Hybrid Recommender Models in E-Learning: A Comprehensive Review of HybridBERT4Rec

UNIVERSITÄT MANNHEIM

Leon Knorr[1]
[1]Mannheim Master of Datascience

Student ID:1902854
Keywords: Education, AI, E-Learning
10th January, 2024

1 Introduction

In the dynamic landscape of today's knowledge-driven society and global economics, the concept of life-long learning has evolved into a cornerstone for personal and professional development. It ensures the competitiveness of individuals in a globalized market, while also aiding in overcoming social challenges, such as demographic change, social cohesion or public health through continuous education. Because of its key-role in overcoming today's challenges, many governments and corporations invested heavily in lifelong learning offerings and frameworks [1]. This investment can be seen by observing the vast landscape of different learning platforms, which have emerged over the last couple of years, ranging from publicly funded E-Learning solutions such as Moodle[2] or ILIAS[3], to private corporate platforms such as Linked-In Learning[4]. As individuals embark on continuous learning journeys on such platforms, the demand for tailored educational experiences has surged. These experiences not only include the recommendation of new content, but also content which aims to aid the user with current learning objectives. Recommender algorithms play a pivotal role in shaping these learning odysseys by providing personalized content recommendations [5].
In this work, the hybrid content recommendation system HybridBERT4Rec, which uses a transformer based approach to collaborative and content-based filtering techniques, is reviewed and adapted to an E-Learning use case.

2 Related Work

3 Sequential Content Recommendation

Traditional content recommendation systems usually observe user interactions as a set of independent and unrelated data points. These systems then compute or form a hidden representation in order to model user interests, which can then be used to predict which items may be relevant for the given user. If an item is considered relevant, it gets recommended. This modeling technique models a user's *general* preferences and interests [6]. But, this is insufficient modelling, as user preferences change over time [6]! For example, let Alice be a user whose general interests reflect romantic films, such as Titanic, Romeo & Juliet or "me before you" and who recently got into the Marvel Universe, and started watching Movies like "Spider-Man". In addition, Alice is now really interested in following up with that series. Alice's movie history contains a total of 10 movies, 9 romantic movies and one Spider-Man movie, with the latter being the most recently watched. A traditional content recommendation system would observe Alice's movie history and model her interests as consisting of 90% romantic films and 10% Marvel. As a result, the recommendation system would assign romantic films

a higher relevance score than a movie that is similar to Spider-Man, leading to unsatisfying recommendations for Alice's *current* interests. Sequential Content Recommendation aims to solve this problem by observing user-interactions as a sequence. Thus, these models not only consider the items in the user history, but also the temporal aspect given by the order in which the items occur [6]. This ordering implies that more recent interactions are more relevant for recommending the next item, but at the same time also covers a user's general interest. As a result, Sequential Recommendation models are able to make accurate recommendations at any point in time, that also reflect temporary spikes of interests, while at the same time being able to recommend content that the user may generally be interested in. By choosing sequences as their data model, these models also gain a lot of flexibility in terms of the use cases they are able to cover. Because Sequential Recommendation models do not predict the relevance of items directly, but predict user-interaction probabilities at a given time-step, they are able to not only make recommendations for the present, but also for the future [6]. This allows these models to predict sequences of items, which can be used to craft recommendation series. E.g. a movie recommendation system can then recommend a complete series of movies which fit to Alice's interest in Marvel films, by predicting not only the next movie the user is likely to interact with but the next n movies. These reasons, render Sequential Recommendation models theoretically superior to traditional recommendation models in terms of content recommendation.

A common approach to realizing Sequential Recommendation models is by using Recurrent

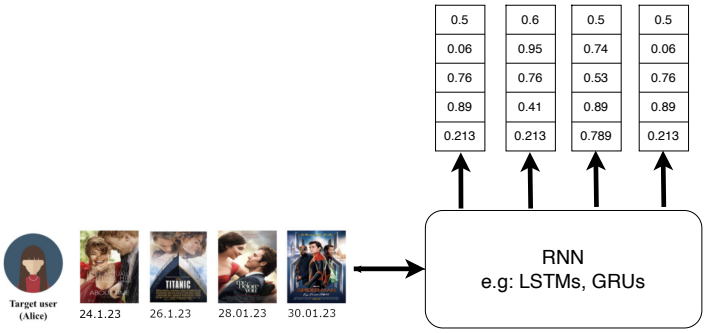

Fig. 1. Sequential Content Recommendation with RNNs [7]

Neural Networks (RNNs), such as Long-Short-Term-Memory (LSTMs) or Gated-Recurrent-Units (GRUs), as depicted in Figure 1. The RNN gets the sequence of user-interactions or items as input, in this case movies, and encodes them into a fixed length vector representation, which gets updated at every time-step in the sequence. The model then predicts the interaction probabilities or rating scores for every item which may be recommended at the next time step [7]. An example, for such an application can be found in Fung Yu et al. DREAM model [7]. However, these models suffer from common RNN problems, such as catastrophic forgetting, vanishing gradients and uni-directionality, as well as their inefficiency when dealing with longer sequences. This motivates the use of transformer based models, more specifically HybridBERT4Rec, which allows to use a sequence based approach to content recommendation without the downsides of RNNs [8].

4 HybridBERT4Rec

HybridBERT4Rec is a hybrid recommendation model, which was originally developed by Chanapa Channarong et al. [8] on a movie recommendation task [8]. It uses a combination of

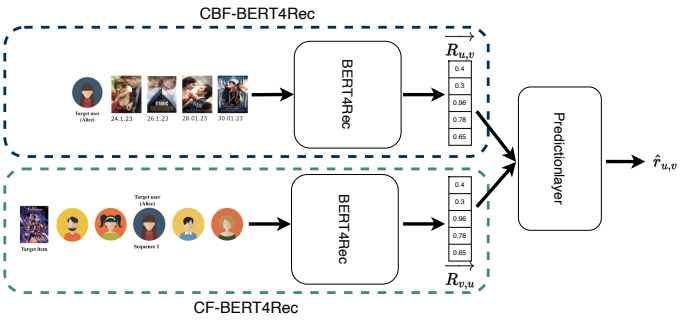

Fig. 2. High level overview of HybridBERT4Recs Architecture. [8]

Collaborative Filtering (CF) and Content Based Filtering (CBF) methods, in order to predict a relevance score for a given target item. As shown in Figure 2, it consists of three main parts:

1. CBF-BERT4Rec: Models Content Based Filtering
2. CF-BERT4Rec: Models Collaborative Filtering
3. Prediction layer: Combines both approaches and predicts the final relevance score $\hat{r}_{u,v}$

Both the CBF- and CF-Part of the model build upon HybridBERT4Recs predecessor BERT4Rec, and use the exact same architecture, while working with different inputs, in order to model their respective filtering techniques.

4.1 BERT4Rec

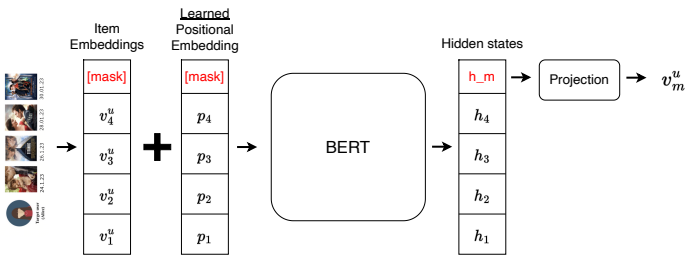

Fig. 3. BERT4Rec Architecture, taking item embeddings v_t^u from user u's history as input and predicts the next item v_m^u, u is likely to interact with [9].

BERT4Rec is a content recommendation model, which is based on the Bidirectional Encoder Representations from Transformers (BERT) model in order to facilitate Content Based Filtering. It has been developed in order to predict the next item a user is likely to interact with from a sequence of item embeddings from a user's interaction history. As depicted in Figure 3 The model consists of a learned positional embedding layer, an unmodified BERT transformer and a projection layer. In order to train BERT4Rec, Fei Sun et al. [9] applied Masked Language Modelling, a common training technique from Natural Language Processing and transferred it to the domain of recommendation models. As a result, during training, one or more items in the input sequence are masked at random and replaced with a special `[mask]` token. Then, the learned positional embeddings are added to the item embeddings, and the sequence gets passed on to the standard BERT transformer. The resulting hidden representation, of the masked item(s) is then passed through the prediction layer, which consists of a two layer feed forward neural network with GELU activations and a softmax layer, as given by equation 1.

This layer then produces an output distribution over the target items the model is supposed to rank.

$$P(v) = \text{softmax}(\texttt{GELU}(h_t^L W_p + b_p) E_T + b_O)[9] \tag{1}$$

In Equation 1, h_t^L marks the hidden representation of the masked token, W_p is the weight matrix of the linear layer, b_p and b_O are bias terms and E_T is the item embedding matrix, which is also used to convert the items in the input sequence to item embeddings. The embedding matrix was reused in order to alleviate overfitting and reduce model size [9].
This training objective allows the transformer to learn strong and meaningful item representations from their contexts. In the domain of recommendation models, this translates to learning how to construct the item representation of the next optimal item a user would want to interact with from the user's history. The prediction layer then essentially calculates similarity scores between this optimal item representation and the target items.
Despite its upsides, using BERT4Rec in a standalone manner, still comes with a limitation. As the prediction is solely based on extracting user historical patterns which can be seen as characteristics the user favors, the model can't recommend items from categories that aren't included in the user history already. In other words, it can't help a user discover new content, which is a general downside of using CBF in a standalone setting [8]. HybridBERT4Rec overcomes this issue by taking a hybrid approach to content recommendation by not only focusing on CBF but also using Collaborative Filtering techniques. But, before presenting how HybridBERT4Rec achieves Collaborative Filtering, the usage of BERT4Rec is put into context by laying out how HybridBERT4Rec achieves CBF using BERT4Rec in the following subsection.

4.2 CBF-BERT4Rec

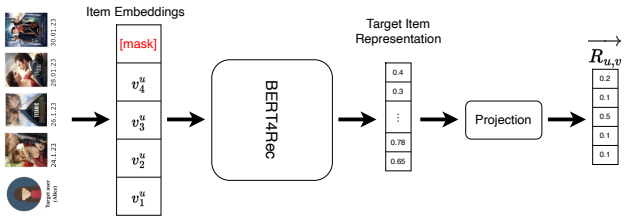

Fig. 4. CBF-HybridBERT4Rec Architecture, taking item embeddings from a user u history as input and predicts a "target user profile $\overrightarrow{R_{u,v}}$" [8].

As laid out earlier, the responsibility of CBF-BERT4Rec is to realize Content Based Filtering. In other words, it aims at extracting the target user representation, which describes the users preferences and interests. As shown in Figure 4, it achieves this by deploying BERT4Rec directly without any modifications. The input to the model is then given by the sequence of items in the target users history in chronological order. Then random item(s) are masked during the training process, during inference, the `[mask]` token is appended to the sequence and the model calculates the hidden representation for the masked token. This representation is then passed through the projection layer, as described earlier. The resulting distribution is then a distribution of all items over the "optimal item", expressed as the interaction probability of the target user with all items. Chanapa Channarong et al. [8] call this distribution the "target user profile $\overrightarrow{R_{u,v}}$"

4.3 CF-BERT4Rec

In order to overcome the aforementioned limitations of only using CBF, HybridBERT4Rec also uses CF techniques. Implementing CF is the main responsibility of the CF-BERT4Rec part of HybridBERT4Rec. CF-BERT4Rec achieves collaborative filtering by constructing the representation of a target item based on the target user and its set of neighbors. As shown in Figure 5, it also deploys BERT4Rec unmodified similar to CBF-BERT4Rec, but it uses completely different data. The input to CF-BERT4Rec is constructed by creating a sequence of all users who have interacted with the target item in chronological (in the case of movies, this is equivalent to all users who have rated the target movie). This may or may not include the target user. Again during inference, the `[mask]` token is appended to the sequence, and it is passed through BERT4Rec. The model will then construct a representation of the target item, reflecting the "optimal user" for this item, based on the neighboring users in the user sequence, which is passed on to the prediction layer. The prediction layer then yields a distribution of all neighboring users over the "optimal user", expressed as user-similarity probabilities between the neighboring users and the target user, which is also called the "target item profile $\overrightarrow{R_{v,u}}$" [8].

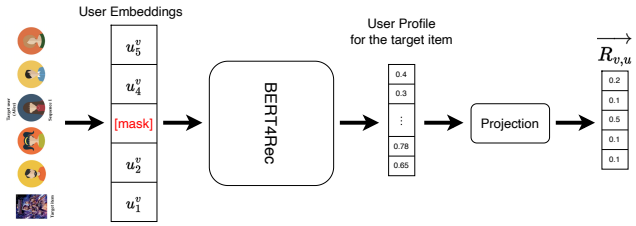

Fig. 5. CF-HybridBERT4Rec Architecture, taking user embeddings from all users that have rated the target item v as input and predicts the "target item profile $\overrightarrow{R_{v,u}}$" [8].

This approach to collaborative filtering is quite unusual, as modeling the *set* of neighbors as a sequence is questionable. The argument that the set of neighbors change over time and some might be more important than others does not hold in the same way as with content based filtering in collaborative filtering. Even though a users preferences might change overtime, and thus his current interests might not match with his past interests anymore, and he might thus not be a close neighbor anymore, his inclusion in the set of neighbors with an equal weighting is still valid. The reason for this is, that the primary goal of the inclusion of Collaborative Filtering is to enable *exploration*. The recommender system is supposed to help the user explore new content by sacrificing the overall precision of the recommendations. Thus, the user who has moved from his past interests, is still as important as before, as the target user might be interested in following this user's direction as the system might not be aware of a specific interest of the target user. Because of that, the inclusion of a transformer based model in this case is questionable, as collaborative filtering in general only requires the calculation of user-user similarities between neighbors and the target user [8]. As a result, the collaborative filtering part could have been implemented with a smaller and more efficient model, than with BERT4Rec. The authors also don't provide much reasoning as to why the use of a transformer based sequence recommendation model is beneficial for collaborative filtering [8].

4.4 Prediction Layer

The last layer in HybridBERT4Rec is the Prediction Layer. It combines the target item profile $\overrightarrow{R_{v,u}}$ from the CF-part with the target user profile $\overrightarrow{R_{u,v}}$ from the CBF-part of the model

through Generalized Matrix Factorization (GMF), in order to predict a final relevance score $\hat{r}_{u,v}$, which target user u would assign to the target item v. The computation performed by this layer is given by Equation 2, where $\odot$ denotes the element-wise product of both vectors [8].

$$\hat{r}_{u,v} = \sigma(WR(u,v) + b), \text{ with } R(u,v) = R_{uv} \odot R_{vu} \tag{2}$$

In contrast to traditional Matrix Factorization, generalized matrix factorization is based on neural networks with sigmoid activations, which are trained with a log-loss. This allows the model to learn the interaction function between the user- and item profile from data. This grants the model the ability to express a form of Matrix Factorization that isn't uniform and non-linear. In other words, it is able to assign different weights to different latent dimensions of the user-item vector $R(u,v)$ and is able to model non-linear and more complex structures than traditional Matrix Factorization [10].

4.5 Strengths & Weaknesses

This transformer based architecture comes with several strengths:

1. **Highly parallelizable:** The model can be efficiently parallelized in several ways. First, it inherits the sequential in depth characteristics of transformer models. This means that the models internal computations can be efficiently parallelized on a per-layer basis, as the computation for time step t at layer l is independent of the result of the computation for time step $t+1$ or $t-1$ at layer l.

2. **Bi-directionality:** Because HybridBERT4Rec is based on BERT, it inherits BERTs Bi-directionality. This allows the model to compute more expressive and meaningful representations of items, as these representations are not only dependent on past contents ($t < t_{present}$) but also on future ($t > t_{present}$) contents, providing a more sophisticated and information rich context for each token in the input sequence.

3. **Sequential modelling:** It inherits the strengths of sequential content recommendation models, as explained in Section 3. However, as layed out in Section 4.3, the use of a sequential approach to Collaborative Filtering remains questionable.

4. **Independent Execution & Caching:** HybridBERT4Recs CF- and CBF-part can be executed independent of each other, as no data flow is required between both parts. If this is combined with the ability to cache the target item- and user- profiles, then the model allows for dynamic profile calculations whenever needed. For example, it is only necessary to calculate the target user profile once for each item and similarly, the target item profile once for each user, as long as there is no update to either of them. If an update to one of them occurs, for example user u watches a movie, and it is appended to his history, then *only* the target item profile for user u needs to be updated. There is no need to run the model for every user and every item again. At recommendation time, the cacheability of the user- and item-profiles also reduces the models time and hardware requirements, as only the comparatively small and shallow prediction-layer needs to be executed in order to retrieve new recommendations. Both, independent execution and caching grant the model a high flexibility in terms of model deployment and applications.

Despite its numerous strengths, the model also includes some significant weaknesses:

1. **Limited Sequence Length:** Inheriting the characteristics of transformer models also means inheriting their weaknesses, in particular the limited sequence length. Because

the per-layer complexity of the self-attention mechanisms of $O(n^2 * d)$, where n denotes the sequence length and d the representation dimension, self-attention architectures scale badly in terms of sequence length [11]. This combined with memory constraints of modern hardware limits the sequence length of any transformer based model. In the domain of recommendation systems, this translates to limiting the maximum length of user histories, and potential neighbors to form both the target user- and target item profiles. In the case of the movie recommendation example, used throughout this paper, it limits the length of a users watch history and the maximum number of ratings that can be taken into consideration in order to form a user's potential neighbors.

2. **Hardware Requirements:** Because of HybridBERT4Recs hybrid nature, combining both Collaborative Filtering and Content Based Filtering, it also uses two seperate BERT based transformer models to model each approach. In the worst case, this translates to executing a transformer model for each of our users, another transformer model for each of our items and the prediction layer for each user-item combination, *if intermediate results are cached*! If no caching is present, both transformer models and the prediction layer would have to be executed *for every user and item combination*. In both cases, the model requires lots of processing power and memory in order to run. However, both processing and memory requirements do decrease significantly if the aforementioned optimization opportunities for dynamic recalculations, independent execution and caching are used. In addition, transformer models need comparatively large datasets if trained from scratch [8, 9, 11].

4.6 Performance & Experiments

Chanapa Channarong et al. [8] evaluated HybridBERT4Rec on three different datasets:

- MovieLens1M [12]: A movie recommendation dataset aimed at building and evaluating recommendation algorithms. It is composed of one million ratings from 6000 users on 4000 movies, with each user being responsible for at least 20 ratings.
- Yelp [13]: A dataset consisting of eight million ratings of restaurant and service business reviews, where each user created at least four ratings.
- GoodReads [14]: A book review dataset consisting of one million ratings, where each user has been responsible for at least 70 ratings.

As evaluation metrics, they adopted the Hit Ratio (HR) of the top k predictions, which measures the ranking accuracy, and the Normalized Discounted Cumulative Gain (NDCG), which measures the recommendation list quality of the top k positions. Along HybridBERT4Rec, they reported the performance of four additional models:

- Caser [15]: Caser is a unidirectional CBF approach using Convolutional Neural Networks (CNNs), which predicts the top-N ranked items that a user will likely interact with.
- GRU4Rec [16]: GRU4Rec is a unidirectional CBF approach using Recurrent Neural Networks (RNNs), more specifically Gated Recurrent Units, which predicts the next item embeddings at every time step.
- SAS4Rec [17]: SAS4Rec is a unidirectional CBF approach, which uses a transformer-based architecture to predict the next item a user will interact with.

- BERT4Rec [9]: As covered in Section 4.1, it is a bidirectional transformer based architecture, which predicts the next item a user is likely to interact with.

The results for NDCG@50 reported by Chanapa Channarong et al. [8] are depicted in Figure 6. In their experiments, HybridBERT4Rec outperforms all other models across all three datasets,

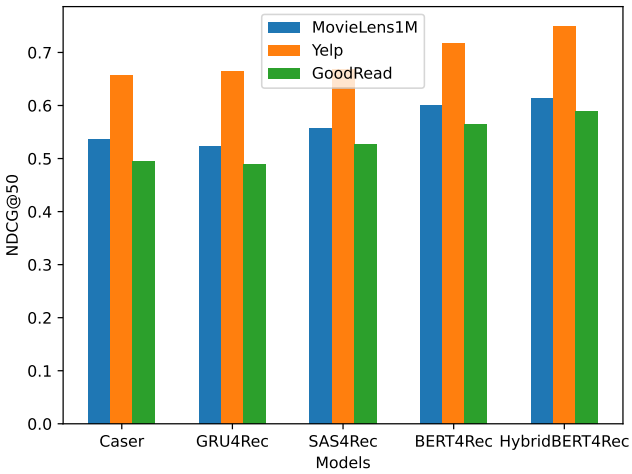

Fig. 6. Performance comparison of different recommender models on three datasets as published by the authors of HybridBERT4Rec [8].

with BERT4Rec being the second-best model and Caser performing the worst. The same result was obtained for the HR and all other values of k for both metrics, hence these results haven't been directly reported in this paper again. For a detailed look at the numbers, please refer to Channarong et al. original paper [8]. The results reported are not surprising, given that the *Hybrid* bidirectional HybridBERT4Rec model was evaluated solely against unidirectional (except for BERT4Rec) *Content Based Filtering* models. With that, the experiments are not representative in order to evaluate HybridBERT4Rec performance. The main takeaways presented by the experiments are that BERT4Recs bidirectional approach performs better than its unidirectional competitors and that the inclusion of Collaborative Filtering into Content Based Filtering models is effective and can increase recommendation performance. But, it remains unclear how HybridBERT4Rec performs in comparison to other hybrid recommendation models.

In addition, the authors don't provide much information about how the data for training and testing has been partitioned for these experiments. They mention, that some authors had to be pruned from the datasets, because their histories were too short for model training, but other than that, no further information is provided. As a result, HybridBERT4Recs generalization performance also remains unknown, along with questions like: Does it suffer from the cold start problem? How does it handle new and unseen items / users? How does it handle domain transfer after training? Does it inherit the fine-tuning capabilities transformers are well known for [18]? Without these questions answered the real world applicability of this model is unknown, as it is unclear how it would handle such a dynamic and fast-paced environment.

5 Applying HybridBERT4Rec in an E-Learning Environment

In this section, HybridBERT4Rec is going to be transferred into an E-Learning Environment. First, a trivial solution resolving around Video-content based E-Learning platforms such as Linked-In Learning [4] is presented. After that, a more involved setting, which reflects the use on an E-Learning platform like ILIAS [3], is defined, followed by a discussion about how HybridBERT4Rec could be adapted and evaluated.

5.1 The Trivial Solution

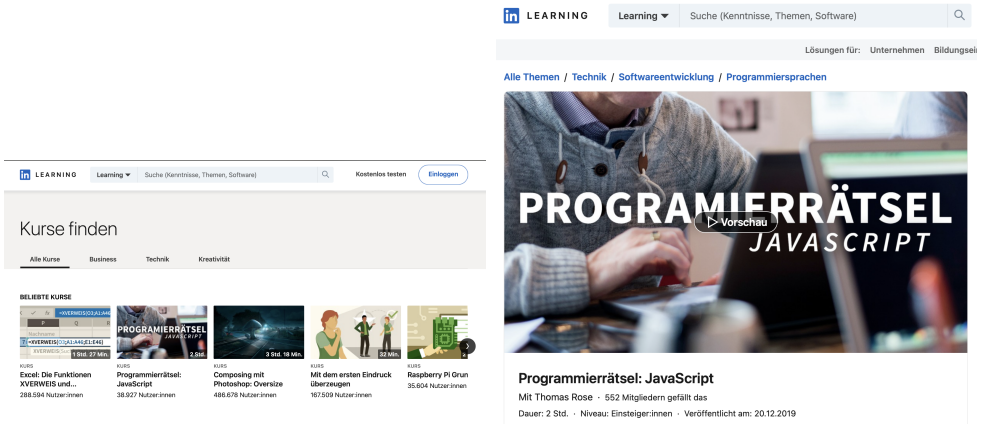

Fig. 7. Linked-In Learning landing-page and course overview [4].

As HybridBERT4Rec was originally developed and tested on the MovieLens1M [12] dataset, applying it in a video based setting which logs user watch histories and rating is trivial. An example for such an environment is Linked-In Learning [4]. As shown in Figure 7, Linked-In Learning offers video based learning courses, which can be "liked", it also logs a history including which courses have been started and completed. In this setting, the history logged by the platform can be used directly to form the inputs for the CBF-BERT4Rec part of HybridBERT4Rec, similarly as with the original movie data. For the CF-Part, the "like" of a course can be seen as rating information. With that, every user who liked a course can be defined as a neighbor and be used in order to construct the user sequence for the CF-Part. As a result, the CF-Part can also be used directly, showing that HybridBERT4Rec can be applied to such a scenario without any necessary changes or adaptations, due to its similarity to movie recommendations.

5.2 Recommending Exercises to Create Personalized Learning Experiences for Students

However, if the E-Learning environment is more involved, HybridBERT4Rec can't be applied directly anymore without carefully defining its input and output data. To showcase such a scenario, an exercise recommendation task in an inverted classroom setting is considered. In this setting, students watch videos about different topics, which are clustered into Learning Objectives. After working through the material for each Learning Objective, students are asked to assign the Learning Objective a difficulty level. The recommendation algorithm should then recommend exercises to the student (user) which (presumably) fit their skill level. A more

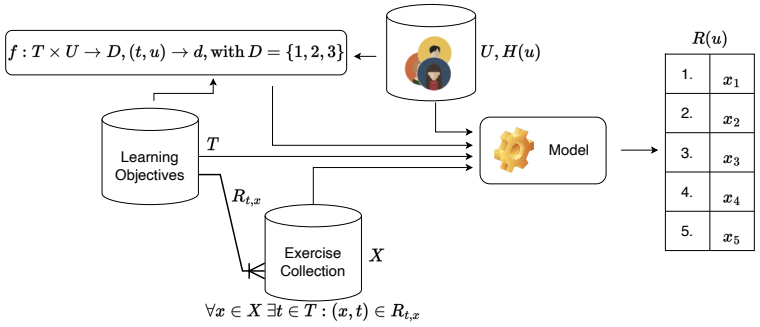

Fig. 8. The Setting, consisting of a user collection U and their histories $h(u)$, a collection of learning objectives T and a collection of exercises X, which can be used to predict a ranking $R(u)$ for a given user u.

formal definition of this setting is shown in Figure 8. Let U represent the platform's user collection, encompassing every user (student) on the platform, alongside their exercise history $H(u)$. The collection T includes Learning Objectives, storing each learning objective with its associated videos available on the platform. Similarly, the exercise collection X comprises every exercise available on the platform. The difficulty rating d a user $u \in U$ assigns to learning objective $t \in T$ is then given by $f : T \times U \to D, (t, u) \to d,$ with $D = \{1, 2, 3\}$. Here, each number in D represents a difficulty level:

- 1: Easy, the student u encounters no issues solving complex tasks related to learning objective t,
- 2: Medium, the student u experiences some difficulty but has grasped the core principles assigned to learning objective t,
- 3: Difficult, the student u faces significant challenges in solving tasks related to learning objective t and has not yet understood the core principles.

It is important to note that the learning objective collection and exercise collection exhibit a one-to-many relationship denoted by $R_{t,x}$. This relationship signifies, that for every exercise x in the exercise collection, there must exist one learning objective t in the learning objective collection, such that the tuple (x, T) is in the relation $R_{t,x}$. In other words, every exercise has to be assigned to exactly one learning objective, whereas one learning objective can have multiple exercises assigned to it, this relation is then stored in $R_{t,x}$.
The task of the recommendation model is then to process the available data and produce a ranking $R(u)$ for target user u, which encompasses different exercises, which fit the user's difficulty level for different learning objectives.

5.3 Model Adaption

In order to adapt HybridBERT4Rec to work in the defined environment, no architecture changes are needed. But, the input sequences are not directly given by the use-case and have to be constructed from different parts of the available data. First, the input sequence for the CBF-Part of the model is constructed. For this, the exercise history of a user $u \in U$ has to be defined. Its definition is given in Equation 3.

$$H(u) := (\{(x_i, t_j, s_k) | (x_i, t_j) \in R_{t,x}\}, \leq) \tag{3}$$

The exercise history of a user u consists of triples in the form of (x_i, t_j, s_k), where x_i denotes the completed exercise, t_j its assigned learning objective, and s_k the timestamp of the time of completion. For every triple included in the history, it holds that if one would construct a tuple with the exercise x_i and the learning objective t_i, the tuple is included in the one-to-many relation $R_{t,x}$. Additionally, the history is ordered by the timestamp s_k, such that $s_{k-1} \leq s_k$. Using this definition, the input sequence $I(u)$ of user u for the CBF-part is then given by Equation 4.

$$I(u) := (\{x_i | (x_i, t_j, s_k) \in H(u)\}, \leq) \tag{4}$$

It is constructed by extracting each completed exercise, which is present in the history $H(u)$ of user u, while preserving the order of the exercises into a new sequence $I(u)$. In more formal terms, for each exercise x_i in the input sequence $I(u)$, it holds that a triple (x_i, t_j, s_k) is included in $H(u)$ and the sequence is ordered by s_k based on the triple included in $H(u)$, such that $s_{k-1} \leq s_k$. This results in an input sequence composed solely of exercises from the exercise collection, which the user has worked on, in ascending order. As a small note, it does not matter if the sequence is ordered in ascending or descending order, as long as it is ordered such that the neighborhood of x_i is preserved. The reason for this are self-attentions ordering preserving properties [11]. Orderings however, should not be mixed, as the positional encoding will otherwise lose its expressiveness.

Next the user sequence for the CF-Part of the model will be constructed. As in Channarong et al. original proposition, the input is still a sequence of users. However, the users who are considered neighbors have to be carefully selected. Simply considering every user who assigned a difficulty rating to the target learning objective is not sufficient. Because, then users who rated the objective as easy and are able to complete more complex assignments would also be considered neighbors to users who have rated the learning objective as difficult. As a result, the target user may be deemed similar to another user who rated the target learning objective as easy, even though the target user has rated it as difficult. For example, if two users have the same courses, and their difficulty ratings are very close, or even similar for most courses, the model could consider both users as very similar, even though one rated the learning objective as easy and the other one rated it as difficult. This would result in the recommendation of too complex exercises for one and too simple ones for the other user. Because of that, a more restrictive filtering criteria is imposed.

$$u \in N \Longleftrightarrow d_{u,t} = d_{u_m,t} \wedge (x,t) \in \{(x,t) | (x,t,s_k) \in H(u)\} \tag{5}$$

As given in Equation 5, a user is in the set of Neighbors N for target user u_m and the target learning objective t *if and only if*, the difficulty rating $d_{u,t}$, u has assigned to t, is equal to the difficulty rating $d_{u_m,t}$, u_m has assigned t *and* if the tuple (x,t) of the target exercise x and learning objective t is included in the user's history. This filtering criteria restricts the set of neighbors to include only users who gave the same difficulty rating to the target learning objective as the target user. Additionally, it only considers users who have completed the specific exercise that the system aims to rank. In essence, this limits the set of users who can receive a high similarity probability to users who have the same level of difficulty with the same learning objective and who already studied for the target learning objective. This approach provides the model with the capability to recommend exercises that have proven beneficial for users facing similar difficulties, including exercises from other learning objectives.

As a final step, the execution of the model needs to be adapted to the given setting. Algorithm 1 describes how the different parts of the model need to be executed, in order to retrieve the rating which the target user u_m would assign to the target exercise and learning objective (x,t). First, for every user u_m in the user collection U, the CBF-part `cbf_bert4rec` needs to

Algorithm 1 HybridBERT4Rec in an E-Learning Setting

for all $u_m \in U$ **do**
  $r_{x,u_m} = \texttt{cbf\_bert4rec}(H(u_m))$
  for all $(x,t) \in R_{t,x}$ **do**
    $r_{u,x} = \texttt{cf\_bert4rec}(u_m, t, x)$
    $\hat{r}_{u,x} = \texttt{prediction\_layer}(r_{x,u}, r_{u,x})$
  end for
end for

be executed resulting in the target user profile r_{x,u_m}. Then, for every tuple of target exercises and target learning objectives (x,t) present in the Relation $R_{t,x}$ the CF-part `cf_bert4rec` is executed, resulting in a target item profile $r_{u,x}$. Directly after that, the prediction layer is executed using both the target user and item profile in order to retrieve the final relevance score $\hat{r}_{u,x}$, which the target user u_m would assign to the target exercise x. Because of the restrictive filtering objective, which is dependent on the target user u_m, the dynamic independent execution and caching strategies presented in Section 4.5 can't be applied to the same extend in this setting. The complexity for a change in the user history is $O(x)$, with x being the number of exercises in the exercise collection, because all target item profiles also need to be recomputed. For the movie recommendation task on the other hand the complexity is $O(1)$, as both parts can be executed independently during inference. The complexities for a change in the set of neighbors remain $O(1)$ for both tasks. As a result, the model can't be deployed as efficiently in this E-Learning setting as for movie recommendations.
However, this model allows to compose different rankings with only one execution of Algorithm 1. An overall ranking of exercises can be constructed, by sorting the exercises by their relevance scores independent of their assigned learning objectives. This could give students an overview on which topics and exercises should be prioritized in their studies. Additionally, a learning objective specific ranking can also be provided, by grouping the rated exercises by learning objective and sorting them by their relevance scores inside each group. This gives students the opportunity to pick a specific topic / learning objective that they want to study or improve.

5.4 Solving Evaluation

Evaluating the model in the given setting is difficult, as no binary or graded relevance annotations are available. Annotating the whole dataset is also infeasible, as this would require $U \times T \times X$ relevance annotations. Because, annotations are needed for every user, learning objective and exercise combination, if the annotations should also reflect the relevance of exercises for a learning objective which they are not related to, e.g. exercises from statistics might be helpful for understanding the Bayes rule. If this should not be reflected in the dataset, then $U \times X$ relevance annotations would be needed. Still, this is infeasible for large collections. In order to reduce the annotation workload, the method of pooling can be used [19]. The fundamental idea behind pooling, is that for most queries, in this case user, learning objective combinations, only a tiny fraction $N << X$ of exercises is actually relevant. An ideal recommendation system would rank said exercises at the top of the ranking [19]. Because of that, it can be sufficient to annotate only the top n results of every ranking. As a result, only $U \times T \times N$ or $U \times N$ annotations are needed. Using the resulting test set, common metrics for recommendation systems can then be computed, such as the P@k (Precision@k), R@k (Recall@k), NDCG, AP (Average Precision), MAP (Mean Average Precision) etc. However, a shortcoming of such an approach is, that it is not given that all relevant documents are included

in the top n positions of the ranking. As a result, it is possible, that relevant documents aren't annotated and are not considered in the evaluation. Thus, this method only provides an approximation of the recommendation models performance [19]. Nevertheless, using pooling marks a great balance between annotation work and evaluation accuracy if configured well.

6 Conclusion

ABC

7 References

1. Rubenson, K. *Adult Learning and Education* (Academic Press, Feb. 2011).
2. *Startseite — Moodle.org* https://moodle.org/.
3. *Ilias.De* https://www.ilias.de/.
4. *LinkedIn Learning mit Lynda: Onlinekurse aus dem Business-, Technik- und Kreativbereich* https://de.linkedin.com/learning/.
5. Jeevamol, J. & Renumol, V. G. An Ontology-Based Hybrid e-Learning Content Recommender System for Alleviating the Cold-Start Problem. *Educ Inf Technol* **26,** 4993–5022 (July 2021).
6. Wang, S., Hu, L., Wang, Y., Cao, L., Sheng, Q. Z. & Orgun, M. Sequential Recommender Systems: Challenges, Progress and Prospects, 6332–6338 (2019).
7. Yu, F., Liu, Q., Wu, S., Wang, L. & Tan, T. *A Dynamic Recurrent Model for Next Basket Recommendation* in *Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval* (Association for Computing Machinery, New York, NY, USA, July 2016), 729–732.
8. Channarong, C., Paosirikul, C., Maneeroj, S. & Takasu, A. HybridBERT4Rec: A Hybrid (Content-Based Filtering and Collaborative Filtering) Recommender System Based on BERT. *IEEE Access* **10,** 56193–56206 (2022).
9. Sun, F., Liu, J., Wu, J., Pei, C., Lin, X., Ou, W. & Jiang, P. *BERT4Rec: Sequential Recommendation with Bidirectional Encoder Representations from Transformer* Aug. 2019. arXiv: 1904.06690 [cs].
10. He, X., Liao, L., Zhang, H., Nie, L., Hu, X. & Chua, T.-S. *Neural Collaborative Filtering* Aug. 2017. arXiv: 1708.05031 [cs].
11. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. & Polosukhin, I. *Attention Is All You Need* Dec. 2017. arXiv: 1706.03762 [cs].
12. *MovieLens 1M Dataset* https://grouplens.org/datasets/movielens/1m/. Sept. 2015.
13. *Yelp Dataset* https://www.yelp.com/dataset.
14. Wan, M., Misra, R., Nakashole, N. & McAuley, J. *Fine-Grained Spoiler Detection from Large-Scale Review Corpora* in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics* (Association for Computational Linguistics, Florence, Italy, 2019), 2605–2610.
15. Tang, J. & Wang, K. *Personalized Top-N Sequential Recommendation via Convolutional Sequence Embedding* Sept. 2018. arXiv: 1809.07426 [cs].

16. Hidasi, B., Karatzoglou, A., Baltrunas, L. & Tikk, D. *Session-Based Recommendations with Recurrent Neural Networks* Mar. 2016. arXiv: 1511.06939 [cs].

17. Kang, W.-C. & McAuley, J. *Self-Attentive Sequential Recommendation* Aug. 2018. arXiv: 1808.09781 [cs].

18. Radford, A., Narasimhan, K., Salimans, T. & Sutskever, I. Improving Language Understanding by Generative Pre-Training.

19. Suarez, D. P. O., Glavaš, P. D. G. & Ponzetto, P. D. S. P. *Information Retrieval & Web Search* Sept. 2022.