Sequential Recommendation with Probabilistic Logical Reasoning

Huanhuan Yuan[1] , **Pengpeng Zhao**[1*] , **Xuefeng Xian**[2*] and **Guanfeng Liu**[3] and **Victor S. Sheng**[4] and **Lei Zhao**[1]
[1]Soochow University
[2]Suzhou Vocational University
[3] Macquarie University
[4] Texas Tech University
hhyuan@stu.suda.edu.cn, ppzhao@suda.edu.cn, xianxuefeng@jssvc.edu.cn, guanfeng.liu@mq.edu.au, Victor.Sheng@ttu.edu, zhaol@suda.edu.cn

Abstract

Deep learning and symbolic learning are two frequently employed methods in Sequential Recommendation (SR). Recent neural-symbolic SR models demonstrate their potential to enable SR to be equipped with concurrent perception and cognition capacities. However, neural-symbolic SR remains a challenging problem due to open issues like representing users and items in logical reasoning. In this paper, we combine the Deep Neural Network (DNN) SR models with logical reasoning and propose a general framework named **S**equential **R**ecommendation with **P**robabilistic **L**ogical **R**easoning (short for SR-PLR). This framework allows SR-PLR to benefit from both similarity matching and logical reasoning by disentangling feature embedding and logic embedding in the DNN and probabilistic logic network. To better capture the uncertainty and evolution of user tastes, SR-PLR embeds users and items with a probabilistic method and conducts probabilistic logical reasoning on users' interaction patterns. Then the feature and logic representations learned from the DNN and logic network are concatenated to make the prediction. Finally, experiments on various sequential recommendation models demonstrate the effectiveness of the SR-PLR. Our code is available at https://github.com/Huanhuaneryuan/SR-PLR.

1 Introduction

Sequential Recommendation (SR) has been proposed to solve information overload in a variety of real-world applications, including e-commerce, advertising, and other areas. Meanwhile, Deep Neural Network (DNN) is widely used in SR to capture the sequential characteristics of user behaviors and generate accurate recommendations [Hidasi *et al.*, 2016; Kang and McAuley, 2018]. Recent developments in neural-symbolic methods [Shi *et al.*, 2020; Shi *et al.*, 2022; Zhang *et al.*, 2022] have demonstrated competitive performance against DNN-based SR models, thus boosting significant research interest in combining DNN with symbolic learning.

Deep learning and symbolic learning are two different approaches frequently used in the field of artificial intelligence. The former is especially a central concern in current research with the resurgence data-driven learning paradigm. Without explicit common sense knowledge and cognitive reasoning, these data-hungry strategies are typically difficult to generalize [Marcus, 2020]. In contrast, symbolic learning involves the use of logical operators (e.g., AND ($\wedge$), OR ($\vee$) and NOT ($\neg$)) and human-interpretable representations of data (e.g., words or numbers) to perform language understanding or logical reasoning tasks. As for SR, DNN-based works (such as SASRec [Kang and McAuley, 2018], etc.) sort to learning expressive representations of items and generating recommendations by calculating the similarity between the representations of historical interactions and target items. However, rather than only calculating the similarity score, symbolic learning-based models focus more on making predictions based on the users' cognitive reasoning procedure [Chen *et al.*, 2021]. For example, after buying a laptop, a user may prefer to purchase a keyboard rather than a similar laptop.

At the same time, DNN and symbolic learning are complementary, and fusing them properly could combine the strengths of both approaches, thus improving the performance of deep learning models [Shi *et al.*, 2022]. For example, symbolic learning can provide a more flexible logical structure to latent features that are learned from DNN. Additionally, the introduction of deep learning enables end-to-end training of the symbolic learning and reasoning process. However, neural-symbolic learning for SR remains a challenging problem with open issues in the following aspects. First, the most recent models for logical reasoning are embedding-based. The feature description and logical representation are coupled in the same framework, which makes it hard to distinguish which latent feature contributes to feature representation or logical reasoning [Shi *et al.*, 2020]. And intuitively, the representations that work for feature description and logical reasoning in the model may influence the recommendation differently. Second, most of them assume user preferences are static and embed users and items in a deterministic manner, but ignore that the user's tastes are full of uncertainty and evolving by nature, which incurs inaccurate

[*]Corresponding authors.

recommendations [Zhao *et al.*, 2021a].

In this paper, we enhance the classical DNN-based models with logical reasoning and propose a general framework named **S**equential **R**ecommendation with **P**robabilistic **L**ogical **R**easoning (short for SR-PLR). In our framework, feature embedding and logic embedding are disentangled in disparate networks, which enables SR-PLR to be benefited from both similarity matching and logical reasoning. Specifically, for the feature part, we take DNN-based SR methods (such as SASRec, GRU4Rec, etc.) as the backbone to learn the powerful latent representations of sequences. For the logical part, two transfer matrices are mentioned to convert the original feature embedding into several independent Beta distributions to represent historical items. Then two closed probabilistic logical operators (i.e., AND, NOT) are conducted on these distributions to infer the target items' distribution with the KL-divergence (Kullback-Leibler). Finally, the feature representation obtained from traditional SR methods is concatenated with the logical representation sampled from the output Beta distribution to make the prediction.

In a nutshell, the contributions of our paper can be concluded as follows:

- We propose a general framework for sequential recommendation that combines the deep learning method with symbolic learning.
- We develop a probabilistic embedding method for recommendation, and conduct probabilistic logical reasoning on users' interaction behaviors for better capturing the uncertainty and evolution of user tastes.
- We successfully implement our framework to traditional and newly released DNN SR models, and our experimental results show that the performance of all these models can be improved with the help of probabilistic logical reasoning.

2 Related Work

There are multiple topics related to our SR-PLR. Here, we first present some sequential recommendation works, and then introduce some neural-symbolic, probabilistic embedding recommendation models in this section.

2.1 Sequential Recommendation

In early works, Markov chains, Recurrent Neural Network (RNN), and Convolutional Neural Network (CNN) are commonly used in SR. For example, FPMC [Rendle *et al.*, 2010] relies on modeling item-item transition relationships and predicts the next item with the last interaction of a user. GRU4Rec [Hidasi *et al.*, 2016] captures the sequential patterns with a multi-layer Gate Recurrent Unit (GRU) structure and NARM [Li *et al.*, 2017] further imports the item-level attention mechanism into GRU to measure the importance of different items. Simultaneously, Caser [Tang and Wang, 2018] explores the application of CNN for sequential recommendation, which uses two types of convolutional filters to extract the information hidden in the users' sequences. Recently, SASRec [Kang and McAuley, 2018] introduces self-attention into recommendation systems and achieves great success. Beyond that, MLP4Rec [Ma *et al.*, 2019] simply stacks multiple MLP layers to model users' dynamic preferences. S3Rec [Zhou *et al.*, 2020] and DuoRec [Qiu *et al.*, 2022] boost the performance with self-supervised signals. Different from these models, we aim to integrate symbolic learning into these DNN models and endow SR with cognition ability.

2.2 Neural-symbolic Recommendation

Recent neural-symbolic recommendation models can be divided into two categories. One type of neural-symbolic model (e.g., ENRL [Shi *et al.*, 2022] and NS-ICF [Zhang *et al.*, 2022]) aims to build an explainable recommender system with the aid of symbolic learning. They focus on learning interpretable recommendation rules on user and item attributes, and then output how these predictions are generated. Another type (e.g., LINN [Shi *et al.*, 2020], NCR-E [Chen *et al.*, 2021] and GCR [Chen *et al.*, 2022]) explores to represent logical operators with the multilayer perceptron and conducts logical reasoning with the guide of several logical regularizers. In this way, sequential behavior can be presented as a logical expression, so that conducting logical reasoning and prediction in a continuous space. However, different from these methods to conduct reasoning with solely embedding, we disentangle feature and logic embedding in different networks, and then combine feature learning and symbolic learning in a unified framework.

2.3 Probabilistic Embedding for Recommendation

Probability distributions are widely used in representing all the uncertain unobserved quantities in a model (including structural, parametric, and noise-related) and how they relate to the data [Ghahramani, 2015]. There are several works mentioned to model the users and items in probabilistic embeddings. For example, PMLAM [Ma *et al.*, 2020] and DDN [Zheng *et al.*, 2019] represents users and items as learnable Gaussian distributions, and uses Wasserstein distance to estimate whether the user will buy the target item or not. DT4SR [Fan *et al.*, 2021] learns the mean and covariance with different Transformers, which performs effectively for cold-start recommendation, and STOSA [Fan *et al.*, 2022] further proposes novel Wasserstein self-attention based on Gaussian distributions. Furthermore, some works use the deep generative model in SR with probabilistic methods. For example, VSAN [Zhao *et al.*, 2021a] fuse Variational AutoEncoders (VAE) with self-attention networks to capture the long and local dependencies in the behavior sequences. Different from these above probabilistic models, we embed users and items as Beta distributions for not only capturing uncertainty but also facilitating logical reasoning.

3 Formalization

Formally, let $\mathcal{U}$ and $\mathcal{V}$ be user and item sets. Given u's historical behaviors $\mathcal{V}_u = \{v_1, v_2, \cdots, v_m\} \subseteq \mathcal{V}$, the goal of SRs is to predict the next item that $u \in \mathcal{U}$ will interact with. For DNN SR models, they take $\mathcal{V}_u$ as input to predict the most possible top-K items, which can be formulated as $v_t = \arg\max_{v_i \in \mathcal{V}} P(v_{m+1} = v_i \mid \mathcal{V}_u)$, where v_t represents the target item of u.

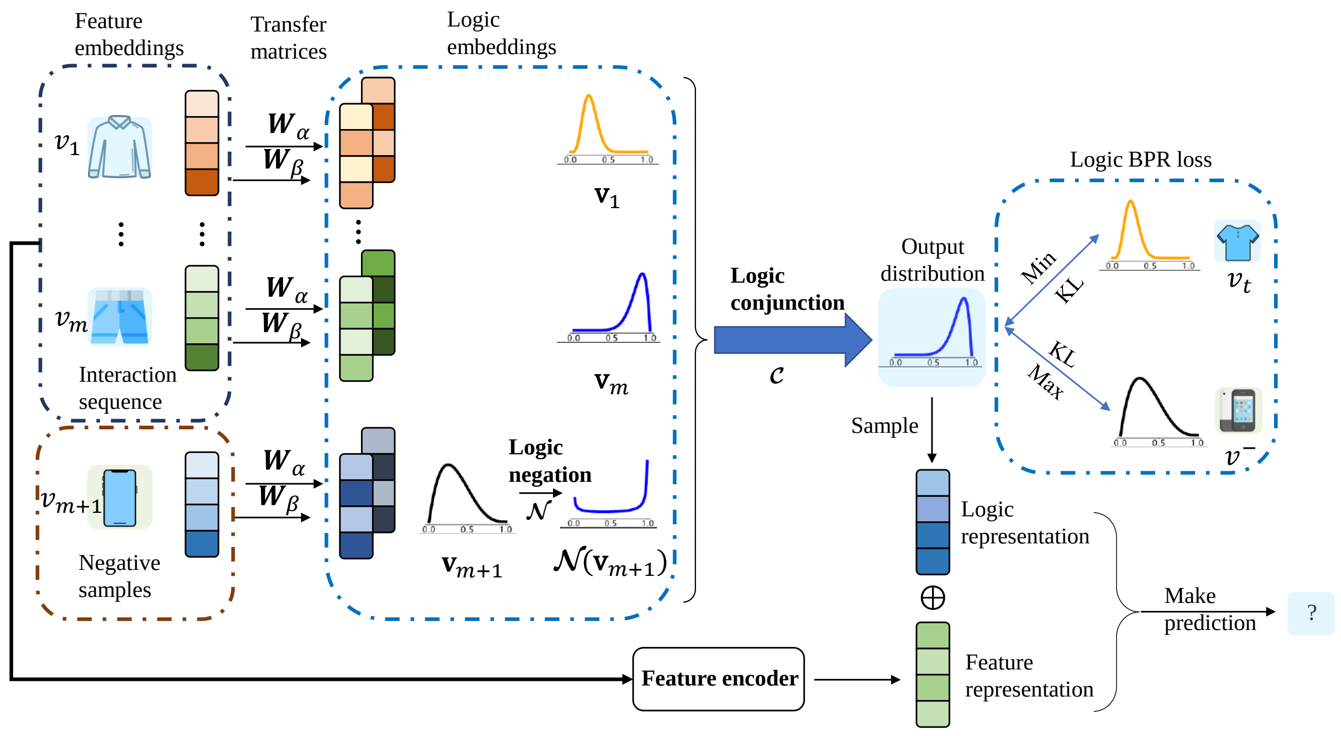


Figure 1: The framework of SR-PLR. $v_1, v_2, \cdots, v_m$ are u's historical items, v_t is a target item and $\mathbf{v}_i$ is their corresponding distributions. v_{m+1} and v^- represent the sampled items that the user does not interact with.

In this paper, we complement classical DNN with logical reasoning. For our logic part, the sequential recommendation task can be seen as a logical reasoning task. For example, let $\mathcal{V}_u^+, \mathcal{V}_u^- \subseteq \mathcal{V}_u$ denote u's clicked and unclicked item sets, $v_1, v_2 \in \mathcal{V}_u^+$ and $v_3 \in \mathcal{V}_u^-$, the logical formula $v_1 \wedge v_2 \wedge (\neg v_3)$ can be used to infer v_t.

4 Methodology

The SR-PLR framework shown as Figure 1 consists of two main parts: feature representation learning and probabilistic logical reasoning. For the feature part, SR-PLR utilizes traditional DNNs (e.g., GRU4Rec, SASRec, etc.) as the feature encoder to extract the feature representations of users' behavior sequences. For the logic part, SR-PLR conducts logical reasoning with a probabilistic method, which will be fully explained from three points: probabilistic logic embedding, probabilistic logical operators and logical reasoning. More details are illustrated in the following parts.

4.1 Feature Representation Learning

We first briefly introduce how to learn feature representations in SR-PLR. Following the most widely used manner in SR models, the backbone feature encoder used in our framework contains two parts: the embedding layer and the feature encoder.

Embedding Layer

In our framework, ID information is all we need to build the model, since side information (e.g., sex, category, etc.) may be not always available in practice. Hence, we embed the whole items' IDs into the same latent space [Kang and McAuley, 2018] and generate the ID item embedding matrix $\mathbf{M} \in R^{|\mathcal{V}| \times d}$, where d is the embedding dimension. Given the u's interaction sequence, the embedding of $\mathcal{V}_u$ is initialized to $\mathbf{e}^u \in R^{m \times d}$ and $\mathbf{e}^u = \{\mathbf{m}_1, \mathbf{m}_2, ..., \mathbf{m}_m\}$, where $\mathbf{m}_k \in R^d$ represents the item's embedding at the position k in the sequence.

Feature Encoder

Given the sequence embedding $\mathbf{e}^u$, a deep neural network model (e.g., SASRec) represented as $f_\theta(\cdot)$ is utilized to learn the representation of the sequence. The output representation of feature encoder $\mathbf{H}_f^u \in R^d$ is calculated as $\mathbf{H}_f^u = f_\theta(\mathbf{e}^u)$, and chosen as the representation of u's behavior sequence.

4.2 Probabilistic Logical Reasoning

As illustrated in the Introduction section, symbolic learning could grant DNN models the capability of cognition. However, how to represent users/items and how to conduct logical reasoning are still worth discussing.

Hence in this paper, we: (1) design a probabilistic logical embedding method for the recommender system to represent users/items in the logical space, (2) apply probabilistic logical operators on these probabilistic embeddings, and (3) get the logical representations for the sequence by using logical reasoning.

Probabilistic Logical Embedding
Different from previous neural-symbolic works that treat the same ID embeddings as both feature representation and logical variable, two different types of embedding are used in SR-PLR to describe features and conduct logical reasoning, respectively.

Specifically, we leverage the transfer matrix to covert ID embeddings into the logic space, and use multiple independent Beta distributions to demonstrate items thus modeling the uncertainty of user's tastes. Where Beta distribution refers to a continuous probability distribution defined on [0, 1], and its Probability Density Function (PDF) is $p_{[(\alpha,\beta)]}(x) = \frac{1}{B(\alpha,\beta)}x^{\alpha-1}(1-x)^{\beta-1}$, where $x \in [0,1]$, shape parameters $\alpha, \beta \in [0, \infty]$, and $B(\alpha,\beta)$ is the beta function. Given ID embeddings, we use two transfer matrices represented as $\mathbf{W}_\alpha$ and $\mathbf{W}_\beta \in R^{d\times d}$ to transfer $\mathbf{M}$ into two shape matrices $\boldsymbol{\alpha}$ and $\boldsymbol{\beta} \in R^{m\times d}$:

$$\boldsymbol{\alpha} = \mathbf{M}\mathbf{W}_\alpha, \boldsymbol{\beta} = \mathbf{M}\mathbf{W}_\beta \tag{1}$$

For v_i, each dimension of $\mathbf{v}_i = [(\boldsymbol{\alpha}_i, \boldsymbol{\beta}_i)] = [(\alpha_{i,1}, \beta_{i,1}), (\alpha_{i,2}, \beta_{i,2}), \cdots, (\alpha_{i,d}, \beta_{i,d})]$ characterizes the uncertainty by an independent Beta distribution instead of a single value, and its PDF $p_{\mathbf{v}_i}(x)$ is denoted as $\mathbf{P}(\mathbf{v}_i) = [p_{[(\alpha_{i,1},\beta_{i,1})]}(x), p_{[(\alpha_{i,2},\beta_{i,2})]}(x), \cdots, p_{[(\alpha_{i,d},\beta_{i,d})]}(x)]$. Following BetaE [Ren and Leskovec, 2020], we clamp all elements in $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ into (0, 10^9] after multiplying transfer matrices. Note that items' representations in SR-PLR are assumed to follow the Beta distribution rather than the Gaussian distribution because we aim to guarantee the probabilistic logical operators on these Beta embeddings are closed. More details will be illustrated in the next section.

Probabilistic Logical Operators
In this section, we define two probabilistic operators on the Beta embedding space, which are called probabilistic negation operator ($\mathcal{N}$) and conjunction operator ($\mathcal{C}$), respectively. Note that since the disjunction operator can be implemented using negation and conjunction with De Morgan's laws [Ren and Leskovec, 2020], only the aforementioned two operators are discussed in our framework.

Probabilistic negation operator: For probabilistic negation operator, $\mathcal{N}(\mathbf{v}_i)$ is defined as the reciprocal of $\mathbf{v}_i$:

$$\mathcal{N}(\mathbf{v}_i) = [(\frac{1}{\alpha_{i,1}}, \frac{1}{\beta_{i,1}}), (\frac{1}{\alpha_{i,2}}, \frac{1}{\beta_{i,2}}), \cdots, (\frac{1}{\alpha_{i,d}}, \frac{1}{\beta_{i,d}})] \tag{2}$$

It can be seen that, different from [Shi *et al.*, 2020] that relies on logical regularizations, $\mathcal{N}$ naturally satisfies $\mathcal{N}(\mathcal{N}(\mathbf{v}_i)) = \mathbf{v}_i$. In addition, as shown in Figure 1, operator $\mathcal{N}$ can inherently reverse high probability density to low density and vice versa [Ren and Leskovec, 2020], which enables $\mathcal{N}(\mathbf{v}_i)$ to represent the dislikes item v_i of users.

Probabilistic conjunction operator: Given a user's behaviors $\mathcal{V}_u = \{v_1, v_2, \cdots, v_m\}$ and their embeddings $\mathbf{v}_1 = [(\boldsymbol{\alpha}_1, \boldsymbol{\beta}_1)], \mathbf{v}_2 = [(\boldsymbol{\alpha}_2, \boldsymbol{\beta}_2)], \cdots, \mathbf{v}_m = [(\boldsymbol{\alpha}_m, \boldsymbol{\beta}_m)]$, we define the output of conjunction $\overline{\mathbf{v}} = \mathcal{C}(\{\mathbf{v}_1, \mathbf{v}_2, \cdots, \mathbf{v}_m\})$, where $\mathcal{C}$ is probabilistic conjunction operator. Following [Ren and Leskovec, 2020], $\overline{\mathbf{v}}$'s distributions can be represented as

$$\overline{\mathbf{v}} = [(\sum_{i=1}^{m} \mathbf{w}_i \odot \boldsymbol{\alpha}_i, \sum_{i=1}^{m} \mathbf{w}_i \odot \boldsymbol{\beta}_i)] \tag{3}$$

where $\sum$ and $\odot$ are the element-wise summation and product, respectively. $\mathbf{w}_i \in R^d$ is a weight vector. Its j-th dimension $w_{i,j}$ satisfy $\Sigma_{i=1}^{m} w_{i,j} = 1$. In this way, the PDF of $\overline{\mathbf{v}}$ is calculated as the weighted product of $\mathcal{V}_u$' PDFs:

$$p_{\overline{\mathbf{v}}}(x) = \prod p_{\mathbf{v}_1}^{\mathbf{w}_1}(x) p_{\mathbf{v}_2}^{\mathbf{w}_2}(x) \cdots p_{\mathbf{v}_m}^{\mathbf{w}_m}(x) \tag{4}$$

where $\prod$ is the element-wise product. In SR-PLR, we adopt the attention mechanism to learn the importance of different items during training:

$$\mathbf{w}_i = \frac{\exp(MLP(\boldsymbol{\alpha}_i \oplus \boldsymbol{\beta}_i))}{\sum_j \exp(MLP(\boldsymbol{\alpha}_j \oplus \boldsymbol{\beta}_j))} \tag{5}$$

where $\oplus$ represents concatenation operation, and $MLP : R^{2d} \to R^d$ is a multilayer perceptron taking the concatenation of $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ as its input thus making our logical operators can be learned end-to-end. Obviously, the defined conjunction operator satisfies the equation $\mathcal{C}(\{\mathbf{v}_1, \mathbf{v}_1, \cdots, \mathbf{v}_1\}) = \mathbf{v}_1$ and does not need any logical regularization.

It should be pointed out that both operators are closed in the Beta embedding space, which makes these operators could be combined in arbitrary ways and prevents exponential computation [Ren and Leskovec, 2020].

Logical Reasoning
After defining logical embedding and operators, it is convenient for us to conduct reasoning on historical items. For example, given $v_1, v_2 \in \mathcal{V}_u^+$ and $v_3 \in \mathcal{V}_u^-$, $v_1 \wedge v_2 \wedge (\neg v_3)$ can be represented as $\mathcal{C}(\{\mathbf{v}_1, \mathbf{v}_2, \mathcal{N}(\mathbf{v}_3)\})$ to calculate the target items' distribution. Generally, for $v_1, v_2, \cdots, v_m \in \mathcal{V}_u^+$ and sampled negative item $v_{m+1}, v_{m+2}, \cdots, v_n \in \mathcal{V}_u^-$, we apply logical operators on them

$$\overline{\mathbf{v}}_u = \mathcal{C}(\{\mathbf{v}_1, \mathbf{v}_2, \cdots, \mathbf{v}_m, \mathcal{N}(\mathbf{v}_{m+1}), \cdots, \mathcal{N}(\mathbf{v}_n)\}) \tag{6}$$

As logical operators $\mathcal{N}$ and $\mathcal{C}$ are closed, $\overline{\mathbf{v}}_u$ is also a Beta embedding to represent the user's preference. Hence, by measuring the KL-divergence distance between $\overline{\mathbf{v}}_u$ and target item's distribution $\mathbf{v}_t$, the reasoning result of logic network can be obtained:

$$\text{Dist}(\overline{\mathbf{v}}_u, \mathbf{v}_t) = \sum_{k=1}^{d} \text{KL}(\mathbf{P}_k(\mathbf{v}_t), \mathbf{P}_k(\overline{\mathbf{v}}_u)) \tag{7}$$

where $\mathbf{P}_k(\overline{\mathbf{v}}_u)$ represents the k-th dimension of $\mathbf{P}(\overline{\mathbf{v}}_u)$. To this end, we utilize a pair-wise loss (e.g., Bayesian Personalized Ranking (BPR) [Rendle *et al.*, 2009]) to optimize the parameters of this logic network:

$$\mathcal{L}_l = \log(\sigma(\text{Dist}(\overline{\mathbf{v}}_u, \mathbf{v}_t) - \text{Dist}(\overline{\mathbf{v}}_u, \mathbf{v}^-)) \tag{8}$$

where $\mathbf{v}^-$ is a embedding of $v^- \in \mathcal{V}_u^-$, and σ is the sigmoid function.

4.3 Training and Prediction

To fuse DNN and logic network together, we are going to concatenate the feature representation $\mathbf{H}_f^u$ with logical representation $\mathbf{H}_l^u$ together to make the final prediction. The most direct way to get logical representations is to take samples from the distribution $\overline{\mathbf{v}}_u = [(\overline{\boldsymbol{\alpha}}_u, \overline{\boldsymbol{\beta}}_u)]$. However, the sample operation is not differentiable, which makes the process

Datasets	Users	Items	Ratings	Avg. Len.	Sparsity
Sports	35,598	18,357	296,337	8.3	99.95%
Toys	19,413	11,925	167,597	8.6	99.93%
Yelp	30,499	20,068	317,182	10.4	99.95%

Table 1: Statistics of datasets.

hard to be trained end-to-end. Hence, in our framework, we choose the mean of distribution $\overline{\mathbf{v}}_u$ to represent the sequence:

$$\mathbf{H}_l^u = \frac{\overline{\boldsymbol{\alpha}}_u}{\overline{\boldsymbol{\alpha}}_u + \overline{\boldsymbol{\beta}}_u} \tag{9}$$

Then, the corresponding prediction matrix $\hat{\mathbf{y}} \in R^{|\mathcal{V}|}$ can be generated by:

$$\hat{\mathbf{y}} = (\mathbf{H}_f^u \oplus \mathbf{H}_l^u)(\mathbf{M} \oplus \mathbf{E})^\top \tag{10}$$

where $\mathbf{E} = \frac{\boldsymbol{\alpha}}{\boldsymbol{\alpha}+\boldsymbol{\beta}}$. And we use the cross-entropy loss to approximate the ground truth $\mathbf{y}$:

$$\mathcal{L}_{Rec} = -\sum_{i=1}^{|\mathcal{V}|} y_i \log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i) \tag{11}$$

At last, the final objective function is:

$$\mathcal{L} = \mathcal{L}_{Rec} + \lambda \mathcal{L}_l \tag{12}$$

where λ is a hyperparameter.

5 Experiments

In this section, we conduct experiments with the aim of answering the following questions: **Q1**: How do our models SR-PLR perform compared with other baselines? **Q2**: What is the influence of key components of SR-PLR? **Q3**: Whether is SR-PLR sensitive to the hyperparameters? **Q4**: How is the robustness of SR-PLR?

5.1 Experimental Settings

Dataset

Experiments are conducted on three publicly available datasets. The statistics of each dataset are given in Table 1.

- **Amazon** [He and McAuley, 2016]: *Amazon Sports and Outdoors* and *Amazon Toys and Games* are two subcategories datasets crawled from Amazon, denoted as Sports and Toys.
- **Yelp**: Yelp is one of the most widely used datasets for business recommendation.

Following previous works [Kang and McAuley, 2018; Tang and Wang, 2018], we use the '5-core' version for all datasets and adopt the leave-one-out method to split these three datasets.

Evaluation Metrics

The widely used Normalized Discounted Cumulative Gain at rank K (NDCG@K) and Hit ratio at rank K (Hit@K) are used as evaluation metrics in our experiments, and we choose K from 5, 10. As recommended by [Krichene and Rendle, 2020], we adopt all-rank evaluation scores throughout the entire item set to ensure that the evaluation process is unbiased. As [Shi *et al.*, 2020; Chen *et al.*, 2021] are evaluated with sampled metrics in the original paper, we compare SR-PLR with our re-implemented version.

Baseline Methods

Baselines used to compare with our models can be categorized into two groups.

DNN based models: For this group, we compare the Recurrent Neural Network (RNN), Convolutional Neural Network (CNN), self-attention based models, and newly released contrastive learning based model. They not only act roles as baselines, but also as the backbone of SR-PLR to demonstrate the effectiveness of combining logical reasoning.

- **GRU4Rec** [Hidasi *et al.*, 2016]: a method utilizing GRU to model user sequential behaviors as a strict order,
- **Caser** [Tang and Wang, 2018]: a method using both horizontal and vertical convolution for sequential recommendation,
- **SASRec** [Kang and McAuley, 2018]: a method based on the self-attention mechanism,
- **DuoRec** [Qiu *et al.*, 2022]: a method developing a positive sampling strategy and using contrastive learning for SR with a model-level augmentation.

Neural-symbolic models: For this group, we compare our framework with some neural-symbolic recommendation methods. As mentioned in [Chen *et al.*, 2021], there are two different versions that base on implicit feedback NCR-I and explicit feedback NCR-E. Since SR-PLR is implicit feedback based, for fair comparisons, we use the implicit feedback versions here for these two neural-symbolic baselines.

- **LINN** [Shi *et al.*, 2020]: a neural collaborative reasoning based recommendation method,
- **NCR-I** [Chen *et al.*, 2021]: a personalized method based on LINN.

Implementation Details

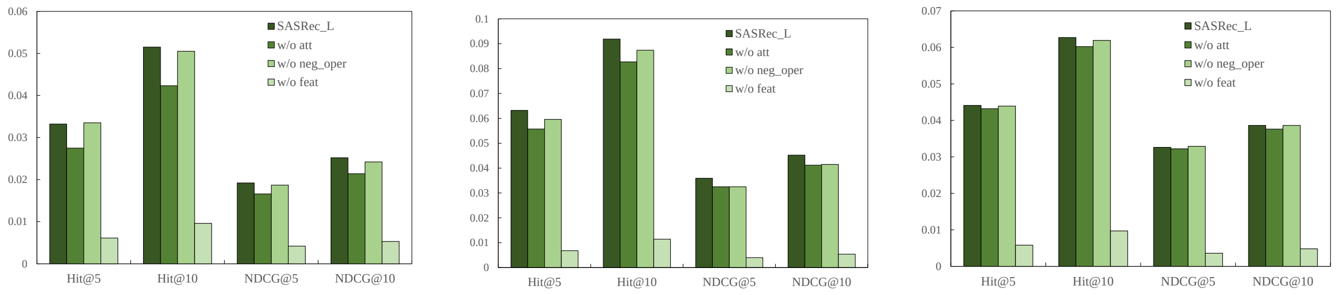
We run all methods in PyTorch [Paszke *et al.*, 2017] with Adam [Kingma and Ba, 2015] optimizer on an NVIDIA Geforce 3070Ti GPU, and all these models are implemented based on RecBole [Zhao *et al.*, 2021b]. The batch size and the dimension of embeddings d are set to 2048 and 64 in our experiments. The max sequential length for all baselines is set as 50. We train all models 50 epochs. In the experiment, we keep all the hyperparameters of backbone models in RecBole unchanged and stack our logic network on them. SR-PLR is trained with a learning rate of 0.002. For the logic network, we set the λ in Eq. (12) as a hyperparameter and select it from [0, 1] with step 0.1. For the negative item number in Eq. (6), we choose it from 1 to 10.

5.2 Overall Performance (Q1)

We report the experimental results of different methods in Table 2. Note that 'XX_L' means the SR-PLR method that uses 'XX' as the backbone. Where the 'N' is short for NDCG and the numbers in bold indicate the better results that are generated by the same feature encoder. As we can see, the methods

Models		Amazon Sports				Amazon Toys				Yelp			
		HIT@5	HIT@10	N@5	N@10	HIT@5	HIT@10	N@5	N@10	HIT@5	HIT@10	N@5	N@10
RNN	GRU4Rec	0.0186	0.0300	0.0121	0.0158	0.0352	0.0519	0.0240	0.0294	0.0217	0.0360	0.0144	0.0189
	GRU4Rec_L	**0.0225**	**0.0353**	**0.0141**	**0.0182**	**0.0387**	**0.0557**	**0.0268**	**0.0323**	**0.0249**	**0.0433**	**0.0160**	**0.0220**
	Impro.	20.97%	17.67%	16.53%	15.19%	9.94%	7.32%	11.67 %	9.86%	14.74%	23.06%	11.11%	16.40%
CNN	Caser	0.0141	0.0226	0.0087	0.0115	0.0183	0.0302	0.0113	0.0151	0.0231	0.0351	0.0164	0.0202
	Caser_L	**0.0173**	**0.0283**	**0.0109**	**0.0144**	**0.0228**	**0.0390**	**0.0132**	**0.0185**	**0.0260**	**0.0372**	**0.0185**	**0.0221**
	Impro.	22.70%	25.22%	25.29%	25.27%	24.59%	22.56%	16.81%	22.52 %	12.55%	5.98%	12.80%	9.41%
Attention	SASRec	0.0317	0.0484	0.0172	0.0226	0.0630	0.0909	0.0354	0.0444	0.0422	0.0595	0.0322	0.0377
	SASRec_L	**0.0332**	**0.0515**	**0.0192**	**0.0252**	**0.0632**	**0.0919**	**0.0359**	**0.0452**	**0.0441**	**0.0627**	**0.0326**	**0.0386**
	Impro.	4.73%	6.40%	11.63%	11.50%	0.32 %	1.10%	1.41%	1.80%	4.50%	5.38%	1.24%	2.39%
DuoRec	DuoRec	0.0328	0.0505	0.0192	0.0249	0.0648	0.0929	**0.0388**	0.0479	0.0434	0.0618	0.0319	0.0378
	DuoRec_L	**0.0342**	**0.0522**	**0.0200**	**0.0257**	**0.0652**	**0.0946**	**0.0388**	**0.0484**	**0.0441**	**0.0624**	**0.0321**	**0.0380**
	Impro.	4.27%	3.37%	4.17%	3.21%	0.62%	1.83%	0%	1.04%	1.61%	0.97%	0.63%	0.53%
Logic	LINN	0.0151	0.0256	0.0101	0.0132	0.0196	0.0320	0.0133	0.0172	0.0215	0.0355	0.0146	0.0192
	NCR-I	0.0162	0.0263	0.0110	0.0146	0.0201	0.0322	0.0135	0.0176	0.0204	0.0354	0.0146	0.0191

Table 2: Overall performance on all datasets. 'XX_L' means the SR-PLR method that uses 'XX' as the backbone and the numbers in bold indicate the better results that are generated by the same feature encoder. 'N' denotes 'NDCG' and 'Impro.' denotes performance improvement compared with backbones.



(a) Sports (b) Toys (c) Yelp

Figure 2: Ablation study of SR-PLR on three datasets.

combining logical reasoning achieve better performance than other comparative models. Besides, there are more findings in these comparative experiments.

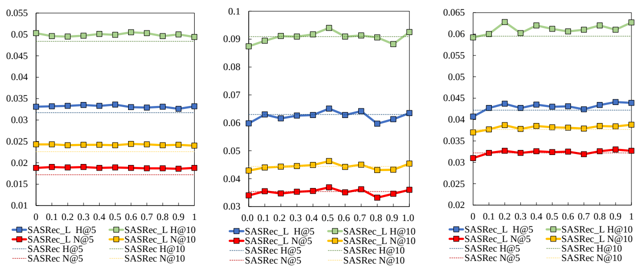
Firstly, SASRec achieves better performance than GRU4Rec and Caser for both NDCG and Hit on all three datasets. It shows that modeling the importance of different interactions is highly effective for describing users' preferences. The main reason is that the items in the sequences contribute differently to final results, e.g., the laptop makes more contributions than clothes in buying the keyboard, which is also why the attention mechanism is used in our probabilistic operator. Secondly, DuoRec performs better than SASRec in most cases, suggesting that pushing the positive views closer and pulling negative pairs away by contrastive learning are meaningful for getting better representations. Thirdly, both neural-symbolic models get similar results with GRU4Rec (consistent with the results reported in [Shi *et al.*, 2020; Chen *et al.*, 2021]). They also leverage deep neural networks to conduct logical reasoning, but only considering the logical equations of items may be not adequate for describing the complex relationship among interactions.

Finally, SR-PLR achieves performance improvement based on four types of feature encoders and performs best on three datasets. We contribute the improvement to the following aspects: (1) combining deep learning and symbolic learning in a dual feature-logic network, and (2) modeling users' dynamic preferences with a probabilistic method. As self-attention based SASRec performs better than RNN and CNN, meanwhile DuoRec is also the development of SASRec, we pay more attention to SASRec and perform more detailed experiments based on SASRec_L.

5.3 Ablation Study (Q2)

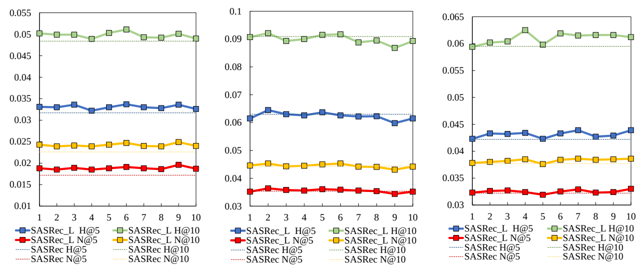
In this section, we conduct three ablation studies on all three datasets. We are going to examine the effectiveness of main three components, which are the attention mechanism in the probabilistic conjunction operator, the probabilistic negation operator and the feature network. Hence, three variants (which are named 'w/o att', 'w/o neg_oper' and 'w/o feat') are designed to compare against SASRec_L, and the results are shown in Figure 2. The details of these variants are as follows:

w/o att. We first evaluate the necessity of item-level attention in our probabilistic conjunction operator. The variant 'w/o att' means to compute sequence distribution by aggregating the hidden states via average pooling operations in Eq. (5). From Figure 2, we can see that our SASRec_L far outperforms its variant without the attention operation on



(a) Sports (b) Toys (c) Yelp

Figure 3: Sensitivity of λ on three datasets.



(a) Sports (b) Toys (c) Yelp

Figure 4: Sensitivity of negative item number on three datasets.

	r		0.1	0.2	0.3	0.4	0.5
Sports	H@5	SASRec	0.0296	0.0292	0.0241	0.0214	0.0197
		SASRec_L	**0.0334**	**0.0310**	**0.0262**	**0.0215**	**0.0198**
	H@10	SASRec	0.0473	0.0462	0.0402	**0.0361**	**0.0327**
		SASRec_L	**0.0499**	**0.0484**	**0.0417**	0.0357	0.0320
	N@5	SASRec	0.0162	0.0158	0.0135	0.0122	0.0118
		SASRec_L	**0.0189**	**0.0185**	**0.0160**	**0.0136**	**0.0124**
	N@10	SASRec	0.0219	0.0213	0.0187	0.0169	0.0160
		SASRec_L	**0.0242**	**0.0241**	**0.0210**	**0.0182**	**0.0163**
Toys	H@5	SASRec	0.0582	0.0581	0.0504	0.0457	0.0384
		SASRec_L	**0.0626**	**0.0600**	**0.0582**	**0.0551**	**0.0446**
	H@10	SASRec	0.0880	0.0872	0.0779	0.0700	0.0594
		SASRec_L	**0.0890**	**0.0888**	**0.0879**	**0.0821**	**0.0733**
	N@5	SASRec	0.0329	0.0327	0.0285	0.0265	0.0228
		SASRec_L	**0.0359**	**0.0343**	**0.0337**	**0.0324**	**0.0271**
	N@10	SASRec	0.0426	0.0421	0.0374	0.0343	0.0295
		SASRec_L	**0.0444**	**0.0436**	**0.0433**	**0.0411**	**0.0363**

Table 3: Robustness analysis on Sports and Toys.

three datasets. It indicates that using a fixed weight matrix will be a bottleneck to modeling the dynamic evolving tastes of users. Our attention based logical operation can adaptively measure the importance score for each item, which helps to improve the model performance.

w/o neg_oper. The negation operator is also an essential part of SR-PLR. For w/o neg_oper, we use positive items and conjunction operator to calculate the distribution of sequences, that is using $\overline{\mathbf{v}}_u = \mathcal{C}(\{\mathbf{v}_1, \mathbf{v}_2, \cdots, \mathbf{v}_m\})$ in Eq. (6). From Figure 2, it can be seen that the recommendation performance can be booted with the help of operator $\mathcal{N}$. Note that there is nearly no work to consider to conclude negative items during representing the interaction sequences except neural-symbolic based models. Our experiments indicate that the negative operator is vital to make logical reasoning complete, and using the unclicked items with the negative operator in symbolic learning may be another way to leverage unlabeled data, which may be useful for the semi-supervised recommendation.

w/o feat. We also evaluate the effectiveness of the feature network. For w/o feat, we alternatively make the prediction by using $\hat{\mathbf{y}} = \mathbf{H}_l^u \mathbf{E}^\top$ for Eq. (10). As shown in Figure 2, only using the probabilistic logic network will result in a performance drop, which indicates the importance of combing feature learning with symbolic learning, and the effectiveness of probabilistic logic network in boosting feature learning.

5.4 Hyper-parameter Sensitivity (Q3)

In this section, we are going to investigate how the performance varied with the changes of λ and the sampled negative item number. In experiments, we vary one of these two hyperparameters and keep others unchanged to investigate the influence of λ and the negative item number.

Sensitivity of λ. For λ, we conduct the experiment in the range from 0 to 1 for all three datasets. The experimental results are shown in Figure 3. In this figure, we compare SASRec_L with SASRec under different λ. It can be seen that adding a logical regularizer, i.e., the $\mathcal{L}_l$ in Eq. (8), to optimize the logic network is helpful for the recommendation performance. Essentially, it acts as an auxiliary task to push the sequences' distribution close to the positive items' distributions and far away from the negative ones. In addition, it matters to carefully find a suitable value for each dataset, and values around 0.5 are recommended.

Sensitivity of sampled negative item number. For the negative item used for operator $\mathcal{N}$, we choose it ranges from 1 to 10, and the experimental results are indicated in Figure 4. It can be seen that, in most cases, introducing negative items is effective for logical reasoning, especially in Sports and Yelp. As we use implicit feedback in SR-PLR, the sampled negative items may be not truly disliked by the user, hence the number of negative items is set relatively small during training to maintain the semantic of sequence.

5.5 Robustness Analysis (Q4)

In the logic network, we endow SR-PLR with the capability of modeling uncertainty by using the Beta distribution to represent the items. To verify this point, we randomly mask some items in the sequence with probability r during training and evaluate the performance of models in the original test dataset. Ideally, the item distribution contains more information than the static embedding, which makes SR-PLR could maintain its superiority with distorted sequences. As the results in Table 3 illustrated, though both models' performances slowly drop, SASRec_L still beats SASRec in most cases when r varies from 0.1 to 0.5 in two datasets, which indicates SR-PLR is more robust by using the probabilistic method.

6 Conclusion

In this paper, we proposed a general framework named SR-PLR to combine deep learning with symbolic learning via a Beta embedding method. Our main idea is two-fold: One is to disentangle items' embeddings and endow the basic DNN based sequential recommendation with cognitive capability; Another is to model the uncertainty and dynamic tastes of users. Therefore, we designed a dual feature-logic network and applied probabilistic logical operators on the items' Beta embeddings. Experiments on three real-world datasets showed that SR-PLR exhibits significant improvement against traditional baselines and neural-symbolic models. Our future work will consider modeling content information and knowledge graph with more interpretable and transferable logic neural networks in sequential recommendation, and that may lead this framework to be more transparent and achieve significant performance.

Acknowledgments

This research is partially supported by the NSFC (61876117, 62176175), the major project of natural science research in Universities of Jiangsu Province (21KJA520004), Suzhou Science and Technology Development Program (SYC2022139), the Priority Academic Program Development of Jiangsu Higher Education Institutions.

References

[Chen *et al.*, 2021] Hanxiong Chen, Shaoyun Shi, Yunqi Li, and Yongfeng Zhang. Neural collaborative reasoning. In *WWW 2021*, pages 1516–1527, 2021.

[Chen *et al.*, 2022] Hanxiong Chen, Yunqi Li, Shaoyun Shi, Shuchang Liu, He Zhu, and Yongfeng Zhang. Graph collaborative reasoning. In *WSDM 2022*, pages 75–84, 2022.

[Fan *et al.*, 2021] Ziwei Fan, Zhiwei Liu, Shen Wang, Lei Zheng, and Philip S. Yu. Modeling sequences as distributions with uncertainty for sequential recommendation. *CIKM 2021*, page 3019–3023, 2021.

[Fan *et al.*, 2022] Ziwei Fan, Zhiwei Liu, Yu Wang, Alice Wang, Zahra Nazari, Lei Zheng, Hao Peng, and Philip S. Yu. Sequential recommendation via stochastic self-attention. In *WWW 2022*, pages 2036–2047, 2022.

[Ghahramani, 2015] Zoubin Ghahramani. Probabilistic machine learning and artificial intelligence. *Nature*, 521(7553):452–459, 2015.

[He and McAuley, 2016] Ruining He and Julian McAuley. Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In *WWW 2016*, pages 507–517, 2016.

[Hidasi *et al.*, 2016] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. Session-based recommendations with recurrent neural networks. In *ICLR 2016, arXiv:1511.06939*, 2016.

[Kang and McAuley, 2018] Wang-Cheng Kang and Julian J. McAuley. Self-attentive sequential recommendation. In *ICDM 2018*, pages 197–206, 2018.

[Kingma and Ba, 2015] Diederik P. Kingma and Jimmy Ba. Adam: a method for stochastic optimization. In *ICLR 2015, arXiv:1412.6980*, 2015.

[Krichene and Rendle, 2020] Walid Krichene and Steffen Rendle. On sampled metrics for item recommendation. In *KDD 2020*, pages 1748–1757, 2020.

[Li *et al.*, 2017] Jing Li, Pengjie Ren, Zhumin Chen, Zhaochun Ren, Tao Lian, and Jun Ma. Neural attentive session-based recommendation. In *CIKM 2017*, pages 1419–1428, 2017.

[Ma *et al.*, 2019] Chen Ma, Peng Kang, and Xue Liu. Hierarchical gating networks for sequential recommendation. In *KDD 2019*, pages 825–833, 2019.

[Ma *et al.*, 2020] Chen Ma, Liheng Ma, Yingxue Zhang, Ruiming Tang, Xue Liu, and Mark Coates. Probabilistic metric learning with adaptive margin for top-k recommendation. In *KDD 2020*, pages 1036–1044, 2020.

[Marcus, 2020] Gary Marcus. The next decade in AI: four steps towards robust artificial intelligence. *arXiv:2002.06177*, 2020.

[Paszke *et al.*, 2017] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in PyTorch. 2017.

[Qiu *et al.*, 2022] Ruihong Qiu, Zi Huang, Hongzhi Yin, and Zijian Wang. Contrastive learning for representation degeneration problem in sequential recommendation. In *WSDM 2020*, pages 813–823, 2022.

[Ren and Leskovec, 2020] Hongyu Ren and Jure Leskovec. Beta embeddings for multi-hop logical reasoning in knowledge graphs. In *NIPS 2020*, pages 19716–19726, 2020.

[Rendle *et al.*, 2009] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. BPR: Bayesian personalized ranking from implicit feedback. In *UAI 2009*, pages 452–461, 2009.

[Rendle *et al.*, 2010] Steffen Rendle, Christoph Freudenthaler, and Lars Schmidt-Thieme. Factorizing personalized Markov chains for next-basket recommendation. In *WWW 2010*, pages 811–820, 2010.

[Shi *et al.*, 2020] Shaoyun Shi, Hanxiong Chen, Weizhi Ma, Jiaxin Mao, Min Zhang, and Yongfeng Zhang. Neural logic reasoning. In *CIKM 2020*, pages 1365–1374, 2020.

[Shi *et al.*, 2022] Shaoyun Shi, Yuexiang Xie, Zhen Wang, Bolin Ding, Yaliang Li, and Min Zhang. Explainable neural rule learning. In *WWW 2022*, pages 3031–3041, 2022.

[Tang and Wang, 2018] Jiaxi Tang and Ke Wang. Personalized top-n sequential recommendation via convolutional sequence embedding. In *WSDM 2018*, pages 565–573, 2018.

[Zhang *et al.*, 2022] Wei Zhang, Junbing Yan, Zhuo Wang, and Jianyong Wang. Neuro-symbolic interpretable collaborative filtering for attribute-based recommendation. In *WWW 2022*, pages 3229–3238, 2022.

[Zhao *et al.*, 2021a] Jing Zhao, Pengpeng Zhao, Lei Zhao, Yanchi Liu, Victor S. Sheng, and Xiaofang Zhou. Variational self-attention network for sequential recommendation. In *ICDE 2021*, pages 1559–1570, 2021.

[Zhao *et al.*, 2021b] Wayne Xin Zhao, Shanlei Mu, Yupeng Hou, Zihan Lin, Yushuo Chen, Xingyu Pan, Kaiyuan Li, Yujie Lu, Hui Wang, Changxin Tian, Yingqian Min, Zhichao Feng, Xinyan Fan, Xu Chen, Pengfei Wang, Wendi Ji, Yaliang Li, Xiaoling Wang, and Ji-Rong Wen. Recbole: Towards a unified, comprehensive and efficient framework for recommendation algorithms. In *CIKM 2021*, pages 4653–4664, 2021.

[Zheng *et al.*, 2019] Lei Zheng, Chaozhuo Li, Chun-Ta Lu, Jiawei Zhang, and Philip S. Yu. Deep distribution network: Addressing the data sparsity issue for top-n recommendation. *SIGIR 2019*, page 1081–1084, 2019.

[Zhou *et al.*, 2020] Kun Zhou, Hui Wang, Wayne Xin Zhao, Yutao Zhu, Sirui Wang, Fuzheng Zhang, Zhongyuan Wang, and Ji-Rong Wen. S3-rec: Self-supervised learning for sequential recommendation with mutual information maximization. In *CIKM 2020*, pages 1893–1902, 2020.