Does Every Data Instance Matter? Enhancing Sequential Recommendation by Eliminating Unreliable Data

Yatong Sun[1] , **Bin Wang**[1] , **Zhu Sun**[2] and **Xiaochun Yang**[1*]
[1]Northeastern University, China
[2]Macquarie University, Australia
yatong@stumail.neu.edu.cn, {binwang, yangxc}@mail.neu.edu.cn, z.sun@mq.edu.au

Abstract

Most sequential recommender systems (SRSs) predict next-item as target for each user given its preceding items as input, assuming that each input is related to its target. However, users may unintentionally click on items that are inconsistent with their preference. We are the first to empirically verify that SRSs can be misguided with such unreliable instances (i.e. targets mismatch inputs). This inspires us to design a novel SRS **B**y **E**liminating un**R**eliable **D**ata (BERD) guided with two observations: (1) unreliable instances generally have high training loss; and (2) high-loss instances are not necessarily unreliable but uncertain ones caused by blurry sequential patterns. Accordingly, BERD models both loss and uncertainty of each instance via a Gaussian distribution to better distinguish unreliable instances; meanwhile an uncertainty-aware graph convolution network is exploited to assist in mining unreliable instances by lowering uncertainty. Experiments on four real-world datasets demonstrate the superiority of our proposed BERD.

1 Introduction

Sequential recommender systems (SRSs) generally predict each user's next-item given the preceding (and succeeding) item sequence [Pei *et al.*, 2017; Yuan *et al.*, 2020]. Thereby, a training instance of SRSs is typically constructed by an input item sequence and its next-item as target. Such training procedure implicitly assumes that each input is related to its target, which might not always hold. For example, we may accidentally click on songs or videos that we are not interested in, or receive recommendations from friends who are inconsistent with our preference. Such behaviors could cause unreliable instances (i.e. targets mismatch inputs), which has been rarely investigated by previous studies.

To fill this gap, we first verify the existence and severity of unreliable instances by evaluating seven state-of-the-art SRSs. The results unveils that SRSs trained without unreliable data detected by our proposed data reliability measurement reach higher accuracy with a maximum 5.71% lift on NDCG, which firmly emphasizes the necessity of eliminating unreliable data for performance-enhanced SRSs. Based on these, we further conduct analysis and gain two insightful observations: (1) unreliable instances are prone to have higher training loss; and (2) high-loss instances are not necessarily unreliable. They can also be uncertain instances caused by blurry patterns of input sequences that are correlated with uncertain users whose preferences shift frequently, or uncertain items that co-occur with various subsequent items.

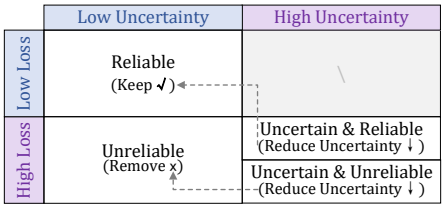


Figure 1: Classification of instances based on loss and uncertainty.

Next, we exploit both loss and uncertainty as guidance to better distinguish unreliable instances, as depicted in Fig. 1. First, instances with low loss and low uncertainty can be kept as *reliable* ones. Second, instances with high loss but low uncertainty can be cautiously removed as *unreliable* ones. Third, instances with high loss and high uncertainty are either *uncertain & reliable* or *uncertain & unreliable* instances. Hence, an SRS is expected to be capable of lowering the uncertainty, so as to identify and handle unreliable instances as indicated by the arrows in Fig. 1. Lastly, low loss and high uncertainty are conceptually contradictory, as high uncertainty is generally caused by blurry patterns of input sequences which are hard to fit with high training loss.

However, there are two challenges for achieving such effects: (1) how to accurately model loss and uncertainty of per-instance; and (2) how to mine unreliable instances from high uncertainty ones. To this end, we propose a novel framework BERD that enhances SRS **B**y **E**liminating un**R**eliable **D**ata. Specifically, BERD models the input of each instance as a Gaussian distribution, with its mean representing the extracted dynamic user preference for loss calculation, and its covariance representing the uncertainty of the extracted preference. An extended objective function is designed to properly endow uncertainty to every training instance. Meanwhile, we equip BERD with a novel uncertainty-aware graph

*Corresponding Author

convolution network (UGCN) so as to reduce the uncertainty of instances with blurry sequential patterns. As such, uncertain instances (either reliable or unreliable) can be clearly identified and properly handled as depicted in Fig. 1.

In sum, our contributions are threefold: **(1)** we are the first to define unreliable instances (i.e. targets mismatch inputs) in SRSs; by verifying their presence and severity, we highlight the necessity of eliminating them for performance-enhanced SRSs; **(2)** we conduct analysis gaining two insightful observations, which inspire us to design BERD to detect unreliable instances by modelling per-instance loss and uncertainty via a Gaussian distribution. Moreover, a novel UGCN facilitates to mine unreliable instances from high uncertainty ones by lowering instance uncertainty; and **(3)** extensive experiments on four real-world datasets demonstrate the superiority of BERD. Additionally, detailed ablation study further confirms the effectiveness of each module of BERD.

2 Related Works

Early SRSs adopt Markov Chains to model users' dynamic preference, such as FPMC [Rendle *et al.*, 2010]. Recently, deep learning advances are more prevalent. For example, Caser [Tang and Wang, 2018] and GRU4Rec [Cho *et al.*, 2014] exploit CNN and RNN to model users' sequential behaviours, respectively. Later, researchers find it is essential to highlight relevant input items given the target. Typical works include NARM [Li *et al.*, 2017], RUM [Chen *et al.*, 2018], SASRec [Kang and McAuley, 2018] and GARG [Wu *et al.*, 2020], which adopt attention mechanisms; MCPRN [Wang *et al.*, 2019] proposes a multi-channel purpose routing network; HRL [Zhang *et al.*, 2019] leverages the hierarchical reinforcement learning. BERT4Rec [Sun *et al.*, 2019] and GRec [Yuan *et al.*, 2020] verify the benefits of bidirectional context for dynamic preference modelling. Besides, HGN [Ma *et al.*, 2019], MA-GNN [Ma *et al.*, 2020] and M3R [Tang *et al.*, 2019] reveal the usefulness of modelling long-term user interests.

Previous works, however, posit each input sequence as a whole should be related to its target, thus fail to handle unreliable instances (i.e. targets mismatch inputs). Although several studies have investigated noisy ratings [Burke *et al.*, 2006; Li *et al.*, 2019], they are mainly designed for the rating prediction task with static preference and beyond our scope.

3 Reliability Analysis on Data Instances

In SRSs, each user u chronologically interacts with a list of items $S^u = [i^u_1, i^u_2, \dots, i^u_{|S^u|}]$, where i^u_m is the m-th item u interacts with ($1 \le m \le |S^u|$). During training, an SRS aims to predict u's next item as target, given her id and L preceding items as input. Formally, the training instance for u to predict her t-th interaction is denoted by an input-target pair $< \{u, S^u_{t,L}\}, i^u_t >$, where $S^u_{t,L} = [i^u_{max(1,t-L)}, \dots, i^u_{t-2}, i^u_{t-1}]$. The instance with target i^u_t mismatching its input $S^u_{t,L}$ is called an unreliable instance.

Measurement of Data Reliability

To quantify the reliability of each instance, i.e., the matching degree between inputs and targets, we propose a data reliability measurement with the aid of item attributes (e.g., category and tag) and item co-occurrence relation. Assume there are A attributes and N items. Each item i is characterized by two tf-idf vectors $\boldsymbol{q}^a_i \in \mathbb{R}^A$ and $\boldsymbol{q}^c_i \in \mathbb{R}^N$, treating attributes and co-occurred items as terms, respectively. Taking instance $< \{u, S^u_{t,L}\}, i^u_t >$ as an example, we define the similarity between $S^u_{t,L}$ and i^u_t as the averaged cosine similarity between i^u_t and each item within $S^u_{t,L}$, given by,

$$\text{sim}^a(S^u_{t,L}, i^u_t) = \frac{1}{|S^u_{t,L}|} \sum\nolimits_{j \in S^u_{t,L}} \frac{\boldsymbol{q}^a_j \cdot \boldsymbol{q}^a_{i^u_t}}{||\boldsymbol{q}^a_j|| \, ||\boldsymbol{q}^a_{i^u_t}||}, \quad (1)$$

where $\cdot$ indicates vector dot product; $||\cdot||$ denotes the L_2 norm; $\text{sim}^a(\cdot)$ indicates the similarity measured by item attributes; $\text{sim}^c(\cdot)$ can be analogically calculated with $\boldsymbol{q}^c_i$. We then sample a set of negative items $\mathcal{I}^-_u$ ($\mathcal{I}^-_u \cap S^u = \varnothing$) and rank the target i^u_t among the negative ones to get the relative matching degree between the targets and inputs:

$$r^a = \frac{1}{|\mathcal{I}^-_u|} \sum\nolimits_{j^- \in \mathcal{I}^-_u} \left[\text{sim}^a(S^u_{t,L}, i^u_t) - \text{sim}^a(S^u_{t,L}, j^-)\right]_+, \quad (2)$$

where $[x]_+$ is an indicator function whose value is 1 with $x > 0$, and 0 otherwise; r^a indicates the relative matching degree between the target and its input w.r.t. attributes, for example, $r^a = 0.7$ means the target matches the input better than 70% of negative items. r^c holds similar calculation and meaning w.r.t. item co-occurrence. The measurement classifies the instances with both r^a and r^c lower than 0.9 as unreliable.

Existence and Severity of Unreliable Data

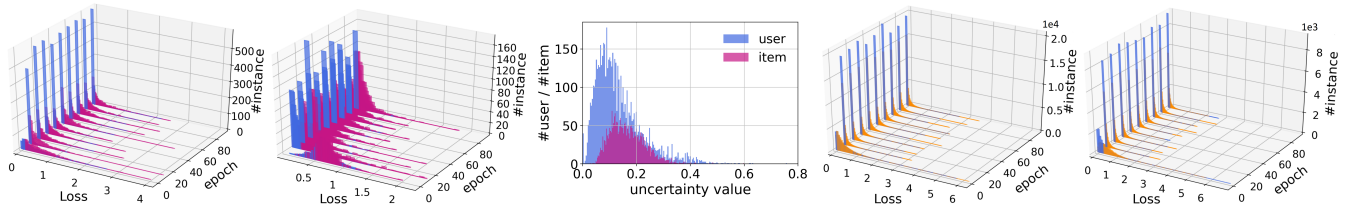
To verify the existence and severity of unreliable instances, we evaluate the performance of seven state-of-the-art SRSs on four real-world datasets (see Table 2) under two views w.r.t. the training process: **Origin** – directly uses the original training instances; **Filter** – removes the unreliable data from training sets based on our proposed reliability measurement, where the column 'U-Ratio' in Table 2 shows their respective percentage w.r.t. the total training instances. The final recommendation results are reported in Fig. 4, where both views are also considered during test (i.e. Test-Origin and Test-Filter). We can observe that SRSs trained with filtered datasets (i.e. Train-Filter) consistently reach higher accuracy with a maximum lift of 3.88%, 5.71% on NDCG under both test views, respectively. This validates the existence and severity of unreliable instances, as well as shows the necessity of eliminating them for performance-enhanced SRSs.

Characteristics of Unreliable Data

Although the eliminated instances are generally unreliable based on our proposed reliability measurement, the reliance on high-quality auxiliary information restricts its generality. Therefore, we further conduct an in-depth analysis on the detected unreliable instances, whereby two insightful observations are noted. These help us discover intrinsic and distinctive characteristics of unreliable instances.

Obs. 1. *Unreliable instances generally have high training loss; but high-loss instances are not necessarily unreliable.*

Figs. 2(a-b) show the loss distribution of reliable (blue bars) and unreliable (red bars) instances at different epochs when training Caser on ML-1M and CD. Similar trends can be



(a) ML-1M (b) CD (c) ML-1M (d) ML-1M User (e) ML-1M Item

Figure 2: (a-b) are the loss distribution of reliable (blue) and unreliable (red) instances at different epochs when training Caser on ML-1M and CD, respectively; (c) shows the uncertainty value of user (blue) and item (red) on ML-1M; (d-e) depicts the loss distribution of reliable instances on ML-1M, where yellow (blue) bars correspond to instances that are related (unrelated) to uncertain users (d) or items (e).

found with other SRSs and datasets. We note that (1) the loss of unreliable instances is generally higher than that of reliable ones; and (2) the high-loss area also contains blue bars, suggesting that high-loss instances are not necessarily unreliable. Obs. 1 implies that modelling per-instance loss is helpful but not sufficient for distinguishing unreliable instances.

Obs. 2. *High-loss also can be caused by reliable but uncertain instances with blurry sequential patterns. Such patterns are correlated with uncertain users or items.*

In SRSs, there are always uncertain users whose preferences shift frequently, or uncertain items that are complex and co-occur with various succeeding items. The instances whose inputs are associated with such uncertain users/items could have blurry sequential patterns, thus being hard to fit and leading to high training loss. Formally, the uncertainty of a user (or an item) can be measured by the ratio of unreliable instances over the total number of instances the user (or item) holds. A higher ratio suggests a higher uncertainty. Their existence can be reflected in Fig. 2(c), where the uncertainty of around 6% users and 4% items is higher than 0.3. We then plot the training loss of reliable instances on ML-1M, where yellow bars indicate instances that are related to these uncertain users (Fig. 2(d)) or items (Fig. 2(e)), while blue bars hold the opposite case. We note that yellow bars are generally distributed to the right of blue bars, indicating that uncertain users and items do increase the loss of reliable instances, confirming that uncertain users/items bring uncertainty into sequential patterns, thus leading to high training loss.

To summarize, the two observations suggest that high loss can be caused by either unreliable or uncertain instances, which inspire us to leverage both training loss and uncertainty to help distinguish unreliable instances as depicted by Fig. 1 , instead of relying on side information.

4 The Proposed BERD

In this section, we propose a novel neural framework – BERD – that boosts the recommendation performance of SRSs **B**y **E**liminating un**R**eliable **D**ata.

4.1 Framework Overview

Fig. 3 depicts the overall framework of our proposed BERD, mainly composed of four modules, including (1) Sequence Modelling with UGCN – it extracts input sequential patterns with reduced uncertainty on the base of uncertainty-aware

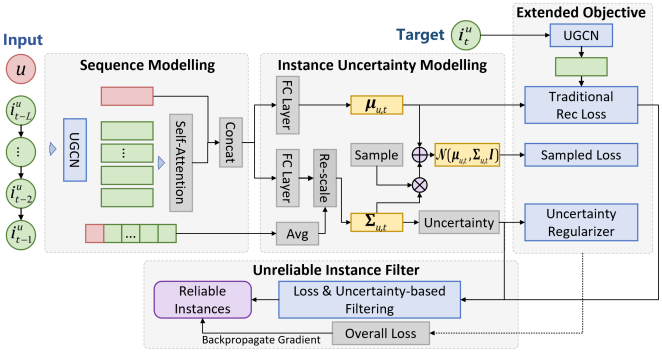


Figure 3: Illustration of our proposed BERD framework.

graph convolution network; (2) Instance Uncertainty Modelling – it models input sequential patterns as a Gaussian Distribution, with its mean representing the extracted dynamic user preference, and its covariance representing the uncertainty of the extracted preference (i.e. instance uncertainty); (3) Extended Objective Function – it helps endow proper uncertainty to every training instance by extending the conventional recommendation objective with a sampled loss and an uncertainty regularizer; and (4) Unreliable Instance Filter – it eliminates unreliable instances with high loss but low uncertainty. Among these modules, Instance Uncertainty Modelling and Extended Objective Function accurately model the training loss and uncertainty of each instance; Sequence Modelling with UGCN endeavors to mine unreliable instances from high loss & high uncertainty ones by reducing the instance uncertainty. As a result, the instances with high loss but low uncertainty can be cautiously removed as unreliable instances by the Unreliable Instance Filter.

4.2 Sequence Modelling with UGCN

To extract more accurate sequential patterns from $\{u, S^u_{t,L}\}$ (input), it is fundamental to get accurate user and item embeddings, especially for uncertain users and items which blurs input sequential patterns. To this end, we propose UGCN ensuring that users and items with high uncertainty will contribute less to their neighbors but be compensated more. Formally, we build a graph that includes users and items as nodes, while the edges are constituted by user-item interaction and item-item adjacency relationships. Each user u (or item i) is initially encoded as an embedding vector $\boldsymbol{e}^0_u \in \mathbb{R}^d$ (or $\boldsymbol{e}^0_i \in \mathbb{R}^d$)

and an uncertainty factor f_u (or f_i), where d denotes the embedding size. Take user (node) u as an example, UGCN propagates u's embedding by restricting that her neighboring nodes with higher uncertainty contribute less:

$$\boldsymbol{e}_u^k = \sum_{j \in \mathcal{M}_u} \frac{1-\sigma(f_j)}{\sqrt{|\mathcal{M}_u|}\sqrt{|\mathcal{M}_j|}} \boldsymbol{e}_j^{k-1}, \tag{3}$$

where $\boldsymbol{e}_u^k (k \geq 1)$ is the refined embedding of u after k layers' propagation; $\mathcal{M}_u$ is the set of items u has interacted with; $\mathcal{M}_j$ is the set of users who have interacted with item j and the adjacent items of j; $\sigma(\cdot)$ is the sigmoid function to rescale the uncertainty factors into $(0, 1)$; thus $1-\sigma(\cdot)$ ensures that high-uncertainty neighbors affect u less. After K layers' propagation, we aggregate the embeddings obtained at each layer to form the final representation of u:

$$\boldsymbol{e}_u = (1-\sigma(f_u))\boldsymbol{e}_u^0 + \sigma(f_u)\sum\nolimits_{k=1}^{K} \frac{1}{k+1}\boldsymbol{e}_u^k, \tag{4}$$

where $\boldsymbol{e}_u^0$ and $\boldsymbol{e}_u^k (k \geq 1)$ contain the information from u and her neighbors, respectively. Thus their weights $1-\sigma(f_u)$ and $\sigma(f_u)$ ensure that u's final representation $\boldsymbol{e}_u$ will be compensated more by u's neighbors if u has high uncertainty. Similarly, item $i \in S_{t,L}^u$ is also embedded into $\boldsymbol{e}_i$ according to Eqs. 3 and 4. As such, UGCN restricts the propagation of uncertain information while encourages the spreading of certain signals between users and items. This facilitates to produce accurate representations especially for uncertain users and items, allowing Sequence Modelling to extract sequential patterns with lower uncertainty. In particular, we stack the item embedding $\boldsymbol{e}_i, i \in S_{t,L}^u$ into matrix $\boldsymbol{E}_t^u \in \mathbb{R}^{L\times d}$ with sinusoidal positional encoding [Vaswani *et al.*, 2017], which is then fed into a multi-head self-attention network [Vaswani *et al.*, 2017] to extract u's short-term interest:

$$\begin{aligned} &\text{MH}(\boldsymbol{E}_t^u) = \text{Concat}(\text{head}_1, \text{head}_2, ..., \text{head}_h)\boldsymbol{W}^O, \\ &\text{head}_j = \text{softmax}(\frac{(\boldsymbol{E}_t^u\boldsymbol{W}_j^Q)(\boldsymbol{E}_t^u\boldsymbol{W}_j^K)^\top}{\sqrt{d/h}})\boldsymbol{E}_t^u\boldsymbol{W}_j^V, \end{aligned} \tag{5}$$

where h is the number of heads; $\boldsymbol{W}_j^Q, \boldsymbol{W}_j^K, \boldsymbol{W}_j^V \in \mathbb{R}^{d\times d/h}$ and $\boldsymbol{W}^O \in \mathbb{R}^{d\times d}$ are learnable projection matrices. The L-th row of $\text{MH}(\boldsymbol{E}_t^u) \in \mathbb{R}^{L\times d}$ encodes u's short-term interest, which is further concatenated with her general preference to denote the sequential pattern of user u at time step t:

$$\boldsymbol{s}_{u,t} = \text{Concat}(\boldsymbol{e}_u, \text{MH}(\boldsymbol{E}_t^u)_L). \tag{6}$$

4.3 Instance Uncertainty Modelling

The obtained sequential pattern $\boldsymbol{s}_{u,t}$ merely represents user u's preference at time step t. In our study, we aim to additionally model the uncertainty of the extracted preference, which is used to represent instance uncertainty and better distinguish unreliable instances. To this end, Instance Uncertainty Modelling aims to model u's preference at time step t as the mean $\boldsymbol{\mu}_{u,t}$ of a Gaussian Distribution, while its covariance $\boldsymbol{\Sigma}_{u,t}$ represents the uncertainty of the extracted preference (i.e. instance uncertainty). As modeling a full covariance matrix is expensive, we constrain it to be diagonal,

$$\begin{aligned} &\boldsymbol{\mu}_{u,t} = \boldsymbol{W}_1\boldsymbol{s}_{u,t} + \boldsymbol{b}_1, \\ &\boldsymbol{\Sigma}_{u,t} = \text{ReLU}(\boldsymbol{W}_2\boldsymbol{s}_{u,t} + \boldsymbol{b}_2)\sigma(\text{avg}(f_u, f_{i_{t-L}^u}, ..., f_{i_{t-1}^u})), \end{aligned} \tag{7}$$

where $\boldsymbol{W}_1, \boldsymbol{W}_2 \in \mathbb{R}^{2d\times d}$ and $\boldsymbol{b}_1, \boldsymbol{b}_2 \in \mathbb{R}^d$ are trainable parameters; $\text{ReLU}(\cdot)$ is the activation function [Nair and Hinton, 2010] to restrict the values of $\boldsymbol{\Sigma}_{u,t}$ to be nonnegative; $\text{avg}(\cdot)$ is the average function encoding user/item uncertainty into instance uncertainty. Next, we quantify the instance uncertainty $h_{u,t}$ (i.e. the uncertainty of u's preference at time step t) as the entropy of a multivariate Gaussian distribution,

$$h_{u,t} = \frac{1}{2}\sum\nolimits_{j=1}^{d} log(2\pi e(\boldsymbol{\Sigma}_{u,t})_j), \tag{8}$$

where j indices each dimension of $\boldsymbol{\Sigma}_{u,t}$.

4.4 Extended Objective Function

To accurately model recommendation loss and uncertainty of each instance, we expand the conventional recommendation objective with a sampled loss and an uncertainty regularizer. Specifically, the conventional objective encourages the model to rank target item i_t^u higher than the sampled negative item i' by maximizing the gap between their predicted scores:

$$\mathcal{L}_{rec} = \sum_u \sum_{i_t^u \in S^u} \sum_{i' \notin S^u} -log(\sigma(\boldsymbol{\mu}_{u,t}^\top \boldsymbol{e}_{i_t^u} - \boldsymbol{\mu}_{u,t}^\top \boldsymbol{e}_{i'})). \tag{9}$$

To seamlessly accommodate the learning of instance uncertainty, we introduce an additional sampled loss, which aims to push Z samples from $\mathcal{N}(\boldsymbol{\mu}_{u,t}, \boldsymbol{\Sigma}_{u,t}\boldsymbol{I})$ to be close to the target item i_t^u and far away from the negative item i':

$$\mathcal{L}_{sam} = \lambda \sum_u \sum_{i_t^u \in S^u} \sum_{i' \notin S^u} \frac{1}{Z}\sum_{z=1}^{Z} -log(\sigma(\boldsymbol{p}_{u,t}^{z\top} \boldsymbol{e}_{i_t^u} - \boldsymbol{p}_{u,t}^{z\top} \boldsymbol{e}_{i'})), \tag{10}$$

where λ balances the importance of the sampled loss; $\boldsymbol{p}_{u,t}^z$ denotes the embedding of the z-th sample, generated with a reparameterization trick [Kingma and Welling, 2014]:

$$\boldsymbol{p}_{u,t}^z = \boldsymbol{\mu}_{u,t} + \boldsymbol{\Sigma}_{u,t}^{\frac{1}{2}} \odot \epsilon, \quad \epsilon \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}), \tag{11}$$

where $\odot$ is the element-wise multiplication. However, simply training with $\mathcal{L}_{rec}$ and $\mathcal{L}_{sam}$ is not sufficient. Specifically, large covariance will enforce the generated samples far away from each other based on Equ. 11, thus leading to high $\mathcal{L}_{sam}$ according to Equ. 10. Hence, during optimization, the model will instinctively incentivize all covariance to be zero for a reduced $\mathcal{L}_{sam}$. To avoid such trivial solution, we constrain the model to maintain certain degree of instance uncertainty with an uncertainty regularizer:

$$\mathcal{L}_{reg} = \sum\nolimits_u \sum\nolimits_{i_t^u \in S^u} \max(0, \gamma - h_{u,t}), \tag{12}$$

where γ is used to control the degree of uncertainty to maintain, and we thus do not introduce additional regularization coefficients. Accordingly, the final objective function is,

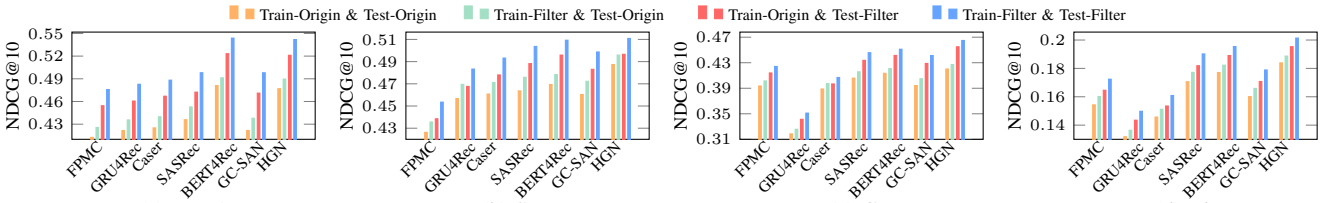
$$\arg\min_{\Theta} \mathcal{L} = \mathcal{L}_{rec} + \mathcal{L}_{sam} + \mathcal{L}_{reg}, \tag{13}$$

where Θ denotes the parameter set of BERD.

Minimizing $\mathcal{L}$ encourages BERD to properly endow high uncertainty to uncertain instances, and *vice versa*. The rationale behind is that – for certain & reliable instances, their mean vectors (e.g. $\boldsymbol{\mu}_{u,t}$) are generally close to target vectors (e.g. $\boldsymbol{e}_{i_t^u}$). Therefore, reducing their uncertainty can effectively help decrease $\mathcal{L}_{sam}$ as the generated sample (e.g.

Datasets	Metrics	BPRMF	FPMC	GRU4Rec	Caser	SASRec	BERT4Rec	GC-SAN	HGN	BERD	*Improve*
ML-1M	HR@10	0.5861	0.6319	0.6453	0.6503	0.6673	0.6958	0.6639	0.6935	**0.7437****	6.88%
	NDCG@10	0.3341	0.4126	0.4215	0.4251	0.4359	0.4811	0.4217	0.4769	**0.5125****	6.53%
	MRR	0.2747	0.3432	0.3549	0.3613	0.3641	0.4212	0.3621	0.4172	**0.4517****	7.24%
Steam	HR@10	0.6367	0.6764	0.7159	0.7185	0.7358	0.7468	0.7224	0.7523	**0.8022****	6.63%
	NDCG@10	0.3540	0.4263	0.4567	0.4608	0.4636	0.4694	0.4604	0.4874	**0.5458*****	11.98%
	MRR	0.2887	0.3634	0.3901	0.3935	0.3934	0.3962	0.3930	0.4176	**0.4757*****	13.91%
CD	HR@10	0.5382	0.6245	0.5493	0.6184	0.6464	0.6543	0.6377	0.6623	**0.6995****	5.62%
	NDCG@10	0.3213	0.3936	0.3185	0.3889	0.4059	0.4135	0.3943	0.4202	**0.4555****	8.40%
	MRR	0.2771	0.3473	0.2676	0.3340	0.3421	0.3507	0.3397	0.3581	**0.4071*****	13.68%
Elect	HR@10	0.2554	0.2990	0.2637	0.2893	0.3174	0.3290	0.3054	0.3346	**0.3601****	7.62%
	NDCG@10	0.1372	0.1543	0.1318	0.1457	0.1706	0.1771	0.1601	0.1839	**0.2023*****	10.01%
	MRR	0.1140	0.1358	0.1163	0.1282	0.1493	0.1544	0.1397	0.1609	**0.1763*****	9.57%

Table 1: Performance comparison of all models with original training and test sets, where the best performance is boldfaced and the runner up is underlined. *Improve* means the relative improvement of BERD over the runner up. Statistical significance of pairwise differences of BERD vs. the runner up is determined by a paired t-test (** for $p \leq 0.01$ and *** for $p \leq 0.001$).



(a) ML-1M (b) Steam (c) CD (d) Elect

Figure 4: Performance of SRSs on original and filtered datasets w.r.t. NDCG@10.

$p^z_{u,t}$) will be concentrated around the target. For certain & unreliable instances, their input is certain with salient sequential patterns as certain & reliable ones, thus their uncertainty can be collaboratively reduced. For uncertain instances, their mean vectors are normally far away from target vectors, thus lowering their uncertainty cannot ensure a reduced $\mathcal{L}_{sam}$. Recall that $\mathcal{L}_{reg}$ is introduced to maintain certain degree of uncertainty. Hence, the model has to allocate high uncertainty to uncertain instances, as it is a smarter way to minimize $\mathcal{L}$.

4.5 Unreliable Instance Filter

Till now, we have obtained the recommendation loss $\mathcal{L}_{rec}$ (Equ. 9) and uncertainty $h_{u,t}$ (Equ. 8) for each instance from the Instance Uncertainty Modelling and Extended Objective Function; meanwhile, the total number of uncertain instances is comprised by the Sequence Modelling module via UGCN. Next, we remove these unreliable instances with the Unreliable Instance Filter. Specifically, in each epoch, we rank all the instances according to their $\mathcal{L}_{rec}$ and $h_{u,t}$ in a descending order, respectively. Then, the instances ranked at top α of $\mathcal{L}_{rec}$ and bottom half of $h_{u,t}$ are considered as unreliable instances, and their gradient will not be backpropagated to the model parameters in this epoch.

5 Experiments and Analysis

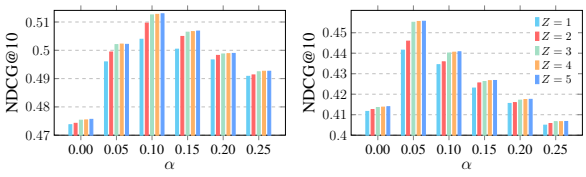
Datasets. We adopt four datasets varying w.r.t. domain, size, sparsity level, and ratio of unreliable data, as shown in Table 2. `ML-1M` [Harper and Konstan, 2015] is a popular dataset for movie recommendation. `Steam` [Kang and McAuley, 2018] is a game recommendation benchmark collected from *Steam*. `CD` and `Elect` are product review datasets crawled from *Amazon* [McAuley and Leskovec, 2013] for cd and electronics, respectively. Following [Huang *et al.*, 2018], we discard inactive users with fewer than 7 interactions.

Datasets	#User	#Item	#Instance	Density	U-Ratio	Attribute (Auxiliary Info)
ML-1M	6,040	3,416	925,542	5.253%	13.45%	actor,country,director,genre,tag
Steam	60,575	5,970	3,903,109	1.079%	7.34%	publisher, tag, genre, developer
CD	12,871	10,078	209,159	0.161%	5.79%	category
Elect	22,502	10,649	201,973	0.084%	10.68%	category

Table 2: Statistics of the datasets, where the column 'U-ratio' indicates the respective percentage of unreliable instances.

Baselines. `BPRMF` [Rendle *et al.*, 2009] is a traditional matrix factorization model with a pairwise ranking loss; `FPMC` [Rendle *et al.*, 2010] is a classic SRS that combines matrix factorization and Markov Chains; `GRU4Rec` [Hidasi *et al.*, 2016] and `Caser` [Tang and Wang, 2018] adopt RNN and CNN to extract users' dynamic preferences, respectively; `SASRec` [Kang and McAuley, 2018] and `GC-SAN` [Xu *et al.*, 2019] use attention to highlight salient input items, while GC-SAN additionally adopts GNN to obtain accurate item representations; `BERT4Rec` [Sun *et al.*, 2019] considers bidirectional context to model users' dynamic preferences; and `HGN` [Ma *et al.*, 2019] applies a hierarchical gating network to fuse long-term interest for dynamic preference modelling.

Evaluation Protocol. We follow the same evaluation protocol as in [Sun *et al.*, 2019]. To be more specific, for each user, we split the last two interactions (instances) into validation and test sets, respectively, while the rest are used for training. To improve the test efficiency, we pair each target item in the test set with 100 negative items that the user has



(a) ML-1M (b) CD

Figure 5: Effect of the unreliable filter ratio α and sample size Z.

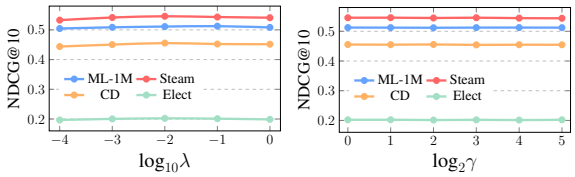


Figure 6: Effect of sampled loss weight λ and uncertainty margin γ.

not interacted with, and rank the target item among the 101 items. The negative items are sampled according to the distribution of item popularity. HR, NDCG and MRR are adopted to evaluate the ranking quality. Generally, higher metric values indicate better ranking performance.

Parameter Settings. We adopt Xavier [Glorot and Bengio, 2010] initializer and Adam [Kingma and Ba, 2015] optimizer with $d = 50$; the learning rate $\eta = 0.01$ with batch size of 8192; the weight of sampled loss $\lambda = 0.01$ and uncertainty margin $\gamma = 1$; the number of propagation layers $K = 2$; the input length $L = 5$; the sample size $Z = 4$; the filter ratio $\alpha = 0.05$ for Steam and CD, and $\alpha = 0.1$ for ML-1M and Elect; the head number of self-attention is set to 2.

Overall Comparison. Table 1 shows the performance of all methods, where several major findings are noted: **(1)** BPRMF performs the worst among all methods as it fails to model sequential behavior; **(2)** deep learning based SRSs (e.g., SASRec) generally achieves better performance than conventional SRSs (i.e. FPMC); **(3)** the fact that SASRec and GC-SAN are superior to GRU4Rec and Caser implies the usefulness of highlighting relevant input items with attention mechanism; **(4)** BERT4Rec and HGN are the best baselines, indicating that considering bidirectional context and long-term preference assists in reducing the negative impact of unreliable instances; and **(5)** BERD consistently achieves the best performance, demonstrating its superiority against all the counterparts. Moreover, BERD trained under Train-Origin view defeats all baselines under the Train-Filter view as depicted in Fig. 4, which further confirms that eliminating unreliable instances w.r.t. recommendation loss and uncertainty via our end-to-end BERD is more efficient than the heuristic reliability measurement.

Ablation Analysis. To examine the efficacy of each module of BERD, six variants are compared as shown in Table 3. First, (a) without uncertainty modelling is significantly surpassed by (f) – our final BERD, verifying the necessity of modelling both loss and uncertainty to eliminate unreliable data. Second, (b) abandons uncertainty regularizer and performs closely to (a), as it fails to allocate uncertainty properly. Third, (c) replaces UGCN with a classic GCN; (d-e) enhance

Variants of BERD	ML-1M		CD	
	HR@10	NDCG@10	HR@10	NDCG@10
(a) UGCN+$\mathcal{L}_{rec}$	0.7225	0.4913	0.6772	0.4238
(b) UGCN+$\mathcal{L}_{rec}$+$\mathcal{L}_{sam}$	0.7246	0.4942	0.6825	0.4267
(c) GCN+$\mathcal{L}_{rec}$+$\mathcal{L}_{sam}$+$\mathcal{L}_{reg}$	0.7312	0.5007	0.6849	0.4372
(d) GCN$_p$+$\mathcal{L}_{rec}$+$\mathcal{L}_{sam}$+$\mathcal{L}_{reg}$	0.7359	0.5053	0.6916	0.4489
(e) GCN$_f$+$\mathcal{L}_{rec}$+$\mathcal{L}_{sam}$+$\mathcal{L}_{reg}$	0.7395	0.5066	0.6927	0.4501
(f) UGCN+$\mathcal{L}_{rec}$+$\mathcal{L}_{sam}$+$\mathcal{L}_{reg}$	**0.7437**	**0.5125**	**0.6995**	**0.4555**

Table 3: Ablation analysis on BERD.

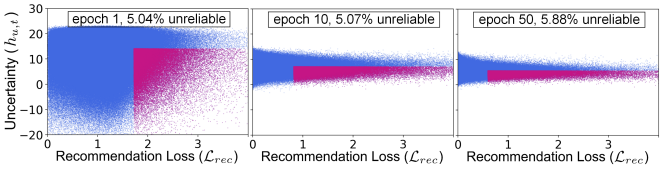


Figure 7: Visualization of instances at different epochs on ML-1M.

GCN with the propagation (Equ. 3) and fusion (Equ. 4) operation of UGCN, respectively. The performance comparison among (c-f) firmly validates the superior design of UGCN.

Hyper-parameters Analysis. The filter ratio α, sample size Z, sampled loss weight λ and uncertainty margin γ are four vital hyper-parameters in BERD. Their impacts are illustrated by Figs. 5 and 6, which indicate that the best choice for α is around $0.05 \sim 0.10$ depending on the datasets; $Z > 3$ is sufficient to help reach good performance; the optimal setting for λ ranges from 0.01 to 0.1; and BERD is less sensitive to γ in comparison with other hyper-parameters.

Visualization. Fig. 7 depicts the instance distribution w.r.t. $\mathcal{L}_{rec}$ (Equ. 9) and $h_{u,t}$ (Equ. 8) at different training epochs on ML-1M, where the red points denote unreliable instances. As the epoch increases from 1 to 50, the ratio of detected unreliable instances climbs up from 5.04% to 5.88% while the number of high-uncertainty instances gradually drops (the maximum of $h_{u,t}$ decreases from 20 to 10). It exhibits that BERD is capable of progressively mining unreliable instances from high-uncertainty ones.

6 Conclusion

This paper first verified that SRSs can be misguided by the unreliable instances (i.e. targets mismatch inputs). To ease this issue, we propose a novel neural model BERD guided by two insightful observations to eliminate unreliable data for performance-enhanced SRS. BERD modelled both loss and uncertainty via a Gaussian distribution, whereby instances with high loss but low uncertainty can be cautiously removed as unreliable instances; meanwhile the novel UGCN is designed to help mine unreliable instances from the high loss & high uncertainty ones by lowering instance uncertainty. Extensive experiments verified the superiority of BERD.

Acknowledgments

The work is partially supported by the National Key Research and Development Program of China (2020YFB1707900), National Natural Science Foundation of China (62072088), Ten Thousand Talent Program (ZX20200035), and Liaoning Distinguished Professor (XLYC1902057).

References

[Burke *et al.*, 2006] Robin Burke, Bamshad Mobasher, Chad Williams, and Runa Bhaumik. Classification features for attack detection in collaborative recommender systems. In *KDD*, pages 542–547, 2006.

[Chen *et al.*, 2018] Xu Chen, Hongteng Xu, Yongfeng Zhang, Jiaxi Tang, Yixin Cao, Zheng Qin, and Hongyuan Zha. Sequential recommendation with user memory networks. In *WSDM*, pages 108–116, 2018.

[Cho *et al.*, 2014] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. In *EMNLP*, pages 1724–1734, 2014.

[Glorot and Bengio, 2010] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *AISTATS*, pages 249–256, 2010.

[Harper and Konstan, 2015] F Maxwell Harper and Joseph A Konstan. The movielens datasets: History and context. *ACM transactions on interactive intelligent systems (TIIS)*, 5(4):1–19, 2015.

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

[Huang *et al.*, 2018] Jin Huang, Wayne Xin Zhao, Hongjian Dou, Ji-Rong Wen, and Edward Y Chang. Improving sequential recommendation with knowledge-enhanced memory networks. In *SIGIR*, pages 505–514. ACM, 2018.

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

[Kingma and Ba, 2015] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015.

[Kingma and Welling, 2014] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. In *ICLR*, 2014.

[Li *et al.*, 2017] Jing Li, Pengjie Ren, Zhumin Chen, Zhaochun Ren, Tao Lian, and Jun Ma. Neural attentive session-based recommendation. In *CIKM*, pages 747–755, 2017.

[Li *et al.*, 2019] Dongsheng Li, Chao Chen, Zhilin Gong, Tun Lu, and Ning Gu. Collaborative filtering with noisy ratings. In *SDM*, pages 747–755, 2019.

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

[Ma *et al.*, 2020] Chen Ma, Liheng Ma, Yingxue Zhang, Jianing Sun, Xue Liu, and Mark Coates. Memory augmented graph neural networks for sequential recommendation. In *AAAI*, pages 5045–5052, 2020.

[McAuley and Leskovec, 2013] Julian McAuley and Jure Leskovec. Hidden factors and hidden topics: understanding rating dimensions with review text. In *RecSys*, pages 165–172, 2013.

[Nair and Hinton, 2010] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *ICML*, pages 807–814, 2010.

[Pei *et al.*, 2017] Wenjie Pei, Jie Yang, Zhu Sun, Jie Zhang, Alessandro Bozzon, and David M. J Tax. Interacting attention-gated recurrent networks for recommendation. In *CIKM*, pages 1459–1468, 2017.

[Rendle *et al.*, 2009] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. Bpr: Bayesian personalized ranking from implicit feedback. In *UAI*, 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*, pages 811–820, 2010.

[Sun *et al.*, 2019] Fei Sun, Jun Liu, Jian Wu, Changhua Pei, Xiao Lin, Wenwu Ou, and Peng Jiang. Bert4rec: Sequential recommendation with bidirectional encoder representations from transformer. In *CIKM*, pages 1441–1450, 2019.

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

[Tang *et al.*, 2019] Jiaxi Tang, Francois Belletti, Sagar Jain, Minmin Chen, Alex Beutel, Can Xu, and Ed H. Chi. Towards neural mixture recommender for long range dependent user sequences. In *WWW*, pages 1782–1793, 2019.

[Vaswani *et al.*, 2017] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NIPS*, pages 5998–6008, 2017.

[Wang *et al.*, 2019] Shoujin Wang, Liang Hu, Yan Wang, Quan Z. Sheng, and Longbing Cao. Modeling multi-purpose sessions for next-item recommendations via mixture-channel purpose routing networks. In *IJCAI*, pages 3771–3777, 2019.

[Wu *et al.*, 2020] S. Wu, Y. Zhang, C. Gao, K. Bian, and B. Cui. Garg: Anonymous recommendation of point-of-interest in mobile networks by graph convolution network. *Data Science and Engineering (DSE)*, 5(2), 2020.

[Xu *et al.*, 2019] Chengfeng Xu, Pengpeng Zhao, Yanchi Liu, Victor S Sheng, Jiajie Xu, Fuzhen Zhuang, Junhua Fang, and Xiaofang Zhou. Graph contextualized self-attention network for session-based recommendation. In *IJCAI*, pages 3940–3946, 2019.

[Yuan *et al.*, 2020] Fajie Yuan, Xiangnan He, Haochuan Jiang, Guibing Guo, Jian Xiong, Zhezhao Xu, and Yilin Xiong. Future data helps training: Modeling future contexts for session-based recommendation. In *WWW*, pages 303–313, 2020.

[Zhang *et al.*, 2019] Jing Zhang, Bowen Hao, Bo Chen, Cuiping Li, Hong Chen, and Jimeng Sun. Hierarchical reinforcement learning for course recommendation in moocs. In *AAAI*, pages 435–442, 2019.