A Guided Learning Approach for Item Recommendation via Surrogate Loss Learning

Ahmed Rashed
ahmedrashed@ismll.uni-hildesheim.de
Information Systems and Machine Learning Lab, University of Hildesheim, Germany

Josif Grabocka
grabocka@informatik.uni-freiburg.de
Department of Computer Science, University of Freiburg, Germany

Lars Schmidt-Thieme
schmidt-thieme@ismll.uni-hildesheim.de
Information Systems and Machine Learning Lab, University of Hildesheim, Germany

ABSTRACT

Normalized discounted cumulative gain (NDCG) is one of the popular evaluation metrics for recommender systems and learning-to-rank problems. As it is non-differentiable, it cannot be optimized by gradient-based optimization procedures. In the last twenty years, a plethora of surrogate losses have been engineered that aim to make learning recommendation and ranking models that optimize NDCG possible. However, binary relevance implicit feedback settings still pose a significant challenge for such surrogate losses as they are usually designed and evaluated only for multi-level relevance feedback. In this paper, we address the limitations of directly optimizing the NDCG measure by proposing a guided learning approach (GuidedRec) that adopts recent advances in parameterized surrogate losses for NDCG. Starting from the observation that jointly learning a surrogate loss for NDCG and the recommendation model is very unstable, we design a stepwise approach that can be seamlessly applied to any recommender system model that uses a point-wise logistic loss function. The proposed approach guides the models towards optimizing the NDCG using an independent surrogate-loss model trained to approximate the true NDCG measure while maintaining the original logistic loss function as a stabilizer for the guiding procedure. In experiments on three recommendation datasets, we show that our guided surrogate learning approach yields models better optimized for NDCG than recent state-of-the-art approaches using engineered surrogate losses.

CCS CONCEPTS

• **Computing methodologies** → **Artificial intelligence**; **Learning from implicit feedback**; • **Information systems** → **Recommender systems**.

KEYWORDS

Collaborative Filtering, Recommender Systems, Surrogate Loss Learning, Learning to Rank

SIGIR '21, July 11–15, 2021, Virtual Event, Canada

ACM ISBN 978-1-4503-8037-9/21/07…$15.00
https://doi.org/10.1145/3404835.3462864

ACM Reference Format:
Ahmed Rashed, Josif Grabocka, and Lars Schmidt-Thieme. 2021. A Guided Learning Approach for Item Recommendation via Surrogate Loss Learning. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '21), July 11–15, 2021, Virtual Event, Canada.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3404835.3462864

1 INTRODUCTION

Recommender systems aim to provide a personalized ranked list of items to users that they are likely to interact with through clicks, views, or purchases. To achieve this goal, two crucial design choices have to be defined, which are the type of feedback setting on which the model will be deployed and the ranking quality measure for which we want to optimize the model. One famous and widely used example of such measures is the Normalized Discounted Cumulative Gain (NDCG) [7, 8]; however, since NDCG is non-differentiable, most of the models are usually optimized by different proxy loss functions depending on the used feedback setting.

In this work we focus on implicit feedback settings where we have only observed and unobserved interactions; hence only binary relevance feedback exists, and all observed interactions are equally important. Implicit feedback interactions currently dominate most recommender systems settings such as clicks, views, and purchases as they are generally much easier to collect and do not require any user-side explicit actions.

In such settings, models are usually optimized for maximizing the ranking measure, such as the NDCG by maximizing the difference between the likelihood scores of positive (observed) and negative (unobserved) pairs of users and items. To solve this task, point-wise logistic loss function (log loss) [7] or pair-wise loss functions such as BPR [17] are usually selected for the training procedure. Also, given the absence of multi-level relevance feedback in such settings, point-wise and pair-wise losses are generally preferred over other proxy list-wise ranking losses, such as the smoothly approximated NDCG [2, 15], Gumbel approximated NDCG [1] and sorting networks [5], even though they can not capture the list-wise nature of the target ranking measure for which the model to be optimized. This preference is mainly because the proxy list-wise losses mostly rely on having explicit multi-level relevance feedback, and they were not evaluated on implicit binary relevance settings.

In this paper, we propose a guided learning step-wise approach (GuidedRec) that allows recommender system models to directly optimize the NDCG instead of using proxy loss functions by adapting, stabilizing and extending the Learning Surrogate Losses approach

by (Grabocka et al. 2019) for optimizing the NDCG measure. The proposed GuidedRec approach can be used to optimize any recommender model that utilizes a point-wise logistic loss function seamlessly and without requiring any modifications to the model itself. To optimize the true NDCG function, GuidedRec uses an independent surrogate loss model that is trained in parallel to learn and maximize the NDCG loss using the output scores of the recommender model. The GuidedRec approach also utilizes the original logistic loss function as a secondary task to stabilize the optimization phase of the surrogate model. The contributions of this paper can be summarized as follows:

- We show that learning a surrogate loss for NDCG with the Learning Surrogate Losses approach is unstable and does not yield competitive results.
- We propose an alternative guided learning approach for surrogate losses that learns the surrogate loss in parallel to learning the recommender model using the original logistic loss to stabilize the learning phase.
- We apply learning surrogate losses and guided learning surrogate losses for the first time to NDCG.
- In experiments on three recommendation datasets we show that our guided surrogate learning approach yields models better optimized for NDCG than recent state-of-the-art engineered surrogate losses.

2 RELATED WORK

Deciding on which optimization strategy to use in the task of item recommendation depends on two main factors. The first one is the target feedback setting on which the model will be deployed; this can be explicit, implicit, or a hybrid setting. The second factor is the target ranking measure that we will want to optimize and use for performance evaluation.

In implicit feedback settings there exist only observed and unobserved interactions and multiple types of proxy loss functions have been used to optimize the final non-differentiable NDCG ranking measure. Two widely used functions are the point-wise logistic loss [7] and the pair-wise BPR loss [17]. These functions rely on maximizing the difference between the likelihood scores of positive and negative interactions. They are also mostly used in models that output a single likelihood score for any user-item pair fed as an input, such as matrix factorization [17] and deep learning models [7]. Another candidate proxy loss function is the multinomial logistic loss, which can be used in auto-encoder based models [9]. Although these proxy functions are used in most of the recent state-of-the-art models, they still cannot capture the list-wise nature of the target NDCG ranking measure in terms of the relative preference between items in the same list and the relative position of the positive items in the final ranked list. A possible appealing solution to such limitations is using list-wise loss functions that mathematically approximate the NDCG measure, such as the smoothly approximated NDCG [2, 15], Gumbel approximated NDCG [1] or using a differentiable sorting network [5] that can address the non-differentiable sorting operation in the NDCG metric. However, these functions mainly rely on having explicit multi-level relevance feedback values, and they were not evaluated on binary relevance feedback settings before. We further shed light on these state-of-the-art loss functions in the experiments section, as we propose to use them as baselines. Our empirical results on multiple datasets also showed that they sometimes perform slightly better than other point-wise proxy functions in binary-relevance settings.

A possible approach to solve the different limitations of these different proxy functions is to directly learn and optimize the final target measure through surrogate loss learning [4]. However, this approach suffers from stability issues similar to Generative Adversarial Networks (GANs), and it needs to be further extended to be applicable for list-wise measures such as the NDCG.

In this work, we extend and address the stability problem of the surrogate loss learning approach for item recommendation tasks by proposing a guided learning approach (GuidedRec) for directly optimizing the NDCG measure in implicit feedback settings. To mitigate the stability issue, we propose using the point-wise logistic loss function as a stabilizer for the guidance procedures, which achieves a significant improvement over multiple state-of-the-art proxy loss functions for ranking items. We also limit our focus on the logistic loss function as a stabilizing function; however, we note that following the same notation, other loss functions can also be subsequently derived and exploited as alternatives.

3 PROBLEM DEFINITION

In item recommendation tasks, there exist a set of users $U := \{u_1, u_2, ..., u_N\}$, a set of items $I := \{i_1, i_2, ..., i_M\}$, and a sparse binary interaction matrix $R \in \mathbb{R}^{N \times M}$ that indicate user's implicit preferences on items. The main goal of the item recommendation task is to fill the missing entries in R using a recommender system model $\hat{r} : U \times I \rightarrow \mathbb{R}$ in order to present users with short personalized ranked lists of relevant items based on their likelihood scores. To do so, models have to be optimized with respect to maximizing the ranked list's NDCG quality measure

$$\text{DCG@k} = \sum_{p=1}^{k} \frac{rel_p}{\log_2(p+1)} \tag{1}$$

$$\text{NDCG@k} = \frac{\text{DCG}_k}{\text{IDCG}_k} \tag{2}$$

where rel_p is the relevance of the item at position p. IDCG_k is the best possible value for DCG for the best possible ranking of relevant items at threshold k.

4 PROPOSED METHOD

The proposed GuidedRec approach can be divided into two main components. The first component is the wrapper that reshapes the list-wise inputs required for optimizing the NDCG ranking measure into pair-wise inputs suitable for recommender models that utilize point-wise logistic loss functions. The wrapper also converts the point-wise predicted relevance scores into list-wise relevance scores suitable for calculating the NDCG measure. The second component is the surrogate loss model that is trained in parallel to simultaneously approximate the true NDCG function and to maximize the prediction performance through maximizing the NDCG measure of the predicted lists of items.

Figure 1 illustrates the architecture of the GuidedRec and each component will be discussed in detail in the following subsections.

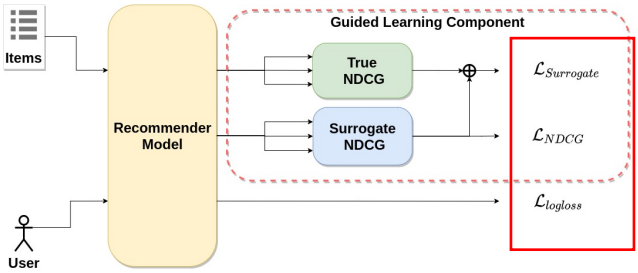

Figure 1: GuidedRec Architecture

4.1 GuidedRec Wrapper

To optimize the list-wise NDCG loss through GuidedRec, we have to optimize the recommender system model using a list-wise batch of inputs $(\mathcal{U}, \mathcal{T}, \mathcal{Y}) := \{(u_1, t_1, y_1), ..., (u_{B_L}, t_{B_L}, y_{B_L})\}$ where B_L is the list-wise batch size, u_j is the target user, t_j is a list of K items $t_j := \{i_1^{(j)}, ..., i_K^{(j)}\}$, and $y_j := \{r_1^{(j)}, ..., r_K^{(j)}\}$ is the list of ground truth binary relevance scores for items in t_j with respect to user u_j.

On the other hand, recommender models that utilize a point-wise loss function such as the log loss usually inputted with a pair-wise batch of B_P inputs $(\mathcal{U}, \mathcal{I}, \mathcal{R}) := \{(u_1, i_1, r_1), ..., (u_{B_P}, i_{B_P}, r_{B_P})\}$ to predicts a single likelihood score $\hat{r}_{ui}$ for each user-item pair as follows:

$$\hat{r}_{ui} = f(u, i; \theta_f) \tag{3}$$

where f is the recommender system model with parameters θ_f, u represents a user's one-hot encoded input vector, and i represents an item's one-hot encoded input vector.

To apply GuidedRec to such models without having to modify them, we simply reshape the sampled list-wise batch $(\mathcal{U}, \mathcal{T}, \mathcal{Y})$ into a pair-wise batch of size $B_L \times K$ by repeating u_j for each item in t_j and utilizing their ground-truth scores $r_k^{(j)}$ in y_j. We denote this preprocessing function as $Q_{in} : (\mathcal{U}, \mathcal{T}, \mathcal{Y}) \rightarrow (\mathcal{U}, \mathcal{I}, \mathcal{R})$. Similarly, we reshape back the output scores $\hat{\mathcal{R}} \in \mathbb{R}^{(B_L \times K) \times 1}$ into $\hat{\mathcal{Y}} \in \mathbb{R}^{(B_L \times K)}$ to recover the list-wise shape of the predicted scores using the postprocessing function $Q_{out} : \mathbb{R}^{(B_L \times K) \times 1} \rightarrow \mathbb{R}^{B_L \times K}$.

4.2 Surrogate Loss Model

Given the list-wise predicted scores $\hat{y} \in \hat{\mathcal{Y}}$ and ground-truth scores $y \in \mathcal{Y}$, we define the add-on surrogate loss model that estimate the NDCG of the predicted scores as follows:

$$x = zip(y, \hat{y}) = \{(r_1, \hat{r}_1), ..., (r_K, \hat{r}_K)\}, x \in R^{K \times 2} \tag{4}$$

$$\phi_{\text{MLP}}(x) = v(Flatten(x); \theta_v) \tag{5}$$

$$z = g(x; \theta_g) \tag{6}$$

$$\phi_{\text{NFM}}(x) = e\left(\frac{1}{K} z z^T \mathbf{1}; \theta_e\right) \tag{7}$$

$$\hat{\text{NDCG}}(y, \hat{y}) = h(\phi_{\text{NFM}}(x) \odot \phi_{\text{MLP}}(x); \theta_h) \tag{8}$$

where g, v, e, and h are a series of non-linear fully connected layers with weights θ_g, θ_v, θ_e, and θ_h respectively, and $\odot$ denotes the element-wise product of two vectors. $Flatten(x)$ is a function that flatten the input tensor x to a one-dimensional array. $\phi_{\text{NFM}}(x)$ is a non-linear factorization machine component inspired by the architecture of the NFM model [6] that capture the bi-linear interactions between the latent features of the predicted and ground-truth scores pairs in x. Finally, $\hat{\text{NDCG}}(y, \hat{y})$ represents the final predicted NDCG score that is calculated by having an ensemble of a multi-layer perception component $\phi_{\text{MLP}}(x)$ and the non-linear factorization machine component $\phi_{\text{NFM}}(x)$. Both components are inputted with the series of predicted and ground-truth scores pairs x and their output latent features are multiplied together element-wise before being fed to the final NDCG scoring function h. Figure 3d illustrates the final architecture of the surrogate loss model. It is worthy to note that we tried different architectures that are shown in Figure 3, however, we found out that ensemble version performed best on all datasets. Comparisons between different architectures of the surrogate model will be discussed in detail in Section 5.4.

4.3 Optimizing GuidedRec

In this section we explain the end-to-end learning procedure of GuidedRec. Given the recommender system model f and the surrogate loss model $\hat{\text{NDCG}}(\mathcal{Y}, \hat{\mathcal{Y}})$ with weights $\Theta := \{\theta_g, \theta_v, \theta_e, \theta_h\}$, we define a two step optimization procedure that is applied to every input batch $(\mathcal{U}, \mathcal{T}, \mathcal{Y})$.

In the first step we optimize the recommender model f by minimizing its original log loss function in Equation (9) while fixing the surrogate model weights Θ. We also utilize a weighting parameter β that controls the importance of the log loss function to the overall learning procedure. After optimizing the original log loss function, we retrieve the reshaped predicted relevance scores $\hat{\mathcal{Y}}$ to calculate the true NDCG score $NDCG(\mathcal{Y}, \hat{\mathcal{Y}})$ [7] using the ground-truth scores $\mathcal{Y}$.

$$\mathcal{L}_{\text{logloss}}(\mathcal{U}, \mathcal{I}, \mathcal{R}) = -\beta\left(\sum_{(u,i) \in (\mathcal{U}, \mathcal{I}, \mathcal{R})} r_{ui} \log(\hat{r}_{ui}) + (1 - r_{ui}) \log(1 - \hat{r}_{ui})\right) \tag{9}$$

In the second step, we follow a similar bi-level optimization approach to the one proposed by [4]. Once the true NDCG score is calculated, we first optimize the surrogate model to approximate the true NDCG score by minimizing Equation (10) while fixing the model weights θ_f of the recommender system. After that, we optimize the recommender system model f to maximize the predicted NDCG score of the surrogate loss model $\hat{\text{NDCG}}(\mathcal{Y}, \hat{\mathcal{Y}})$ by minimizing Equation (11) while fixing the surrogate model weights Θ.

$$\mathcal{L}_{\text{Surrogate}}(\mathcal{Y}, \hat{\mathcal{Y}}) = \sum_{(y, \hat{y}) \in \text{zip}(\mathcal{Y}, \hat{\mathcal{Y}})} \left(\text{NDCG}(y, \hat{y}) - \hat{\text{NDCG}}(y, \hat{y})\right)^2 \tag{10}$$

$$\mathcal{L}_{\text{NDCG}}(\mathcal{Y}, \hat{\mathcal{Y}}) = \sum_{(y, \hat{y}) \in \text{zip}(\mathcal{Y}, \hat{\mathcal{Y}})} -\hat{\text{NDCG}}(y, \hat{y}) \tag{11}$$

The full pseudo-code of GuidedRec is described in Algorithm 1.

Algorithm 1: GuidedRec ($\mathcal{U}, \mathcal{I}, \mathcal{R}$)

input : A set of users $\mathcal{U}$, a set of items $\mathcal{I}$ and their ground-truth relevance scores $\mathcal{R}$
output : The predicted items scores $\hat{\mathcal{Y}}$

Initialize the surrogate model and the recommender system model parameters Θ and θ_f
for *E epochs* **do**
  Draw a list-wise batch $(\mathcal{U}, \mathcal{T}, \mathcal{Y})$
  Convert the list-wise batch to pair-wise batch using Q_{in}
  Predict items scores $\hat{\mathcal{R}}$ using Eq. (3)
  Update θ_f by mnimizing Eq. (9)
  Convert the pair-wise predicted scores $\hat{\mathcal{R}}$ to list-wise scores $\hat{\mathcal{Y}}$ using Q_{out}
  Calculate the average true NDCG value using Eq. (1) and (2)
  Predict the surrogate NDCG value using Eq. (8)
  Update Θ by minimizing Eq. (10) while fixing θ_f
  Update θ_f by maximizing Eq. (11) while fixing Θ
end

5 EXPERIMENTS

In this section, multiple experiments were conducted to evaluate the performance of the GuidedRec approach and to answer the following research questions.

RQ1 How well does GuidedRec perform compared to the state-of-the-art ranking-based surrogate loss functions?
RQ2 What is the best candidate architecture for the surrogate loss model?
RQ3 Can we learn the NDCG measure from scratch without using a stabilizing loss function?

5.1 Datasets

In order to evaluate the performance of GuidedRec, we used the following real-world datasets:

(1) MovieLens 100K[1]: This is a well-known dataset for movies rating prediction which contains 100,000 ratings.
(2) MovieLens 1M: This is a larger version of the MovieLens 100K with 1,000,209 interactions.
(3) MovieTweeting [3]: This dataset consists of movies ratings that were collected from a stream of tweets on Twitter. This dataset is extremely sparse and it is much more challenging to learn useful patterns from it, hence a simple naive Top-Popular model usually outperforms all other baselines on it. We use this dataset not just for performance comparison but also for checking the loss functions robustness against overfitting.

We transformed all the datasets into implicit feedback binary interactions that indicate whether the user has rated the item or not similar to [7]. Datasets statistics are shown in Table 1.

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

Table 1: Datasets Statistics

Dataset	Items	Users	Interactions
MovieLens 100k	1,682	943	100,000
MovieLens 1M	3,952	6,040	1,000,209
MovieTweeting	7,505	10,455	50,000

5.2 Evaluation Protocol

To evaluate the GuidedRec performance on the item recommendation task, we utilize the widely used leave-one-out evaluation protocol [7]. In this protocol, we hold out the last two positive interactions for every user for validation and testing while using the rest for training. For evaluation, we rank the positive test item among different 99 negatively sample items that were not interacted by the user. Finally, for each user, we truncate the ranked list at a threshold value of 10 to match our training sampling strategy indicated in section 5.3, and we measure the overall quality using the average Hit-Ratio (HR) and the Normalized Discounted Cumulative Gain (NDCG) across all users.

To measure the statistical significance of the results, we repeat this process five times, and we report the average metrics across all runs along with their statistical significance using the p-values of a paired t-test. The optimal hyper-parameters for the GuidedRec approach have been estimated via grid search on the validation set. The best hyper-parameter settings and the utilized grid search space are discussed in section 7.2 in the Appendix.

5.3 Performance comparison with state-of-the-art ranking-based loss functions (RQ1)

In this section we conduct multiple experiments to compare the performance of using the GuidedRec learning procedure against the widely used log loss function and a battery of state-of-the-art ranking loss functions from the TensorFlow TFRanking library [14].

To have a fair comparison, we utilized the same batch sampling strategy for training and a fixed state-of-the-art GraphRec model [16] as a base recommender system model for all loss functions. The optimal hyper-parameters for the fixed GraphRec model have been estimated via grid search on the validation set using the log loss function. It is worth noting that we also tried to estimate the optimal model hyper-parameters separately for each loss function, but no differences were found. The best hyper-parameter settings and the utilized grid search space will be discussed in section 7.2 in the Appendix. Regarding the sampling strategy, we sample 10 items for each user in the batch with an equal ratio of positive and negative items. All loss functions were optimized with the ADAM optimizer.

5.3.1 *Baselines Loss Functions.*

(1) **Logloss** [7, 14]: This is the widely used logistic loss function for binary classification.
(2) **ListMLE** [18]: A famous and widely used list-wise likelihood loss function for learning-to-rank tasks.
(3) **Softmax** [14]: A list-wise Softmax Cross-Entropy loss function for learning-to-rank with binary relevance scores.

Table 2: Performance comparison of the GuidedRec learning approach

Loss Function	MovieLens 100k HR@10	MovieLens 100k NDCG@10	MovieLens 1M HR@10	MovieLens 1M NDCG@10	MovieTweeting HR@10	MovieTweeting NDCG@10
Logloss	0.628	0.368	0.681	0.413	0.723	0.567
Gumbel-NDCG [1]	0.633	0.372	0.684	0.410	0.722	0.567
Approx-NDCG [2, 15]	0.618	0.355	0.684	0.409	0.714	0.559
ListMLE [18]	0.618	0.366	0.675	0.401	0.723	0.567
Softmax [14]	0.620	0.346	0.656	0.374	0.724	0.568
Pair-wise logloss [14]	0.645	0.377	0.688	0.409	0.724	0.568
Neural-Sort [5]	0.645	0.379	0.663	0.385	0.726	0.567
GuidedRec w/o logloss	0.437	0.241	0.292	0.170	0.724	0.567
GuidedRec w/ logloss	**0.661****	**0.386****	**0.695***	**0.422****	**0.732****	**0.571****
Benchmark Models						
TopPopular	0.406	0.219	0.467	0.260	0.724	0.565
NeuMF [7]	0.621	0.356	0.660	0.393	0.724	0.567

Significantly outperforms the best baseline at the: ** 0.01 and * 0.05 levels.

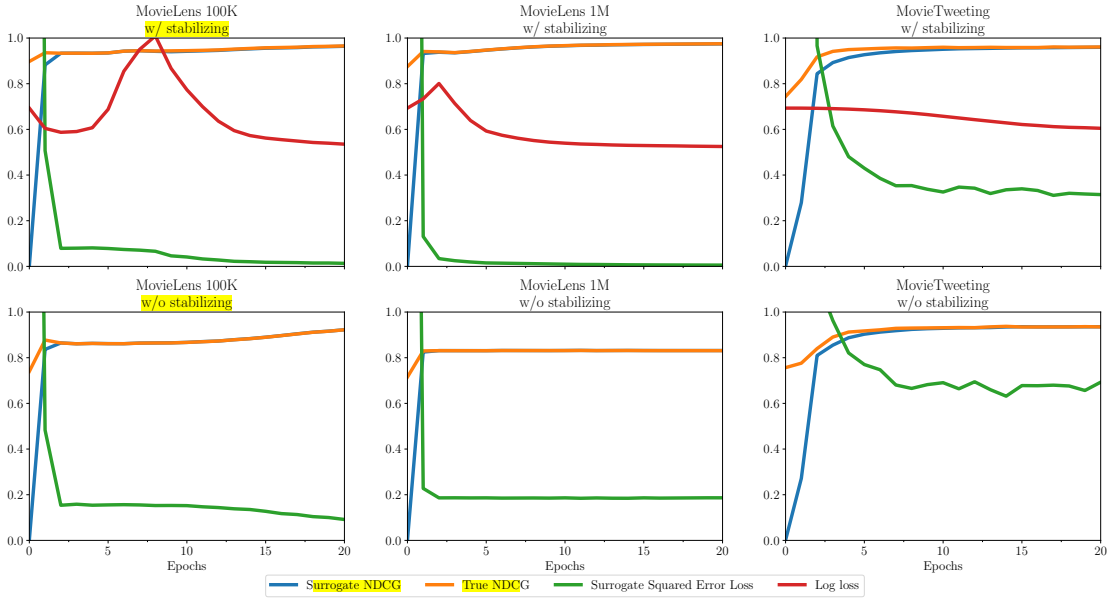

Figure 2: Training Losses Curves

(4) **Pair-wise Logistic Loss** [14]: A pair-wise logistic for learning-to-rank with binary relevance scores.
(5) **Approx-NDCG** [2, 15]: A state-of-the-art list-wise ranking loss that represents a smooth approximation of the NDCG measure.
(6) **Gumbel-NDCG** [1]: A state-of-the-art list-wise ranking loss that further extends and improves the Approx-NDCG approach using stochastic sampling with Gumbel-Max method [11–13]
(7) **NeuralSort** [5]: A state-of-the-art list-wise ranking loss that applies a softmax cross entropy loss on the true and predicted items ranks which are calculated through differentiable sorting networks.

To measure the loss functions robustness against overfitting and to compare it against recent benchmark results, we compare their performance against a non-personalized TopPopular model that recommends the most popular items to everyone and against the NeuMF model [7].

5.3.2 Results. Results in Table 2, show that optimizing the base GraphRec model with GuidedRec while maintaining the logistic loss as a stabilizing loss function achieves statistically significant improvements in the NDCG measure across the three datasets. Interestingly the results also show significant improvements in the secondary HitRatio measure even though we only optimized the NDCG measure. This effect is mainly due to the high correlation

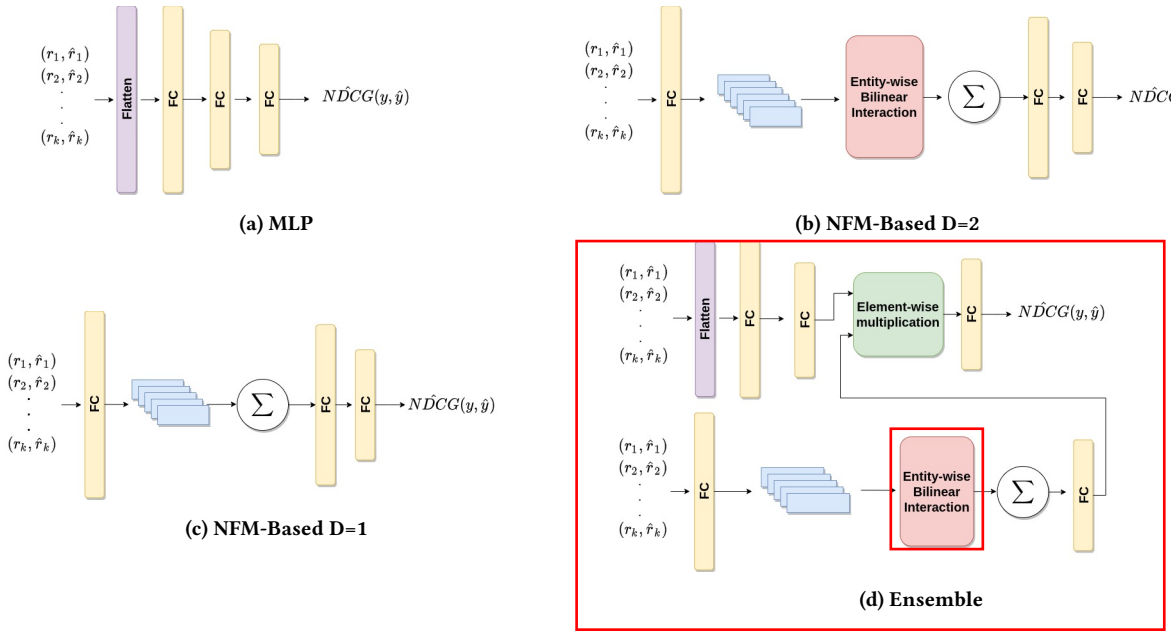

Figure 3: Different candidate architectures for the surrogate loss model

Table 3: Performance comparison between surrogate loss models with leaky and non-leaky gradients on the MovieLens datasets

Loss Function	Gradients	MovieLens 100k HR@10	MovieLens 100k NDCG@10	MovieLens 1M HR@10	MovieLens 1M NDCG@10
Logloss	-	0.628	0.368	**0.681**	**0.413**
GuidedRec w/o logloss	Non-Leaky	0.437	0.241	0.292	0.170
GuidedRec w/o logloss	Leaky	**0.643**	**0.369**	0.676	0.405

between the two measures. Results on the MovieTweeting dataset also shows that nearly all baseline loss functions overfitted on the training set, and they were outperformed by the simple TopPopular model as expected. However, this was not the case with GuidedRec, which indicates that it was more robust, and this allows it to outperform all baselines loss functions and the benchmark baseline models.

On the other hand, results show that optimizing the GuidedRec's surrogate loss model without using a stabilizing loss function as proposed by [4] suffers a severe instability on the MovieLens datasets and the optimization procedure always gets stuck in local optima. This is because using log loss as a stabilizing function allows the surrogate loss model to start learning the true NDCG measure from a relatively good and close starting point to the true loss curve. While without using the log loss as a stabilizing function, the surrogate loss model will start from a possible unfavored random point to the true loss curve, which increases the probability of being stuck in local optima.

These stabilizing effect of using the log loss function in the GuidedRec approach can also be seen clearly in the training loss curves in Figure 2. On the MovieLens datasets, the log loss function first starts to increases in order to counteract the decrease in the surrogate model's squared-error loss until both losses reach a stable point. Once they reach a stable point, both functions start to decrease simultaneously while the true and surrogate NDCGs increase. This counteracting behavior induces a regularizing effect on the surrogate loss model, which allows it to overcome the local optima and achieve higher NDCG values compared to the model version that does not use a log loss functions that are indicated in the second row of Figure 2. The same figure also shows that the counteracting effect is significantly higher on the MovieLens datasets compared with the MovieTweeting dataset, which indicates that the MovieTweeting has a smoother NDCG loss surface.

Further experimental results using different evaluation metrics and threshold values are shown and discussed in section 7.1 in the Appendix due the space constraints.

5.4 Surrogate Loss Model Architecture (RQ2)

To select the best candidate architecture for the surrogate loss model; we conducted a comparative study between performance of the different candidates on the MovieLens 100K dataset. We compared four possible architecture candidates in this study; a simple MLP-based fully connected approach, a permutation invariant neural factorization machine based approach with second-degree

Table 4: Performance comparison between different architectures for the surrogate loss on the MovieLens 100K

Model	HR@10	NDCG@10
MLP	0.641	0.375
NFM-Based D=1	0.647	0.377
NFM-Based D=2	0.646	0.378
Ensemble	**0.661****	**0.386****

Significantly outperforms the best baseline at the: ** 0.01 and * 0.05 levels.

bi-linear interactions between latent features, a permutation invariant neural factorization machine based approach with first-degree feature interactions, and an ensemble of the first two candidates. All architecture candidates are shown in Figure 3

Results in Table 4 show that the permutation invariant NFM-based architectures are slightly better than the MLP-based architecture in terms of HitRatio and NDCG. Results also show that using an ensemble of the MLP architecture and the NFM architecture with second degree features interactions provide the best results across most of the evaluation metrics.

5.5 Learning NDCG measure from scratch using leaky-gradients (RQ3)

To answer the third research question, we compared the performance of using GuidedRec to optimize a GraphRec model with and without alternatively fixing the models' weights in the bi-level optimization phase. We also omitted the log loss from GuidedRec to eliminate any side effects from the stabilizing function. To differentiate between the two options, we indicate the one without fixing models' weights as having a leaky-gradients during the bi-level optimization compared with non-leaky gradients for the one with weights fixation.

Table 3 shows performance comparison on the MovieLens datasets between the two GuidedRec versions and the standard log loss function. Results show that the surrogate loss model with Leaky-gradients is much more stable than the non-leaky version, and it is even slightly better than the log loss function on the MovieLens 100k. On the other hand, it is only slightly worse than the log loss on the MovieLens 1M.

These results indicate that the surrogate loss model of GuidedRec approach can be stabilized by just leaking the gradients in the bi-level optimization. However, the achieved performance is still inferior compared to using a dedicated stabilizing function, such as a log loss along with the non-leaky version. On the other hand, these findings suggest the possibility of using the leaky surrogate loss model for modeling evaluation metrics that are non-differentiable and having no proxy loss functions for stabilizing.

6 CONCLUSION

In this paper, we propose GuidedRec, a guided learning approach that can be applied to any recommender system model that outputs point-wise likelihood scores for user-item pairs. The proposed approach aims to guide the recommender system model to optimize the true NDCG measure by training and optimizing a parallel independent surrogate loss model that is trained to learn the true NDCG measure. We also proposed using the original logistic point-wise loss function to stabilize the learning procedure. Experiments on three real-world recommender system datasets show that optimizing recommender models through the GuidedRec can significantly outperform the state-of-the-art and the widely used proxy loss functions for optimizing the NDCG measure. In future works, we plan to extend this model for other non-differentiable evaluation metrics for item recommendation and learning-to-rank tasks.

ACKNOWLEDGMENTS

This work is co-funded by the industry project "Data-driven Mobility Services" of ISMLL and Volkswagen Financial Services.(https://www.ismll.uni-hildesheim.de/projekte/dna_en.html)

REFERENCES

[1] Sebastian Bruch, Shuguang Han, Michael Bendersky, and Marc Najork. 2020. A Stochastic Treatment of Learning to Rank Scoring Functions. In *Proceedings of the 13th International Conference on Web Search and Data Mining*. 61–69.

[2] Sebastian Bruch, Masrour Zoghi, Michael Bendersky, and Marc Najork. 2019. Revisiting Approximate Metric Optimization in the Age of Deep Neural Networks. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1241–1244.

[3] Simon Dooms, Toon De Pessemier, and Luc Martens. 2013. MovieTweetings: a Movie Rating Dataset Collected From Twitter. In *Workshop on Crowdsourcing and Human Computation for Recommender Systems, CrowdRec at RecSys 2013*.

[4] Josif Grabocka, Randolf Scholz, and Lars Schmidt-Thieme. 2019. Learning Surrogate Losses. *CoRR* abs/1905.10108 (2019). arXiv:1905.10108 http://arxiv.org/abs/1905.10108

[5] Aditya Grover, Eric Wang, Aaron Zweig, and Stefano Ermon. 2019. Stochastic Optimization of Sorting Networks via Continuous Relaxations. In *International Conference on Learning Representations*. https://openreview.net/forum?id=H1eSS3CcKX

[6] Xiangnan He and Tat-Seng Chua. 2017. Neural factorization machines for sparse predictive analytics. In *Proceedings of the 40th International ACM SIGIR conference on Research and Development in Information Retrieval*. ACM, 355–364.

[7] 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.

[8] Kalervo Järvelin and Jaana Kekäläinen. 2002. Cumulated gain-based evaluation of IR techniques. *ACM Transactions on Information Systems (TOIS)* 20, 4 (2002), 422–446.

[9] Dawen Liang, Rahul G Krishnan, Matthew D Hoffman, and Tony Jebara. 2018. Variational autoencoders for collaborative filtering. In *Proceedings of the 2018 World Wide Web Conference*. 689–698.

[10] Ilya Loshchilov and Frank Hutter. 2017. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101* (2017).

[11] R Duncan Luce. 2012. *Individual choice behavior: A theoretical analysis*. Courier Corporation.

[12] Chris J. Maddison, Andriy Mnih, and Yee Whye Teh. 2016. The Concrete Distribution: A Continuous Relaxation of Discrete Random Variables. *CoRR* abs/1611.00712 (2016). arXiv:1611.00712 http://arxiv.org/abs/1611.00712

[13] Chris J Maddison, Daniel Tarlow, and Tom Minka. 2014. A* sampling. In *Advances in Neural Information Processing Systems*. 3086–3094.

[14] Rama Kumar Pasumarthi, Sebastian Bruch, Xuanhui Wang, Cheng Li, Michael Bendersky, Marc Najork, Jan Pfeifer, Nadav Golbandi, Rohan Anil, and Stephan Wolf. 2019. TF-Ranking: Scalable TensorFlow Library for Learning-to-Rank. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (Anchorage, AK). 2970–2978.

[15] Tao Qin, Tie-Yan Liu, and Hang Li. 2010. A general approximation framework for direct optimization of information retrieval measures. *Information retrieval* 13, 4 (2010), 375–397.

[16] Ahmed Rashed, Josif Grabocka, and Lars Schmidt-Thieme. 2019. Attribute-aware non-linear co-embeddings of graph features. In *Proceedings of the 13th ACM Conference on Recommender Systems*. 314–321.

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

[18] Fen Xia, Tie-Yan Liu, Jue Wang, Wensheng Zhang, and Hang Li. 2008. Listwise Approach to Learning to Rank: Theory and Algorithm. In *Proceedings of the 25th International Conference on Machine Learning* (Helsinki, Finland) *(ICML '08)*. Association for Computing Machinery, New York, NY, USA, 1192–1199. https://doi.org/10.1145/1390156.1390306

Table 5: Best found hyper-parameters for GuidedRec

Dataset	v	g	e	h	$\mathcal{L}_{logloss}$		$\mathcal{L}_{NDCG}$		$\mathcal{L}_{Surrogate}$		Epochs
					λ	lr	λ	lr	λ	lr	
ML 100K	[20,20]	[20]	[20]	[8]	1×10^{-5}	1×10^{-3}	1×10^{-6}	1.3×10^{-3}	8×10^{-5}	2×10^{-3}	440
ML 1M	[20,20]	[20]	[20]	[8]	1×10^{-5}	5×10^{-4}	1×10^{-5}	6.5×10^{-4}	7×10^{-7}	1.2×10^{-3}	430
MVT	[20,20]	[20]	[20]	[8]	1×10^{-5}	1×10^{-4}	2×10^{-6}	2×10^{-4}	7×10^{-7}	1.2×10^{-3}	111

Table 6: Runtime comparison between GuidedRec and other proxy loss functions

Loss Function	Average batch runtime in seconds
Logloss	4.185
Gumbel-NDCG [1]	12.451
Approx-NDCG [2, 15]	4.709
ListMLE [18]	4.557
Softmax [14]	4.680
Pair-wise logloss [14]	5.567
Neural-Sort [5]	4.812
GuidedRec w/ logloss	16.525

A SUPPLEMENTARY RESULTS AND EVALUATION METRICS

In this section, we present complete experimental results for the comparative experiment discussed in Section 5.3 using different evaluation metrics and threshold values. Results in Tables 7, 8, and 9, show that optimizing the base GraphRec model with GuidedRec while maintaining the log loss as a stabilizing loss function achieves statistically significant improvements in the NDCG measure across the three datasets regardless of the threshold value K even though we only trained the model on lists of size 10.

B REPRODUCIBILITY OF THE EXPERIMENTS

The source code of GuidedRec is available at our GitHub repository[2]. Regarding the base recommender system models, we used ADAM optimizer with weight decay λ [10]. The batch sizes were 250, 250, 1000 for MovieLens 100K, MovieLens 1M, and MovieTweeting respectively. We tuned the hyper-parameters of all models using a grid-search on the log loss as follows:

- **NeuMF**: We tested prediction factors of [8, 16, 32, 64], layers numbers of [1, 2, 3, 4], ADAM weight decays $\lambda_{logloss}$ that range from 0.00000001 to 0001, and learning rates that range from 0.0001 to 0.002. The best hyper-parameters were prediction layers = [16, 16, 8, 8], learn rate =1×10^{-3}, and weight decay $\lambda_{logloss} = 1 \times 10^{-5}$ for MovieLens 100K; prediction layers = [64, 64, 32, 16], learn rate =5×10^{-4}, and weight decay $\lambda_{logloss} = 1 \times 10^{-5}$ for MovieLens 1M; and prediction layers = [16, 16, 8, 8], learn rate =1×10^{-4}, and weight decay $\lambda_{logloss} = 1 \times 10^{-5}$ for MovieTweeting
- **GraphRec**:We tested different embedding sizes that range from 5 to 150, ADAM weight decays $\lambda_{logloss}$ that range from 0.00000001 to 0001, and learning rates that range from 0.0001 to 0.002. The best hyper-parameters were embedding size = 20, learn rate =1×10^{-3}, and weight decay $\lambda_{logloss} = 1 \times 10^{-5}$ for MovieLens 100K; embedding size = 45, learn rate =5×10^{-4}, and weight decay $\lambda_{logloss} = 1 \times 10^{-5}$ for MovieLens 1M; and embedding size = 10, learn rate =1×10^{-4}, and weight decay $\lambda_{logloss} = 1 \times 10^{-5}$ for MovieTweeting

On the other hand, for tuning the GuidedRec surrogate loss model we tested a different number of layers numbers of [1, 2, 3, 4], with sizes that range from 5 to 150, learning rates that range from 0.0001 to 0.002, β that range from 10 to 1×10^{-12}, ADAM weight decays λ_{NDCG} and $\lambda_{Surrogate}$ that range from 0.00000001 to 0001, and activation functions of [Linear, Leaky_ReLU, Sigmoid, Relu, Tanh, and CReLU]. The best-found hyper-parameters for the GuidedRec approach are indicated in Tables 5. We also used $\beta = 1.0$ and Tanh activation on all datasets.

Finally, regarding the baseline loss functions, we used the best-found hyper-parameters of the base models to optimize them, and their best number of epochs were as follows:

- **Logloss**: The best number of epochs were 320, 156, and 43 for GraphRec on MovieLens 100K, MovieLens 1M and MovieTweeting respectively.
- **Gumbel-NDCG**: The best number of epochs were 464, 190, and 37 for GraphRec on MovieLens 100K, MovieLens 1M, and MovieTweeting respectively.
- **Approx-NDCG**: The best number of epochs were 320, 191, and 18 for GraphRec on MovieLens 100K, MovieLens 1M and MovieTweeting respectively.
- **ListMLE**: The best number of epochs were 44, 9, and 41 for GraphRec on MovieLens 100K, MovieLens 1M, and MovieTweeting respectively.
- **Softmax**: The best number of epochs were 266, 17, and 57 for GraphRec model on MovieLens 100K, MovieLens 1M, and MovieTweeting respectively.
- **Pair-wise logloss**: The best number of epochs were 65, 29, and 100 for GraphRec model on MovieLens 100K, MovieLens 1M, and MovieTweeting respectively.
- **Neural-Sort**: The best number of epochs were 151, 53, and 29 for GraphRec model on MovieLens 100K, MovieLens 1M, and MovieTweeting respectively.

C RUN-TIME COMPARISON

Training an independent surrogate loss model and calculating the true NDCG ground truth values for each batch induce some runtime overhead on top of the average batch runtime. To measure such overhead we compared the average runtime of a batch using all

[2]https://github.com/ahmedrashed-ml/GuidedRec

Table 7: Performance comparison on the MovieLens 100k dataset

Loss Function	Main Metrics		Supplementary Metrics			
	HR@10	NDCG@10	HR@5	HR@20	NDCG@5	NDCG@20
Logloss	0.628	0.368	0.449	0.781	0.310	0.407
Gumbel-NDCG [1]	0.633	0.372	0.456	0.794	0.315	0.413
Approx-NDCG [2, 15]	0.618	0.355	0.443	0.782	0.299	0.397
ListMLE [18]	0.618	0.366	0.454	0.772	0.313	0.405
Softmax [14]	0.620	0.346	0.440	0.796	0.288	0.391
Pair-wise logloss [14]	0.645	0.377	0.460	0.813	0.318	0.420
Neural-Sort [5]	0.645	0.379	0.459	**0.815**	0.318	**0.422**
GuidedRec w/o logloss	0.437	0.241	0.290	0.628	0.193	0.289
GuidedRec w/ logloss	**0.661****	**0.386****	**0.476***	0.803	**0.326***	**0.422**
Benchmark Models						
TopPopular	0.406	0.219	0.257	0.601	0.170	0.268
NeuMF [7]	0.621	0.356	0.446	0.790	0.299	0.398

Significantly outperforms the best baseline at the: ** 0.01 and * 0.05 levels.

Table 8: Performance comparison on the MovieLens 1M dataset

Loss Function	Main Metrics		Supplementary Metrics			
	HR@10	NDCG@10	HR@5	HR@20	NDCG@5	NDCG@20
Logloss	0.681	0.413	0.514	0.824	0.359	0.450
Gumbel-NDCG [1]	0.684	0.410	0.510	0.838	0.354	0.449
Approx-NDCG [2, 15]	0.684	0.409	0.511	0.837	0.352	0.448
ListMLE [18]	0.675	0.401	0.504	0.825	0.345	0.439
Softmax [14]	0.656	0.374	0.473	0.822	0.314	0.416
Pair-wise logloss [14]	0.688	0.409	0.512	**0.845**	0.352	0.449
Neural-Sort [5]	0.663	0.385	0.489	0.815	0.329	0.424
GuidedRec w/o logloss	0.292	0.170	0.212	0.363	0.144	0.188
GuidedRec w/ logloss	**0.695***	**0.422****	**0.524****	0.842	**0.366****	**0.459****
Benchmark Models						
TopPopular	0.467	0.260	0.322	0.650	0.214	0.306
NeuMF [7]	0.660	0.393	0.490	0.824	0.338	0.435

Significantly outperforms the best baseline at the: ** 0.01 and * 0.05 levels.

Table 9: Performance comparison on the extremely sparse MovieTweeting dataset

Loss Function	Main Metrics		Supplementary Metrics			
	HR@10	NDCG@10	HR@5	HR@20	NDCG@5	NDCG@20
Logloss	0.723	0.567	0.647	0.795	0.542	0.585
Gumbel-NDCG [1]	0.722	0.567	0.646	0.795	0.542	0.585
Approx-NDCG [2, 15]	0.714	0.559	0.641	0.789	0.535	0.578
ListMLE [18]	0.723	0.567	0.646	0.796	0.542	0.586
Softmax [14]	0.724	0.568	0.646	0.796	0.542	0.586
Pair-wise logloss [14]	0.724	0.568	0.647	0.797	0.543	0.586
Neural-Sort [5]	0.726	0.567	0.647	0.798	0.541	0.585
GuidedRec w/o logloss	0.724	0.567	0.645	0.795	0.542	0.585
GuidedRec w/ logloss	**0.732****	**0.571****	**0.649****	**0.809****	**0.544****	**0.590****
Benchmark Models						
TopPopular	0.724	0.565	0.643	0.800	0.539	0.585
NeuMF [7]	0.724	0.567	0.646	0.798	0.542	0.586

Significantly outperforms the best baseline at the: ** 0.01 and * 0.05 levels.

loss functions against the GuidedRec approach on the MovieLens 1M dataset. Results in table 6 show that the average batch runtime using GuidedRec is around 3 to 4 times slower than the proxy loss functions which means there will be a slight trade-off between accuracy and average runtime when considering the GuidedRec approach. This experiment was done on a machine with two E5-2670v2 Xeon CPUs and 128 GB of RAM.