Personalization through User Attributes for Transformer-based Sequential Recommendation

ELISABETH FISCHER, Data Science Chair, Julius-Maximilians University Würzburg
ALEXANDER DALLMANN, Data Science Chair, Julius-Maximilians University Würzburg
ANDREAS HOTHO, Data Science Chair, Julius-Maximilians University Würzburg

ACM Reference Format:
Elisabeth Fischer, Alexander Dallmann, and Andreas Hotho. 2023. Personalization through User Attributes for Transformer-based Sequential Recommendation. 1, 1 (July 2023), 13 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

A successful recommender system needs to learn a user's preferences to give relevant recommendations. A number of neural network architectures has been shown to learn and model the interest of a user from sequences of user interactions successfully, starting from Recurrent Neural Networks (RNNs) [12] and Convolutional Neural Networks (CNNs) [21] up to transformer based models like SASRec [14] and BERT4Rec [19]. These models are especially useful in recommendation settings where the user is anonymous, as they only rely on the history of previous interactions and don't require any other information about the user. However, often additional data characterizing the items or the user is present, which can be explored to improve the recommender's predictions. The combination of user attributes and sequence information could help with the cold-start problem, but especially user attributes have not been explored much yet. For a user returning after a long time with few or none recent interactions, attributes from the user profile could still produce meaningful recommendations and anonymous users or users without profiles could still receive recommendations based on their interaction.

The inclusion of additional item information has already been shown to improve recommendation in several models. For example, in [23] a 3D-CNN is used to learn a character-wise text-representation of the item and in Novabert [18] embeddings of categorical item attributes are integrated directly in the transformer layer as key and query. In contrast it is hard to find work on the inclusion of user attributes, not the least because of the lack of publicly available datasets, as releasing user information is often a privacy issue. However, there is the work of Chen et al. [3], who propose an uni-directional transformer model using user profile information to increase the click-through-rate, although the paper doesn't focus on the user information. They embed the sequence with a transformer and concatenate an embedding of the user and other information to the output of the transformer.

While their setup is similar, our goal is the prediction of the next item and our approach to the problem differs as follows: At each step of the sequence the user decides which item to take, therefore the representation of the user in

Authors' addresses: Elisabeth Fischer, elisabeth.fischer\spacefactor\@m{}informatik.uni-wuerzburg.de, Data Science Chair, Julius-Maximilians University Würzburg; Alexander Dallmann, dallmann\spacefactor\@m{}informatik.uni-wuerzburg.de, Data Science Chair, Julius-Maximilians University Würzburg; Andreas Hotho, hotho\spacefactor\@m{}informatik.uni-wuerzburg.de, Data Science Chair, Julius-Maximilians University Würzburg.

Manuscript submitted to ACM

our model should also influence each item separately. Additionally, the user representation and the sequence contain different types of information, so it would be reasonable to treat them as different data types.

However, it has been already shown that transformers themselves have the capability to handle different types of data successfully. In VisualBert [17] image and language representations are used as combined input to the transformer, so the transformer can learn the relationship between both. Combining the user and the sequence in a similar way could allow the transformer to learn the influence of the user for each sequence step. Thus we want to investigate whether the transformer itself can handle merging the user information into the sequence and compare the performance to approaches merging the user information before and after the transformer.

We therefore propose a modification for including a representation of the user for transformer-based recommendation models. First, we create an embedding of the user's attributes in addition to the usual item embeddings. Then, we concatenate this user embedding to the beginning of the sequence of embedded item ids. This modified sequence now contains the user embedding as the first entry, followed by each item embedding and is used as an input for the transformer layer, so the attention between each item and the user can be computed.

To test our approach we adapt the well known BERT4Rec model as well as an uni-directional transformer model inspired by the SASRec model, but utilizing a cross entropy loss instead of a negative sampling based loss function as this loss performs better on our data. We also adapt two baselines: KeBERT4Rec merges the attributes pre-fusion, which means before the transformer layer creates the fused sequence representation. Inspired by [2] we create anothere model using a post-fusion merge of the attributes on the output of the transformer layer.

We evaluate our models on a proprietary dataset from a large fashion webshop. Additionally, we provide results for Movielens-1m, an publicly available movie recommendation dataset. The Movielens-1m dataset contains user attributes like gender, age and occupation that can be used for sequential recommendation. As both datasets contain no numerical attributes for the user, we limit our evaluation to the use of categorical attributes.

Overall, our contributions are as follows: 1) We propose a way of including categorical user attributes into transformer-based recommendation models with exemplary adaptations for BERT4Rec and a SASRec-inspired model. 2) We evaluate the performance on a real-life dataset from an ecommerce online shop and on the Movielens-1m dataset.

In Section 2 we give an overview over the related work for sequential recommendation. We define the task at hand in Section 3, followed by the description of our approach in Section 4. Our datasets are described in Section 5, while we present our experiments and results in Section 6. Finally, we conclude the paper in Section 7.

2 RELATED WORK

This section covers related work on sequential recommendation with a focus on deep learning methods and architectures using additional information for users and items.

Over time different neural network architectures have been introduced for modeling sequences of user interactions. GRU4Rec [12] was introduced by Hidasi et al. as the first model for sequential recommendation using Recurrent Neural Networks to model the item sequence and was later improved by introducing a new ranking loss in [11]. However, the authors of [20] show that a cross-entropy loss can perform even better. Other architectures proposed Convolutional Neural Networks, for example Caser [21] or combinations of RNNs and CNNs [26]. Originally introduced for tasks in Natural Language Processing, attention [24] has also been adapted for sequential session recommendation. The first model using attention to encode the sequence was NARM [16], which is based on a GRU layer. In contrast, SASRec [14] uses an uni-directional transformer to encode the sequence and was trained through sampling negative items for a binary cross-entropy loss. Inspired by BERT [8], the authors of [19] adapted the masked training task [22] and bidirectional

transformers for sequential recommendation. Their model BERT4Rec was able to outperform state-of-the-art methods on several datasets.

To further improve recommendation performance, several modifications of these models have been published, which try to leverage additional item information. For examples, in [23] CNNs are using 3D convolutions to include character-wise item descriptions. In [13] textual and visual item features are fed to different GRUs and output is combined for recommendation afterwards. In [2] the authors use the Latent Cross technique to included context in a GRU model in both a pre-fusion and post-fusion setting. For bi-directional transformer models, NovaBert [18] infuses item attribute embeddings directly in the transformer as query and value, while [9] allow the integration of categorical item attributes in BERT4Rec through adding the item embeddings in the embedding layer. The framework Transformers4Rec [7] transfers some more recent NLP models like XLNet [27] for sequential recommendation and allows the integration of categorical and numerical item features. Another type of information about the sequence and the items is explored in MEANTIME [5], where the authors embed the temporal information of the interactions. While these works focus mostly on additional item information, there are some papers taking special interest in the user. CASER [21] for example creates a separate user embedding from the user ids. A similar approach in combination with self-attention is done by [25], but both models only rely on the identity of the user and not on attributes of the user. FairSR [15] includes user attributes to make recommendations more fair and uses a CNN and the construction of a preference graph in a multi-task learning setting to achieve this. In [4] the authors propose a system for next basket recommendation, which models dynamic user attributes.

Most similar to our work is the work of Chen et al. [3], who use an uni-directional transformer and embed user attributes and other features separately from the sequence. The sequence including the target item is encoded with the transformer. The embedded features and the embedded item representation are concatenated and fed through three layers of MLPs for a binary classification of whether the target item is clicked. In contrast, we want to predict the next item, so the target item is not included in our input sequence. We only use the last output of the transformer for our prediction and concatenate the user and the sequence before the transformer layer, as we want investigate the transformers ability to learn the relations between the user and each separate item in the sequence.

3 PROBLEM SETTING

In this section, we introduce the problem setting and the notation used in this paper, which closely follows [9]. In this work we aim to solve the task of recommending items to a user given the sequence of previous item interactions. We define the set of users as $\mathcal{U} = \{u_1, u_2, \ldots, u_{|\mathcal{U}|}\}$ and the item set with $\mathcal{V} = \{v_1, v_2, \ldots, v_{|\mathcal{V}|}\}$. Therefore, for each user u who interacted at the relative time step t with an item $v_t^u \in \mathcal{V}$, we can denote the sequence of interactions as $s_u = [v_1, v_2 \ldots, v_n]$. The set of all sessions is defined as $\mathcal{S}_u = \{s_{u_1}, s_{u_2}, \ldots, s_{u_{|\mathcal{U}|}}\}$. For each user $u \in \mathcal{U}$ we also have information in the form of categorical attributes which we write as $a_u = \{a_1, a_2, \ldots, a_{|A|}\}$. With $\mathcal{A}$ we denote all possible attributes.

We can now formally define our task as the prediction of the next item v_{n+1} for every $v_t^u \in \mathcal{S}_u$ with the additional user information a_u.

4 USER ATTRIBUTES PERSONALIZATION FOR TRANSFORMERS

In this section we specify the details for our base models and our original approach leveraging user attributes. In our architectures we integrate only categorical user attributes into transformer-based models for sequential recommendation.

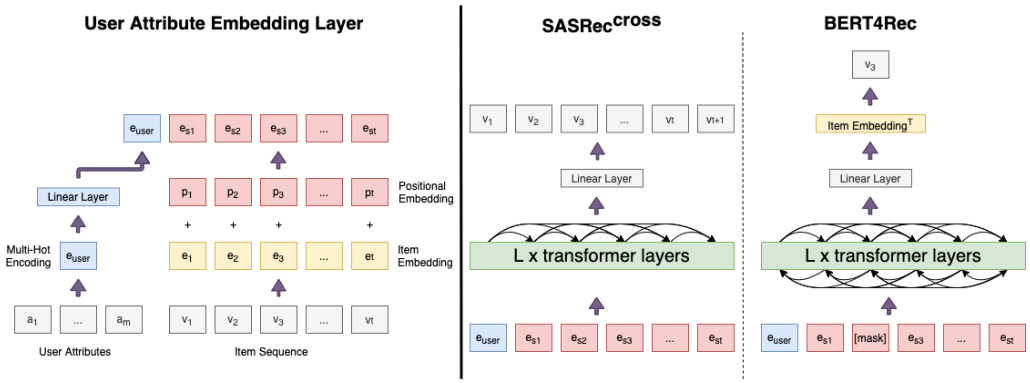


Fig. 1. The user attribute embedding layer and the modified architectures for BERT4Rec and SASRec. The user representation is prepended to the item representations in the embedding layer and fed to the transformer layers. For the transformer layers, SASRec contains only the forward connections, while BERT4Rec also contains backward connections.

We adapt two well known recommendation models for our experiments: SASRec [14] as one of the first recommendation models using transformers and BERT4Rec [19] as a bidirectional model with the Cloze training task.

The original SASRec^{neg} [14] uses a negative sampling loss for training. But in our preliminary experiments we noticed that the SASRec model was not able to learn much (all metrics smaller than the most popular baseline) for the Fashion dataset. As the cross-entropy loss has worked well for other types of recommendation models (BERT4Rec, GRUs) in [20] and [19], we change the training task: Instead of a binary classification with negative item sampling, we implement a classification of all items with a cross-entropy loss. We denote this architecture with SASRec^{cross} and the original implementation with SASRec^{neg}. We name the architectures including user attributes U-BERT4Rec and U-SASRec^{cross} respectively and show them in Figure 1. [1]

4.1 Layers

For both U-SASRec^{cross} and U-BERT4Rec the model consists of three different layers, like in their original implementations: (i) the *embedding layer*, which learns a representation of the input (i.e., the sequence of item identifier and the user attributes). The resulting representation is then fed to (ii) a *Transformer layer*, that consists of L layers of transformer blocks (as introduced in [24]), which the final output is put through (iii) the *projection layer*, which maps the output of the last transformer back to the item space using a linear layer with softmax activation function.

Embedding Layer: The embedding of the interaction sequence $s_u = [v_1, v_2 \ldots, v_n]$ follows the setup in BERT4Rec and SASRec with some modifications for the user. The embedding layer has a size of d and consists of the following parts: (i) the embedding $E_{\mathcal{V}} \in \mathbb{R}^{|\mathcal{V}| \times d}$ of the item identifier v and (ii) the embedding $E_P \in \mathbb{R}^{N \times d}$ of the timestep t, marking the position of the item in the sequence. This encodes the position for the transformers, with N being the maximum length of the input sequence. (iii) One multi-hot encoding of the user attributes $E_{|A|} \in \{0,1\}^{|A|}$ and (iv) a linear layer $l_A(x) = Wx + b$ with the weight matrix $W \in \mathbb{R}^{|A| \times d}$ and bias $b \in \mathbb{R}^d$ for scaling the encoded attributes E_A to the hidden size d.

The item embeddings $e_t = v_t E_{\mathcal{V}}$ and the positional embeddings $p_t = tE_P$ are summed up for each time step t. This gives us $e_s = [e_1 + p_1, e_2 + p2, \ldots, e_n + p_n]$ as the list of embedded items. The embedding of a user is created by applying

[1] Our code is available at notincludedyet.

l_A to the multi-hot encoding of a user's attributes, formalized as $e_u = l_A(E_A(a_u))$. We prepend the user embedding e_u to e_s, giving us $e = [e_u, e_{s_1}, e_{s_2}, \ldots, e_{s_n}]$ for the final output of the embedding layer and input for the transformer.

Transformer Layers: Each of the L transformer layer contains $N + 1$ transformers (maximum length of the sequence plus one for the user representation) with the hidden size d. In the U-SASRec^{cross} model, the connections between the transformer layers are unidirectional and pointing forwards in the sequence, while in the U-BERT4Rec model they are connected bidirectional.

Projection Layer: To map the output of the L-th transformer layer back to the item ids, a linear layer is used. As the implementation for U-BERT4Rec and U-SASRec^{cross} differs, the details for the projection layer and the training can be found in the following sections.

4.2 BERT4Rec

BERT4Rec uses the Cloze masking task for training. This means the hidden state of the L-th Transformer layer h_t^L at time step t is used to predict the masked item v_t. BERT4Rec uses a linear layer with the GELU activation function [10] and layer normalization, followed by the transposed item embedding $E_{\mathcal{V}}$ to project the output back to the item space. Therefore we use $o_0 = \sigma(h_t^L W_1)$ with $W_1 \in \mathbb{R}^{d \times d}$ for the linear layer with $\sigma = GELU$, followed by $o_2 = o_1 E_{\mathcal{V}}^T$ for the projection (bias ommitted for readability).

4.3 SASRec^{cross}

As we compute a full ranking over all items instead and use the normal cross-entropy loss to train the model, the projection layer changes as follows:

The hidden state of the L-th Transformer layer h_t^L at time step t is used to predict the item v_{t+1}. To compute the complete ranking over the items, we use the linear projection $o = h_t^L W$ with $\bar{W} \in \mathbb{R}^{d \times |\mathcal{V}|}$.

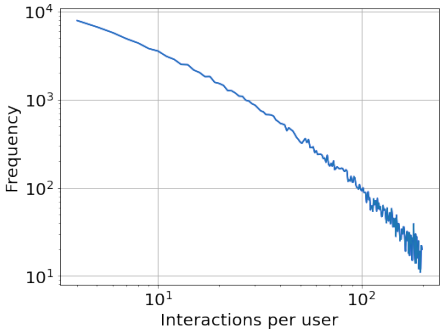
5 DATASETS

In this section we describe the two datasets used for evaluation. We give information on the preprocessing, some statistics and show the distribution of available user attributes.

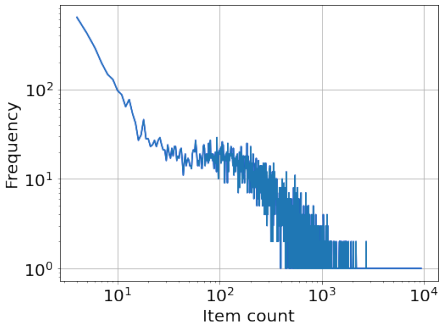
As an established dataset for sequential recommendation we use the Movielens-1m[2] dataset. Movielens-1m contains ratings for movies from an online platform and provides information about the users gender, age and occupation. Unfortunately, none of the larger datasets from Movielens give this information. We apply the same preprocessing steps as BERT4Rec [19]. For training, we use the commonly used leave-out-out split, where the penultimate item in a sequence is used for validation and the final item in a sequence is used for testing the model. The attribute distribution for the Movielens-1m dataset is shown in Figure 3b.

The second dataset (Fashion) contains user interactions with product pages from a big online fashion store collected over the duration of 45 days. The dataset only contains interactions with pages which are linked to a product, interactions with technical pages (e.g., account pages) and other content pages (e.g., overview) are not included. As for user attributes, we collect information from the following three categories: The gender, the age group and the shopper type. The gender can be one of male, female or other/unknown. The age is also a categorical value, as only a general age group is provided, e.g., users between 20-29 years are assigned the same age group. The shopper types define groups of users with specific interests or behaviours. This could be a higher interest in a specific brand or sports or the tendency to buy

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



(a) Distribution of the number of item interactions per user.



(b) Distribution of the number of interactions per item

Fig. 2. Item and session length distributions for the Fashion dataset.

Table 1. Statistics for the train, validation and test split of the Fashion dataset.

dataset	$\lvert\mathcal{U}\rvert$	$\lvert\mathcal{V}\rvert$	$\lvert\mathcal{A}\rvert$	#Interactions	Unknown items
Train	93,248	9,560	34	2,4m	
Validation	21,651	5,770	34	0,4m	1.7%
Test	46,391	7,429	34	1,1m	12.6%

Table 2. Statistics of the two preprocessed datasets Movielens-1m and Fashion.

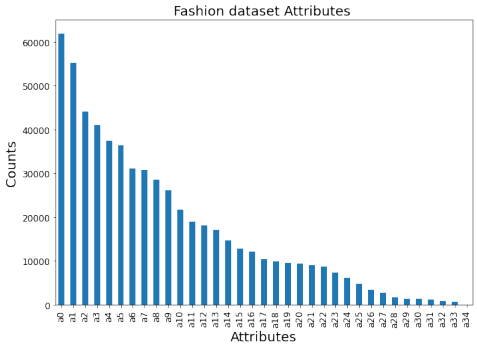
dataset	$\lvert\mathcal{U}\rvert$	$\lvert\mathcal{V}\rvert$	$\lvert\mathcal{A}\rvert$	#Interactions	Avg.length	Density
Movielens-1m	6,040	3,706	18	1m	165.6	4.46 %
Fashion	119,776	10,502	34	3.5m	26.4	0.02 %

mostly bargains and products on sale or a mix of different characteristics. Therefore, a user can fall into multiple of these shopper types at once. Figure 3a shows the distribution of user attributes over the complete dataset.

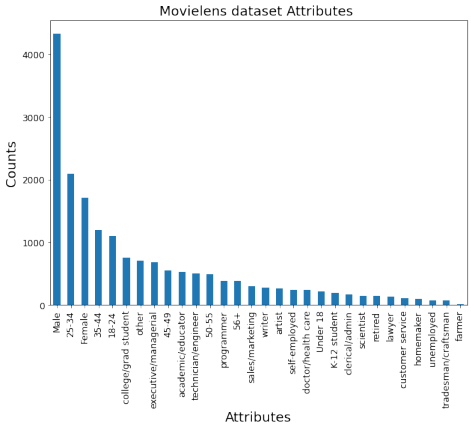
If the same product is interacted with multiple times in a row, we remove all but one interaction - a product can still be in the sequence multiple times, just not immediately after itself. Furthermore, we drop sequences with less than 3 and more than 200 interactions, as in [19]. While the average number of interactions or session length is about 26.4, most of the sessions are on the shorter side, with close to a long tail distribution for long sessions, as shown in Figure 2a. We show the frequency of the interactions per items in Figure 2b: There are few items with a high number of interactions, and far more items with a low interaction count. Overall, the density (describing the average number of unique item interactions per user) of the dataset is at 0.02% and far lower than in the Movielens-1m dataset.

We split the dataset by time with 28 days for training, 3 for validation and 14 days for the test set, as shown in Table 1. We can also see that the training item set differs hugely from the items seen in test and validation. Only about 60% of the training items are seen in validation and 1.7% items are new and unknown. In the test period, we see about 66% of the items in the training set, but we also find that 12.6% of the items haven't been seen before. Items not seen in training will be mapped to an special token (<unknown>).

We show the statistics about both the preprocessed Movielens-1m and the Fashion dataset in Table 2.



(a) Distribution over all users for gender, age and shopper type attributes for the Fashion dataset.



(b) Distribution over all users for gender, age and occupation attributes for the Movielens-1m dataset.

Fig. 3. Item and session length distributions for the Fashion dataset.

6 EXPERIMENTS

In this session we introduce the setup for our evaluation. We present a comparison of our models as well as further experiments on the influence of the session length, the performance in a cold-start setting and an ablation study of the attributes.

6.1 Evaluation Setup

To evaluate the influence of the added user representation, we compare the results for both U-BERT4Rec and U-SASRec^{cross} with the models without user information and two baseline models for merging attributes. For evaluation, we report the *Hit Ratio (HR)* and the *Normalized Discounted Cumulative Gain (NDCG)* at k = 1,5,10. Instead of using sampled metrics, we report the full metrics as it has been shown that sampled metrics are not reliable for ranking the models we're considering in [6]. We also use the Student's t-test to verify the statistical significance of the improvements for α-level 0.01 and 0.05. We train our model with the AdamOptimizer and unless specified otherwise use the parameters as in [19].

6.2 User Attribute Baselines

We use two baselines for including user attributes in our model to cover both options of merging information before and after the transformer layer.

6.2.1 KeBERT4Rec. The authors of KeBERT4Rec [9] describe an extension to BERT4Rec for including item keywords in the embedding layer as the transformers input. They create a multi-hot encoding from the keywords for each item, scale it with a linear layer to the dimension d of the transformer and add it element-wise to each item embedding to create the transformer input. We follow their approach and add the user attributes as attributes for each item in the sequence instead. Using the notations from section 4, this gives us k_t as the keyword embedding for an item, e_t as the item embedding and p_t as the positional embedding at t. The embedded sequence is then computed as $e_s = [e_1 + p_1 + k_1, e_2 + p2 + k_2, \ldots, e_n + p_n + k_n]$. We set k_t to our user embedding e_u for all timesteps t.

6.2.2 Latent Cross Post-Fusion Context. In [2] the authors introduce the Latent Cross technique and successfully integrate context post-fusion, after the sequence has been transformed by the GRU layer. The context is multiplied elementwise to the output of the GRU, serving as a mask. Inspired by this, we adapt a baseline as follows: The output of the transformer layer h_t is modified for each sequence step t to $h_t^* = h_t \odot e_u$, with $\odot$ as the elementwise multiplication. h_t^* is then used as the input for the projection layer.

6.3 Hyperparameters

6.3.1 Fashion dataset. We select the hyperparameters for the Fashion dataset by running a parameter study on the base models without user information. The best parameters are then used to train all models with user attributes. To limit the computation time and resource usage, we set the maximum session length to 50 after first experiments. We sample uniformly at random from $[1, 4]$ with step size=1 for the transformer heads, $[1, 4]$ and step size = 1 for the number of transformer layers, $[0.1, 0.5]$ and step size = 0.1 for the dropout, $[16, 48]$ and step size = 4 for the hidden size. For the learning rate we sample log-uniformly between $[0.01, 0.0001]$. We end up with transformer heads = 2, transformer layers = 2, dropout = 0.1 for both models. For the BERT4Rec models we use a learning rate of 0.003, a hidden size of 18 and train for 200 epochs. For all SASRec^{cross} models we end up with a learning rate of 0.001, a hidden size of 20 and train for 100 epochs. We run an additional hyperparameter study for U-BERT4Rec to tune the model and report the best model as U-BERT4Rec $_{tuned}$. The hidden size is increased to 48 for this model. Due to time limits we could not conclude further parameter studies.

6.3.2 Movielens-1m dataset. For the Movielens-1m dataset we set our hyperparameter settings as in [19] and use transformer heads = 2, transformer layers = 2, dropout = 0.2, batch size = 128, hidden size = 32 accordingly for all experiments and train for 200 epochs.

6.4 Results

In Table 3 we show the results of the base models without additional information for both datasets. We also report the most-popular baseline (POP). As expected, the popularity baseline is outperformed by both models by far, but the comparison between both models shows mixed results. On Movielens-1m the performance of SASRec^{cross} is more than doubled compared to BERT4Rec, but on the Fashion dataset the differences are far smaller. For HR@5 the difference is not significant and for HR@10 BERT4Rec is even significantly better. Overall, SASRec^{cross} shows itself to be a competitive model to BERT4Rec, with great improvements for the Movielens-1m dataset. Previously, the original SASRec has been reported to perform worse than BERT4Rec, but in [6] SASRec already does perform better for the HR@10 when evaluated on fully computed metrics. Therefore, the performance improvement of SASRec^{cross} on our data is not surprising, but gives reason to explore SASRec and SASRec^{cross} for other datasets in the future.

In Table 4 we show the models leveraging user attributes in comparison to those without.

For the Fashion dataset we can see significant improvements in HR@1, NDCG@5 and NDCG@10 for the all models using user attributes. The results for HR@5 and HR@10 show slightly worse performance for U-BERT4Rec, but the differences are non-significant. However, if we compare the hyperparameter-tuned U-BERT4Rec $_{tuned}$ model to the baseline we see significant improvements over all metrics. The slightly bigger hidden size for this model might give the model the capacity to encode more of the user information. But over all, the improvements for both KeBERT4Rec and LC-BERT4Rec are higher than the ones for U-BERT4Rec, and are significant in all cases. The results for the SASRec^{cross}

Table 3. Hitrate and NDCG for BERT4Rec and SASRec^{cross} and the popularity baseline evaluated on the Fashion and Movielens-1m datasets. Significance is marked by *for $p < 0.01$ and $^{+}$for $p < 0.05$ for the comparison between BERT4Rec and SASRec^{cross}.

Dataset	Metric	POP	B4R	SASRec^{cross}
Fashion	HR@1	0.000	0.079	**0.114***
	HR@5	0.003	0.274	**0.279**
	HR@10	0.011	**0.358***	0.356
	NDCG5	0.002	0.180	**0.199***
	NDCG@10	0.004	0.207	**0.225***
Movielens-1m	HR@1	0.003	0.013	**0.051***
	HR@5	0.005	0.050	**0.155***
	HR@10	0.011	0.103	**0.230***
	NDCG5	0.004	0.031	**0.103***
	NDCG@10	0.006	0.043	**0.128***

models are mixed: All models perform worse regarding the HR@10, U-SASRec^{cross} even significantly. The highest performance gain is achieved by KeSASRec^{cross}, followed by the LC-SASRec^{cross} model.

On the Movielens-1m dataset we also get inconclusive results. We see significant improvements for all but HR@1 between BERT4Rec and U-BERT4Rec. For KeBert4Rec and LC-Bert4Rec only the HR@10 improves significantly, while the other metrics show no improvement or even worse results, even though these are non significant. Looking at the SASRec^{cross} models, we see slightly lower results in comparison to the baseline for all models, part of them significant. Only the Latent-Cross Post Fusion model does not show a significant decrease in any of the metrics, but it still performs worse in absolute numbers.

Nevertheless, the general results on the Movielens-1m dataset show the same trend as on the Fashion dataset: Adding user attributes can improve the performance, but between the three different methods of integration attributes, there is no clear winner. The performance increase varies, depending on the choosen dataset and the base model. Finetuning a model can increase the performance significantly.

Both the BERT4Rec and the SASRec^{cross} model can profit from the additional user information, with the exception of SASRec^{cross} for the Movielens-1m dataset, where all of the methods perform worse.

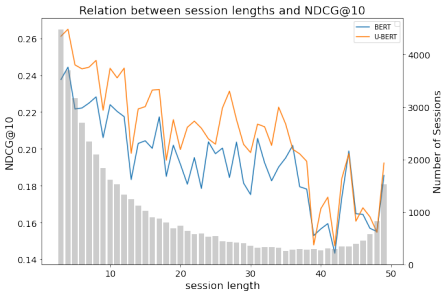
Comparing both model types also give ambiguous results: For the Fashion dataset KeBert4Rec and LC-Bert4Rec perform best, but are outperformed by KeSASRec^{cross} for HR@1. On the Movielens-1m dataset, the performance of SASRec^{cross} is far higher than BERT4Rec. SASRec^{cross} does not profit from the user attributes at all, but is still the best model for the dataset overall. Looking at the different metrics, we can also see that the gains for NDCG tend to be higher than the gains in the hitrate, or that the hitrate even drops. This can be seen for example with the U-BERT4Rec model, but this trade-of between better ranking and fewer hits seems to be a general trend for the models. Due to time and resource limits we could only conclude the hyperparameter study for U-BERT4Rec on the Fashion dataset and as it shows that we could further improve results, the natural next step would be to conduct further studies for the other models and also for the Movielens-1m dataset.

6.5 Session Length Influence

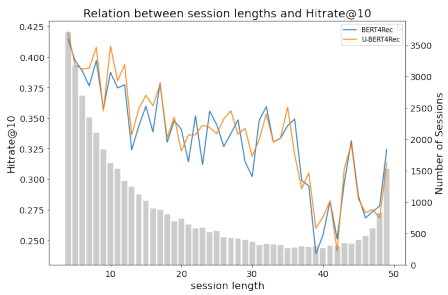
As we have seen, the inclusion of user attributes can improve the recommendation quality. To understand how the models use the additional information, we take a look at the NDCG and Hitrate for the different session lengths for

Table 4. Comparison of NDCG and Hitrate between the BERT4Rec based models with user attributes and between the SASRec^{cross} based models with user attributes for both datasets. Significance is marked by *for $p < 0.01$ and $^{+}$for $p < 0.05$ for the different to BERT4Rec and SASRec^{cross} respectively.

Dataset	Metric	B4R	UB4R	KeB4R	LC-B4R	UB4R$_t$	SAS^c	U-SAS^c	KeSAS^c	LC-SAS^c
Fashion	HR@1	0.079	0.095*	**0.122***	0.106*	*0.109**	0.114	0.121*	**0.124***	0.123*
	HR@5	0.274	0.273	**0.284***	0.281*	*0.281**	0.279	0.278	**0.282***	0.280
	HR@10	0.358	0.357	0.361*	**0.362***	*0.363**	**0.356**	0.351*	0.355	0.355
	NDCG5	0.180	0.186*	**0.207***	0.196*	*0.198**	0.199	0.204*	**0.207***	0.206*
	NDCG@10	0.207	0.214*	0.231*	**0.222***	*0.225**	0.225	0.227*	**0.231***	0.230*
Ml-1m	HR@1	0.013	**0.014**	0.008	0.011	-	**0.051**	0.046^{+}	0.047^{+}	0.048
	HR@5	0.050	**0.061***	0.051	0.050	-	**0.155**	0.147^{+}	0.150	0.151
	HR@10	0.087	**0.103***	0.102*	0.098*	-	**0.230**	0.226	0.228	0.227
	NDCG@5	0.031	**0.037***	0.030	0.030	-	**0.103**	0.097*	0.099^{+}	0.100
	NDCG@10	0.043	**0.051***	0.046	0.046	-	**0.128**	0.122*	0.124^{+}	0.124



(a) NDCG@10 for the different lengths of the input sequences for U-BERT4Rec $_{tuned}$ and BERT4Rec.



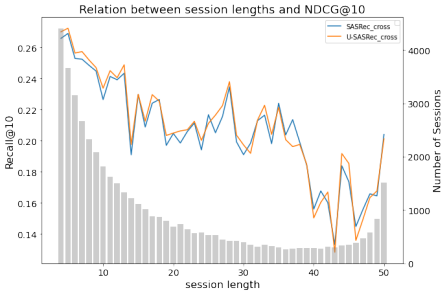
(b) Recall@10 for the different lengths of the input sequences for U-BERT4Rec $_{tuned}$ and BERT4Rec.

Fig. 4. Metrics shown for each input session length for U-BERT4Rec on the Fashion dataset.

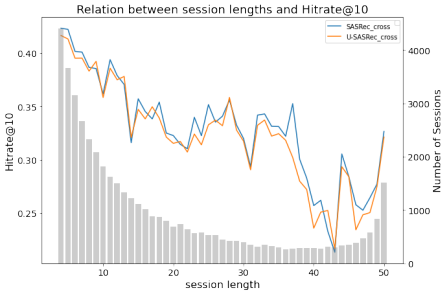
U-BERT4Rec $_{tuned}$ and U-SASRec^{cross} in the Fashion dataset. Figure 4 shows the NDCG@10 and the HR@10 plotted with respect to the length of the input sequence. In Figure 4a we see that the performance of U-BERT4Rec is higher than BERT4Rec over most sessions, only at a session length of 40 they perform similar. The number of sessions with greater lengths is relatively small [3] and the better performance for the many shorter sessions manages to increase the overall metrics score. Looking at the Hitrate we see only a slight improvement for shorter sessions, while for longer sessions the difference is marginalized or even reversed.

In Figure 5 we see a similar pattern in the plots for SASRec^{cross} and U-SASRec^{cross}. For the NDCG we find a small improvement for sessions up to a length between 35-40, but the Hitrate decreases slightly over all of the session lengths. The results for Movielens-1m, while not reported here, show a similar pattern in both metrics. Overall, we find that the inclusion of user attributes tends to primarily improve the ranking of the items, especially for shorter sessions. These results show that there is a trade-off between an significantly improved ranking and a slightly worse Hitrate for U-BERT4Rec and U-SASRec^{cross}.

[3]The number of sessions at the length of 50 is higher, as any longer session will be cut of at 50 and therefore contribute to the count.



(a) NDCG@10 for the different lengths of the input sequences for SASRec^{cross} and U-SASRec^{cross}.



(b) Recall@10 for the different lengths of the input sequences for SASRec^{cross} and U-BERT4Rec.

Fig. 5. Metrics shown for each input session length for U-SASRec^{cross} on the Fashion dataset.

Table 5. NDCG and Hitrate for U-BERT4Rec and U-SASRec^{cross} on the Fashion dataset in a cold-start setting with no user interactions.

Dataset	Metric	POP	UB4R$_{tuned}$	USASRec^{cross}
Fashion Users only	HR@1	0.000	0.001	0.001
	HR@5	0.003	0.010	0.005
	HR@10	0.011	0.021	0.011
	NDCG5	0.002	0.005	0.003
	NDCG@10	0.004	0.009	0.005

6.6 Recommendations without user interactions

One advantage of the way U-BERT4Rec and U-SASRec^{cross} integrate the user attributes is the ability to make recommendations even if no interactions are available. To show the performance in a cold-start setting, we remove all item interactions from the Fashion dataset and keep only the user attributes and the target item. We do use one dummy item with the <unknown> token for technical reasons. Table 5 shows the performance for U-BERT4Rec $_{tuned}$ and U-SASRec^{cross} for the modified test data. For both models the performance drops drastically, as it was to be expected. While the absolute numbers are not high, the metrics are more than doubled in comparison to POP for U-BERT4Rec from e.g., HR@10 of 0.011 to 0.021. The U-SASRec^{cross} model is also beating the most-popular baseline, but with less difference. This is in line with the previous results, where the U-SASRec^{cross} model is not able to use the user information as effectively as U-BERT4Rec. Overall, both models perform better than the most popular baseline and can improve recommendations in this cold-start setting.

6.7 Ablation Study

We conclude our experiments with an ablation study of the user attributes, limited to the U-BERT4Rec model due to time restrictions. For each of the attribute categories of age, gender and occupation/shopper type we train the U-BERT4Rec model separately. For the Fashion dataset, we use the parameter of U-BERT4Rec $_{tuned}$ to train the models and also compare to U-BERT4Rec $_{tuned}$. We show the results in Table 6.

On the Movielens-1m dataset all attributes can improve the model's performance slightly, except for the HR@1 metric. The improvements through the gender and age are similar, but for gender the results are more significant. The occupation has a small, but positive influence on the metrics. This would align with the intuition that an occupation

Table 6. NDCG and Hitrate for ablation study of different user attributes for U-BERT4Rec on the Movielens-1m dataset and U-BERT4Rec $_{tuned}$ on the Fashion dataset. Significance is marked by *for $p < 0.01$ and $^{+}$for $p < 0.05$.

Dataset	Metric	B4R	UB4R$_{gender}$	UB4R$_{age}$	UB4R$_{occ|type}$	UB4R$_{(tuned)}$
Fashion	HR@1	0.079	0.117*	**0.121***	0.115*	0.109 *
	HR@5	0.274	0.283*	**0.286***	0.281*	0.281*
	HR@10	0.358	0.362*	**0.363***	0.362*	0.363*
	NDCG5	0.180	0.204*	**0.207***	0.201*	0.198*
	NDCG@10	0.207	0.229*	**0.232***	0.227*	0.225*
Movielens-1m	HR@1	0.013	0.012	0.012	0.011	**0.014** *
	HR@5	0.050	**0.061** *	0.058 *	0.054	**0.061***
	HR@10	0.087	0.105 *	**0.106** *	0.102 *	0.103 *
	NDCG5	0.031	0.036 $^{+}$	0.035	0.033	**0.037** *
	NDCG@10	0.043	0.050 *	0.050 *	0.048 *	**0.051***

has the least connection to movie preferences, while there are movies targeted at specific age and gender demographics. The combination of all attributes gives the best results for this dataset.

For the Fashion dataset we see that gender, age and type improve the results significantly over all metrics. Age seems to be the more important attribute, as the performance is higher regarding all metrics. However, the final U-BERT4Rec model with the combination of all three features performs worse than the separate models (except for the HR@10). This shows that the available information is not yet optimally used and that feature selection becomes an important step when using user attributes. The investigation of more and different attribute combinations is therefore a candidate for future investigations.

7 CONCLUSION

While item attributes have been explored to improve sequential item recommendation, the same has not been done for user attributes yet. In this paper, we experiment with the integration of categorical user attributes to transformer-based models. We adapt BERT4Rec and create SASRec^{cross}, a SASRec-inspired model with a cross-entropy loss for our experiments. We propose a model where the user representation is merged to the sequence by the transformers and compare this to pre- and post-fusion approaches. Experiments on two datasets show that the models can profit to different, but significant, extends from user attributes. Especially the ranking of the recommendations is improved, while the gains in hitrate are lower. We find there is no best approach overall, as the best model is depending on the dataset and base model. Our original approach allows us to give better recommendations for users with no interactions though. Several directions remain to be explored: We can conduct hyperparameter studies to further verify our findings, as well as experiment with different combinations of attributes for a better feature selection. Variants or combinations of the pre- or postfusion approaches are also of interest as well as applying the models to the recently published H&M dataset [1]. As the SASRec^{cross} model performs quite well, it would be also interesting to investigate the performance on more datasets.

REFERENCES

[1] 2022. H&M personalized fashion recommendations. https://www.kaggle.com/competitions/h-and-m-personalized-fashion-recommendations

[2] Alex Beutel, Paul Covington, Sagar Jain, Can Xu, Jia Li, Vince Gatto, and Ed H Chi. 2018. Latent cross: Making use of context in recurrent recommender systems. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*. 46–54.

[3] Qiwei Chen, Huan Zhao, Wei Li, Pipei Huang, and Wenwu Ou. 2019. Behavior Sequence Transformer for E-Commerce Recommendation in Alibaba. In *Proceedings of the 1st International Workshop on Deep Learning Practice for High-Dimensional Sparse Data*. ACM. https://doi.org/10.1145/3326937.3341261

[4] Yongjun Chen, Jia Li, Chenghao Liu, Chenxi Li, Markus Anderle, Julian McAuley, and Caiming Xiong. 2021. Modeling Dynamic Attributes for Next Basket Recommendation. *2022 CARS Workshop* (2021).

[5] Sung Min Cho, Eunhyeok Park, and Sungjoo Yoo. 2020. MEANTIME: Mixture of attention mechanisms with multi-temporal embeddings for sequential recommendation. In *Fourteenth ACM Conference on Recommender Systems*. 515–520.

[6] Alexander Dallmann, Daniel Zoller, and Andreas Hotho. 2021. A Case Study on Sampling Strategies for Evaluating Neural Sequential Item Recommendation Models. In *Fifteenth ACM Conference on Recommender Systems*. 505–514.

[7] Gabriel de Souza Pereira Moreira, Sara Rabhi, Jeong Min Lee, Ronay Ak, and Even Oldridge. 2021. Transformers4Rec: Bridging the Gap between NLP and Sequential/Session-Based Recommendation. In *Fifteenth ACM Conference on Recommender Systems*. 143–153.

[8] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Association for Computational Linguistics, Minneapolis, Minnesota, 4171–4186. https://doi.org/10.18653/v1/N19-1423

[9] Elisabeth Fischer, Daniel Zoller, Alexander Dallmann, and Andreas Hotho. 2020. Integrating Keywords into BERT4Rec for Sequential Recommendation. In *KI 2020: Advances in Artificial Intelligence*.

[10] Dan Hendrycks and Kevin Gimpel. 2016. Gaussian Error Linear Units (GELUs). http://arxiv.org/abs/1606.08415 cite arxiv:1606.08415Comment: Trimmed version of 2016 draft.

[11] Balázs Hidasi and Alexandros Karatzoglou. 2018. Recurrent neural networks with top-k gains for session-based recommendations. In *Proceedings of the 27th ACM international conference on information and knowledge management*. 843–852.

[12] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. 2016. Session-based Recommendations with Recurrent Neural Networks. In *ICLR (Poster)*, Yoshua Bengio and Yann LeCun (Eds.).

[13] Balázs Hidasi, Massimo Quadrana, Alexandros Karatzoglou, and Domonkos Tikk. 2016. Parallel Recurrent Neural Network Architectures for Feature-rich Session-based Recommendations. In *Proceedings of the 10th ACM Conference on Recommender Systems* (Boston, Massachusetts, USA) *(RecSys '16)*. ACM, New York, NY, USA, 241–248. https://doi.org/10.1145/2959100.2959167

[14] Wang-Cheng Kang and Julian McAuley. 2018. Self-attentive sequential recommendation. In *2018 IEEE International Conference on Data Mining (ICDM)*. IEEE, 197–206.

[15] Cheng-Te Li, Cheng Hsu, and Yang Zhang. 2022. FairSR: Fairness-Aware Sequential Recommendation through Multi-Task Learning with Preference Graph Embeddings. *ACM Trans. Intell. Syst. Technol.* 13, 1, Article 16 (feb 2022), 21 pages. https://doi.org/10.1145/3495163

[16] Jing Li, Pengjie Ren, Zhumin Chen, Zhaochun Ren, Tao Lian, and Jun Ma. 2017. Neural attentive session-based recommendation. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*. 1419–1428.

[17] Liunian Harold Li, Mark Yatskar, Da Yin, Cho-Jui Hsieh, and Kai-Wei Chang. 2019. VisualBERT: A Simple and Performant Baseline for Vision and Language. arXiv:1908.03557 [cs.CV]

[18] Chang Liu, Xiaoguang Li, Guohao Cai, Zhenhua Dong, Hong Zhu, and Lifeng Shang. 2021. Non-invasive Self-attention for Side Information Fusion in Sequential Recommendation. *arXiv preprint arXiv:2103.03578* (2021).

[19] Fei Sun, Jun Liu, Jian Wu, Changhua Pei, Xiao Lin, Wenwu Ou, and Peng Jiang. 2019. BERT4Rec: Sequential Recommendation with Bidirectional Encoder Representations from Transformer. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management - CIKM 19*. ACM Press. https://doi.org/10.1145/3357384.3357895

[20] Yong Kiam Tan, Xinxing Xu, and Yong Liu. 2016. Improved recurrent neural networks for session-based recommendations. In *Proceedings of the 1st workshop on deep learning for recommender systems*. 17–22.

[21] Jiaxi Tang and Ke Wang. 2018. Personalized top-n sequential recommendation via convolutional sequence embedding. In *Proceedings of the eleventh ACM international conference on web search and data mining*. 565–573.

[22] Wilson L. Taylor. 1953. "Cloze procedure": a new tool for measuring readability. *Journalism & Mass Communication Quarterly* 30 (1953), 415–433.

[23] Trinh Xuan Tuan and Tu Minh Phuong. 2017. 3D Convolutional Networks for Session-based Recommendation with Content Features. In *Proceedings of the Eleventh ACM Conference on Recommender Systems* (Como, Italy) *(RecSys '17)*. ACM, New York, NY, USA, 138–146. https://doi.org/10.1145/3109859.3109900

[24] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*. 5998–6008.

[25] Liwei Wu, Shuqing Li, Cho-Jui Hsieh, and James Sharpnack. 2020. SSE-PT: Sequential recommendation via personalized transformer. In *Fourteenth ACM Conference on Recommender Systems*. 328–337.

[26] Chengfeng Xu, Pengpeng Zhao, Yanchi Liu, Jiajie Xu, Victor S.Sheng S.Sheng, Zhiming Cui, Xiaofang Zhou, and Hui Xiong. 2019. Recurrent Convolutional Neural Network for Sequential Recommendation. In *The World Wide Web Conference on - WWW'19*. ACM Press. https://doi.org/10.1145/3308558.3313408

[27] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. *Advances in neural information processing systems* 32 (2019).