Determinantal Point Process Likelihoods for Sequential Recommendation

Yuli Liu
Australian National University & Data61, CSIRO
Canberra, Australia
yuli.liu@anu.edu.au

Christian Walder
Data61, CSIRO &
Australian National University
Canberra, Australia
christian.walder@data61.csiro.au

Lexing Xie
Australian National University & Data61, CSIRO
Canberra, Australia
lexing.xie@anu.edu.au

ABSTRACT

Sequential recommendation is a popular task in academic research and close to real-world application scenarios, where the goal is to predict the next action(s) of the user based on his/her previous sequence of actions. In the training process of recommender systems, the loss function plays an essential role in guiding the optimization of recommendation models to generate accurate suggestions for users. However, most existing sequential recommendation techniques focus on designing algorithms or neural network architectures, and few efforts have been made to tailor loss functions that fit naturally into the practical application scenario of sequential recommender systems.

Ranking-based losses, such as cross-entropy and Bayesian Personalized Ranking (BPR) are widely used in the sequential recommendation area. We argue that such objective functions suffer from two inherent drawbacks: *i)* the dependencies among elements of a sequence are overlooked in these loss formulations; *ii)* instead of balancing accuracy (quality) and diversity, only generating accurate results has been over emphasized. We therefore propose two new loss functions based on the Determinantal Point Process (DPP) likelihood, that can be adaptively applied to estimate the subsequent item or items. The DPP-distributed item set captures natural dependencies among temporal actions, and a quality *vs.* diversity decomposition of the DPP kernel pushes us to go beyond accuracy-oriented loss functions. Experimental results using the proposed loss functions on three real-world datasets show marked improvements over state-of-the-art sequential recommendation methods in both quality and diversity metrics.

CCS CONCEPTS

• **Information systems** → **Recommender systems**.

KEYWORDS

Sequential Recommendation, Determinantal Point Process, Loss Function, Diversity

SIGIR '22, July 11–15, 2022, Madrid, Spain

ACM ISBN 978-1-4503-8732-3/22/07…$15.00
https://doi.org/10.1145/3477495.3531965

ACM Reference Format:
Yuli Liu, Christian Walder, and Lexing Xie. 2022. Determinantal Point Process Likelihoods for Sequential Recommendation. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '22), July 11–15, 2022, Madrid, Spain.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3477495.3531965

1 INTRODUCTION

Traditional recommender systems, *e.g.*, *Top-N* recommendation [8, 21], strive to model users' preferences towards items based on historical interactions between users and items. The modeling process only depends on static interactions and ignores sequential dependencies with the assumption that all user-item interactions are equally important. However, this might not hold in real-world scenarios [9], where the next action(s) of a user are generated according to his/her previous behavioral sequence. To model users' evolving interests and adjust recommendation methods to real applications, the customized time-relevant recommender system (*i.e.*, sequential recommendation) is proposed and has become a popular topic in academic research and online applications.

Sequential dependencies among items are relevant for next item(s) prediction. For example, if it is known that a user has browsed "*shirt*, *hat* and *watch*" (as shown in Figure 1), the conventional recommender typically provides content-similar or category-similar items [28], such as more watches or clothes. As for sequential recommendation, considering dependencies and the temporal order of a previous behavioral sequence, the recommendation model tends to capture the temporal dependency of previous actions on target items (*i.e.*, *sequence dependence*), and thereby determines that the user is actually trying to find clothing accessories (*e.g.*, *headphones* and *glasses*) in this period of time.

The core consideration of recommending the item or items is the user preference-based ranking of items. Similar to most learning-to-rank tasks (*e.g.*, Web search and question answering), commonly used ranking-based loss functions such as pointwise ranking (cross-entropy) and pairwise ranking (BPR) are widely deployed for the training of recommendation models. In this paper, we argue the insufficiency of applying ranking-based loss functions in sequential recommendation from three aspects.

Firstly, we regard estimating the next T items (*i.e.*, target items in Figure 1) as a task of predicting the temporally subsequent set, according to the historical sequence. In this scenario, the dependency among targets is noteworthy. However, the ranking-based objective function independently compares each target to its true label (pointwise ranking), to a negative item (pairwise ranking), or

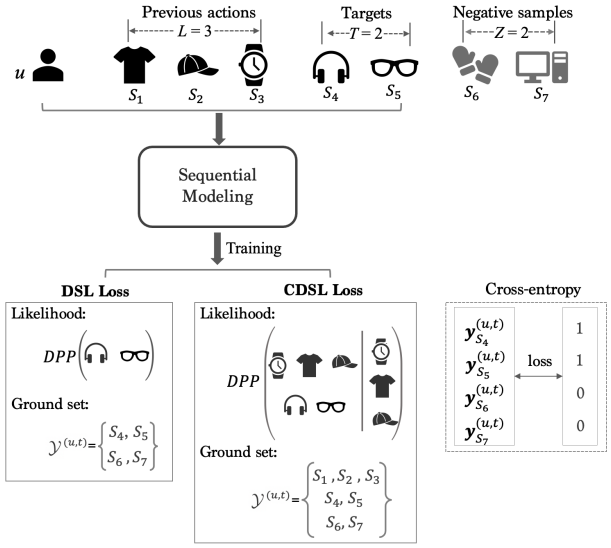


Figure 1: A simplified diagram comparing the loss functions based on DPP-distributed set likelihoods with a basic cross entropy loss function.

to a perfect ordering list (listwise ranking). This manipulation neglects the dependency among targets (*targets dependency*), thereby affecting the set prediction capacity of sequential recommendations. For example, based on the previous sequence in Figure 1, the loss calculated assuming independent targets (*headphones* or *glasses*) merely makes sequential models capture the dependencies in two sequences {*shirt*, *hat*, *watch*, *headphones*} or { *shirt*, *hat*, *watch*, *glasses*}. This means that part of the *sequence dependency* is neglected, due to the omission of *targets dependency*. On the other hand, the inherent goal of next-item recommender systems is to select or generate personalized item sets for specific users. However, ranking-based loss functions enforce the rank or estimated score of targets, leading sequential recommendations to provide high-scored items (*i.e.*, *top-N* prediction), without considering the feasibility of selecting whole targets as a generated set. In this work, instead of stressing the ranking of target items in loss functions, we propose to focus on enhancing the likelihood of target sets, *i.e.*, endowing the target set with a high probability of being selected/recommended.

Based on the Determinantal Point Process (DPP) [23, 24], we propose the DPP Set Likelihood-based loss function (DSL). We regard the temporal set prediction task as drawing a subset from a user-sequence specified ground set (denoted as $\mathcal{Y}^{(u,t)}$ in Figure 1) according to the DPP probability measure. $\mathcal{Y}^{(u,t)}$ is the combination of target items and negative samples for each user's sequence at time step t. It means that the drawn subset represents the probability of inclusion for user-preferred targets and exclusion for non-relevant items. In this way, the set likelihood-based loss function considers the correlations among observed targets, and enforces that the chance of suggesting the target set should be higher than that of selecting the negative one.

Secondly, existing sequential recommendations that try to capture the *sequence dependency* (on the entire sequence) *e.g.*, {*shirt*, *hat*, *watch*, *headphones*, *glasses*} in Figure 1, mainly focus on modeling the dependency in neural layers [22, 39] or Markov chains [15, 33]. However, in a more challenging part, *i.e.*, the training loss, the dependencies among temporal actions are neglected. DSL considers *targets dependency* modeling in loss calculation, but the dependency between previous actions and targets is not directly modeled. Specifically, DSL is proposed for the scenario of predicting the next items (*i.e.*, $T > 1$ in Figure 1).

To model the entire dependency of a sequence and generalize DPP likelihood-based loss to next item(s) estimation (*i.e.*, predicting next one or more targets in training), a new Conditional DPP Set Likelihood (CDSL) objective is designed. We condition a DPP on the event that elements (items) in the previous set are observed. Specifically, given the previous actions, the target item(s) subset is selected (distributed as a DPP) and negative samples are excluded. In this case, the user-sequence specified ground set contains items of the entire sequence ($L + T$) and corresponding negative samples Z. Conditioning a DPP on observed previous items helps recommendation models capture more correlations is more powerful for subset drawing. Experiments on different datasets demonstrated these intuitions.

Thirdly, ranking-based losses mainly aim at improving the relevance score of target items, whereas they account little for the diversity that is an important factor for avoiding users being bored with the narrow scope of topics of recommendations. This causes recommender systems to unduly lean on accuracy recommendations [46] while ignoring the diversity.

The Determinantal Point Process (DPP), as an elegant probabilistic model, is endowed with the property of modeling the repulsive correlations (dissimilarity) of elements in a subset. This property makes DPP a natural choice for diversity-promoting recommender systems, as it facilitates providing items that are diversified. Unlike previous studies that apply DPP for diversification in the process of building model architectures [11, 43, 44] or generating diverse and relevant items by maximum a posteriori (MAP) inference [4, 46], we embed the recommendation diversity in the set likelihood-based loss functions by employing the quality (accuracy or relevance) *vs.* diversity decomposition of DPP kernel.

We summarize the contributions of this work as follows:

- A new perspective (*i.e.*, set selection/prediction instead of ranking items) on loss functions for sequential recommendations is contributed.
- Two objective functions, DSL and CDSL bring *targets dependency* and *sequence dependency* into training loss and fit naturally into the next items and item(s) prediction.
- The diverse kernel $\mathbf{K}$ learned from the observed item sets and personalized relevance scores predicted by the sequential model play roles of diversity and quality in the diversity-aware loss functions, respectively.

2 PROPOSED LOSS FUNCTIONS

The simplified illustration of DPP set likelihood-based loss functions (DSL and CDSL) is shown in Figure 1. The most apparent difference between the commonly used pointwise ranking loss (binary cross entropy) and the proposed loss functions (DSL and CDSL) is that the cross entropy is a type of accuracy comparison loss [34], *i.e.*, measuring the probability error between estimated scores and true labels, whereas DSL and CDSL calculate the loss measure by DPP

set distribution comparison, *i.e.*, the comparison between inclusion probability for targets and exclusion probability for negative samples. In addition, the proposed DSL and CDSL directly apply the output results of sequential models to calculate the proposed losses and no specific changes to the model framework are required. This indicates that DSL and CDSL can be flexibly employed in different sequential models.

2.1 Preliminaries

Before we present the proposed loss functions, we briefly review the preliminaries of ranking-based losses and DPP.

2.1.1 Ranking-based losses. Like other learning to rank tasks, the core of recommender systems is the relevance-based ranking of items[18]. Ranking can be pointwise, pairwise, or listwise.

There are different types of pointwise-based losses, such as cross-entropy, squared loss, and MRR optimization [37]. In this section, we only present the binary cross-entropy loss, as different pointwise-based losses share the similar concept and cross-entropy is the most common and representative one [16].

The binary cross-entropy loss for next items (T targets) prediction is defined as

$$\ell = \sum_{u} \sum_{t \in C^u} \sum_{i \in \mathcal{D}_t^u} -\log\left(\sigma\left(\boldsymbol{r}_i^{(u,t)}\right)\right) + \sum_{j \notin \mathcal{D}_t^u} -\log\left(1-\sigma\left(\boldsymbol{r}_j^{(u,t)}\right)\right), \tag{1}$$

where $\boldsymbol{r}_i^{(u,t)}$ is the estimated preference score of user u on target item i in the sequence of time step t, and σ usually represents the *sigmoid* activation function. C^u is a collection of specific time steps of user u. To make it applicable to next items prediction, $\mathcal{D}_t^u$ is used to denote the next T target items. The cross-entropy format can be derived from the likelihood function

$$p(\mathcal{S} \mid \Theta) = \prod_{u} \prod_{t \in C^u} \prod_{i \in \mathcal{D}_t^u} \sigma\left(\boldsymbol{r}_i^{(u,t)}\right) \prod_{j \notin \mathcal{D}_t^u} \left(1-\sigma\left(\boldsymbol{r}_j^{(u,t)}\right)\right) \tag{2}$$

by taking the negative logarithm. From this equation we see that the cross-entropy loss independently estimates the ranking or score, ignoring dependency among target items.

Pairwise ranking (BPR) [32] is also widely used in learning-to-rank tasks, formulated as:

$$\ell = \sum_{u} \sum_{t \in C^u} \sum_{i \in \mathcal{D}_t^u} -\ln \sigma\left(\boldsymbol{r}_{(u,i,j)}^t\right), \tag{3}$$

where $\boldsymbol{r}_{(u,i,j)}^t = \boldsymbol{r}_{(u,i)}^t - \boldsymbol{r}_{(u,j)}^t$. Item j is an item that user u shows no interest. This formulation is derived from the maximum a posteriori (MAP) estimator of the model parameters (Θ):

$$\underset{\Theta}{\operatorname{argmax}} \prod_{u} \prod_{t \in C^u} \prod_{i \in \mathcal{D}_t^u} p\left(i >_{u,t} j \mid \Theta\right) p(\Theta), \tag{4}$$

based on the Bayesian formulation and the independence assumption of targets and users. This means that pairwise ranking loss also takes no account of targets dependency and sequence dependency. The above formulations are defined in the next T targets scenario, which can be easily transformed for the next item task.

Instead of treating either each rating (cross-entropy) or each pairwise comparison (BPR) as an independent instance, listwise-based ranking [2] directly assigns a loss to an entire item list (or a *Top-N* list) by employing the cross-entropy to measure the distance between a predicted list and perfect ordering. The listwise learns a ranking dependent on the position of items in the relevance score-ordered lists. However, as interactions in recommender systems are usually of binary relevance (with no perfect ordering list) and the permutation probabilities are computationally expensive, listwise ranking is not used often [18] in sequential recommendations. Therefore, listwise ranking is not compared with here.

2.1.2 Determinantal Point Process. DPP is known and widely used within the machine learning community, *e.g.*, diversity-promoting recommendations [46] and neural conversation [36]. It provides tractable and efficient means to balance quality and diversity within a subset. Formally, a determinantal point process $\mathcal{P}$ on a discrete set $\mathcal{Y} = \{1, 2, ..., M\}$ is a probability measure on $2^{\mathcal{Y}}$, the set of all subsets of $\mathcal{Y}$). When $\mathcal{P}$ gives nonzero probability to the empty set, there exists a matrix $\mathbf{L} \in \mathbb{R}^{M \times M}$, such that for every subset $Y \subseteq \mathcal{Y}$, the probability is $\mathcal{P}(Y) \propto \det(\mathbf{L}_Y)$, where $\mathbf{L}$ is a real, positive semi-definite kernel matrix indexed by the elements of $\mathcal{Y}$. In the context of sequential recommendation, $\mathcal{Y}$ is the entire item (movies or products) catalog, and Y is the subset of items that users interact with. The notation $\mathbf{L}_Y$ denotes the submatrix of kernel $\mathbf{L}$ restricted to only those rows and columns which are indexed by Y.

The marginal probability of including one element S_i is $\mathcal{P}(S_i) = \mathbf{L}_{ii}$. That is, the diagonal of $\mathbf{L}$ gives the marginal probabilities of inclusion for individual elements of $\mathcal{Y}$. The probability of selecting two items S_i and S_j is $\mathbf{L}_{ii}\mathbf{L}_{jj} - \mathbf{L}_{ij}^2 = \mathcal{P}(S_i)\mathcal{P}(S_j) - \mathbf{L}_{ij}^2$. The entries of the kernel $\mathbf{L}$ usually measures similarity between pairs of elements in $\mathcal{Y}$. Thus, highly similar elements are unlikely to appear together. In this way, repulsive correlations (*i.e.*, diversity) with respect to a similarity measure are captured. However, an important factor — personalized relevance to items in recommendations are not reflected in the similarity-based kernel matrix. To adapt DPP to user preferences- and diversity-aware sequential recommendation tasks, we bring a quality *vs.* diversity decomposition of the DPP kernel by independently modeling diversity and quality.

2.2 Quality *vs.* Diversity Kernel

Similarity measures reflect the primary qualitative characterization of the DPP as diversifying processes. However, in practical recommendations, diversity needs to be balanced against underlying user preferences (quality) for different items. The decomposition of the DPP kernel $\mathbf{L}$ offers an intuitive trade-off between quality of items in $\mathcal{Y}$ and a global model of diversity, whose entries can be written $\mathbf{L}_{ij} = q_i \phi_i^\top \phi_j q_j$. $q_i \in \mathbb{R}^+$ is a quality term of item i and $\phi_i \in \mathbb{R}^D$ is a D-dimensional vector of normalized diversity features (in practice, it is usually represented by feature vectors) [24]. In sequential recommendations, user preferences on items of a sequence can be naturally treated as the quality measure. On the other hand, we need a diversity kernel to capture the diversity of items, which can be learned from the observed diverse item sets [12]. To reduce the computational complexity of calculating a $|\mathcal{Y}| \times |\mathcal{Y}|$ matrix, the diversity kernel is represented using the low-rank factorization as $\mathbf{K} = \mathbf{V}^\top \mathbf{V}$, where $\mathbf{V}$ is a $M \times D$ matrix and $D << |\mathcal{Y}|$. From now on we use $\mathbf{K}$ to denote the *diversity kernel*, and $\mathbf{L}$ is the final kernel that is balanced by quality and diversity.

We first learn *diversity kernel* $\mathbf{K}$ from observed interactions (instances in training datasets) before training sequential models. To

capture the co-occurrence of diverse items, we use the following diverse sets generation process to diversify item sets: *i)* before sequentially selecting items for a diverse set, we assign each item in a user's interaction list with the same probability of being selected; *iii)* every time an item i is selected. Its category c_i will be stored. The probability of all items in c_i will decay ($decay = 0.5$). Because items are randomly selected according to their corresponding probabilities, the decay setting facilitates generating item sets with different categories (*i.e.*, diversity). Five distinct items are selected for each item subset; *iii)* repeat the last step until the observable positive items of the user have all been selected at least once.

For each positive item set, a corresponding negative set is sampled, which contains negative items that share the same categories with items of the positive set, but have no interaction with the user.

Based on these ground-truth diverse sets, the diversity kernel $\mathbf{K}$ can be learned following [43], with the log-likelihood formulation:

$$\ell = \sum_{\left(T^{(+)}, T^{(-)}\right) \in \mathcal{T}} \log\det\left(\mathbf{K}_{T^{(+)}}\right) - \log\det\left(\mathbf{K}_{T^{(-)}}\right), \tag{5}$$

where $T^{(+)}$ is an observed diverse set and $T^{(-)}$ represents the corresponding set that contains negative items and $\mathcal{T}$ denotes the set of paired sets used for training.

In sequential recommendation, the training loss is calculated by accumulating the difference comparison between targets and predictions from each sequence of users in a batch. To compare the likelihood between the true set and negative set in each sequence, we define the following user-sequence specified kernel:

$$\mathbf{L}^{(u,t)} = \mathrm{Diag}\left(\mathbf{R}^{(u,t)}\right) \cdot \mathbf{K}^{(u,t)} \cdot \mathrm{Diag}\left(\mathbf{R}^{(u,t)}\right), \tag{6}$$

where $\mathbf{R}^{(u,t)}$ represents user u's preferences on specific items (*sequence ground set, i.e.*, $\mathcal{Y}^{(u,t)}$ in Figure 1) depending on the observed sequence of time step t, and $\mathbf{K}^{(u,t)}$ is the submatrix of diversity kernel indexed by the *sequence ground set*. Similarly, $\mathbf{L}^{(u,t)}$ is used to denote the submatrix of $\mathbf{L}$ DPP kernel. We name it as *sequence kernel*, as it is confined to a user sequence. As a shorthand, we will remove the superscript (u, t) that represents a specific ground set associated to a user sequence when the meaning is clear.

2.3 DPP Set Likelihood-Based Loss

We first introduce the intuitive idea of using DPP set likelihood comparison as a loss function for sequential recommendations. Given a sequence, the targets and negative items can be regarded as the user-sequence specified ground set, and we can treat the measurement of loss as the negative log probability of selecting targets as a DPP. This means that the target set is supposed to receive a higher probability of being drawn by $\mathcal{P}$ (mentioned in Section 2.1.2) than that of negative items. The normalization constant for $\mathcal{P}$ follows from the observation that $\sum_{Y' \subseteq \mathcal{Y}} \det(\mathbf{L}_{Y'}) = \det(\mathbf{L} + \mathbf{I})$, where $\mathbf{I}$ is the identity matrix [12]. The probability of $\det(\mathbf{L}_Y)$ is normalized by all possible item sets $Y' \subseteq \mathcal{Y}$. Therefore, we have the following probability of inclusion for target set Y_T in a user sequence:

$$\mathcal{P}(Y_T) = \frac{\det\left(\mathbf{L}_{Y_T}\right)}{\det(\mathbf{L} + \mathbf{I})}, \tag{7}$$

The ground set of this sequence is $\mathcal{Y}^{(u,t)}$ as shown in the DSL part of Figure 1. Y_T denotes target items set in a sequence. More precisely, $\mathbf{I}$ is identity matrix of the *sequence ground set*, *i.e.*, with ones in the diagonal positions. It represents a $|\mathcal{Y}^{(u,t)}| \times |\mathcal{Y}^{(u,t)}|$ matrix. In DSL, *sequence ground set* contains targets and negative samples in a user's sequence instance.

Table 1: Statistics of the datasets.

Dataset	#Users	#Items	#Interactions	#Categories
ML-1M	6.0k	3.4k	1.0M	18
Anime	73.5k	12.2k	1.0M	43
Beauty	52.0k	57.2k	0.4M	213

Based on the introduction above, we can see that DSL is expected to capture the dependency among targets. In this work, DSL is only applied to the next items (target items $T > 1$) estimation. This is to the fact that if only one item is in the target set, the significance of considering *targets dependency* is lost. Besides, DSL ignores the *sequence dependency* (correlations between previous items and targets). To make the DPP-based loss function more viable, we further propose CDSL.

2.4 Conditional DPP Set Likelihood-Based Loss

To capture entire *sequence dependency* in loss functions, we try to formulate the motivation of drawing a DPP set (T targets) conditioning on the observed previous set (L items). Given the previous items (Y_L) in a user-specified sequence, we want to obtain the probability of drawing an entire sequence ($Y_{L\cup T}$), *i.e.*, L previous items and T targets, which is formulated as:

$$\mathcal{P}(Y_{L\cup T} \mid Y_L) = \frac{\det\left(\mathbf{L}_{Y_{L\cup T}}\right)}{\det\left(\mathbf{L} + \mathbf{I}_{\overline{Y}_L}\right)}, \tag{8}$$

where $\mathbf{L}$ is actually the *sequence kernel* on $\mathcal{Y}^{(u,t)}$ that is composed of previous items, targets and negative samples (as shown in the CDSL part of Figure 1). $\mathbf{I}_{\overline{Y}_L}$ is the matrix with ones in the diagonal entries indexed by elements of $\mathcal{Y}^{(u,t)} - \overline{Y}_L$ and zeros everywhere else. This conditional distribution is derived according to [1, 24]. If you are interested, you can learn more from the literature. From this equation, we can see that even if T equals 1 (next item scenario), CDSL can still be applied to utilize the dependency by calculating DPP set ($Y_{L\cup T}$) probability. This means that CDSL is more adaptable.

Based on Equation (7) and Equation (8), DSL and CDSL loss functions for next T targets and next target(s) estimations can be obtained by taking the negative log-likelihood.

3 EXPERIMENTS

We comprehensively evaluate the proposed DSL and CDSL loss functions in terms of *i)* recommendation quality, *ii)* diversification, and *iii)* training efficiency.[1]

3.1 Datasets

We evaluate our methods on three datasets from three real world applications. **MovieLens**[2] is a widely used benchmark movie rating dataset. We use the version (*ML-1M*) that includes one million user ratings to 18 categories of movies. **Amazon** contains a series of

[1] Our source code can be found at https://github.com/l-lyl/DPPLikelihoods4SeqRec.
[2] https://grouplens.org/datasets/movielens/1m/

datasets introduced in [30], comprised of a large corpora of product reviews crawled from *Amazon.com*. We consider *Beauty* category in this work, which is notable for its high sparsity and variability. There are 213 categories of *Beauty* products. **Anime**[3] consists of user ratings to anime in *myanimelist.net*. All items in *Anime* are grouped into 43 categories. The statistics of the datasets are shown in Table 1. The reasons why we select these three datasets are: *i)* they are common public datasets in the sequential recommendation field. In particular, *Beauty* and *ML-1M* are also chosen as experimental datasets in two state-of-the-art baselines [22, 39] analyzed in this work. Selecting them helps to keep the comparison as fair as possible; *ii)* the diversity (category coverage of recommended items) is also considered in the proposed loss functions. To evaluate the diversity metric, we need to choose datasets that contain the category attribute of items. As the category attribute is not considered in the selected baselines, we process the datasets by closely following their settings, instead of directly requiring the processed datasets from the authors. For all datasets, we use timestamps to determine the sequence order of actions. We use the 10-core setting for experiments to ensure the quality of the dataset.

Referring to previous work [14, 22, 33], the most recent T actions (last T items) in each user's sequence are used for testing. The remaining actions (except the T items) are split into two parts: *i)* first 90% of them as the training set and *ii)* the next 10% of actions as the validation set. To better imitate the real-world sequential recommendation scenario and evaluate DSL and CDSL more comprehensively, the test set size of each dataset is not fixed but changes with the number of targets (*i.e.*, T in Figure 1) in training sequences. For instance, when considering the single next target ($T = 1$) in training, we only leave the last item of each user's interactions for testing, and the remaining actions are used for training and validating. Similarly, when $T = 3$, the test set contains each user's last three items.

3.2 Baselines

As mentioned in Section 2, the proposed loss functions can be flexibly deployed in various sequential recommendation methods without modifying their model architecture. Under the same experimental settings, if the performance of reworked models (replacing a baseline's original loss function with DSL or CDSL) is improved, the superiority of our loss functions is demonstrated. In this paper, three baselines are taken into consideration:

Caser [39]: A CNN-based method that applies convolutional operations on the embedding matrix of some most recent items, achieves the *state-of-the-art* sequential recommendation performance. The *binary cross-entropy loss* is used for one or more targets.

MARank [51]: A multi-order attentive neural model to extend user's general preference by fusing individual- and union-level temporal item-item correlations is designed for next-item recommendation. BPR loss for one target is adopted.

SASRec [22]: This is a *state-of-the-art* self-attention based next-item recommendation model, which adaptively assigns weights to previous items at each time step. The *binary cross-entropy loss* is calculated for each fixed-length sequence. If the length of a user's sequence is less than the maximum length n, *'padding'* items will be added. This means that the targets and previous items in a fixed-length sequence are both considered when calculating loss.

The above description provides three reasons for the selection of these baselines: *i)* they are recently proposed state-of-the-art sequential recommendation methods; *ii)* they are representative works, and many new models are built based on them [26, 27, 38, 49]; and *iii)* different settings of loss function are used. As our loss functions are designed for the sequential recommendation scenario with the intuition of capturing sequential dependencies, general recommendation models are not selected as baselines. In addition, it has been demonstrated that non-sequential models that neglect the sequence order of actions are inferior to sequential models in dealing with sequential recommendation tasks [39, 51].

3.3 Performance Comparison

To explore the influences of *sequence dependency* and *targets dependency*, as well as to analyze performances of the proposed loss functions in coping with different length of targets (next item and next items), we set different numbers of last items (*i.e.*, T) of the training sequence instances as targets that receive estimated relevance scores for loss calculation. The target length T is in $\{1, 2, 3, 5\}$. As mentioned before, we divide datasets according to the target length. For each real-world dataset (*ML-1M*, *Anime*, or *Beauty*), the last 1, 2, 3, and 5 items of each user's temporal interactions are left for testing, and remaining actions for training and validation. Therefore we obtain four groups (1, 2, 3, and 5 unobserved items) of divided data (test, training, and validation) from each real-world dataset. Correspondingly, four groups of diverse item sets are observed from different training data of each dataset, which are separately used to learn the corresponding diverse kernel matrix $\mathbf{K}$. In this setting, we imitate the scenarios of next item and next T items recommendation, as well as test the ability of DSL and CDSL to cope with different sizes of missing actions.

Three groups of evaluation metrics are adopted for the comprehensive analysis of the recommendation performance:

Quality. We use two common *Top-N* metrics, Recall@N (Re) and NDCG@N (Nd) to evaluate the quality (relevance) of top-scored results, N $\in \{3, 5, 10\}$.

Diversity. Diversification is also considered in the proposed loss functions. Therefore, we evaluate the recommendation diversity by the widely used metric — category coverage (CC), which is calculated by the number of categories covered by *top-N* items divided by the total number of categories available in the dataset [31, 46]. A higher value of CC@N means the *Top-N* items are more diverse.

Trade-off. To evaluate the overall performance in quality and diversity, a harmonic F-score metric (F) is employed [5], where $\text{F@N} = 2 \times quality\text{@N} \times diversity\text{@N} / (quality\text{@N} + diversity\text{@N})$. And $quality$@N represents the mean of Re@N and Nd@N, and $diversity$@N denotes CC@N.

Following previous work [29, 41, 42], an early stopping strategy is applied to avoid overfitting, *i.e.*, for each experiment we stop training if Nd@5 on the validation set does not increase for 10 successive epochs. To demonstrate the superior effectiveness of our loss functions that aim at sequential recommendation tasks, we deploy DSL and CDSL in three representative works (baselines). Table 2 reports the comparison of experimental results between the reworked models and **Caser** under different settings of target length

[3] https://www.kaggle.com/CooperUnion/anime-recommendations-database

Table 2: Performance comparison based on Caser. Significant improvements over Caser under the same setting are designated as * >5%, ** >10%, and *** >20%. The bold values, marked ‿ and __ denote the Caser achieves better performance than DSL and CDSL, than DSL and than CDSL, respectively.

Dataset		Method	Quality						Diversity			Trade-off		
			Re@3	Re@5	Re@10	Nd@3	Nd@5	Nd@10	CC@3	CC@5	CC@10	F@3	F@5	F@10
ML-1M	T=1	Caser	0.0595	0.0892	0.1398	0.0430	0.0551	0.0713	0.2493	0.3493	0.5086	0.0850	0.1196	0.1748
		DSL	-	-	-	-	-	-	-	-	-	-	-	-
		CDSL	0.0636*	0.0905	0.1388	0.0460*	0.0571	0.0724	0.2618*	0.3632	0.5277	0.0906*	0.1227	0.1760
	T=3	Caser	0.0506	0.0768	0.1313	0.1034	0.1294	0.1634	0.2501	0.3498	0.5045	0.1177	0.1593	0.2281
		DSL	0.0551*	0.0825*	0.1369	0.1129*	0.1400*	0.1756*	0.2505	0.3466	0.4969	0.1258*	0.1684*	0.2377
		CDSL	0.0561**	0.0783	0.1329	0.1151**	0.1362*	0.1718*	0.2581	0.3570	0.5118	0.1285*	0.1649	0.2348
	T=5	Caser	0.0432	0.0678	0.1151	0.1467	0.1805	0.2210	0.2549	**0.3640**	**0.5371**	0.1383	0.1852	0.2560
		DSL	0.0458*	0.0723*	0.1204	0.1513	0.1878	0.2284	0.2533	0.3604	0.5346	0.1419	0.1911	0.2630
		CDSL	0.0459*	0.0694	0.1195	0.1553*	0.1861	0.2296	0.2585	0.3616	0.5326	0.1448	0.1888	0.2629
Anime	T=1	Caser	0.2381	0.3231	0.4443	0.1829	0.2179	0.2572	0.2548	0.3517	0.4947	0.2305	0.3058	0.4105
		DSL	-	-	-	-	-	-	-	-	-	-	-	-
		CDSL	0.2700**	0.3452*	0.4681*	0.2115**	0.2423**	0.2820*	0.2519	0.3400	0.4866	0.2462*	0.3152	0.4236
	T=3	Caser	0.1667	0.2326	0.3332	0.3553	0.3770	0.4144	0.2548	0.3463	0.4858	0.2579	0.3242	0.4225
		DSL	0.1814*	0.2567**	0.3643*	0.3588	0.4035*	0.4397*	0.2636	0.3606	0.4971	0.2668	0.3447*	0.4445*
		CDSL	0.1846**	0.2636**	0.3841**	0.3736*	0.4190**	0.4589**	0.2591	0.3539	0.4877	0.2687	0.3474*	0.4522*
	T=5	Caser	0.1456	0.1977	0.2917	0.4226	0.4636	0.4990	**0.2581**	0.3541	0.4971	0.2705	0.3420	0.4404
		DSL	0.1521	0.2134*	0.3170*	0.4575*	0.4984*	0.5292*	0.2550	0.3546	0.4979	0.2777	0.3552	0.4575
		CDSL	0.1537*	0.2140*	0.3160*	0.4601*	0.5043*	0.5320*	0.2551	0.3590	0.4926	0.2786	0.3591*	0.4557
Beauty	T=1	Caser	0.0846	0.1076	0.1508	0.0538	0.0632	0.0806	0.0422	0.0614	0.1012	0.0524	0.0718	0.1080
		DSL	-	-	-	-	-	-	-	-	-	-	-	-
		CDSL	0.0944**	0.1310***	0.1817***	0.0716***	0.0864***	0.1029***	0.0441	0.0639	0.1034	0.0576**	0.0805**	0.1198**
	T=3	Caser	0.0414	0.0615	0.0975	0.0782	0.0928	0.1035	**0.0463**	0.0662	0.0994	0.0528	0.0703	0.0999
		DSL	0.0568***	0.0840***	0.1388***	0.1045***	0.1209***	0.1420***	0.0459	0.0668	0.1073	0.0585**	0.0809**	0.1216***
		CDSL	0.0496***	0.0760***	0.1289***	0.0831	0.1024**	0.1260***	0.0452	0.0657	0.1074	0.0538	0.0757*	0.1166**
	T=5	Caser	0.0434	0.0677	0.1219	0.1258	0.1472	0.1758	0.0433	0.0637	0.1032	0.0573	0.0800	0.1219
		DSL	0.0482**	0.0727*	0.1306*	0.1320	0.1568*	0.1822	0.0441	0.0629	0.1008	0.0592	0.0813	0.1226
		CDSL	0.0554***	0.0837***	0.1430**	0.1468**	0.1683**	0.1954**	0.0440	0.0640	0.1052	0.0613*	0.0849*	0.1297*

Table 3: Performance comparison based on MARank. Significant improvements over MARank under the same setting are designated as * >5%, ** >10%, and *** >20%. The bold values, marked ‿ and __ denote the MARank achieves better performance than DSL and CDSL, than DSL and than CDSL, respectively.

Dataset		Method	Quality						Diversity			Trade-off		
			Re@3	Re@5	Re@10	Nd@3	Nd@5	Nd@10	CC@3	CC@5	CC@10	F@3	F@5	F@10
ML-1M	T=1	MARank	0.0582	0.0864	0.1470	0.0437	0.0552	0.0746	0.2627	0.3716	0.5407	0.0853	0.1189	0.1839
		DSL	-	-	-	-	-	-	-	-	-	-	-	-
		CDSL	0.0679**	0.1017**	0.1622**	0.0500**	0.0639**	0.0832**	0.2711	0.3798	0.5504	0.0968**	0.1360**	0.2007*
	T=3	MARank	0.0498	0.0742	0.1222	0.1066	0.1308	0.1645	0.2651	0.3745	0.5451	0.1208	0.1609	0.2270
		DSL	0.0524*	0.0779	0.1303*	0.1105	0.1352	0.1687	0.2684	0.3759	0.5456	0.1250	0.1660	0.2347
		CDSL	0.0528*	0.0773*	0.1308	0.1093	0.1335	0.1701	0.2743	0.3848	0.5535	0.1251	0.1655	0.2366
Anime	T=1	MARank	0.1875	0.2518	0.3737	0.1433	0.1696	0.2090	0.2664	0.3536	0.4696	0.2041	0.2641	0.3596
		DSL	-	-	-	-	-	-	-	-	-	-	-	-
		CDSL	0.2211**	0.3039***	0.4453**	0.1683**	0.2023**	0.2477**	0.2617	0.3346	0.4734	0.2233*	0.2882*	0.4001**
	T=3	MARank	0.1346	0.1731	0.2616	0.2542	0.2930	0.3361	**0.2521**	**0.3381**	**0.4585**	0.2195	0.2759	0.3618
		DSL	0.1876***	0.2573***	0.3826***	0.3644***	0.4072***	0.4487***	0.2476	0.3327	0.4570	0.2610**	0.3325***	0.4353***
		CDSL	0.1738***	0.2417***	0.3591***	0.3361***	0.3805***	0.4264***	0.2458	0.3320	0.4517	0.2503**	0.3212**	0.4202**
Beauty	T=1	MARank	0.1103	0.1524	0.2310	0.0830	0.1004	0.1259	0.0423	0.0612	0.0967	0.0588	0.0825	0.1254
		DSL	-	-	-	-	-	-	-	-	-	-	-	-
		CDSL	0.1183*	0.1595	0.2214	0.0925**	0.1092*	0.1292	0.0415	0.0603	0.0981	0.0596	0.0832	0.1258
	T=3	MARank	0.0543	0.0830	**0.1542**	0.0966	0.1138	**0.1434**	0.0417	0.0602	0.0964	0.0537	0.0747	**0.1170**
		DSL	0.0606**	0.0889*	0.1428	0.1001	0.1201*	0.1413	0.0421	0.0611	0.0991	0.0553	0.0771	0.1168
		CDSL	0.0552	0.0856	0.1348	0.0975	0.1156	0.1324	0.0425	0.0603	0.0970	0.0546	0.0754	0.1124

(T). DSL and CDSL in Table 2 represent the reworked models that are constructed through replacing the original *binary cross-entropy* loss of Caser by DSL and CDSL, respectively. Similarly, Table 3 and 4 present performance comparisons based on **MARank** and **SASRec**, respectively. Under a setting T of a dataset, experiments of the baseline and reworked models are carried out using the same training instances and divided data.

Based on these three tables, we aim to demonstrate that deploying DSL or CDSL in existing sequential recommendation models can achieve better performance than using other loss functions,

Table 4: Performance comparison based on SASRec. Significant improvements over SASRec under the same setting are designated as * >5%, ** >10%, and *** >20%. The bold values, marked‿ and _ denote the SASRec achieves better performance than DSL and CDSL, than DSL and than CDSL, respectively.

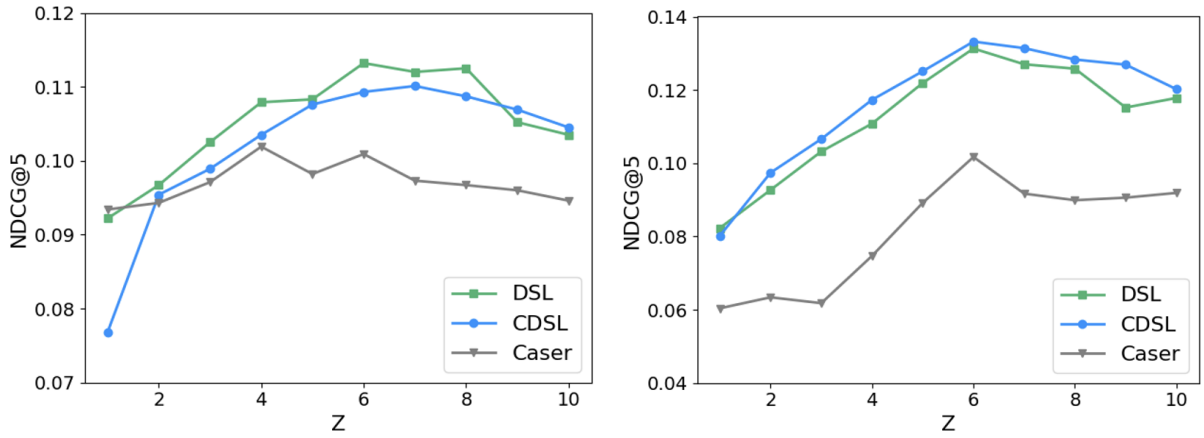
Dataset		Method	Quality						Diversity			Trade-off		
			Re@3	Re@5	Re@10	Nd@3	Nd@5	Nd@10	CC@3	CC@5	CC@10	F@3	F@5	F@10
ML-1M	T=1	SASRec	0.0484	0.0725	0.1208	0.0351	0.0442	0.0601	0.2526	0.3325	0.4616	0.0717	0.0993	0.1513
		DSL	-	-	-	-	-	-	-	-	-	-	-	-
		CDSL	0.0490	0.0730	0.1217	0.0354	0.0449	0.0604	0.2732**	0.3522**	0.4755	0.0731	0.1010	0.1528
	T=3	SASRec	0.0330	0.0501	**0.0891**	0.0596	0.0792	0.1110	0.2656	0.3480	0.4804	0.0789	0.1090	0.1656
		DSL	0.0350*	0.0519	0.0888	0.0658**	0.0827	0.1131	0.2665	0.3511	0.4847	0.0848*	0.1129	0.1671
		CDSL	0.0331	0.0507	0.0869	0.0613	0.0794	0.1089	0.2707	0.3543	0.4880	0.0804	0.1098	0.1631
Anime	T=1	SASRec	0.2628	0.3593	0.5108	0.2007	0.2402	0.2888	0.2293	0.3093	0.4336	0.2305	0.3045	0.4160
		DSL	-	-	-	-	-	-	-	-	-	-	-	-
		CDSL	0.2777*	0.3881*	0.5235	0.2160*	0.2611*	0.3049*	0.2339	0.3165	0.4459	0.2402	0.3205*	0.4295
	T=3	SASRec	0.1677	0.2324	0.3505	0.2510	0.2985	**0.3809**	0.2331	0.3152	0.4462	0.2205	0.2882	0.4019
		DSL	0.1749	0.2334	0.3705*	0.2413	0.2855	0.3642	0.2406	0.3250	0.4419	0.2232	0.2885	0.4012
		CDSL	0.1865**	0.2413	0.3530	0.2654	0.3086	0.3761	0.2384	0.3223	0.4493	0.2320*	0.2967	0.4025
Beauty	T=1	SASRec	0.0873	0.1238	0.2079	0.0665	0.0815	0.1086	0.0379	0.0527	0.0782	0.0508	0.0696	0.1047
		DSL	-	-	-	-	-	-	-	-	-	-	-	-
		CDSL	0.1032**	0.1421**	0.2286	0.0804***	0.0962**	0.1239**	0.0386	0.0507	0.0780	0.0543*	0.0711	0.1081
	T=3	SASRec	0.0645	0.0990	0.1627	0.0829	0.1103	0.1425	**0.0394**	0.0532	0.0808	0.0513	0.0705	0.1056
		DSL	0.0686*	0.0992	0.1565	0.0894*	0.1101	0.1433	0.0383	0.0533	0.0828	0.0516	0.0706	0.1066
		CDSL	0.0679*	0.1072*	0.1646	0.0909**	0.1187*	0.1528*	0.0389	0.0547	0.0834	0.0522	0.0737	0.1093

thus indicating the superiority of our loss functions. Therefore, tuning model parameters is not one of our focuses. In each comparison, we directly use the default model parameters of the baseline. This means that we keep the comparison fair. As Figure 1 shows, the number of negative samples (Z) is an important component of the user sequence ground set, which is also necessary for the selected baselines and most other related models [15, 20, 38] to calculate the loss. For all experiments of baselines and reworked models, we set $Z = 2$ when $T = 1$ and $Z = T$ in other cases. The effects of Z on performance are also analyzed and compared based on Figure 2. Besides, the previous item length L equals 5 when $T = 1$, and $L = 6$ when $T > 1$ to ensure that there are enough training sequences.

In Table 2-4, some important results are marked for clear comparison and emphasis. The value in bold means that a baseline achieves better performance than both DSL and CDSL on an experimental dataset under the setting of T (T test items and T targets). Besides, the marker‿ of a result denotes that a baseline only achieves the higher value than DSL, and the underline means that a baseline beats CDSL under a specific setting. The superscripts *, **, and *** indicate that a reworked model (DSL or CDSL) improves by >5%, >10%, and >20%, respectively over the corresponding baseline *w.r.t.* a metric (in the same column) under a target length T. For instance, in Table 2 when $T = 3$ of *ML-1M* dataset, the Re@3 result of CDSL (Caser with CDSL loss) shows 0.0561, which is 10.87% higher than baseline Caser's Re@3 result (0.0506). Therefore 0.0561** is presented.

We can see that there is no DSL result when $T = 1$ in Table 2-4. This is because DSL is designed for the next T targets estimation as explained in Section 2.3. When there is only one target in loss calculation, DSL cannot capture the targets dependency for model training, which is the core intuition of DSL. Therefore, it is not necessary to test DSL when $T = 1$. As for CDSL, even if there is only one target in the training sequence, the conditional set likelihood can be treated as the probability of selecting a DPP-distributed item set (previous items and a target) given the previous items. This means that only *sequence dependency* is considered when $T = 1$, and both *sequence dependency* and *targets dependency* are taken into account when $T > 1$. The main observations from Table 2-4 are as follows:

- Aligning these three tables, we can see that deploying DSL or CDSL on the state-of-the-art sequential recommendation models significantly improves recommendation quality (recall and NDCG) *w.r.t.* different *Top-N* on different datasets. This suggests that set likelihood-based objective functions that consider sequence dependency and targets dependency are more adaptable to sequential recommendation tasks than commonly used ranking-based loss functions.
- In only one case — MARank model on Beauty dataset when $T = 3$, the value of *trade-off* metric F@10 is slightly higher than DSL and CDSL. Under two settings (ML-1M dataset when $T = 3$ and Anime dataset when $T = 3$), SASRec outperforms the reworked DSL or CDSL version *w.r.t.* trade-off metric. In addition to these three cases, DSL and CDSL achieve better trade-off performance than the original loss functions of baselines. In some cases, the reworked models outperform baselines in F-score by a large margin (over 20%), *e.g.*, reworked MARank on Anime when $T = 3$. These demonstrate that DSL and CDSL can better handle the balance between quality and diversity by DPP kernel decomposition, *i.e.*, facilitating recommending relevant and diverse results.
- When the original models achieve better diversification (CC metric) than their reworked versions, *e.g.*, experiments on Anime dataset ($T = 1$) in Table 2, their corresponding quality performances (Recall and NDCG at the same top N) are clearly inadequate. This observation shows that state-of-the-art baselines sacrifice recommendation relevance for diversity (*i.e.*, sub-optimally trade-off between quality and diversity).
- In those few instances that baseline obtains better recommendation quality than DSL or CDSL in three tables, they are all at



(a) ML-1M dataset (b) Beauty dataset

Figure 2: Affects of negative sample length Z on ML-1M and Beauty datasets.

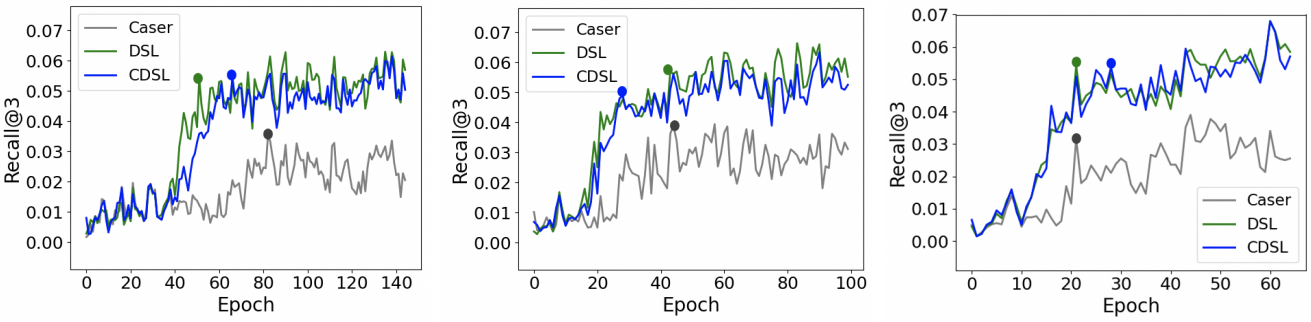
top-5 or top-10 (*e.g.*, Re@10 on ML-1M when $T = 1$ in Table 2). This indicates that in a few cases (only 5 values in Table 2-4), DSL and CDSL may focus more on providing the 'really' relevant (*e.g.*, top-3) items with a high probability of being drawn as a DPP set but ignore the relatively less relevant ones in a list.

- By comparing the performances under different settings of target length T in Table 1, $T = 3$ setting generally yields the most significant improvement compared to the baseline Caser. The reason might be that as the number of target items increases (*e.g.*, $T = 5$), noisy or confusing dependencies between the targets is considered, as thus affecting recommendations. In view of this phenomenon, Table 3 and 4 only present the results under $T = 1$ and $T = 3$ settings.
- Among three experimental datasets in Table 2-4, DSL and CDSL, in general, achieve the most significant improvement on Beauty dataset compared to the corresponding baseline. We think this is because on shopping Web platforms, users tend to search or buy products (items) that have tight correlations over a period. This verifies the importance of dependencies in the sequence or among targets to some extent, because dependencies are considered in the proposed loss functions.
- Compared to DSL in three tables, CDSL in general achieves better overall performance (F@N) when $T > 1$. This further verifies the importance of considering sequence dependency and targets dependency in the loss function, as CDSL considers both of them. Although CDSL cannot beat DSL in most cases with respect to quality metrics, it has a distinct advantage in diversification (CC@N). This finding tells us that taking previous actions into account (*i.e.*, increasing the size of the ground set $\mathcal{Y}^{(u,t)}$) is likely to make the decomposition of user-sequence specified DPP kernel care more about diversity than quality, thus leading CDSL to achieve better diversity performance. This can also be treated as an advantage of CDSL, because it is the distinct diversity result that contributes to the better overall performance.

Among all observations, there is no comparison between the performances of any two methods under the same setting but in different tables. This is because our purpose of presenting Table 2-4 is to demonstrate the superiority of DSL and CDSL through analyzing the performance improvement brought by using DSL or CDSL compared to the corresponding baseline with its original loss function under the same setting. Therefore, there is no need to compare the results of different tables. In the design of Caser, T targets are considered in loss calculation, so we can directly feed training instances of L previous items and T targets into its *binary cross-entropy loss*. Although the loss functions in MARank and SASRec are designed only for next one target, we still can deploy DSL or CDSL in them by modifying the training sequences and expand the original losses that consider one target to comparing T targets (Equation 1 and Equation 3) when $T > 1$. As the architecture of baselines is not altered, performance comparisons are still fair. Essentially, we only compare the loss functions.

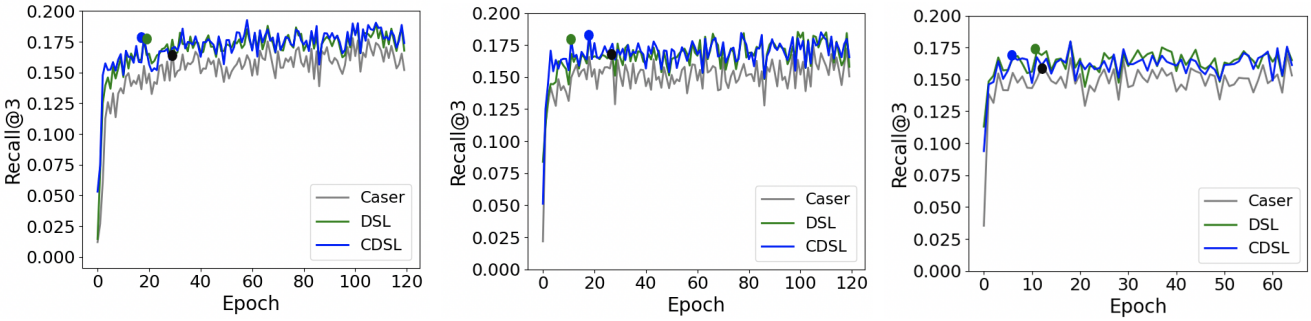
We use Figure 2 to investigate the effect of negative sample length Z. It shows that for all models (Caser and its reworked versions — DSL and CDSL) on ML-1M and Beauty under the setting of $T = 2$ and $L = 5$, with the increase of Z the overall trend is that the NDCG@5 performance grows rapidly at first and then gradually drops. For all experiments reported in Table 2-4, Z is set according to the target length T like most recommendation models do, instead of searching the optimal value of Z by tuning Z. This helps to keep the comparisons fair and the training efficient. In most cases, DSL and CDSL perform better than Caser and the improvement over Caser increases as the value of Z gets larger. When Z equals 1, the performance of CDSL on ML-1M is not better than Caser. This may be because there is not enough pattern/correlation for learning a competent DPP probability measure, when conditioning on 5 previous items for drawing a DPP set with 7 items ($L + T$) that almost equals the item number of the ground set ($L + T + Z = 8$). These observations indicate that negative samples are not only important for the original loss function but also for DSL and CDSL.

Figure 3 shows the test performance (Recall@3) of each training epoch of three models with different learning rates (`lr`) on Beauty. Figure 4 presents the same type of performance comparison on Anime. The value of 0.001 is the default setting of `lr` for the Caser baseline, which is also used for DSL and CDSL versions, with the same setting ($T = 3$, $L = 5$, and $Z = 3$) for all experiments. The solid



(a) lr=0.0005 (b) lr=0.001 (c) lr=0.0015

Figure 3: Test performance of each epoch with different learning rate on Beauty.



(a) lr=0.0005 (b) lr=0.001 (c) lr=0.0015

Figure 4: Test performance of each epoch with different learning rate on Anime.

circle on a performance curve of each subfigure indicates the best performance before the early stopping. In Figure 3 and 4, we use Recall@3 metric to judge whether the early stopping strategy is triggered instead of using NDCG@5 as mentioned above. The solid circle has the same color as the corresponding model's performance curve. Two main observations can be made: *i)* the proposed loss functions achieve better performance almost at any epoch in six subfigures (three `lr` settings on two datasets) compared to the baseline, indicating the superiority of our loss functions in a more detailed way; *ii)* in general, the proposed loss functions prefer a smaller learning rate compared to the original loss, and they learn faster (*i.e.*, trigger the early stopping earlier) in most cases. As the reworked models of other baselines share similar trends with Caser on different datasets, other subfigures with different settings are not presented.

Table 5 shows the training efficiency comparison between MARank (using pairwise loss) and reworked models with our loss functions on two datasets under different settings of target length. In the table, `Time` means average training time per epoch (in seconds), and `#Epochs` represents the number of epochs needed to converge (achieve best NDCG@5 performance before early stopping). `Total` records the total time used in a training process `Time` × `#Epochs`. Besides, `pct.` denotes the *percent change* between our reworked models and the baseline. All experiments in Table 5 are conducted using a single Tesla K40 GPU under the same setting. We can see that the reworked models with DSL or CDSL take more time per epoch than MARank, but converge faster (fewer epochs are needed in training process) and perform better (shown in Table 3). This observation is also made in Figure 3 and 4. In general, DSL and CDSL only need around 10% more time to coverage than that of MARank, but significant improvements are achieved (as shown in Table 2-4). In our experiments, when we compare the training efficiency based on Caser or SASRec (using pointwise ranking), similar observations to those of Table 5 are made. Therefore, comparisons based on other baselines and datasets are not presented.

From the analysis of Table 2-4 and Figure 2, we can draw a clear conclusion about how to make a choice between DSL and CDSL in certain circumstances. If a sequential recommender system emphasizes relevance and training efficiency, DSL is a better choice. On the other hand, in the case where overall performance between quality and diversity is the main concern (*e.g.*, diversity-promoting recommendation [28, 46]) and training efficiency is not the primary factor, choosing CDSL is more advisable.

4 RELATED WORK

Three lines of work are closely related to ours, which are listed as follows.

Sequential Recommendation. The main factor that sets the sequential recommendation apart from conventional recommender systems (*e.g.*, *Top-N* recommendation and collaborative filtering) is the consideration of sequential dependency. To capture sequential dependency for the user interaction sequence completion, Markov Chains based [15, 33] and Recurrent Neural Networks based [22,

Table 5: Training efficiency comparison on two datasets.

Datasets		MARank			DSL				CDSL			
		Time	#Epochs	Total	Time	#Epochs	Total	pct.	Time	#Epochs	Total	pct.
Anime	T=1	22.03	35	771.05	-	-	-	-	48.67	17	827.39	7.30%
	T=3	25.37	38	964.06	52.72	19	1001.68	3.90%	55.15	19	1047.85	8.69%
Beauty	T=1	1.02	40	40.80	-	-	-	-	1.88	23	43.24	5.98%
	T=3	1.06	42	44.52	1.94	23	44.62	0.22%	1.98	25	49.50	11.19%

38, 39] methods have dominated the literature and real-word applications. In this paper, we propose DSL and CDSL loss functions targeting next target(s) estimation. The next-item recommendation is a common task [39, 52] that calculate a loss based on next one target in the training process. The 'next items' in this paper actually means the next T targets for estimation in training. By preparing user sequences with L previous items and T targets for training, we aim to introduce the targets dependency into the loss formulation, and then facilitate recommendation models providing users with a set of relevant results. In this setting, next items recommendation is not the same as next-basket recommendation that predicts the next basket for a user based on a set of sequential baskets (*e.g.*, shopping baskets or transactions), as the basket length (number of products/items in basket) is not fixed. The session-based recommendation [18, 25] is a kind of non-personalized sequential recommendation, which uses sequential session data without user profile/identifier for recommendation. We believe that our loss functions are also adaptive to next-basket and session-based recommendations (both consider the temporal dependency), which will be explored in the future.

Loss Functions in Sequential Recommendation. In the sequential recommendation literature, two types of loss functions are widely employed, *i.e.*, binary cross-entropy and pairwise ranking (BPR). Many different variations are also developed based on these two loss functions, *e.g.*, softmax cross-entropy [7] and pairwise hinge loss [19]. BPR is usually limited to having only one target and one negative sample at each time step [35, 39, 51], but some studies have expanded the BPR formulation to a set-wise Bayesian approach [33, 40] to consider more comparisons. The seminal session-based recommendation model GRU4Rec [18] proposes the TOP1 loss which is similar to the BPR loss and an improved version GRU4Rec$^+$ [17] further boosts the recommendation performance using a special case of binary cross-entropy, *i.e.*, categorical cross-entropy. Recently, inspired by the widely used contrastive loss [13, 50] in the computer vision tasks, a line of work [47, 48] employs the contrastive loss in sequential recommendation models, which is defined similarly to the softmax cross entropy loss. The nature of the above-mentioned loss functions is still the relevance-based ranking [17], and the dependencies in sequence and among targets are ignored. In addition, the temporal dependency is complex [45], merely considering the dependency of items' ranking positions in a list (*e.g.*, list-wise ranking) is not enough [3]. Therefore, we treat the sequential recommendation as a task of set selection using the elegant probabilistic model — DPP, instead of directly comparing the rankings of items.

Determinantal Point Process. Employing DPP to diversify recommendations is a sensible choice, as DPP can measure the diversity of a set by describing the probability for all subsets of entire items. However, the main application of DPP that uses the maximum a posteriori (MAP) to generate diverse elements is NP-hard and thus computationally expensive [28]. With the development of a novel algorithm that accelerates MAP inference for DPP based on the fast inference method [4], some practical DPP-based diversity-promoting methods [10, 46] are proposed to facilitate traditional recommendations providing diverse and relevant results. In addition to using MAP inference for item generation, there are some works [43, 44] proposed to parameterize the DPP kernel on entire items and then learn the kernel matrix as item representations for recommendation. This work is quite different from previous studies, as we directly propose two types of loss functions (DSL and CDSL) based on the likelihoods of DPP set. By considering special dependency correlations in temporal sequences, DSL and CDSL can fit naturally into different sequential recommendation pipelines.

5 CONCLUSION

In this work, we proposed two novel DPP set likelihood-based loss functions (DSL and CDSL) aiming at sequential recommendations, which enlighten a new perspective (set selection) to formulate the temporal training objectives. As such, the *sequence dependency* and *targets dependency* are considered in the loss formulations. In addition, through pre-learning a diverse kernel from observed browsing history and using the quality *vs.* diversity decomposition of user-sequence specified DPP kernel, the proposed loss functions are pushed to be diversity-aware. By simply replacing the original loss of state-of-the-art sequential recommendation models with our DSL or CDSL objective functions, significant improvements can be received, which demonstrates the superiority and flexibility of the proposed loss functions. Experimental results help us analyze the different advantages of the two loss functions and provide insight into making selection between them in different circumstances. DSL and CDSL can also be applied to other temporal sequence related tasks, such as clinical events prediction [6, 20] and next-basket recommendation, which will be explored in the future.

Acknowledgement: This research is supported in part by the Australian Research Council Project DP180101985.

REFERENCES

[1] Alexei Borodin and Eric M Rains. 2005. Eynard–Mehta theorem, Schur process, and their Pfaffian analogs. *Journal of statistical physics* 121, 3 (2005), 291–317.
[2] Zhe Cao, Tao Qin, Tie-Yan Liu, Ming-Feng Tsai, and Hang Li. 2007. Learning to rank: from pairwise approach to listwise approach. In *Proceedings of the 24th international conference on Machine learning*. 129–136.
[3] Lei Chen, Le Wu, Kun Zhang, Richang Hong, and Meng Wang. 2021. Set2setRank: Collaborative Set to Set Ranking for Implicit Feedback based Recommendation. *arXiv preprint arXiv:2105.07377* (2021).

[4] Laming Chen, Guoxin Zhang, and Hanning Zhou. 2018. Fast greedy map inference for determinantal point process to improve recommendation diversity. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*. 5627–5638.

[5] Peizhe Cheng, Shuaiqiang Wang, Jun Ma, Jiankai Sun, and Hui Xiong. 2017. Learning to recommend accurate and diverse items. In *Proceedings of the 26th international conference on World Wide Web*. 183–192.

[6] Edward Choi, Mohammad Taha Bahadori, Andy Schuetz, Walter F Stewart, and Jimeng Sun. 2016. Doctor ai: Predicting clinical events via recurrent neural networks. In *Machine learning for healthcare conference*. PMLR, 301–318.

[7] Paul Covington, Jay Adams, and Emre Sargin. 2016. Deep neural networks for youtube recommendations. In *Proceedings of the 10th ACM conference on recommender systems*. 191–198.

[8] Mukund Deshpande and George Karypis. 2004. Item-based top-n recommendation algorithms. *ACM Transactions on Information Systems (TOIS)* 22, 1 (2004), 143–177.

[9] Hui Fang, Danning Zhang, Yiheng Shu, and Guibing Guo. 2020. Deep learning for sequential recommendation: Algorithms, influential factors, and evaluations. *ACM Transactions on Information Systems (TOIS)* 39, 1 (2020), 1–42.

[10] Lu Gan, Diana Nurbakova, Léa Laporte, and Sylvie Calabretto. 2020. Enhancing recommendation diversity using determinantal point processes on knowledge graphs. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2001–2004.

[11] Mike Gartrell, Victor-Emmanuel Brunel, Elvis Dohmatob, and Syrine Krichene. 2019. Learning nonsymmetric determinantal point processes. *arXiv preprint arXiv:1905.12962* (2019).

[12] Mike Gartrell, Ulrich Paquet, and Noam Koenigstein. 2017. Low-rank factorization of determinantal point processes. In *Thirty-First AAAI Conference on Artificial Intelligence*.

[13] Raia Hadsell, Sumit Chopra, and Yann LeCun. 2006. Dimensionality reduction by learning an invariant mapping. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, Vol. 2. IEEE, 1735–1742.

[14] Ruining He, Wang-Cheng Kang, and Julian McAuley. 2017. Translation-based recommendation. In *Proceedings of the eleventh ACM conference on recommender systems*. 161–169.

[15] Ruining He and Julian McAuley. 2016. Fusing similarity models with markov chains for sparse sequential recommendation. In *2016 IEEE 16th International Conference on Data Mining (ICDM)*. IEEE, 191–200.

[16] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural collaborative filtering. In *Proceedings of the 26th international conference on world wide web*. 173–182.

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

[18] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. 2015. Session-based recommendations with recurrent neural networks. *arXiv preprint arXiv:1511.06939* (2015).

[19] Cheng-Kang Hsieh, Longqi Yang, Yin Cui, Tsung-Yi Lin, Serge Belongie, and Deborah Estrin. 2017. Collaborative metric learning. In *Proceedings of the 26th international conference on world wide web*. 193–201.

[20] Haoji Hu and Xiangnan He. 2019. Sets2sets: Learning from sequential sets with neural networks. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1491–1499.

[21] Yifan Hu, Yehuda Koren, and Chris Volinsky. 2008. Collaborative filtering for implicit feedback datasets. In *2008 Eighth IEEE International Conference on Data Mining*. Ieee, 263–272.

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

[23] Alex Kulesza and Ben Taskar. 2011. k-DPPs: Fixed-size determinantal point processes. In *ICML*.

[24] Alex Kulesza and Ben Taskar. 2012. Determinantal point processes for machine learning. *arXiv preprint arXiv:1207.6083* (2012).

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

[26] Jiacheng Li, Yujie Wang, and Julian McAuley. 2020. Time interval aware self-attention for sequential recommendation. In *Proceedings of the 13th international conference on web search and data mining*. 322–330.

[27] Yang Li, Tong Chen, Peng-Fei Zhang, and Hongzhi Yin. 2021. Lightweight Self-Attentive Sequential Recommendation. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*. 967–977.

[28] Yile Liang and Tieyun Qian. 2021. Recommending Accurate and Diverse Items Using Bilateral Branch Network. *arXiv preprint arXiv:2101.00781* (2021).

[29] Zhiwei Liu, Yongjun Chen, Jia Li, Philip S Yu, Julian McAuley, and Caiming Xiong. 2021. Contrastive self-supervised sequential recommendation with robust augmentation. *arXiv preprint arXiv:2108.06479* (2021).

[30] Julian McAuley, Christopher Targett, Qinfeng Shi, and Anton Van Den Hengel. 2015. Image-based recommendations on styles and substitutes. In *Proceedings of the 38th international ACM SIGIR conference on research and development in information retrieval*. 43–52.

[31] Shameem A Puthiya Parambath, Nicolas Usunier, and Yves Grandvalet. 2016. A coverage-based approach to recommendation diversity on similarity graph. In *Proceedings of the 10th ACM Conference on Recommender Systems*. 15–22.

[32] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2012. BPR: Bayesian personalized ranking from implicit feedback. *arXiv preprint arXiv:1205.2618* (2012).

[33] Steffen Rendle, Christoph Freudenthaler, and Lars Schmidt-Thieme. 2010. Factorizing personalized markov chains for next-basket recommendation. In *Proceedings of the 19th international conference on World wide web*. 811–820.

[34] U Ruby and V Yendapalli. 2020. Binary cross entropy with deep learning technique for image classification. *International Journal of Advanced Trends in Computer Science and Engineering* 9, 10 (2020).

[35] Richard Socher, Danqi Chen, Christopher D Manning, and Andrew Ng. 2013. Reasoning with neural tensor networks for knowledge base completion. *Advances in neural information processing systems* 26 (2013).

[36] Yiping Song, Rui Yan, Yansong Feng, Yaoyuan Zhang, Dongyan Zhao, and Ming Zhang. 2018. Towards a neural conversation model with diversity net using determinantal point processes. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 32.

[37] Harald Steck. 2015. Gaussian ranking by matrix factorization. In *Proceedings of the 9th ACM Conference on Recommender Systems*. 115–122.

[38] Fei Sun, Jun Liu, Jian Wu, Changhua Pei, Xiao Lin, Wenwu Ou, and Peng Jiang. 2019. BERT4Rec: Sequential recommendation with bidirectional encoder representations from transformer. In *Proceedings of the 28th ACM international conference on information and knowledge management*. 1441–1450.

[39] Jiaxi Tang and Ke Wang. 2018. Personalized top-n sequential recommendation via convolutional sequence embedding. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*. 565–573.

[40] Chao Wang, Hengshu Zhu, Chen Zhu, Chuan Qin, and Hui Xiong. 2020. SetRank: A setwise Bayesian approach for collaborative ranking from implicit feedback. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 34. 6127–6136.

[41] Wenjie Wang, Minlie Huang, Xin-Shun Xu, Fumin Shen, and Liqiang Nie. 2018. Chat more: Deepening and widening the chatting topic via a deep model. In *The 41st international acm sigir conference on research & development in information retrieval*. 255–264.

[42] Xiang Wang, Xiangnan He, Meng Wang, Fuli Feng, and Tat-Seng Chua. 2019. Neural graph collaborative filtering. In *Proceedings of the 42nd international ACM SIGIR conference on Research and development in Information Retrieval*. 165–174.

[43] Romain Warlop, Jérémie Mary, and Mike Gartrell. 2019. Tensorized determinantal point processes for recommendation. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1605–1615.

[44] Mark Wilhelm, Ajith Ramanathan, Alexander Bonomo, Sagar Jain, Ed H Chi, and Jennifer Gillenwater. 2018. Practical diversified recommendations on youtube with determinantal point processes. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*. 2165–2173.

[45] Qitian Wu, Yirui Gao, Xiaofeng Gao, Paul Weng, and Guihai Chen. 2019. Dual sequential prediction models linking sequential recommendation and information dissemination. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 447–457.

[46] Qiong Wu, Yong Liu, Chunyan Miao, Binqiang Zhao, Yin Zhao, and Lu Guan. 2019. PD-GAN: Adversarial Learning for Personalized Diversity-Promoting Recommendation.. In *IJCAI*, Vol. 19. 3870–3876.

[47] Xu Xie, Fei Sun, Zhaoyang Liu, Shiwen Wu, Jinyang Gao, Bolin Ding, and Bin Cui. 2020. Contrastive Learning for Sequential Recommendation. *arXiv preprint arXiv:2010.14395* (2020).

[48] Zhe Xie, Chengxuan Liu, Yichi Zhang, Hongtao Lu, Dong Wang, and Yue Ding. 2021. Adversarial and Contrastive Variational Autoencoder for Sequential Recommendation. In *Proceedings of the Web Conference 2021*. 449–459.

[49] Chengfeng Xu, Pengpeng Zhao, Yanchi Liu, Jiajie Xu, Victor S Sheng S. Sheng, Zhiming Cui, Xiaofang Zhou, and Hui Xiong. 2019. Recurrent convolutional neural network for sequential recommendation. In *The world wide web conference*. 3398–3404.

[50] Shuo Yang, Wei Yu, Ying Zheng, Hongxun Yao, and Tao Mei. 2019. Adaptive semantic-visual tree for hierarchical embeddings. In *Proceedings of the 27th ACM International Conference on Multimedia*. 2097–2105.

[51] Lu Yu, Chuxu Zhang, Shangsong Liang, and Xiangliang Zhang. 2019. Multi-order attentive ranking model for sequential recommendation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 5709–5716.

[52] Fajie Yuan, Alexandros Karatzoglou, Ioannis Arapakis, Joemon M Jose, and Xiangnan He. 2019. A simple convolutional generative network for next item recommendation. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*. 582–590.