Attribute-Aware Sequential Recommendation Model for Used Car Auctions

Shereen Elsayed[1,*], Ngoc Son Le[1,*], Ahmed Rashed[2,*], Lukas Hestermeyer[2], Radoslaw Wlodarczyk[2], Maximilian Stubbemann[1], and Lars Schmidt-Thieme[1]

[1] Information Systems and Machine Learning Lab (ISMLL) & VWFS Data Analytics Research Center (VWFS DARC), University of Hildesheim, Hildesheim, Germany
{elsayed,sle,schmidt-thieme}@ismll.uni-hildesheim.de
[2] Volkswagen Financial Services AG, Braunschweig, Germany
{ahmed.galal.ahmed.rashed,Lukas.Hestermeyer,Radoslaw.Wlodarczyk}@vwfs.com

Abstract. In used cars auction systems, users can buy vehicles through fixed-price rounds or participate in auction rounds where they place bids, with each item typically awarded to the highest bidder. This auction setup presents a challenge for recommender systems, as it involves sequential recommendation of unique items, where each item is available for sale only once in both fixed-price and auction rounds. Although this scenario is highly relevant, it has received limited attention in existing sequential recommendation research. Moreover, this challenge relates to the cold start problem encountered by many recommendation models. In this work, we aim to address the unique item sequential recommendation problem by developing an attribute-aware model for next-item prediction. Specifically, we introduce the **A**ttribute-Aware **S**equential **R**ecommendation **M**odel (**ASRM**), which is designed to handle unique item data and effectively leverage item attributes in the absence of item IDs. To further enhance performance in this context, we propose an improved version, **ASRM++**. Our experiments, conducted on a dataset from Volkswagen Financial Services' used car center, demonstrate that ASRM significantly outperforms existing state-of-the-art models for unique item recommendation. Additionally, we present A/B test results from the deployed ASRM model to validate its effectiveness.

Keywords: Attribute-aware recommendation · Auction systems · Sequential recommendation.

1 Introduction

Recommender systems have become essential for nearly all online platforms today. Over the past decade, the number of recommender system models has grown rapidly, including various types such as sequential recommendation [2] [11] [12], context-aware recommendation, and attribute-aware recommendation [7] [10]. Auction platforms raise unique challenges due to their specific characteristics.

* * All three authors contributed equally to this work.

In business-to-business (B2B) used car auction systems, users can purchase vehicles in two types of rounds: a fixed-price round or a bidding round, where users place bids and the item is sold to the highest bidder. Once sold, an item is removed from the platform, meaning each item is unique and becomes unavailable after the sale in both fixed-price and bidding rounds. As a result, relying on item IDs in such systems is not possible, as it is similar to the cold start problem with constantly unseen items. In this context, item attributes are essential for learning user preferences, as each item must be represented by its attributes rather than a persistent ID.

Sequential recommendation models have rapidly advanced due to their significance. Early approaches relied on convolutional neural networks (Caser) [12] and Gated Recurrent Units (GRU4Rec) [2]. A pivotal model, SASRec [4], uses a transformer encoder to build a simple yet effective recommendation model that is trained in an autoregressive manner. Several models have since built upon SASRec, such as S^3Rec [13], which incorporates pre-training to improve learning, and TiSASRec [5], which accounts for time intervals between interactions. Other methods integrate item or user attributes, including NOVA [6] that enhances BERT4Rec [11] to leverage the side information in the data , CARCA [7] which is an attribute and context-aware model that applies cross-attention for item scoring and ProxyRCA [10] further refines CARCA by enhancing the training protocol and introducing proxy-based item embeddings, allowing less frequently observed items to benefit from the information of more frequent ones.

Although many sequential recommendation models have been developed, few address the challenge of unique item settings. In this work, we aim to leverage a sequential recommendation model that focuses on item attributes instead of item IDs. Inspired by the CARCA model architecture, we propose an attribute-aware sequential recommendation model specifically designed for car auction systems, which can learn user preferences based on the features of historical items. Additionally, our model integrates both sales and bid interactions within a multi-task learning framework. The main contributions of the paper can be summarized as follows:

- We introduce an attribute-aware sequential recommendation model (ASRM) tailored for recommending unique items in auction systems. The model utilizes both bids and sales interactions for training, where user bids serve as an auxiliary task.
- We enhance several components of the proposed model and present an improved version called ASRM++.
- We demonstrate superior offline results compared to several state-of-the-art models, highlighting the effectiveness of our approach in unique item recommendation scenarios.
- We present online A/B test results, showcasing the real-world impact of our approach when deployed for actual users.

2 Related work

2.1 Attribute and Context-Aware Recommendation

Attribute and context-aware recommendations aim at enhancing the quality of recommendations by leveraging both item features and contextual information, moving beyond mere item IDs. This is especially crucial in certain domains where all the item IDs are unique, such as in the cases of auctions. Early works introduced Factorization Machines (FM) [9], which models contextual feature interactions between every pair of variables in the dataset. Later, DeepFM [1] unified FM with deep learning to enable higher-order feature interactions. Attention-based methods, such as NOVA [6], proposed to incorporate item attributes and interaction contexts in a non-invasive way, by separating the embeddings of the item IDs and its contextual information, thus preserving the integrity of the original item embeddings. S^3Rec [13] captured the correlation between items and their attributes more effectively by modeling these relationships during both the pre-training and fine-tuning stages of its self-supervised learning protocol. Most recently, CARCA [7] and ProxyRCA [10] achieved state-of-the-art performance in recommendation tasks by adopting a holistic method that incorporate the contextual and attribute-based information deeply within the collaborative filtering process. ProxyRCA further addresses the cold start problem by allowing the less frequent items to benefit from the well-trained proxy embeddings.

2.2 Sequential Recommendation

Sequential recommendation focuses on predicting the next item that the user is supposed to interact with, given the sequence of his historical interactions. In contrast to traditional collaborative filtering methods that do not consider the order of interactions, sequential recommendation methods aim to model evolving user preferences over time. Earlier methods, such as FM [9], were based on Markov Chains and modeled item-item transitions based on the last item interaction. The emergence of deep learning techniques has allowed more complex user-item interactions to be modeled. Recurrent Neural Networks (RNNs) based models, particularly Gated Recurrent Units (GRU) [2] [3], were used to capture longer-term dependencies of the user behavior. With the early success of self-attention mechanisms and the transformer architecture in sequence encoder tasks, such as machine translation, it has naturally been extended to sequential recommendation tasks. SASRec [4] was one of such early examples and its extension SASRec$_F$ [13] that further incorporated the item attributes by fusing it together with the item ID.

3 Attribute-Aware Unique-Item Recommendation Problem Formulation

In this paper, we address the following problem: Given a set of *items* $\mathcal{I} = \{i_1, \ldots, i_{|\mathcal{I}|}\}$, where each item $i \in \mathcal{I}$ is represented by an *attribute-vector* $v_i \in \mathbb{R}^A$,

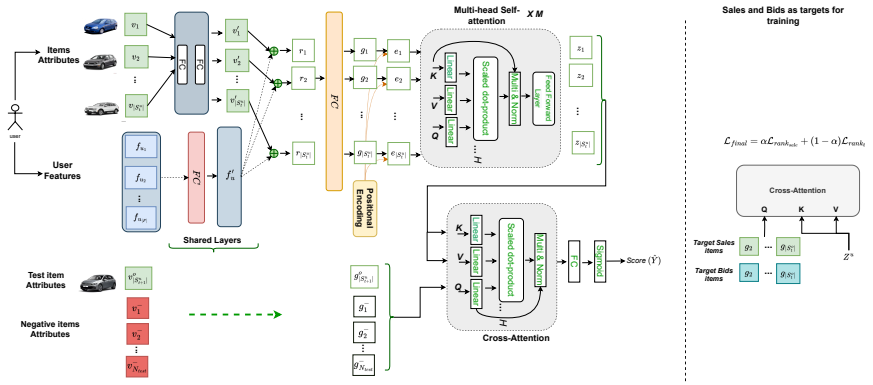

Fig. 1: Illustration of the deployed Attribute-aware Model Architecture

and a set of users $\mathcal{U} = \{u_1, \ldots, u_{|\mathcal{U}|}\}$, where each user $u \in \mathcal{U}$ has an *feature-vector* $f_u \in \mathbb{R}^F$. Each user u has a sequence of sales interactions $S^u = (S_1^u, \ldots, S_{|S^u|}^u)$ and a sequence of bids interactions $B^u = (B_1^u, \ldots, B_{|B^u|}^u)$. The objective is to predict the likelihood scores for a set of target items $\mathcal{T}^o = \{t_1^o, \ldots, t_{|\mathcal{T}^o|}^o\}$ for each user u. The model parameters are optimized by maximizing the log-likelihood (or minimizing cross-entropy) of the observed target item over all users. The bidding behavior serves as auxiliary information to help more accurately model user actions over time. In the context of auction systems, our problem is distinct from typical sequential recommendation scenarios in that each item in every interaction is unique. Formally, for any two interactions S_i^u and $S_j^{u'}$, where $u, u' \in U$ and $(u, i) \neq (u', j)$, we have $S_i^u \neq S_j^{u'}$.

4 ASRM: The Deployed Attribute-Aware Sequential Model for Unique Item Recommendation

4.1 Input Encoding

To form the item latent representation, we first embed item attributes in the input sequence using two fully connected layers to obtain the initial embeddings as:

$$v_j' = (v_j W + b)\, W' + b' \tag{1}$$

where $W \in \mathbb{R}^{A \times d'}, W' \in \mathbb{R}^{d' \times d}$ are the weight matrices $b \in \mathbb{R}^{d'}, b' \in \mathbb{R}^d$ are the bias vectors and d is the items embedding dimension.

For user features, we apply a fully connected layer to encode the user's features as:

$$f_u' = f_u W_f + b_f \tag{2}$$

where $W_f \in \mathbb{R}^{F \times d}$ is a weight matrix, F is the number of user features, and d is the items embedding dimension and b_f is the bias term. Notice that in this case, user features are the same for all items in the input sequence.

Afterward, we concatenate items, and user embeddings to get the combined item encoding r_j. Then, feed it to another fully connected layer for a refined representation:

$$r_j = \text{concat}_{col}\left(v'_j, f'_u\right), \qquad g_j = r_j W_r + b_r \tag{3}$$

where $W_r \in \mathbb{R}^{2d \times d}$ is the weight matrix, d is the layer embedding dimension, and b_r is the bias term. For simplicity, we use the **same dimension** d for all embedding layers. Finally, a learnable positional embedding $p_j \in \mathbb{R}^d$ is added to the final embedding to indicate the position in the input interactions sequence:

$$e_j = g_j + p_j \tag{4}$$

4.2 Multi-head Self-Attention Block

The ASRM model leverages a multi-head self-attention mechanism applied across the latent embeddings of all items. Let $\mathcal{E}^u := (e_1, \ldots, e_{|S^u|}) \in (\mathbb{R}^d)^{|S^u|}$ be the encoded sequence. In the following, we interpret this sequence as a matrix $\mathcal{E}^u \in \mathbb{R}^{|S^u| \times d}$. The first layer in the MHA block is a linear projection on each of the $Query$, $Keys$, and $Values$ as follows:

$$\begin{aligned} Q, K, V = LeakyReLU(\mathcal{E}^u W_Q), LeakyReLU(\mathcal{E}^u W_K), \\ LeakyReLU(\mathcal{E}^u W_V) \end{aligned} \tag{5}$$

where $\mathbf{W}^Q$, $\mathbf{W}^K$, $\mathbf{W}^V \in \mathbb{R}^{d \times d}$ represent the linear projection matrices. Then each of the output embedding is split over the number of heads H to obtain $\mathbf{W}_h^Q$, $\mathbf{W}_h^K$, $\mathbf{W}_h^V \in \mathbb{R}^{d \times \frac{d}{H}}$ of the head at index h. A multi-head self-attention is applied afterward to get the attention across all items in the sequence. This allows the model to capture dependencies between different items in the sequence, resulting in a latent representation for each item:

$$C^u = \text{SA}(\mathcal{E}^u) = \text{concat}_{col}\left(\text{Att}(Q_h, K_h, V_h)\right)_{h=1:H} \tag{6}$$

Here, $C^u \in \mathbb{R}^{|S^u| \times d}$ represents the column-wise concatenation of the attention heads. Afterward, a multiplicative residual is applied between C^u and Q, then we apply a normalization layer to obtain the output $C^{u'}$ as follows:

$$C^{u'} = Normalize(C^u * \mathcal{E}^u) \tag{7}$$

Finally, we have the row-wise feed-forward layers to obtain the component's output representations $Z^u \in \mathbb{R}^{|S^u| \times d}$ as follows. For all $i \in \{1, \ldots, |S^u|\}$ we have:

$$\begin{aligned} Z_i^u &= \text{FFN}(C_i^{u'}) \\ &= LeakyReLU(C_i^{u'} W^Z + b^Z) W^{Z'} + b^{Z'} \end{aligned} \tag{8}$$

where W^Z, $W^{Z'} \in \mathbb{R}^{d \times d}$ are the weight matrices of the two feed-forward layers, while b^Z and $b^{Z'} \in \mathbb{R}^d$ are their bias row-vectors. **LeakyReLU** is the activation function applied after the first layer.

4.3 Next Item Prediction with ASRM

Given Z^u, we predict for a given a set of target items represented by the attribute-matrix $\mathcal{T}^o$ the likelihood via cross-attention.

Cross Attention To derive the latent embedding G^o for the target items, their attribute vectors are passed through the shared item encoding layers up to Equation 3. However, the fully connected layer applied after concatenating the user features is skipped on the target branch. For item ranking, we apply cross-attention between Z^u and G^o as follows: We get a linear projection on each of the $Query$, $Keys$ and $Values$ similar to the input MHA block, as follows:

$$Q, K, V = LeakyReLU(G^o W_Q), LeakyReLU(Z^u W_K), \\ LeakyReLU(Z^u W_V) \tag{9}$$

where $\mathbf{W}^Q$, $\mathbf{W}^K$, $\mathbf{W}^V \in \mathbb{R}^{d \times d}$ represent the linear projection matrices. Then each of the output embedding is split over the number of heads H to obtain $\mathbf{W}_h^Q$, $\mathbf{W}_h^K$, $\mathbf{W}_h^V \in \mathbb{R}^{d \times \frac{d}{H}}$ of the head at index h. A multi-head self-attention is applied afterward to get the attention across all items in the sequence. This allows the model to capture dependencies between different items in the sequence, resulting in a latent representation for each item:

$$X^o = \mathrm{CA}(\mathcal{E}^u) = \mathrm{concat}_{col}\left(\mathrm{Att}(Q_h, K_h, V_h)\right)_{h=1:H} \tag{10}$$

Here, $X^o \in \mathbb{R}^{|\mathcal{T}^o| \times d}$ represents the column-wise concatenation of the attention heads. Afterward, a multiplicative residual is applied between X^o and Q, then we apply a normalization layer to obtain the output $X^{o'}$ as follows:

$$X^{o'} = Normalize(X^o * G^o) \tag{11}$$

To reduce the output dimension, we apply a fully connected layer with output dimension 1 followed by a sigmoid function to compute a probability score between $[0, 1]$:

$$\hat{Y}_k^o = \sigma(X^{o'}{}_k W_o + b_o) \tag{12}$$

for all $k \in \{1, \ldots, |\mathcal{T}^o|\}$, where σ is the sigmoid function, $W_o \in \mathbb{R}^{d \times 1}$ is the weight matrix of the output layer to reduce the output dimension to one value and $b \in \mathbb{R}$ is the bias term.

In the context of used-car recommendations, we have given a sequence of used-cars bought by a dealer and the main task is to rank a sequence of used cars based on their likelihood to be bought by this dealer.

4.4 ASRM Model Optimization

Bids and sales as model targets Given a sequence of positive sales target items from the user's input and an equally long set of negative items sampled from a pool excluding the user's input sequence, we define the binary cross-entropy objective as follows:

$$\mathcal{L}_{Sales} = -\sum_{\mathcal{T}^o \in \mathcal{S}} \sum_{t=0}^{|\mathcal{T}^o|} [log(\hat{Y}_t^{o^{(+sale)}}) + (log(1 - \hat{Y}_t^{o^{(-)}}))] \tag{13}$$

As previously discussed, positive target items can also be selected from the user's bid sequence, as these reflect additional items of interest. Thus, we define the auxiliary loss for bids as follows:

$$\mathcal{L}_{Bids} = -\sum_{\mathcal{T}^o \in \mathcal{S}} \sum_{t=0}^{|\mathcal{T}^o|} [log(\hat{Y}_t^{o^{(+bid)}}) + (log(1 - \hat{Y}_t^{o^{(-)}}))] \tag{14}$$

where $\hat{Y}_t^{O^{(+)}}$ are the output scores for the positive samples (sales or bids) and $\hat{Y}_t^{o^{(-)}}$ are the output scores for the negative samples, $\mathcal{S}$ is the set of all sequences, and $|\mathcal{T}^o|$ is the length of the target sequence.
The final model loss is computed as a weighted sum of the sales and bids losses:

$$\mathcal{L} = \alpha \mathcal{L}_{Sales} + (1 - \alpha)\mathcal{L}_{Bids} \tag{15}$$

Training Protocol Following the CARCA [7] and SASRec [4] models training protocol, we define a fixed sequence length L in which we select the input sequence as the most recent L items for each user, the sequence length can be obtained by truncation or padding in some cases. The target positive items sequence has the same sequence length L and is formed once using the shifted sequence of the sales items $[v_{|S_t^u|-L+1}, \ldots, v_{|S_t^u|}]$ and once using the most recent L bidding items to have this multi-task learning paradigm between the sales and the bids and positive target items. On the other hand, the negative sequence is formed as random unseen items of sequence length L, thus the number of negative samples N_{train} is L in this case.

5 Enhanced ASRM for Unique Item Recommendation (ASRM++)

5.1 Input Encoding

To form the item latent representation, we first embed item attributes in the input sequence using two fully connected layers and GeLU activation to obtain the initial embeddings as:

$$v_j^{'} = GeLU(v_j W + b)W^{'} + b^{'} \tag{16}$$

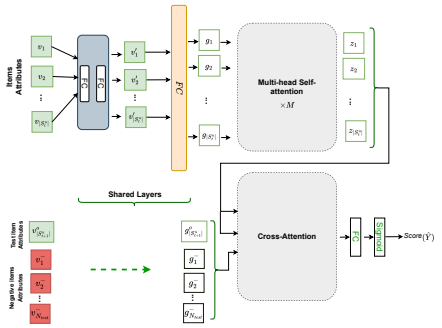

Fig. 2: Illustration of ASRM++ model

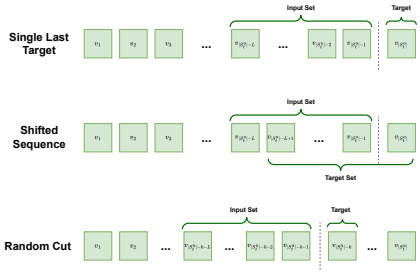

Fig. 3: Different training protocols

where $W \in \mathbb{R}^{A \times d^{'}}, W^{'} \in \mathbb{R}^{d^{'} \times d}$ are the weight matrices $b \in \mathbb{R}^{d^{'}}, b^{'} \in \mathbb{R}^{d}$ are the bias vectors and d is the items embedding dimension.

Then, feed it to another two fully connected layers for a deeper representation:

$$g_j = v_j^{'} W_{v'} + b_{v'} \tag{17}$$

where $W_{v'} \in \mathbb{R}^{d \times d}$ is the weight matrix, d is the layer embedding dimension, and $b_{v'}$ is the bias term. For simplicity, we use the **same dimension** d for all embedding layers.

Multi-head Self-Attention Similar to the ASRM model a multi-head self-attention is applied across the latent embeddings of all items. Let $G^u := (g_1, \ldots, g_{|S^u|}) \in (\mathbb{R}^d)^{|S^u|}$ be the encoded sequence. In the following, we interpret this sequence as a matrix $G^u \in \mathbb{R}^{|S^u| \times d}$. A normalization layer is applied on G^u before it is fed into the multi-head self-attention as follows:

$$G^{u^{'}} = Normalize(G^u) \tag{18}$$

We utilize multi-head self-attention to apply attention across all items in the sequence. This allows the model to capture dependencies between different items in the sequence, resulting in a latent representation for each item.:

$$C^u = \mathrm{SA}(G^{u^{'}}) = \mathrm{concat}_{col}\left(\mathrm{Att}(G^{u^{'}}\mathbf{W}_h^Q, G^{u^{'}}\mathbf{W}_h^K, G^{u^{'}}\mathbf{W}_h^V)\right)_{h=1:H} \mathbf{W}^P \tag{19}$$

where $\mathbf{W}_h^Q$, $\mathbf{W}_h^K$, $\mathbf{W}_h^V \in \mathbb{R}^{d \times \frac{d}{H}}, \mathbf{W}^P \in \mathbb{R}^{d \times d}$ represent the linear projection matrices of the head at index h, and H is the number of heads. Here, $C^u \in \mathbb{R}^{|S^u| \times d}$ represents the column-wise concatenation of the attention heads.

Then additive residual is applied between C^u and Q to obtain the output $C^{u^{'}}$:

$$C^{u^{'}} = C^u + G^u \tag{20}$$

Lastly, we have the row-wise feed-forward layers to obtain the component's output representations $Z^u \in \mathbb{R}^{|S^u| \times d}$ as follows. For all $i \in \{1, \ldots, |S^u|\}$ we have:

$$\begin{aligned} Z_i^u &= \mathrm{FFN}(C^{u'}{}_i) \\ &= GeLU(C^{u'}{}_i W^Z + b^Z) W^{Z'} + b^{Z'} \end{aligned} \tag{21}$$

where W^Z, $W^{Z'} \in \mathbb{R}^{d \times d}$ are the weight matrices of the two feed-forward layers, while b^Z and $b^{Z'} \in \mathbb{R}^d$ are their bias row-vectors. **GeLU** is the activation function applied after the first layer.

5.2 Next Item Prediction with ASRM++

Given Z^u, and the latent embedding G^o of the target item attribute vector. We predict the likelihood via cross-attention.

Cross Attention Simply here the item scoring is calculated similarly to the original ASRM model, where we apply cross attention between the input sequence and target items. Finally, the output layer is a fully connected layer of output 1 followed by a *Sigmoid* function to limit the output scores between $[0, 1]$, as in Equation 12. However, in this case the scores are of size $(N_{train} + 1)$ as we have one positive item and N_{train} negative items, which is explained in detail in the next section.

5.3 ASRM++ Model Optimization

Unlike the original ASRM, we removed the multi-task learning part that was previously employed. As it is shown in later sections, we show that employing the different training protocol was more effective in learning the model and outperformed the added benefit cause by using the bids information. Thus in this enhanced version we eliminated that part. The binary cross-entropy objective can be defined as follows:

$$\mathcal{L} = -\sum_{\mathcal{T}^o \in \mathcal{S}} [log(\hat{Y}^{o^{(+)}}) + \sum_{t=0}^{N_{train}} (log(1 - \hat{Y}_t^{o^{(-)}}))] \tag{22}$$

where $\hat{Y}^{o^{(+)}}$ are the output scores for the positive sale sample and $\hat{Y}_t^{o^{(-)}}$ are the output scores for the negative samples N_{train} in our case we set N_{train} to 100. $\mathcal{S}$ is the set of all sequences.

Enhanced Training Protocol There exist multiple ways to conduct training protocols in recommendation systems. Specifically, the original ASRM followed a shifted sequence training protocol (as shown in Figure 3). In this approach, during training the training sequence $[v_{|S_t^u|-L}, \ldots, v_{|S_t^u|-1}]$ is shifted to form the

target sequence $[v_{|S_t^u|-L+1}, \ldots, v_{|S_t^u|}]$. This technique has also been implemented in previous works [4, 13], with the main difference being that the ASRM model uses cross-attention and does not enforce a causality constraint during both the training and inference phases, allowing bidirectionality in both cases. Despite not leveraging autoregressive tasks due to leakage, the ASRM training protocol has significantly outperformed the training protocol that uses only the last item in the sequence as the target, as well as the case where causality is enforced during the training phase. A newly introduced training protocol, as proposed by [10], addresses the discrepancies between training and inference by using only the final item of a randomly cut sequence as the target. We chose this approach to further increase the variety of target items that appear during training, which has shown to be particularly advantageous for long interaction sequences in the VWFS dataset.

Hard Negative Training Sampling To ensure a realistic assessment of model performance at deployment, during evaluation we would sample 2,000 negative samples that occurred close in time to the test item. This approach mirrors the scenario in auction datasets with unique items, where only unsold items still available for bidding are considered for recommendation during deployment. The original ASRM, however, employed a standard random negative sampling protocol during training, which led to a mismatch between the training and evaluation phases. To address this, one of the key improvements introduced was to apply the same time-aware hard negative sampling protocol during training to align it with the test phase. Specifically, during the training phase the negative items are being chosen to be the ones sold within the same month as the target item.

6 Deployment Infrastructure

To deploy the ASRM model in production, we utilized the AWS step functions to build our training and inference pipelines. Those pipelines are called every weekend to train and deploy a new model using the recent sales and bids data. We also cache the recommendation scores for all expected dealers-vehicle pairs for faster retrieval during the following weekdays.

6.1 Online Training Pipeline

In the training pipeline shown in Figure 4, we first retrieve the latest sales, bids, and dealer data from AWS-Athena. Afterward, we feed this data to a SageMaker pre-processing job that filters and converts the vehicle raw features, sales, and bids data to interaction tuples, and numerical features. Once the data is pre-processed we spawn a SageMaker training job that trains the model on the pre-processed data using the best-found hyperparameters at that time and it saves the model artifact in AWS S3. Once the model is trained we initiate the inference step function which pre-computes and caches the recommendation

scores for all expected dealer-vehicle pairs of the upcoming week. It is worth noting that we employ multiple intermediary validation steps in our training pipeline that check the correctness of the generated preprocessed data, whether the model performance is above a predefined accuracy threshold, and whether the inference job finished successfully or not.

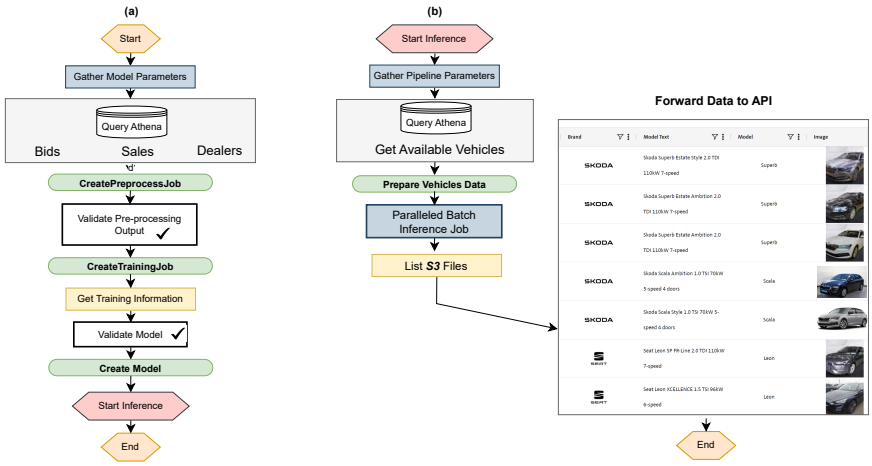

Fig. 4: AWS Training (a) and Inference Pipeline (b)

6.2 Online Inference Pipeline

In our setup, the vehicles that we expect to show on our website are known at least one month beforehand. This allows our inference pipeline shown in Figure 4 to retrieve, preprocess, and utilize the AWS SageMaker batch inference to precompute all the possible dealer-vehicle pairs that can exist in the next upcoming week. These scores are stored as a per-dealer binary file on S3 that contains vehicle scores. This file is then retrieved by our API gateway once it receives a recommendation request from the frontend for a specific dealer.

7 Experiments

7.1 Experimental Settings

Dataset The dataset captures a B2B setting where vehicles are sold to dealers. Collected between January 1, 2016, and December 31, 2023, it includes data from 5,916 users/dealers and records 1,224,364 interactions, each corresponding to a unique vehicle. Each vehicle is described by 109 distinct features, including

Method	Components			HR@10	NDCG@10
	Multi-task	Random-cut	Cross-Att.		
MultiRec [8]	✓	✗	✗	0.1329 ± 2.3e−3	0.0917 ± 4e−4
SASRec [4]	✗	✗	✗	0.1556 ± 6e−4	0.0740 ± 3e−4
CARCA [7]	✗	✗	✓	0.1808 ± 7e−4	0.0966 ± 1.1e−3
ProxyRCA [10]	✗	✓	✓	0.2135 ± 9.2e−3	0.1168 ± 2.8e−3
ASRM (deployed)	✓	✗	✓	0.2027 ± 3.4e−3	0.1052 ± 5e−4
ASRM++ (ours)	✓	✓	✓	0.2321 ± 3.6e−3	0.1289 ± 6e−4
Improv.(%)				8.71%	10.36%

HR@25	NDCG@25	HR@50	NDCG@50
0.2153± 2.7e−3	0.0917 ± 8e−4	0.2941 ± 4.3e−3	0.1068 ± 1.4e−3
0.2751 ± 1.5e−3	0.1031 ± 4e−4	0.3750 ± 1.3e−3	0.1223 ± 1e−4
0.2892 ± 3.4e−3	0.1229 ± 4e−4	0.3742 ± 6.2e−3	0.1393 ± 3e−4
0.3396 ± 8e−4	0.1475 ± 6e−4	0.4477 ± 2.9e−3	0.1683 ± 1e−3
0.3185 ± 3.9e−3	0.1332 ± 6e−4	0.4221 ± 1.3e−2	0.1532 ± 2.4e−3
0.3581 ± 3e−3	0.1595 ± 1.4e−3	0.4605 ± 1.1e−3	0.1792 ± 1e−3
5.45%	8.14%	2.86%	6.48%

Table 1: Model performance and comparison against baselines on VWFS dataset. Where ASRM is the deployed model which we later illustrate A/B test results, and ASRM ++ is the improved version of the model. The best results are reported in red and the second best in blue.

model, brand, color, and gear type. Additionally, the dataset includes 11,573,971 bid interactions, which provide supplementary information for each user. It is essential to note that all user/dealer data is anonymized, with confidentiality ensured in compliance with GDPR requirements.

Implementation Details The ASRM and ASRM++ models were optimized using binary cross-entropy loss through AdamW, and with the weight decay tuned between the range of 0.0 and 1.0. The learning rate was tuned within the range of 1e-6 and 0.1, dropout rate between 0.0 and 0.5 and the random cut probability between 0.0 and 1.0. The number of heads was searched among the set of {2, 4, 8}, hidden dimension {64, 128, 256} and the batch size was kept at 128. The tuning job was performed with the AWS SageMaker Hyperparameter-Tuner using Bayesian optimization. The implementation is publicly available in our code repository[***].

7.2 Evaluation Protocol

We use a leave-one-out mechanism, training and validating the model with the entire sequence except the last interaction, which is used for testing. We sample 2000 negative items N_{test} within the same month of the corresponding positive item date and assess model performance with Hit Ratio (HR@N) and Normalized Discounted Cumulative Gain (NDCG@N). Higher HR and NDCG values indicate better performance. We report the mean and standard deviation of results from three separate runs to ensure statistical robustness.

[***] https://github.com/sonngocl22/ASRM-improved/

Baselines We compare our proposed method against various attribute-aware sequential recommendation methods, that can rely only on items attributes and remove the items ids.

- **MultiRec** [8]: A multi-relational model built for unique item recommendation in auction systems, that can leverage items attributes and bids interactions.
- **SASRec** [4]: A model that applies multi-head self-attention to capture the sequential pattern in the users' history, then applies dot product for calculating the items scores. In this case we replace the items ids in the original model to be only items attributes to be applicable in unique item setting.
- **CARCA** [7]: A context-aware sequential recommendation approach employing cross-attention between user profiles and items for score prediction. For this experiment we removed the items ids part from the model.
- **ProxyRCA** [10]: A state-of-the-art attribute-aware model that introduced using proxy-based embedding to allow less frequent items to benefit from more-frequent ones. For this experiment we removed the items ids part from the model.

8 Offline Results

8.1 Model performance against baselines

The experimental results in Table 1 demonstrate that the ASRM model, specifically designed for the unique item recommendation task, surpasses the original CARCA [7], as well as SASRec [4] and MultiRec [8]. Furthermore, the results indicate that the improved model ASRM++ achieves state-of-the-art performance, outperforming the second-best model ProxyRCA [10] by up to 10%, and significantly improves upon the currently deployed ASRM model by more than 20%. This shows the effectiveness of the applied methods in the ASRM model within auction systems, as well as the substantial performance gains achieved by ASRM++ following the applied enhancements.

Table 2: The table illustrates the effect of different model components

Model	HR@10	NDCG@10
w/o Random cut	0.1581	0.0854
w/o Round type	0.1995	0.1096
w/o Hard sampling	0.2194	0.1206
ASRM++	**0.2321**	**0.1289**

Model	HR@10	NDCG@10
ASRM w/ bids	**0.2027**	**0.1052**
ASRM w/o bids	0.1750	0.0916
ASRM++ w/ bids	0.2172	0.1206
ASRM++ w/o Bids	**0.2321**	**0.1289**

8.2 Ablation studies

Effect of negative sampling (having negative items in the same month) The first ablation experiment was to test the effect of hard negative sampling

during training. This approach involves using negative items from the same month as the target item during the training phase, which better aligns with the real workings of the auction bidding system. As demonstrated in Table 2, there is a notable performance improvement of approximately 7%, with HR@10 increasing from 0.2194 to 0.2321 when hard sampling is incorporated into the implementation.

Effect of different training protocols (random cut training protocol) This study aims to examine the impact of the random cut training protocol. When the random cut method is removed, in this case it means only a single target item—the last item in the sequence is considered during training. As shown in Table 2, this has a substantial effect on the performance of the model, changing the HR@10 from 0.2321 to 0.1581.

A useful comparison is shown in the main results in Table 1, highlighting a key difference between ASRM and ASRM++. Specifically, ASRM adopts the training protocol used in SASRec [4] and CARCA [7], which sets the target sequence as a shifted sequence of the input sequence, while ASRM++ adopts the random-cut protocol. Results indicate that by sampling multiple cuts throughout the entire user sequence, the model learns more effectively than when trained solely on the most recent items.

Effects of adding sales round type As previously mentioned, the auction system consists of two different sales round types: the fixed-price round and the bidding round. Although this distinction was known, the deployed version of ASRM did not utilize this information during training. By incorporating the sales round type as a feature both during training and evaluation phases, we were expecting a performance boost. The results of the ablation study, shown in Table 2, confirm that by including the round type information into the model we can significantly increase the model performance. It is important to note that the sales round type will be known during deployment, aligning with whether the user is in a fixed-price or bidding round.

Effect of adding bids as auxiliary behavior on the target side In our approach, we included bids in the target set during training, allowing them to serve as potential targets for users. As shown in Table 2, for the ASRM model this strategy increased the HR@10 from 0.195 to 0.205, reflecting a 5% improvement and demonstrating the effectiveness of this multi-task learning approach in leveraging auxiliary information. However, for the case of ASRM++ including the bids as targets hurt the performance. This could be due to the fact that the inclusion of random cut in ASRM++ helped expose the model to a broader coverage of unique items, compensating for the effects of using bids as targets in the original ASRM.

Impact of random cut frequency on performance The experimental results and ablation study clearly demonstrate the significant impact of the ran-

dom cut training protocol. This effect is attributed to the broader coverage of the unique items in the input sequences and target items. However, because some users have very long interaction histories and only recent interactions have a strong impact on the final prediction, the frequency of using uniformly randomly chosen items as training targets may affect the final outcomes. To examine this hypothesis, we conducted experiments across varying probabilities that a sequence is cut during training. The results, shown in Figure 5, confirm our hypothesis, with overall performance steadily increasing and peaking at a cut probability of 0.8, while cutting sequences 100% of the time led to a slight decrease in the performance. Further experiments with different sequence lengths and the optimal random cut probability revealed that, given adequate item coverage, sequence length has a minimal effect on model performance.

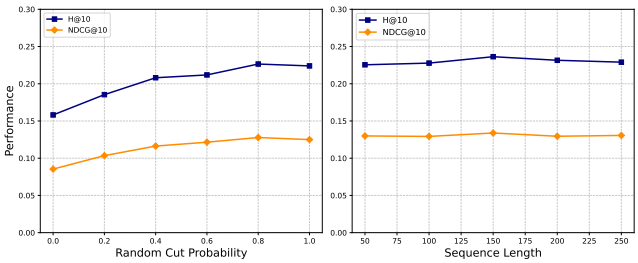

Fig. 5: Performance evaluation across different random cut probabilities and sequence lengths

9 Online A/B Test Results

KPI	Mean			Median			P-Value
	Group B	Group A	improv.	Group B	Group A	improv.	
CTR	**0.7780**	0.7600	2.36%	**0.7500**	0.7450	0.67%	0.6560
CTR on top 25 vehicles	**0.0960**	0.0688	39.53%	**0.0320**	0.0240	33.33%	**0.0007**
Bid-TR	**0.0079**	0.0063	25.39%	0.0	0.0	-	0.1629
Bid-TR on top 25 vehicles	**0.0046**	0.0007	557%	0.0	0.0	-	0.3580
Buy-TR	**0.0005**	0.0003	66.67%	0.0	0.0	-	0.9890
Buy-TR on top 25 vehicles	**4.071e-6**	3.162e-6	28.75%	0.0	0.0	-	0.8229
Adjusted Bid-Buy actions							
Bid-TR on top 25 vehicles	**0.0159**	0.0098	62.24%	0.0	0.0	-	**0.0352**
Buy-TR on top 25 vehicles	**0.0005**	0.0001	400%	0.0	0.0	-	**0.0195**

Table 3: AB Test statistics

We evaluated our ASRM model using an online ab test that ran between 26.6.2023 and 17.10.2023. We employed our recommendation model as a new

option for sorting the main vehicles list on our website stock list page and we compared that behavior against the default behavior of sorting by date. We split the dealers into two groups randomly and tracked the sample mismatch ratio during the test. We tracked multiple KPIs such as clicks, bids, and buy-through rates. Although dealers can directly bid or buy a vehicle from the main list without clicking on the vehicle record to view its detailed information, this is usually rare and dealers prefer to buy or bid on the vehicle after they open its details page. To measure the bid and buy-through rates in such scenarios we included adjusted bid and buy metrics that also track actions on the vehicles pages and link them to their corresponding main list view. Additionally, we measured the differences between the two groups and the statistical significances were computed using Mann-Whitney-U-Tests as most metrics had non-normal distributions. Results in Table 3, show that in general, the treatment group B's KPIs have higher metric values but the effect is statistically significant if we focus on actions on the top 25 records of the main list.

References

1. Guo, H., Tang, R., Ye, Y., Li, Z., He, X.: Deepfm: a factorization-machine based neural network for ctr prediction. arXiv preprint arXiv:1703.04247 (2017)
2. Hidasi, B.: Session-based recommendations with recurrent neural networks. arXiv preprint arXiv:1511.06939 (2015)
3. Jannach, D., Ludewig, M.: When recurrent neural networks meet the neighborhood for session-based recommendation. In: Proceedings of the eleventh ACM conference on recommender systems. pp. 306–310 (2017)
4. Kang, W.C., McAuley, J.: Self-attentive sequential recommendation. In: 2018 IEEE international conference on data mining (ICDM). pp. 197–206. IEEE (2018)
5. Li, J., Wang, Y., McAuley, J.: Time interval aware self-attention for sequential recommendation. In: Proceedings of the 13th international conference on web search and data mining. pp. 322–330 (2020)
6. Liu, C., Li, X., Cai, G., Dong, Z., Zhu, H., Shang, L.: Noninvasive self-attention for side information fusion in sequential recommendation. In: Proceedings of the AAAI conference on artificial intelligence. vol. 35, pp. 4249–4256 (2021)
7. Rashed, A., Elsayed, S., Schmidt-Thieme, L.: Context and attribute-aware sequential recommendation via cross-attention. In: Proceedings of the 16th ACM Conference on Recommender Systems. pp. 71–80 (2022)
8. Rashed, A., Jawed, S., Schmidt-Thieme, L., Hintsches, A.: Multirec: A multi-relational approach for unique item recommendation in auction systems. In: Fourteenth ACM Conference on Recommender Systems. pp. 230–239 (2020)
9. Rendle, S.: Factorization machines. In: 2010 IEEE International conference on data mining. pp. 995–1000. IEEE (2010)
10. Seol, J., Gang, M., Lee, S.g., Park, J.: Proxy-based item representation for attribute and context-aware recommendation. In: Proceedings of the 17th ACM International Conference on Web Search and Data Mining. pp. 616–625 (2024)
11. Sun, F., Liu, J., Wu, J., Pei, C., Lin, X., Ou, W., Jiang, P.: Bert4rec: Sequential recommendation with bidirectional encoder representations from transformer. In: Proceedings of the 28th ACM international conference on information and knowledge management. pp. 1441–1450 (2019)

12. Tang, J., Wang, K.: Personalized top-n sequential recommendation via convolutional sequence embedding. In: Proceedings of the eleventh ACM international conference on web search and data mining. pp. 565–573 (2018)
13. Wu, L., Li, S., Hsieh, C.J., Sharpnack, J.: Sse-pt: Sequential recommendation via personalized transformer. In: Fourteenth ACM Conference on Recommender Systems. pp. 328–337 (2020)