different ways. For example, in both DREAM [3] and MV-RNN [4], the aim of the models is to produce a recommendation list of items that a user may wish to purchase. DREAM applies an RNN by feeding a representation of items obtained from users' historical records to learn a representation of a user's interests at different times. This is then used to calculate a rating score for each user for all items at each time step. Instead of using only the item identification (ID) to represent items, as in DREAM, MV-RNN uses an item representation that includes latent features (item IDs) and directly observable features (images and text description) as an input to an LSTM to capture the users' interests.

Many GRU-based models have been proposed to model user representations [15]–[18]. For example, one user-based GRU [15] integrates user information into a new type of GRU to produce an effective personalized user representation.

Several deep-learning models have been proposed for sequential recommendations, such as a sequential recommendation with user memory (RUM) network [19]. RUM applies MF to predict the rating scores for users and items. Instead of employing traditional user embedding and item embedding as MF inputs, RUM combines users' historical records and traditional user embedding to generate its user embeddings. Another example is the convolutional sequence embedding recommendation model (Caser) [20], which employs a CNN. An item embedding matrix is used to learn sequential patterns of items. The item embedding matrix is a stack of item embeddings in sequence. The sequential patterns of all interacting items are concatenated with the user embedding and fed into the CNN's fully connected layer to obtain the likelihood of the user wanting to interact with the target item at each time step.

2) BERT4Rec

BERT [6] is a successful method for understanding text that achieves state-of-the-art results in text-sequence modeling. It incorporates the encoder parts of a transformer [21] model for bidirectional training. Following the success of BERT, there are various studies incorporating BERT with recommendations [28], [29]. For example, in order to provide more accurate recommendations, [28] employs BERT to extract item embedding vectors from unstructured item description text. However, those methods only utilize BERT for processing text input. Unlike BERT4Rec model [5] which is developed for sequential recommendations (see Fig. 2), aiming to predict the next item with which the user is likely to interact. The model first creates an item sequence based on the user's historical sequence, which is a list of the items that the user has interacted with in chronological order $[v_1, v_2, \ldots, v_t]$. Every token in the item sequence is fed into the embedding layer, which includes both input embedding and position

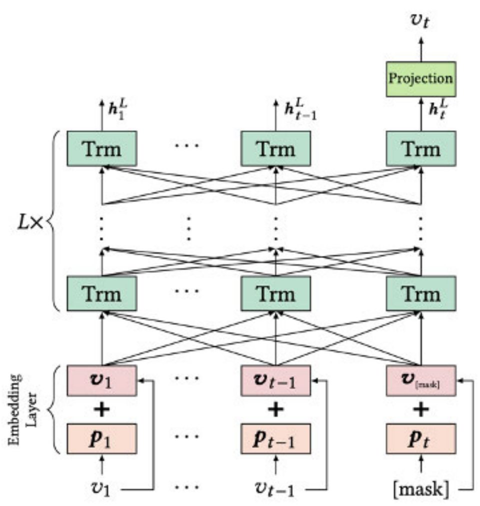


FIGURE 2. BERT4Rec model architecture for predicting the next item that the user might interact with.

embedding, to extract a representation of each token v_i. Note that these item sequences have to randomly mask the item in the sequence and replace it with a special *mask* token before feeding it into the embedding layer. After obtaining the item sequence embedding, it is fed into the BERT model to learn the similarity of every pair of items in the sequence. Finally, BERT4Rec outputs the last token, which indicates the relation or similarity of items to the users' historical sequences. This is the recommendation offered to the user.

The BERT4Rec solely predicts the next item by extracting user historical patterns in terms of the relation of rated items in target users' history data (item sequences). The user historical patterns are basically the general items' style/characteristics that target users favor. However, a limitation of this approach is that the set of recommended items for a target user is quite specific because it cannot recommend a new category of items that the target user has never seen before. It would be better if BERT could also consider other users' behavior toward a target item to increase the variety of recommended items offered to a target user, which is the CF approach. Furthermore, matrix factorization in the current CF technique only considers user-item interaction. However, dominant CF characteristics such as similarity/relation levels do not appear to be used, but BERT has an attention mechanism that can capture the relationship between users in a sequence, where one of them is the target user. As a consequence of inputting a user's sequence into BERT, levels of similarity/relationship between other users who rated the target item(neighbour) and the target user can be extracted. Thus, integrating user sequence enables us to achieve a critical feature in CF, which is to discover the users who are the most similar to your target user (nearest neighbors). Therefore, we propose using BERT to model the sequential interactions from both CBF and CF aspects, aiming for more effective recommendations.

III. OUR PROPOSED HybridBERT4Rec MODEL

From the benefits of sequencing users' historical interactions, we introduce our proposed method, called the HybridBERT4Rec model. It comprises three main parts, namely CF-HybridBERT4rec, CBF-HybridBERT4rec, and a prediction layer (see Fig. 3). For CF-HybridBERT4rec, the raters' sequence (*user* sequence), involving those users (raters) who have rated the target item, is the input used to obtain the set of ''neighbors'' of the target user. For CBF-HybridBERT4rec, the users' historical sequence (*item* sequence), involving those items that the target user has interacted with, is the input used to obtain the similarity value of all items in the sequence with respect to the target item. After receiving both CF-HybridBERT4rec and CBF-HybridBERT4rec results, they are input to the NCF-based prediction layer [11], which outputs the final predicted rating

A. HIGH-LEVEL OF THE PROPOSED METHOD

In both CF-HybridBERT4rec and CBF-HybridBERT4rec, two main models, include three key layers which are Item/User Masking, Embedding Layer, and Attention Layer.

1) ITEM/USER MASKING

In CF-HybridBERT4rec, we aim to extract the target item representation which contains the similarity level between all neighbors and the target user. For each target item, there is a user sequence (see Fig. 6) that includes all users who have rated a target item, including the target user. Thus, we assume that other users besides target user are neighbors. In training, we utilize a random user masking to user sequence, aiming to allow the model to reconstruct the masked user as close to its original embedding as possible. After training is complete, we receive the network which is able to construct the next user representation based on the characteristics of users who interact with the target item. Therefore, in testing, we construct the target item representation by masking the target user and adding it to the end of the user sequence. The target item representation contains a comparison value between each neighboring user in the sequence and the target user. In other words, the target item representation represents the similarity level between all neighbors and the target user.

Conversely, in CBF-HybridBERT4rec, we intend to extract the user representation that represents the target user preference towards the target item. For each target user, it has an item sequence (see Fig. 7) with which is a series of items that the target user interacts. In the training stage, we randomly mask items in the item sequence. The purpose of item masking is similar to user masking, in that it enables the model to rebuild the masked item as closely as possible to its original embedding. After finishing training, we then obtain the network that can predict the next item. Hence, item masking in testing, we merely mask the target item and append it to the end of the sequence in order to predict the target user profile toward the target item.

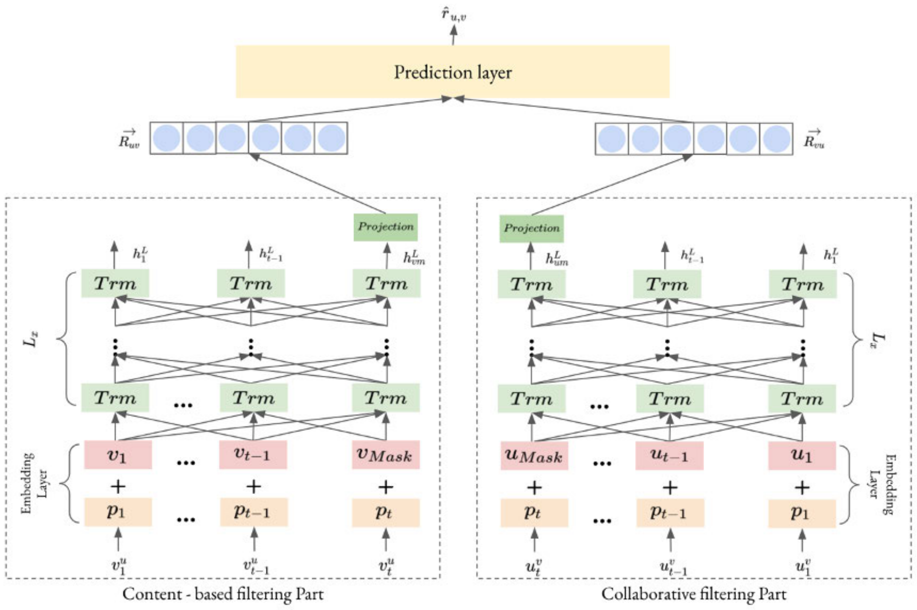


FIGURE 3. The architecture of the HybridBERT4Rec model, which comprises a CBF part, a CF part, and a prediction layer.

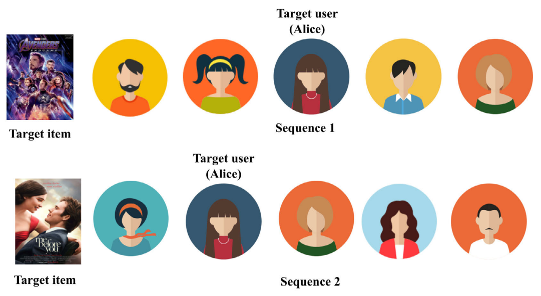


FIGURE 4. Comparison of Item sequence between "Avenger" and "Me Before You."

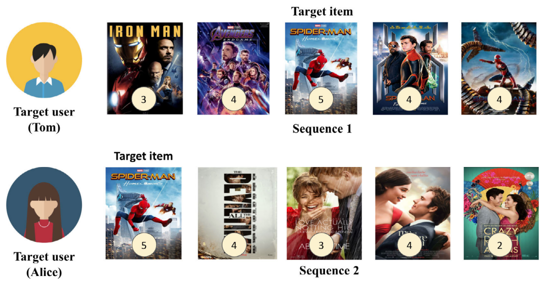


FIGURE 5. Comparison of Item sequence between "Tom" and "Alice."

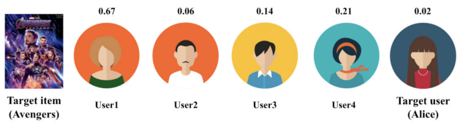


FIGURE 6. Users sequence of the target item(Avenger) with similarity scores.

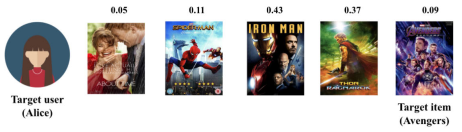


FIGURE 7. Items sequence of the target user(Alice) with similarity scores.

2) EMBEDDING LAYER

Despite the fact that user/item orders in sequence are significant in both CF-HybridBERT4rec and CBF-HybridBERT4rec, the transformer encoder in our model does not employ recurrence or convolution, therefore it cannot take into account the order of the input sequence. In CF-HybridBERT4rec, the ordering of users in user sequence indicates how significant the user who rated the target item (neighbor) is to the target user. Considering Alice is the target user, Figure 4 shows a user sequence of *"Avenger"* and a user sequence of *"Me Before You."* Despite the fact that both sequences contain the target user, the position of the target user differs. Furthermore, the surrounding users of the target user in both movies are of distinct categories of users. The majority of the surrounding users in the *"Avenger"* series are young adults, but the majority of the surrounding users in the *"Me Before You."* sequence are women.

On the other side, CBF-HybridBERT4rec, order of items in item sequence is also important since it can utilize to capture the user preference changed roughly. Suppose the target item is *"Spider-Man: Coming Home"*. As shown in Figure 5, the position of "Spider-Man: Coming Home" in Tom's sequence is different from Alice's sequence. Moreover, if we consider the surrounding movies of *"Spider-Man:*

Coming Home" in both sequences, they are also different even though the rating of *"Spider-Man: Coming Home"* from both sequences is 5. To address this issue, we positional embeddings into a masked user sequence and mask item sequence. Therefore, the embedding layer of our model is the summation of the positional embeddings and user embeddings in CF-HybridBERT4rec and the summation of the positional embeddings and item embeddings in CBF-HybridBERT4rec. We then call this **user sequence embedding** and **item sequence embedding** respectively.

3) ATTENTION LAYER

After obtaining the user sequence embedding, we feed it into BERT. BERT contains an attention mechanism that makes use of three main vectors, namely the queries Q, the keys K, and the values V. To compute the attention, they produce a matching score/similarity score by taking the dot product of Q of the target item/word and the K of the input that we want to compare with the target item or word, and these matching scores then function as the weights of the V vectors during summation. Hence, we suppose Q is the target user and K is the neighboring user. We then receive similarities between other users who rated the target item (neighbor) and the target user. As a result, inputting a user sequence into CF-HybridBERT4rec can not only capture patterns/characteristics of people who interact with target items but also simulate the level of similarity or relationship between other users who rated the target item (neighbor) and the target users. As illustrated in Figure 6, Alice's most similar neighbor in the "Avenger" user sequence is User1, a woman in her early twenties. User2, on the other hand, is an elderly man who has the least in common with Alice.

On the other hand, in CBF-HybridBERT4rec, we feed the obtained item sequence embedding into BERT.We assume Q is the target item and K is the neighboring item to compute the attention. We then receive similarity between other rated items and the target item. In conclusion, CBF-HybridBERT4rec extracts user historical patterns in terms of the relationship of rated items in target users' history data (item sequences). The user's historical patterns are basically the general items' styles or characteristics that target users favor. As you can see in Figure 7, Alice prefers sci-fi or action movies, as well as movies set in the Marvel world.

B. CF

To calculate the similarity between target users and their neighbors, we aim to apply BERT in our model by supplying a *user sequence*. A user sequence is the sequence of all users who have rated the target item, including the target user. The fundamental reason for including the CF approach is that people often obtain the most relevant recommendations from those with similar interests. Therefore, this approach can be expected to improve the target user's recommendation by considering the neighbors of the target user and measuring the similarity between the target user and neighboring users. Next, these measured similarity values are used to *weigh* each neighbor's rating of the target item, and normalized by the average similarity score. Consequently, we can calculate the rating score that target user u assigns to target item v, as

$$\widehat{r}(u, v) = \frac{\sum_{u' \in N} sim_{(u,u')} r_{(u',v)}}{\sum_{u' \in N} sim_{(u,u')}} \tag{4}$$

where N is the set of u's neighbors, $sim(u, u')$ are the similarities between the target user and its neighboring users, and r are the ratings that neighboring users assign to the target item.

As presented in (4), the CF approach requires estimates of the similarity between a target user and each of the user's neighbors with respect to their ratings. To employ this approach in our model, we introduce the neighbor users as *raters*, who are all those users who have rated the target item. For example, consider the movie "Titanic" as an item v. The raters' sequence (user sequence) is the sequence of all users who have rated the movie, namely u_1^v, u_2^v, u_3^v, u_4^v, and u_5^v (see Fig. 8(a)).

After obtaining the set of raters, they are used as an input to the BERT4Rec model. Because the BERT4Rec attention mechanism considers every pair of the input's relations, which here are the similarities among neighbors toward the target user in (4), the output will be that neighbor who is most similar to the target user.

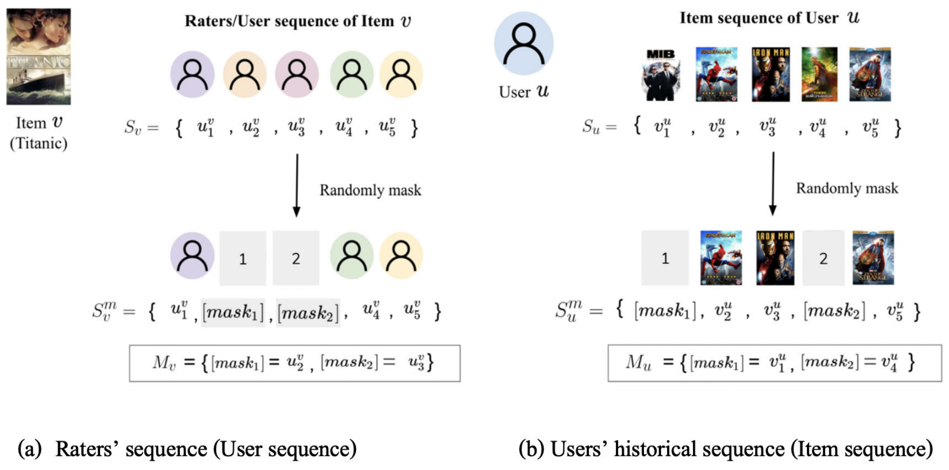
We now consider some details of the BERT4Rec model with respect to the CF part.

1) RANDOM MASKING OF THE USER IN THE RATERS' SEQUENCE/USER SEQUENCE

For target item v and user set $U = \{u_1, u_2, \ldots, u_{|U|}\}$, let $S_v = \{u_1^v, u_2^v, \ldots, u_t^v\}$ be the raters' sequence, where u_t^v is the user who rated item v at time step t. We now apply *mask language modeling* [7]. Given a raters' sequence S_v, we randomly mask the users in the raters' sequence and define the new sequence as $S_v^m = \{u_m^v, u_2^v, \ldots, u_m^v, u_t^v\}$, where u_m^v is a masked user. For example, as shown in Fig. 3(a), we have masked 15% of all users in the sequence at random, then replaced them by a *mask* token. Here, the masked users are u_2^v and u_3^v, which correspond to *User2* and *User3*, respectively, for the raters' sequence of the movie "Titanic".

2) INJECTING POSITIONAL EMBEDDINGS INTO THE MASKED RATERS' SEQUENCE

After randomly masking the input in the user sequence, we extract the user representation in the masked user sequence (S_v^m) by feeding it into the embedding layer. Because the encoder of the transformer in our model involves neither recurrence nor convolution, it cannot consider the order of the input sequence. To address this issue, we must incorporate the position of users in the input sequence. Therefore, the embedding layer of our model is the summation of the user embeddings and the positional embeddings (see Fig. 2). We call this **user sequence embedding** of the target item v and denote it as S_v'.



Raters/User sequence of Item v
Item v (Titanic)
$S_v = \{ u_1^v, u_2^v, u_3^v, u_4^v, u_5^v \}$
Randomly mask
$S_v^m = \{ u_1^v, [mask_1], [mask_2], u_4^v, u_5^v \}$
$M_v = \{[mask_1] = u_2^v, [mask_2] = u_3^v\}$
(a) Raters' sequence (User sequence)
Item sequence of User u
User u
$S_u = \{ v_1^u, v_2^u, v_3^u, v_4^u, v_5^u \}$
Randomly mask
$S_u^m = \{ [mask_1], v_2^u, v_3^u, [mask_2], v_5^u \}$
$M_u = \{[mask_1] = v_1^u, [mask_2] = v_4^u \}$
(b) Users' historical sequence (Item sequence)

FIGURE 8. The inputs for the CF and CBF parts in the HybridBERT4Rec model.

3) FEEDING THE USER SEQUENCE EMBEDDING TO THE STACK OF TRANSFORMERS

Having obtained the user sequence embedding (S_v'), we feed it to a stack of transformer models for training. After exchanging information among the users in the sequence through the L transformer layers that capture the user–user relation, we obtain the final hidden layer output $H_U^L = \{h_{u1}^L, h_{u2}^L, \ldots, h_{um}^L, \ldots, h_{ut}^L\}$, where h_{ut}^L shows the relation or similarity for every pair of users in the raters' sequence with respect to the target item for each query user u_t. The model predicts the masked users (u_m) based on the hidden layer output of $u_m(h_{um}^L)$ (see Fig. 2). The feedforward layer uses Gaussian-error-linear-unit (GELU) [22] activation to generate the neighboring distribution over the masked user, as

$$P(u_m) = softmax(GELU(h_{um}^L W^h + b^h)E^U + b^O) \quad (5)$$

where W^h is the projection weight matrix of the final hidden-layer output H_U^L, b^h is the bias term at the final hidden-layer output, and E^U is the embedding matrix for all users in the set U who have passed the user embedding and positional embedding described above, and b^O is the bias term of the activation function GELU.

In the training step, our goal is to make the model associate the masked users as closely as possible with the original users. We therefore define the loss function as the negative log-likelihood of the masked user on the CF side as

$$L = \frac{1}{|M_v|} \sum_{u_m \in M_v} -logP(u_m = u_m^T | S_v^m) \quad (6)$$

where S_v^m is the masked version of the raters' sequence corresponding to target item v, and M_v is the set of randomly masked users in S_v^m, u_m^T is the true user for the masked user u_m, and $P(\cdot)$ is the probability that was defined in (5).

4) FINDING THE NEIGHBORING DISTRIBUTION OVER THE TARGET USER

After this training step, we obtain the final hidden layer output H'^L_U of the trained BERT model with a suitably weighted neural network that shows the similarity between users in the user sequence. In the prediction step, we want to find the similarity between each neighbor in this user sequence and the target user. We therefore input the neighboring raters' sequences (user sequences) for target item v into our model by appending a *mask* token at the end of the input sequence. The last *mask* token is selected from the set $U - S_v$ and is assumed to be the **target user (u_{m_t})**. This is then input to the embedding layer, which involves both token embedding and position embedding, and on to the trained BERT model. The model predicts the last *mask* token based on the final hidden layer output of the token ($h'^L_{um_t}$). The feedforward layer with GELU activation then generates the distribution of all users over the target user, as

$$P(u_{m_t}) = softmax(GELU(h'^L_{um_t} W^{h'} + b^{h'})E^U + b^O) \quad (7)$$

where $W^{h'}$ is the projection weight matrix of the final hidden-layer output $h'^L_{um_t}$, $b^{h'}$ is the bias term at the final hidden-layer output, E^U is the embedding matrix for all users in the set $U - S_v$, and b^O is the bias term of the activation function GELU.

Finally, we obtain the neighboring distribution over the target user, expressed as the user-similarity probability between the target user and the neighboring users. We call this the **target item v profile ($\overline{R_{vu}}$)** for target user u as shown in Algorithm 1 in CF-HybridBERT4Rec procedure.

In the next subsection, we show how BERT is applied in the CBF part to predict the next item with which the target user is likely to interact.

C. CBF

The CBF approach's fundamental aim is to recommend an item similar to an item that the user has interacted with in the past. Therefore, this approach measures the similarities between the target user's historical items and items that are potential recommendations to the target user by considering the content that describes the items. The BERT4Rec [5] model determines the similarity of every pair of items in

Algorithm 1 HybridBERT4Rec

```
input Set of Users $U$, Set of Items $V$, User embedding matrix
    $E^U$, Item embedding matrix $E^V$, User sequence $S_v$, Item
    sequence $S_u$
output $\hat{r}(u_i, v_i)$
procedure CF-HybridBERT4Rec($U$,$V$,$E^U$,$S_v$)
    for $v_j \in V$ do
        $S^m{}_{v_j} \leftarrow$ randomly mask($S_{v_j}$)
        $S'_{v_j} \leftarrow$ embedding layer($S^m{}_{v_j}$)
        $H'^L_U \leftarrow$ BERT($S'_{v_j}$) as final hidden layer
        for $u_i \in U$ do
            $S^m{}_{v_j,u_i} \leftarrow$ mask($S_{v_j}$,$u_i$)
            $Rv_j, u_i = softmax(GELU(h'^L_{u^m_i} W^h + b^h)E^u + b^O)$
procedure CBF-HybridBERT4Rec($U$,$V$,$E^V$,$S_u$)
    for $u_i \in U$ do
        $S^m{}_{u_i} \leftarrow$ randomly mask($S_{u_i}$)
        $S'_{u_i} \leftarrow$ embedding layer($S^m{}_{u_i}$)
        $H'^L_V \leftarrow$ BERT($S'_{u_i}$) as final hidden layer
        for $v_j \in V$ do
            $S^m{}_{u_i}, v_j \leftarrow$ mask($S_{u_i}$,$v_j$)
            $Ru_i, v_j = softmax(GELU(h'^L_{v^m_j} W^h + b^h)E^v + b^O)$
procedure Prediction($Ru_i, v_j$,$Rv_j, u_i$)
    for $u_i, v_j \in U, V$ do
        $R(u_i, v_i) = Ru_i, v_j \odot Rv_j, u_i$
        $\hat{r}(u_i, v_i) \leftarrow = \sigma(W\dot{R}(u_i, v_i) + b)$
    return $\hat{r}(u_i, v_i)$
```

the sequence and predicts the next item that the target user is likely to interact with by processing the user's historical sequence. For example, consider movies and a target user u. This user's historical sequence (item sequence) is the sequence of all movies that user u has previously rated, namely $v^u_1, v^u_2, v^u_3, v^u_4$, and v^u_5 (see Fig. 4(b)).

To employ the CBF approach in our model, we adopt the BERT4Rec [5] model by measuring the similarity between items in the users' historical sequence and the target item. The detailed steps are similar to those used for the CF-based analysis.

For a target user u and a set of items $V = \{v_1, v_2, \ldots, v_{|V|}\}$, let $S_u = \{v^u_1, v^u_2, \ldots, v^u_t\}$ be user u's historical sequence, where v^u_t is the item that user u rated at time step t. First, we randomly mask items in item sequence, giving $S^m_u = \{v^u_m, v^u_2, v^u_m, \ldots, v^u_t\}$ as the masked item sequence, where v^u_m is a masked item. Only the masked items are collected in M_u. For the example in Fig. 3(b), we mask v^u_1 and v^u_4, which are the movies "MIB" and "Thor", respectively.

Next, we incorporate positional embeddings into a masked item sequence by summing the item embedding and positional embedding in the embedding layer, which we call the **item sequence embedding** for target user u, and denote as S'_u. This item sequence embedding is then fed into BERT for training and we obtain the final hidden-layer output $H^L_V = \{h^L_{v1}, h^L_{v2}, \ldots, h^L_{vm}, \ldots, h^L_{vt}\}$, where h^L_{vt} shows the relation

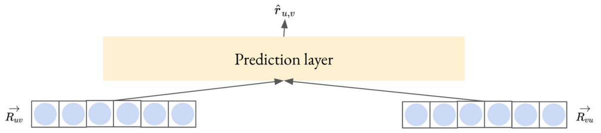


FIGURE 9. The architecture of HybridBERT4Rec at the prediction stage.

or similarity of every pair of items in the historical item sequence for the target user with respect to each query item v_t. The distribution of all items over the masked item v_m is then

$$P(v_m) = softmax(GELU(h^L_{vm} W^h + b^h)E^V + b^O) \quad (8)$$

where W^h is the projection weight matrix of the final hidden-layer output H^L_V, b^h is the bias term at the final hidden-layer output, E^V is the embedding matrix for all items in the set V, and b^O is the bias term for the activation function GELU.

In the CBF training step, our goal is the same as that for the CF part. We define the loss function as the negative log-likelihood of the masked item as

$$L = \frac{1}{|M_u|} \sum_{v_m \in M_u} -logP(v_m = v^T_m | S^m_u) \quad (9)$$

where S^m_u is the masked version of user u's historical interaction sequence, M_u is the set of randomly masked items in S^m_u, v^T_m is the true item for the masked item v_m, and $P(\cdot)$ is the same probability that was defined in (8).

After training, we obtain the final hidden-layer output H'^L_V, which shows the similarity between items in the item sequence. In the testing step, we want to predict the degree to which the target user will interact with the target item. Therefore, we feed the item sequence for target user u into our model by appending a *mask* token at the end of the input sequence. The last *mask* token, **target item (v_{m_t})**, is from set $V - S_u$, which is the set of all items that the target user has neither rated nor seen previously. The feedforward layer with GELU activation then computes the distribution of all items over the target item as

$$P(v_{m_t}) = softmax(GELU(h'^L_{vm_t} W^{h'} + b^{h'})E^V + b^O) \quad (10)$$

where $W^{h'}$ is the projection weight matrix of the final hidden-layer output H'^L_V, $b^{h'}$ is the bias term at the final hidden-layer output, E^V is the embedding matrix for all items in the set $V - S_u$, and b^O is the bias term for the activation function GELU.

Finally, we obtain the distribution of all items over the target item, expressed as the interaction probability of all items with the target user. We call this the **target user** u **profile ($\overline{R_{uv}}$)** toward target item v as shown in Algorithm 1 in CBF-HybridBERT4Rec procedure.

D. PREDICTION STAGE

From Algorithm 1 in Prediction procedure, it show that we are able to extract target item profiles for target user u **($\overline{R_{vu}}$)** from the CF part and target user profiles toward target item v ($\overline{\boldsymbol{R_{uv}}}$) from the CBF part. In this subsection, we present a method for predicting the rating score by using the MF principle [10] used in the NCF [11] approach, as previously described in Section 2.1. Figure 4 shows the mapping function of the first neural CF layer, expressed as

$$R_{(u,v)} = \overline{R_{uv}} \odot \overline{R_{vu}} \tag{11}$$

where $\odot$ denotes the element-wise product of the vectors. In the prediction state, the rating of user u toward item v is computed by

$$\widehat{r}_{u,v} = R_{(u,v)} + b_u + b_v + \alpha \tag{12}$$

where b_u, b_v, and α are the bias of target user u, the bias of target item v, and the global bias, respectively. We use the L2-norm for the loss function, expressed as

$$L = \sum (r_{u,v} - \widehat{r}_{u,v})^2 \tag{13}$$

where $r_{u,v}$ is the true rating and $\widehat{r}_{u,v}$ is the rating prediction.

IV. EXPERIMENTAL EVALUATION

BERT4Rec considers only the historical data for the target user, which is a CBF approach. In contrast, our proposed HybridBERT4Rec method considers not only the CBF approach but also a CF approach, which considers the opinions of the target users' neighbors. In this section, we report on our experiments, which compare HybridBERT4Rec and BERT4Rec.

A. DATASETS

We evaluate the proposed model in terms of three datasets as follows.

- **MovieLens**[1]: This is a dataset that is widely used in recommendation tasks for evaluating recommendation algorithms. In this work, we adopt the MovieLens-1M version, which comprises one million ratings, with each user being responsible for at least 20 ratings.
- **Yelp**[2]: This is a dataset of restaurant reviews and service businesses crawled from Yelp's business in the United States. The dataset comprises eight million ratings, with each user being responsible for at least four ratings.
- **Goodreads**[3] [23]: This is a book-review dataset collected from goodreads.com. It comprises one million ratings, with each user being responsible for at least 70 ratings.

In the CBF part, we group the interaction records by user and build the item sequence for each user by sorting these interaction records according to their timestamps. In the CF part, we group the interaction records by item and build the user sequence for each item by again sorting according to timestamps. The details of the datasets are given in Table 1.

[1]https://grouplens.org/datasets/movielens/1m/

[2]https://www.yelp.com/dataset

[3]https://sites.google.com/eng.ucsd.edu/ucsdbookgraph/reviews

TABLE 1. Summary details of the datasets.

Dataset	#Interactions	#Users	#Items	Sparsity
MovieLens-1M	1,000,209	6,040	3,706	95.53%
Yelp dataset	8,021,122	1,968,703	209,393	99.98%
Goodreads dataset	1,330,981	18,868	25,469	99.72%

The sparsity value indicates the proportion of interactions that are missing and is calculated by

$$\frac{\#Interactions}{\#Users \times \#Items} \tag{14}$$

We have to use sufficient data to be able to create a historical sequence for learning via the BERT model. The full Yelp dataset guarantees only at least four ratings for each user. Therefore, we had to prune some users from the dataset (those who contributed less than 20 ratings) to meet the sufficiency requirement. The MovieLens and Goodreads datasets already met that condition. Table 2 gives the dataset characteristics after this preprocessing.

TABLE 2. Summary details of the datasets after preprocessing.

Datasets	#Interactions	#Users	#Items	Sparsity
MovieLens-1M	1,000,209	6,040	3,706	95.53%
Yelp dataset	3,000,029	57,819	182,225	99.97%
Goodreads dataset	1,330,981	18,868	25,469	99.72%

B. EVALUATION METRICS

To evaluate our proposed method, we consider performance in terms of prediction accuracy. We adopted the hit ratio (HR) as an evaluation metric for measuring the ranking accuracy of the top-K recommendation list for each user. The HR can be expressed as

$$HR = \frac{number\ of\ hits}{number\ of\ hits + number\ of\ misses} \tag{15}$$

where *hits* and *misses* can be calculated by first checking if the item's predicted rating score exceeds a threshold. In our experiments, all datasets had a rating range of 1 to 5, and we set the threshold as 3. If the threshold for an item was not met, we omitted the item. For the remaining items, we then checked whether the item was ranked. If it was among the ranked items, we categorized it as a *hit* and assigned it a value of 1. Conversely, if the item was unranked, we categorized it as a *miss* and assigned it a value of 0.

A second metric adopted was the normalized discounted cumulative gain (NDCG), which measures the top-K recommendation list quality. The NDCG can be expressed as

$$DCG_k = \sum_{i=1}^{k} \frac{2^{rel_i} - 1}{log_2(i+1)}, NDCG_k = \frac{DCG_k}{IDCG_k} \tag{16}$$

where $rel_i = \{1,2,3,4,5\}$ is the actual rating score of an item at the rank position i.

From our experiments, we give results for HR and NDCG with $k = 1, 3, 5, 10, 15, 20, 30, 40, 50$. For all metrics, high values correspond to better performance.

TABLE 3. Comparison characteristic of all models.

	Unidirectional	Bidirectional	CBF	CF	
				without Neighbors	with Neighbors
Caser	✓	-	✓	-	-
GRU4Rec	✓	-	✓	-	-
SASRec	✓	-	✓	-	-
BERT4Rec	-	✓	✓	-	-
HybridBERT4Rec	-	✓	✓	✓	✓

TABLE 4. The NDCG comparison of all comparative models on all datasets.

Datasets	Matric	Caser	GRU4Rec	SAS4Rec	BERT4Rec	HybirdBERT4Rec
MovieLen 1M	NDCG@1	0.2211	0.2147	0.2952	0.3467	**0.3528**
	NDCG@3	0.3538	0.3064	0.3325	0.4593	**0.4595**
	NDCG@5	0.3591	0.3259	0.3472	0.5132	**0.5339**
	NDCG@10	0.3699	0.3448	0.4282	0.5052	**0.5251**
	NDCG@15	0.3983	0.3733	0.427	0.5274	**0.5412**
	NDCG@20	0.4116	0.4048	0.4335	0.5448	**0.5597**
	NDCG@30	0.4258	0.4329	0.4521	0.5780	**0.5859**
	NDCG@40	0.5092	0.4901	0.535	0.5888	**0.5956**
	NDCG@50	0.5372	0.5223	0.5563	0.6006	**0.6128**
Yelp	NDCG@1	0.3593	0.3725	0.3833	0.4075	**0.4356**
	NDCG@3	0.3937	0.4317	0.4440	0.5113	**0.5987**
	NDCG@5	0.4222	0.5349	0.5434	0.5886	**0.6163**
	NDCG@10	0.5391	0.5501	0.5672	0.5940	**0.6179**
	NDCG@15	0.5834	0.5925	0.6118	0.6344	**0.6420**
	NDCG@20	0.6021	0.6111	0.6199	0.6482	**0.6599**
	NDCG@30	0.6172	0.6179	0.6368	0.6732	**0.6802**
	NDCG@40	0.6326	0.6288	0.6523	0.7086	**0.7234**
	NDCG@50	0.657	0.6654	0.6692	0.7176	**0.7490**
GoodRead	NDCG@1	0.2026	0.1899	0.2245	0.2209	**0.2880**
	NDCG@3	0.2892	0.2661	0.2971	**0.3212**	0.3027
	NDCG@5	0.3050	0.2865	0.3426	0.3865	**0.3904**
	NDCG@10	0.3861	0.3557	0.4031	0.4530	**0.4832**
	NDCG@15	0.4556	0.4366	0.4964	0.4644	**0.5091**
	NDCG@20	0.4618	0.4492	0.4631	0.4820	**0.5200**
	NDCG@30	0.4689	0.4572	0.4754	0.5032	**0.5471**
	NDCG@40	0.4764	0.4550	0.5002	0.5332	**0.5555**
	NDCG@50	0.4950	0.4901	0.5266	0.5640	**0.5893**

C. BASELINES & PARAMETER SETTING

To investigate the effectiveness of HybridBERT4Rec, we compare our model with following baselines:

- **Caser** [20]: It is a unidirectional CBF approach CNN-based method that uses a sequence of items that users interacted with in the past to predict top-N ranked items that a user will likely interact.
- **GRU4Rec** [24]: It is a unidirectional RNN-based model that predicts next item embeddings using the user's feedback sequence as input. It only includes the CBF method.
- **SASRec** [25]: It is a unidirectional CBF approach transformer-based model that predicts the next item based on a user's action history.
- **BERT4Rec** [6]: It is a bi-directional model that utilizes user historical sequences to predict next item. It contains solely the CBF approach.

Table 3 compares the characteristics of all methods. Our method HybridBERT4Rec is the only sequential-based recommendation method that contains both the CBF approach which captures the general items' style/characteristics that target users prefer and the CF approach which extracts similarity/relationship levels between other users who rated the target item (neighbor) and the target user in order to predict next item.

For Caser, GRU4Rec, SASRec, and BERT4Rec, we used the actual code and, for all initialization procedures and hyperparameter values, we followed the author's suggestions.

HybridBERT4Rec comprises three main components. The first is the CBF part, which we trained in the same way as for BERT4Rec, using a batch size of 256 sequences and Adam [22] with a learning rate of $1 \times e^{-4}$. Although HybridBERT4Rec's input was the same as that for BERT4Rec, which is the user's historical sequence (item sequence) of the target user, the model goals were different. HybridBERT4Rec's goal is to capture the similarities between items in a sequence to predict items with which the target user will interact. In contrast, BERT4Rec predicts the next item with which the target user will interact. In the CBF part, the maximum sequence length of items (V) was derived by averaging the sequence of all items. It was $V = 200$ for MovieLens-1M and $V = 100$ for both Yelp and Goodreads.

Next, the CF part was trained in the same way as for the CBF part, with the input being the user sequence for the

TABLE 5. The HR comparison of all comparative models on all datasets.

Datasets	Matric	Caser	GRU4Rec	SAS4Rec	BERT4Rec	HybirdBERT4Rec
MovieLen 1M	HR@1	0.2921	0.3147	0.3252	**0.3529**	0.3469
	HR@3	0.4258	0.4064	0.4435	**0.4656**	0.4652
	HR@5	0.4391	0.4925	0.5137	**0.5339**	0.5315
	HR@10	0.3629	0.3941	0.4282	0.5576	**0.5722**
	HR@15	0.4523	0.4733	0.5427	0.6069	**0.6112**
	HR@20	0.5116	0.5048	0.5335	0.6370	**0.6455**
	HR@30	0.6252	0.6341	0.6521	0.6826	**0.6829**
	HR@40	0.5592	0.5921	0.6352	0.6920	**0.7140**
	HR@50	0.6375	0.6293	0.6563	0.7280	**0.7378**
Yelp	HR@1	0.2593	0.2725	0.3133	0.3547	**0.4095**
	HR@3	0.4537	0.4317	0.444	0.4856	**0.5468**
	HR@5	0.5183	0.5349	0.5434	0.5855	**0.6655**
	HR@10	0.5411	0.5501	0.5672	0.6734	**0.6933**
	HR@15	0.5864	0.5555	0.6118	0.6944	**0.7207**
	HR@20	0.6421	0.6521	0.6999	0.7021	**0.7386**
	HR@30	0.6172	0.6179	0.6863	0.723	**0.7514**
	HR@40	0.6326	0.6788	0.7123	0.7545	**0.7968**
	HR@50	0.7520	0.7354	0.7672	0.7986	**0.8132**
GoodRead	HR@1	0.3026	0.2879	0.3245	0.3335	**0.3869**
	HR@3	0.3892	3805	0.4100	0.4388	**0.4570**
	HR@5	0.3305	0.35621	0.3646	0.4463	**0.4846**
	HR@10	0.3962	0.3647	0.4141	0.4900	**0.5032**
	HR@15	0.4512	0.4223	0.4994	0.5023	**0.5191**
	HR@20	0.4611	0.4409	0.4941	0.5151	**0.5320**
	HR@30	0.4329	0.4022	0.4734	0.5387	**0.5571**
	HR@40	0.4564	0.4335	0.5335	0.5627	**0.6077**
	HR@50	0.4725	0.4521	0.5666	0.5909	**0.6253**

target item. (The goal is to capture the relationship between the target user and its neighbors by extracting neighbors that are similar to the target user.) The maximum sequence length of users (U) used here was $U = 300$ for MovieLens-1M, $U = 50$ for Yelp, and $U = 100$ for Goodreads. Again, these values were derived by averaging the sequence across all users.

The third component is the prediction part, in which we apply the NCF approach to predict the rating score via a regression technique. Here, we used two hidden layers, each having 1000 dimensions.

D. EXPERIMENTAL RESULTS

For our experiments, we partitioned the datasets, with 80% used for training and 20% for testing. We evaluated Hybrid-BERT4Rec and four baseline methods via the three datasets listed in Table 2. Table 4 and Table 5 summarized that Hybrid-BERT4Rec receive the best results of all models on three benchmark datasets in term of NDCG and HR. Among CBF baseline, BERT4Rec outperforms Caser, GRU4Rec, SASRec on all datasets indicating that bidirectional training is a more potent tool for sequential learning.

Furthermore, the results show that HybridBERT4Rec outperforms BERT4Rec on all datasets and all evaluation metrics in terms of accuracy. The only difference between the systems is that BERT4Rec considers only target user interactions to find the similarity between the target item and all items that the target user has interacted with. This is the pure CBF approach. In contrast, HybridBERT4Rec not only uses the historical sequence for the target user but also considers the historical sequence of those other users in the system who are neighbors of the target user. This is the alternative CF approach. Therefore, with HybridBERT4Rec clearly outperforming BERT4Rec, this can only be because it adopts both CBF and CF approaches, unlike BERT4Rec, with its CBF-only approach. Additionally, employing BERT in CF and CBF allows for the acquisition of more rich characteristics.

TABLE 6. Time consumption between BERT4Rec and HybridBERT4Rec for training MovieLen-1M dataset.

Model	Time complexity	#Time (Min)
BERT4Rec	$O(dn^2)$	10224
HybridBERT4Rec	$O(dn^2)$	10512

In order to evaluate potential trade-offs between performance/resource consumption, we show comparative time consumption between BERT4Rec and HybridBERT4Rec. Table 6 shows that training HybridBERT4Rec takes slightly longer than training BERT4Rec as it must train on both CF and CBF. For Time complexity of both BERT4Rec and HybridBERT4Rec is $O(dn^2)$ where d is embedding dimension and n is sequence length.

E. IMPACT OF ATTENTION MECHANISM

We are therefore investigating how the attention mechanism influences recommendation performance. Figure 10 show NDCG@, NDCG@10, HR@5, and HR@10 for Hybrid-BERT4rec with and without attention mechanism. The most obvious implication of the results from these sub-figures is that the performance of HybridBERT4rec with attention mechanism outperform HybridBERT4rec without attention.

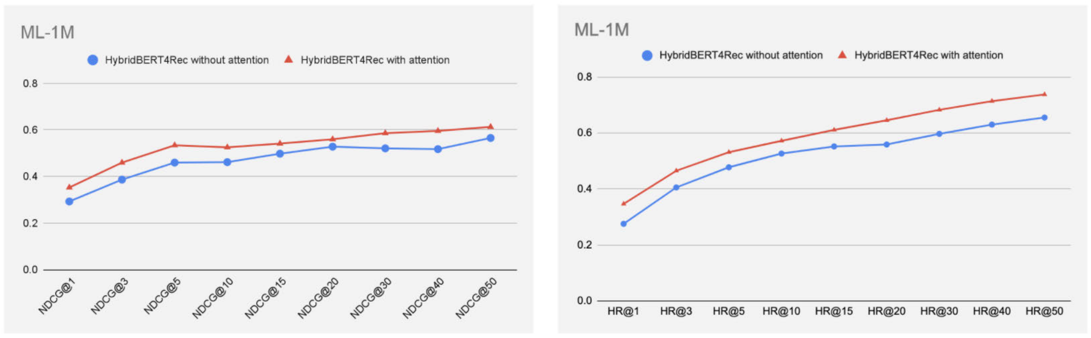
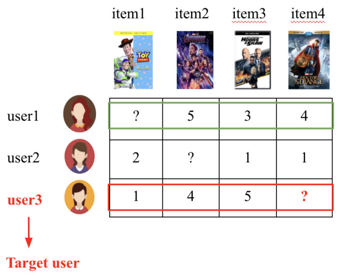


FIGURE 10. Rating matrix between user and item.



	item1	item2	item3	item4
user1	?	5	3	4
user2	2	?	1	1
user3	1	4	5	?

FIGURE 11. Rating matrix between user and item.

To demonstrate why attention mechanism is important, we describe an example situation in which standard memory-based CF approaches are compared to calculation of similarities utilizing attention mechanism. From Figure 11, to compute similarities using memory-based CF techniques. They calculate the similarities between user3 (target user) vector and user1/user2 (neighbor user) vectors. In the same ways, an attention mechanism that makes use of three main vectors, namely the *queries* Q, the *keys* K, and the *values* V. To calculate the attention score, they take the dot product of the Q of the target item/word and the K of the input that we want to compare with target item/word to calculate a matching score/similarity score, and these matching scores then act as the weights of the V vectors during summation. Thus, we suppose Q is the target user and K is the neighboring users, we receive similarities between other users who rated the target item (neighbor) and the target user. Hence, using the attention mechanism is not different from the computation of similarities in standard memory-based CF. Instead of preprocessing by calculating similarities using standard memory-based CF techniques then incorporating them into the network, we can perform end-to-end tasks by applying an attention mechanism from BERT.

V. DISCUSSION

BERT4Rec outperforms other sequential recommendation models such as Caser [20], GRU4Rec [24], and SASRec [25] because those models handle sequences in one direction only. BERT is a deep bidirectional self-attention model that can capture item relations on both sides, left and right, whereas other sequential recommendation models only consider users' historical sequences from left to right. In real situations, a user's behavior depends on the user's current interests, which can evolve in a highly dynamic manner. Therefore, considering only previous items is insufficient in terms of accuracy. For example, suppose $User1$ is currently interested in action movies but has previously only watched romantic movies. If a model considers only previous items, it is going to recommend romantic movies to $User1$ rather than a movie that $User1$ might currently like.

However, BERT4Rec solely predicts the next item by extracting user historical patterns in terms of the relation of rated items in target users' history data (item sequences). The user historical patterns are basically the general items' style/characteristics that target users favor.

On the other hands, our model, HybridBERT4Rec, is composed of three main parts, which are CF-HybridBERT4rec, CBF-HybridBERT4rec, and Prediction layer. From Figure 12, If we assume that the target interaction consists of the target user who is Alice and the target item is Avenger. Our model wants to predict the rating of target interaction, which is a rating that Alice will give to Avenger. Firstly, CBF-HybridBERT4rec, we construct the user representation(Alice) by inputting item sequence that Alice rated as

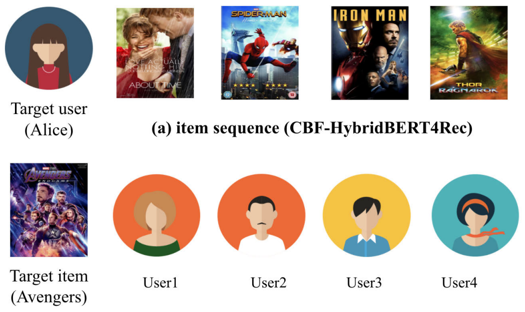

Target user
(Alice)
(a) item sequence (CBF-HybridBERT4Rec)
Target item
(Avengers)
User1
User2
User3
User4
(b) user sequence (CB-HybridBERT4Rec)

FIGURE 12. The input of CF-HybridBERT4rec and the CBF-HybridBERT4rec models.

shown in Figure 12(a). Afterward, we construct item representation(Avenger) by inputting user sequence who rated Avenger as shown in Figure 12(b) into CF-HybridBERT4rec. Finally, prediction layer, we utilize output from both CF-HybridBERT4rec and the CBF-HybridBERT4rec to predict the rating of the target interaction. Then the rating can be calculated in the NCF model by element-wise Alice representation and Avenger's representation. After receiving the predicted rating of target interaction, we use the rating from the actual interaction between Alice and Avenger as a label to back propagate our model.

The experimental results show that our HybridBERT4Rec performs better than BERT4Rec with respect to both HR and NDCG metrics. BERT4Rec only extracts user historical patterns, which are the general item styles/characteristics that target users like. On the other side, HybridBERT4Rec incorporates users sequence which is a sequence of users who rated the target item and also be target user neighbors, in order to capture patterns/characteristics of people who interact with the target item. Moreover, BERT has an attention mechanism that is able to capture the relation between the user in sequence and one of the users in the sequence is the target user. Consequently, feeding a users' sequence into BERT also can extract similarity/relation levels between other users who rated target item(neighbor) and target user. Thus, integrating user sequence allow us to achieve critical feature in CF which is to find the most similar users to your target user (nearest neighbors) and weight their ratings of an item as the prediction of the rating of this item for target user as

$$\hat{r}(user, item) = \frac{\sum sim(user, neighborr(neighbor, item)}{\sum sim(user, neighbor)} \tag{17}$$

whereas BERT4Rec is not able to accomplish that. In addition, utilizing historical data to predict the next items is only CBF technique. The limitation of the BERT4Rec or CBF-base approach is users cannot acquire recommended items that vary from their historical sequence. In other words, the new item characteristics are not recommended because CBF-base models recommend based on the item characteristic that users have been engaged with. The various reasons explained above support the empirical evidence that HybridBERT4Rec is more accurate than BERT4Rec.

VI. CONCLUSION

In this study, we propose HybridBERT4Rec, a hybrid (CBF and CF) recommender system based on BERT that uses other relevant users' historical sequences to help with the recommendation instead of using only the historical sequence for the target user. In the CBF part, we estimate the similarity between a target item and other items in the historical sequence of the target user. We feed the target user's item sequence into the BERT model, with the query being about a target item, and we receive the target user profile, which shows the similarity between the target item and items in the sequence. In addition to this CBF part, we also determine which neighbors are the most similar to the target user in the CF part. Here, we introduce the raters' sequences (user sequences), which is the set of sequences for all users who have rated the target item. We first determine how many "neighbors" among the raters' sequences are similar to the target user. We then feed the raters' sequences into the BERT model, with the query being about the target user, receiving a target item profile. This shows the similarity of neighbors to the target user. After obtaining both the target user profile and the target item profile, we input them into an NCF model that predicts the rating score. Our experimental results show that HybridBERT4Rec consistently outperformed the baseline BERT4Rec model in terms of accuracy.

REFERENCES

[1] K. D. Bollacker, S. Lawrence, and C. L. Giles, "CiteSeer: An autonous web agent for automatic retrieval and identification of interesting publications," in *Proc. 2nd Int. Conf. Auton. Agents (AGENTS)*, 1998, pp. 116–123.

[2] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 6088, pp. 533–536, 1986.

[3] F. Yu, Q. Liu, S. Wu, L. Wang, and T. Tan, "A dynamic recurrent model for next basket recommendation," in *Proc. 39th Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.* Pisa, Italy: Association for Computing Machinery, Jul. 2016, pp. 729–732.

[4] Q. Cui, S. Wu, Q. Liu, W. Zhong, and L. Wang, "MV-RNN: A multi-view recurrent neural network for sequential recommendation," *IEEE Trans. Knowl. Data Eng.*, vol. 32, no. 2, pp. 317–331, Feb. 2020.

[5] F. Sun, J. Liu, J. Wu, C. Pei, X. Lin, W. Ou, and P. Jiang, "BERT4Rec: Sequential recommendation with bidirectional encoder representations from transformer," in *Proc. 28th ACM Int. Conf. Inf. Knowl. Manage.* New York, NY, USA: Association for Computing Machinery, Nov. 2019, pp. 1441–1450.

[6] J. Devlin, M. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proc. NAACL-HLT*. Stroudsburg, PA, USA: Association for Computational Linguistics, 2019, pp. 4171–4186.

[7] W. L. Taylor, "Cloze procedure: A new tool for measuring readability," *Journalism Bull.*, vol. 30, no. 4, pp. 415–433, 1953.

[8] C. Yang, X. Chen, L. Liu, T. Liu, and S. Geng, "A hybrid movie recommendation method based on social similarity and item attributes," in *Proc. 9th Int. Conf. Adv. Swarm Intell.*, Shanghai, China, 2018, pp. 275–285.

[9] R. Turnip, D. Nurjanah, and D. S. Kusumo, "Hybrid recommender system for learning material using content-based filtering and collaborative filtering with good learners' rating," in *Proc. IEEE Conf. e-Learn., e-Manage. e-Services (ICe)*, Nov. 2017, pp. 61–66.

[10] Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," *IEEE Comput.*, vol. 42, no. 8, pp. 30–37, Aug. 2009.

[11] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T. Chua, "Neural collaborative filtering," in *Proc. 26th Int. Conf. World Wide Web, Int. World Wide Web Conf. Committee (IWC)*, Perth, WA, Australia, 2017, pp. 173–182.

[12] Y. LeCun, P. Haffner, L. Bottou, and Y. Bengio, "Object recognition with gradient-based learning," in *Shape, Contour and Grouping in Computer Vision*. Berlin, Germany: Springer, 1998, pp. 1627–1634.

[13] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.

[14] K. Cho, B. van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using RNN encoder–decoder for statistical machine translation," in *Proc. Conf. Empirical Methods Natural Lang. Process. (EMNLP)*. Doha, Qatar: Association for Computational Linguistics, 2014, pp. 1724–1734.

[15] T. Donkers, B. Loepp, and J. Ziegler, "Sequential user-based recurrent neural network recommendations," in *Proc. 11th ACM Conf. Rec. Syst.* New York, NY, USA: Association for Computing Machinery, Aug. 2017, pp. 152–160.

[16] B. Hidasi and A. Karatzoglou, "Recurrent neural networks with top-k gains for session-based recommendations," in *Proc. 27th ACM Int. Conf. Inf. Knowl. Manage.* New York, NY, USA: Association for Computing Machinery, Oct. 2018, pp. 843–852.
[17] J. Li, P. Ren, Z. Chen, Z. Ren, T. Lian, and J. Ma, "Neural attentive session-based recommendation," in *Proc. ACM Conf. Inf. Knowl. Manage.* New York, NY, USA: Association for Computing Machinery, Nov. 2017, pp. 1419–1428.
[18] M. Quadrana, A. Karatzoglou, B. Hidasi, and P. Cremonesi, "Personalizing session-based recommendations with hierarchical recurrent neural networks," in *Proc. 11th ACM Conf. Recommender Syst.* New York, NY, USA: Association for Computing Machinery, Aug. 2017, pp. 130–137.
[19] X. Chen, H. Xu, Y. Zhang, J. Tang, Y. Cao, Z. Qin, and H. Zha, "Sequential recommendation with user memory networks," in *Proc. 11th ACM Int. Conf. Web Search Data Mining*, New York, NY, USA, Feb. 2018, pp. 108–116.
[20] J. Tang and K. Wang, "Personalized top-N sequential recommendation via convolutional sequence embedding," in *Proc. 11th ACM Int. Conf. Web Search Data Mining*, Marina Del Rey, CA, USA, Feb. 2018, pp. 565–573.
[21] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, N. A. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *Proc. Neural Inf. Process. Syst. Conf. (NIPS)*. Red Hook, NY, USA: Curran Associates, 2017, pp. 5998–6008.
[22] P. K. Diederik and L. B. Jimmy, "Adam: A method for stochastic optimization," *CoRR*, vol. abs/1412.6980, 2015.
[23] M. Wan, R. Misra, N. Nakashole, and J. McAuley, "Fine-grained spoiler detection from large-scale review corpora," in *Proc. ACL*, 2019, pp. 2605–2610.
[24] B. Hidasi, A. Karatzoglou, L. Baltrunas, and D. Tikk. "Session-based recommendations with recurrent neural networks," *CoRR*, vol. abs/1511.06939, 2016.
[25] W.-C. Kang and J. McAuley, "Self-attentive sequential recommendation," in *Proc. IEEE Int. Conf. Data Mining (ICDM)*, Nov. 2018, pp. 197–206.
[26] M. Chen, Z. Xu, K. Weinberger, and S. Fei, "Marginalized denoising autoencoders for domain adaptation," in *Proc. 29th Int. Conf. Int. Conf. Mach. Learn. (ICML)*. Madison, WI, USA: Omnipress, 2012, pp. 1627–1634.
[27] P. B. Thorat, R. M. Goudar, and S. Barve, "Survey on collaborative filtering, content-based filtering and hybrid recommendation system," *Int. J. Comput. Appl.*, vol. 110, no. 4, pp. 31–36, Jan. 2015.
[28] I. Islek and S. G. Oguducu, "A hybrid recommendation system based on bidirectional encoder representations," in *Proc. Joint Eur. Conf. Mach. Learn. Knowl. Discovery Databases*. Cham, Switzerland: Springer, 2020, pp. 225–236.
[29] B. Juarto and A. S. Girsang, "Neural collaborative with sentence BERT for news recommender system," *Int. J. Informat. Vis.*, vol. 5, no. 4, pp. 448–455, 2021.

CHANAPA CHANNARONG received the B.S. degree from Chulalongkorn University, Thailand, in 2020, where she is currently pursuing the master's degree with the Department of Mathematics and Computer Science, Faculty of Science. Her research interests include recommender systems and machine learning.

CHAWISA PAOSIRIKUL received the B.S. degree from Chulalongkorn University, Thailand, in 2020. She is currently an Application Development Analyst (Frontend Developer) at Accenture. Her research interests include recommender systems and machine learning.

SARANYA MANEEROJ received the B.S. degree from Chulalongkorn University, Thailand, in 1996, and the M.E. and Dr.Eng. degrees from the University of Electro-Communications, Japan, in 2001 and 2005, respectively. She is currently an Associate Professor at the Department of Mathematics and Computer Science, Faculty of Science, Chulalongkorn University. Her research interests include recommender systems and data mining.

ATSUHIRO TAKASU received the B.E., M.E., and Dr.Eng. degrees from the University of Tokyo, Japan, in 1984, 1986, and 1989, respectively. He is currently a Professor at the National Institute of Informatics, Japan. His research interests include data engineering and data mining.