Breaking the Top-K Barrier: Advancing Top-K Ranking Metrics Optimization in Recommender Systems

Weiqin Yang[†][‡]
Zhejiang University
Hangzhou, China
tinysnow@zju.edu.cn

Jiawei Chen[*][†][‡][§]
Zhejiang University
Hangzhou, China
sleepyhunt@zju.edu.cn

Shengjia Zhang[†][‡]
Zhejiang University
Hangzhou, China
shengjia.zhang@zju.edu.cn

Peng Wu[¶]
Beijing Technology and Business University
Beijing, China
pengwu@btbu.edu.cn

Yuegang Sun
Intelligence Indeed
Hangzhou, China
bulutuo@i-i.ai

Yan Feng[†][‡]
Zhejiang University
Hangzhou, China
fengyan@zju.edu.cn

Chun Chen[†][‡]
Zhejiang University
Hangzhou, China
chenc@zju.edu.cn

Can Wang[†][§]
Zhejiang University
Hangzhou, China
wcan@zju.edu.cn

Abstract

In the realm of recommender systems (RS), Top-K ranking metrics such as NDCG@K are the gold standard for evaluating recommendation performance. However, during the training of recommendation models, optimizing NDCG@K poses significant challenges due to its inherent discontinuous nature and the intricate Top-K truncation. Recent efforts to optimize NDCG@K have either overlooked the Top-K truncation or suffered from high computational costs and training instability. To overcome these limitations, we propose **SoftmaxLoss@K (SL@K)**, a novel recommendation loss tailored for NDCG@K optimization. Specifically, we integrate the quantile technique to handle Top-K truncation and derive a smooth upper bound for optimizing NDCG@K to address discontinuity. The resulting SL@K loss has several desirable properties, including theoretical guarantees, ease of implementation, computational efficiency, gradient stability, and noise robustness. Extensive experiments on four real-world datasets and three recommendation backbones demonstrate that SL@K outperforms existing losses with a notable average improvement of **6.03%**. The code is available at https://github.com/Tiny-Snow/IR-Benchmark.

[*]Corresponding author.
[†]State Key Laboratory of Blockchain and Data Security, Zhejiang University.
[‡]College of Computer Science and Technology, Zhejiang University.
[§]Hangzhou High-Tech Zone (Binjiang) Institute of Blockchain and Data Security.
[¶]School of Mathematics and Statistics, Beijing Technology and Business University.

CCS Concepts

• **Information systems → Recommender systems.**

Keywords

Recommender systems; Surrogate loss; NDCG@K optimization

ACM Reference Format:
Weiqin Yang, Jiawei Chen, Shengjia Zhang, Peng Wu, Yuegang Sun, Yan Feng, Chun Chen, and Can Wang. 2025. Breaking the Top-K Barrier: Advancing Top-K Ranking Metrics Optimization in Recommender Systems . In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.2 (KDD '25), August 3–7, 2025, Toronto, ON, Canada.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3711896.3736866

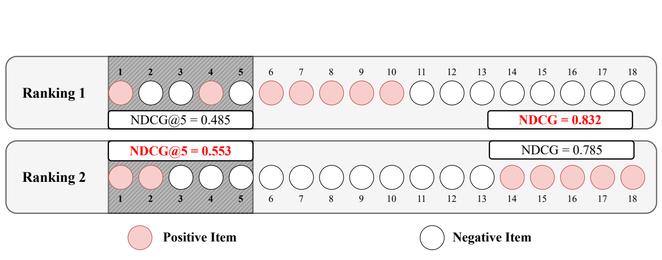
KDD Availability Link:
The source code of this paper has been made publicly available at https://doi.org/10.5281/zenodo.15535932.

1 Introduction

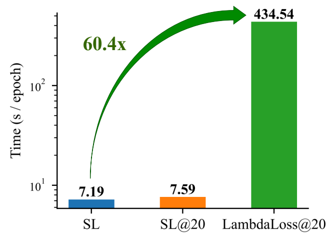
Recommender systems (RS) [11, 16, 17, 34, 61, 63] have been widely applied in various personalized services [45, 53]. The primary goal of RS is to model users' preferences on items and subsequently retrieve a select number of items that users are most likely to interact with [28, 37, 40]. In practice, RS typically display only K top-ranked items to users based on their preference scores. Therefore, *Top-K ranking metrics*, e.g., NDCG@K [25], are commonly used to evaluate recommendation performance. Unlike *full-ranking metrics*, e.g., NDCG [30], which assess the entire ranking list, Top-K ranking metrics focus on the quality of the items ranked within the Top-K positions, making them more aligned with practical requirements.

Challenges. Despite the widespread adoption of the NDCG@K metric, its optimization presents two fundamental challenges:

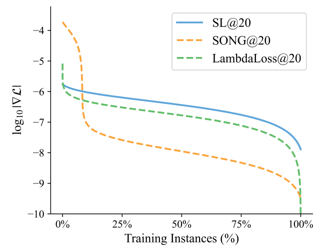
- *Top-K truncation*: NDCG@K involves truncating the ranking list, requiring the identification of which items occupy the Top-K



(a) Inconsistency between NDCG and NDCG@K.



(b) Execution time.



(c) Gradient distribution.

Figure 1: (a) Inconsistency between NDCG and NDCG@K. Ranking 1 and Ranking 2 represent two different ranking lists of the same set of items, where red/white circles denote positive/negative items, respectively. While Ranking 1 has a better NDCG than Ranking 2, it has worse NDCG@5. (b) Execution time comparison. LambdaLoss@K incurs a significantly higher (60 times) computational overhead compared to SL and SL@K on the Electronic dataset (8K items). (c) Gradient distribution comparison. LambdaLoss@K and SONG@K exhibit skewed long-tailed gradient distributions, where top-5% samples contribute over 90% of the overall gradients. In contrast, SL@K achieves a more moderate gradient distribution, where top-5% samples contribute less than 15% of the overall gradients. This leads to better data utilization and training stability.

positions. This necessitates sorting the entire item list, imposing significant computational costs and optimization complexities.

- *Discontinuity*: NDCG@K is inherently discontinuous or flat everywhere in the space of model parameters, which severely impedes the effectiveness of gradient-based optimization methods.

Existing works. Recent studies have introduced two types of NDCG@K surrogate losses to tackle these challenges. However, these approaches still exhibit significant limitations:

- A prominent line of work focuses on *optimizing full-ranking metrics* like NDCG, without accounting for the complex Top-K truncation. Notable among these is Softmax Loss (SL) [69], which serves as an upper bound for optimizing NDCG and demonstrates state-of-the-art performance [4, 62, 68, 71]. Moreover, SL enjoys practical advantages in terms of formulation simplicity and computational efficiency. However, we argue that NDCG is *inconsistent* with NDCG@K — NDCG@K focuses exclusively on a few top-ranked items, while NDCG evaluates the entire ranking list. This discrepancy means that optimizing NDCG does not always yield improvements in NDCG@K and may even lead to performance degradation, as illustrated in Figure 1a. Therefore, without incorporating Top-K truncation, these NDCG surrogate losses could inherently encounter performance bottlenecks.
- Few studies have explored *incorporating Top-K truncation* into NDCG@K optimization. For example, LambdaLoss@K [29] incorporates truncation-aware lambda weights [5, 66] based on ranking positions to optimize NDCG@K, exhibiting superior performance compared to full-ranking surrogate losses like SL [69] and LambdaLoss [66] in learning to rank tasks [40]. Another notable work is SONG@K [51], which employs a ingenious bilevel compositional optimization strategy [64] to optimize NDCG@K with provable guarantees. While these methods have proven effective in other tasks, we find them *ineffective* for recommendation due to the large-scale and sparse nature of RS data. Specifically, LambdaLoss@K requires sorting the entire item list to calculate lambda weights, which is computationally impractical in real-world RS (cf. Figure 1b). Additionally, both LambdaLoss@K and SONG@K exhibit a highly skewed gradient distribution in RS — a few instances dominate the gradients, while the majority contribute negligibly (cf. Figure 1c). This severely hinders effective data utilization and model training.

Our method. Given the critical importance of optimizing NDCG@K and the inherent limitations of existing losses in RS, it is essential to devise a more effective NDCG@K surrogate loss. In this paper, we propose **SoftmaxLoss@K (SL@K)**, incorporating the following two key strategies to address the aforementioned challenges:

- To address the *Top-K truncation* challenge, we employ the *quantile technique* [3, 35]. Specifically, we introduce a Top-K quantile for each user as a threshold score that separates the Top-K items from the remainder. This technique transforms the complex Top-K truncation into a simpler comparison between item scores and quantiles, which circumvents the need for explicit calculations of ranking positions. We further develop a Monte Carlo-based quantile estimation strategy that achieves both computational efficiency and theoretical precision guarantees.
- To overcome the *discontinuity* challenge, we derive an upper bound for optimizing NDCG@K and relax it into a smooth surrogate loss — SL@K. Our analysis proves that SL@K serves as a tight upper bound for $-\log$ NDCG@K, ensuring its theoretical effectiveness in Top-K recommendation.

Beyond its theoretical foundations, SL@K offers several practical advantages: (i) *Ease of implementation*: Compared to SL, SL@K only adds a quantile-based weight for each positive instance, making it easy to implement and integrate into existing RS. (ii) *Computational efficiency*: The adoption of quantile estimation and relaxation techniques incurs minimal additional computational overhead over SL (cf. Figure 1b). (iii) *Gradient stability*: SL@K exhibits more moderate gradient distribution characteristics during training (cf. Figure 1c), promoting effective data utilization and improving model training stability. (iv) *Noise robustness*: SL@K demonstrates enhanced robustness against false positive noise [12, 67], i.e., interactions arising from extraneous factors rather than user preferences.

Finally, to empirically validate the effectiveness of SL@K, we conduct extensive experiments on four real-world recommendation datasets and three typical recommendation backbones. Experimental results demonstrate that SL@K achieves impressive performance improvements of **6.03%** on average. Additional experiments, including an exploration of varying hyperparameter K and robustness evaluations, confirm that SL@K is not only well-aligned with NDCG@K, but also exhibits superior resistance to noise. Moreover, since SL@K is essentially a general ranking loss, it can be seamlessly applied to other information retrieval (IR) tasks. We extend our work to three different IR tasks, including learning to rank (LTR) [49], sequential recommendation (SeqRec) [33], and link prediction (LP) [38]. Empirical results validate the versatility and effectiveness of SL@K across diverse IR tasks.

Contributions. In summary, our contributions are as follows:

- We highlight the significance of optimizing the Top-K ranking metric NDCG@K in recommendation and reveal the limitations of existing losses.
- We propose a novel loss function, SL@K, tailored for Top-K recommendation by integrating the quantile technique and analyzing the upper bound of NDCG@K.
- We conduct extensive experiments on various real-world datasets and backbones, demonstrating the superiority of SL@K over existing losses, achieving an average improvement of 6.03%.
- We extend SL@K to three different IR tasks, validating its versatility and effectiveness beyond conventional recommendation.

2 Preliminaries

In this section, we first present the task formulation (Section 2.1), then highlight the challenges in optimizing NDCG@K (Section 2.2), and finally introduce Softmax Loss (SL) [69] while discussing its limitations in optimizing NDCG@K (Section 2.3).

2.1 Top-K Recommendation

In this work, we focus on the Top-K recommendation from implicit feedback, a widely-used scenario in recommender systems (RS) [60, 74]. Specifically, given an RS with a user set $\mathcal{U}$ and an item set $\mathcal{I}$, let $\mathcal{D} = \{y_{ui} : u \in \mathcal{U}, i \in \mathcal{I}\}$ denote the historical interactions between users and items, where $y_{ui} = 1$ indicates that user u has interacted with item i, and $y_{ui} = 0$ indicates no interaction. For each user u, we denote $\mathcal{P}_u = \{i \in \mathcal{I} : y_{ui} = 1\}$ as the set of positive items, and $\mathcal{N}_u = \mathcal{I} \setminus \mathcal{P}_u$ as the set of negative items. The recommendation task can be formulated as follows: learning user preferences from dataset $\mathcal{D}$ and recommending the Top-K items that users are most likely to interact with.

Formally, modern RS typically infer user preferences for items with a learnable model $s_{ui} = f_\Theta(u, i)$, where $f_\Theta(u, i) : \mathcal{U} \times \mathcal{I} \to \mathbb{R}$ can be any flexible recommendation backbone with parameters Θ, mapping user/item features (e.g., IDs) into their preference scores s_{ui}. Subsequently, the Top-K items with the highest scores s_{ui} are retrieved as recommendations. In this work, we focus not on model architecture design but instead on exploring the recommendation loss. Given that the loss function guides the optimization direction of models, its importance cannot be overemphasized [54].

2.2 NDCG@K Metric

Formulation of NDCG@K. Given the Top-K nature of RS, Top-K ranking metrics have been widely used to evaluate the recommendation performance. This work focuses on the most representative Top-K ranking metric, i.e., NDCG@K (Normalized Discounted Cumulative Gain with Top-K truncation) [25, 30]. Formally, for each user u, NDCG@K can be formulated as follows:

$$\mathrm{NDCG@}K(u) = \frac{\mathrm{DCG@}K(u)}{\mathrm{IDCG@}K(u)}, \ \mathrm{DCG@}K(u) = \sum_{i \in \mathcal{P}_u} \frac{\mathbb{I}(\pi_{ui} \le K)}{\log_2(\pi_{ui} + 1)}, \tag{2.1}$$

where $\mathbb{I}(\cdot)$ is the indicator function, $\pi_{ui} = \sum_{j \in \mathcal{I}} \mathbb{I}(s_{uj} \ge s_{ui})$ is the ranking position of item i for user u, and IDCG@K is a normalizing constant representing the optimal DCG@K with an ideal ranking.

As observed, NDCG@K not only evaluates the number of positive items within the Top-K recommendations (similar to other Top-K metrics, e.g., Recall@K and Precision@K), but also accounts for their ranking positions, i.e., higher-ranked items contribute more to NDCG@K. This makes NDCG@K a more practical metric for recommendation. Therefore, this work focuses on NDCG@K, while we also observe that effectively optimizing NDCG@K can bring improvements on other Top-K metrics like Recall@K (cf. Table 2).

Challenges in optimizing NDCG@K. While NDCG@K is widely applied, directly optimizing it presents significant challenges:

- *Challenge 1: Top-K truncation.* NDCG@K involves truncating the ranking list, as indicated by the term $\mathbb{I}(\pi_{ui} \le K)$ in Equation (2.1). This implies the need to determine whether an item is situated within the Top-K positions. Directly computing this involves sorting all items for each user, which is computationally impractical for RS. Moreover, this truncation introduces highly complex gradient signals, complicating the optimization process.
- *Challenge 2: Discontinuity.* NDCG@K is a discontinuous metric as it incorporates the indicator function and the ranking positions. Furthermore, this metric exhibits flat characteristics across most regions of the parameter space, i.e., the metric remains unchanged with minor perturbations of s_{ui} almost everywhere. This results in the gradient being undefined or vanishing, posing substantial challenges to the effectiveness of existing gradient-based optimization methods [55]. Consequently, a smooth surrogate for NDCG@K is required to facilitate optimization.

2.3 Softmax Loss

Softmax Loss (SL) [69] has achieved remarkable success in RS. Specifically, SL integrates a contrastive learning paradigm [41]. It normalizes the preference scores to a multinomial distribution [9] by Softmax operator, augmenting the scores of positive items as compared to the negative ones [7]. Formally, SL is defined as:

$$\mathcal{L}_{\mathrm{SL}}(u) = -\sum_{i \in \mathcal{P}_u} \log \frac{\exp(s_{ui}/\tau)}{\sum_{j \in \mathcal{I}} \exp(s_{uj}/\tau)} = \sum_{i \in \mathcal{P}_u} \log\left(\sum_{j \in \mathcal{I}} \exp(d_{uij}/\tau)\right), \tag{2.2}$$

where $d_{uij} = s_{uj} - s_{ui}$ is the negative-positive score difference, and τ is a temperature coefficient controlling the sharpness of the Softmax distribution.

Underlying rationale of SL. The success of SL can be attributed to two main aspects: (i) *Theoretical guarantees*: SL has been proven to serve as an upper bound of $-\log$ NDCG [4, 71], ensuring that optimizing SL is consistent with optimizing NDCG, leading to state-of-the-art (SOTA) performance [68]. (ii) *Computational efficiency*: SL does not require accurately calculating the ranking positions, which is time-consuming. In fact, SL can be efficiently estimated through negative sampling [68]. That is, the sum of item j over the entire item set $\mathcal{I}$ in Equation (2.2) can be approximated by sampling a few negative items through uniform [19, 71] or in-batch [32, 69] sampling. These advantages make SL a practical and effective choice for NDCG optimization, demonstrating superior performance and efficiency over other NDCG surrogate methods, including ranking-based (e.g., Smooth-NDCG [10]), Gumbel-based (e.g., NeuralSort [18]), and neural-based (e.g., GuidedRec [52]) methods. Nowadays, SL has been extensively applied in practice, attracting considerable research exploration with a substantial amount of follow-up work.

Limitations of SL. While SL serves as an effective surrogate loss for NDCG, a significant gap remains between NDCG and NDCG@K, which limits its performance. As Figure 1a shows, optimizing NDCG does not consistently improve NDCG@K and sometimes even leads to performance drops. This limitation still exists in more advanced SL-based losses, e.g., AdvInfoNCE [73], BSL [68], and PSL [71]. Therefore, how to bridge this gap and effectively model the Top-K truncation in recommendation loss remains an open challenge.

3 Methodology

To bridge the gap towards NDCG@K optimization, we propose **SoftmaxLoss@K (SL@K)**, a novel NDCG@K surrogate loss. In this section, we first present the derivations and implementation details of SL@K (Section 3.1). Then, we analyze its properties and discuss its advantages over existing losses (Section 3.2).

3.1 Proposed Loss: SoftmaxLoss@K

The primary challenges in optimizing NDCG@K, as discussed in Section 2.2, are the *Top-K truncation* and the *discontinuity*. To tackle these challenges, we introduce the following two techniques.

3.1.1 Quantile-based Top-K Truncation. To address the *Top-K truncation* challenge, we need to estimate the Top-K truncation term $\mathbb{I}(\pi_{ui} \le K)$, which involves estimating the ranking position π_{ui} for each interaction (u, i). However, directly estimating π_{ui} is particularly challenging. Sorting all items for each user to calculate π_{ui} will incur a computational cost of $O(|\mathcal{U}||\mathcal{I}|\log|\mathcal{I}|)$, which is impractical for real-world RS with immense user and item scales.

To overcome this, we borrow the **quantile technique** [20, 35]. Specifically, we introduce a *Top-K quantile* β_u^K for each user u, i.e.,

$$\beta_u^K \coloneqq \inf\{s_{ui} : \pi_{ui} \le K\}. \tag{3.1}$$

This quantile acts as a threshold score that separates the Top-K items from the rest. Specifically, if an item's score is larger than the quantile, i.e., $s_{ui} \ge \beta_u^K$, then item i is Top-K ranked; conversely, $s_{ui} < \beta_u^K$ implies that item i is outside the Top-K positions. Therefore, the Top-K truncation term can be rewritten as:

$$\mathbb{I}(\pi_{ui} \le K) = \mathbb{I}(s_{ui} \ge \beta_u^K). \tag{3.2}$$

This transformation reduces the complex truncation to a simple comparison between the preference score s_{ui} and the quantile β_u^K, thus avoiding the need to directly estimate π_{ui}. This makes the Top-K truncation both computationally efficient and easy to optimize. To handle the complexities of quantile estimation, we further propose a simple Monte Carlo-based quantile estimation strategy in Section 3.1.3, which guarantees both high efficiency and precision.

Notably, while quantile-based techniques have been explored in previous works – e.g., AATP [3] employs quantiles to optimize Top-K accuracy, and SONG@K [51] adopts quantile-related thresholds for bilevel compositional optimization – we adapt this approach specifically for NDCG@K optimization in the context of recommendation. Specifically, we propose a novel tailored recommendation loss and a dedicated quantile estimation strategy, which address the unique challenges in RS. Readers can refer to Sections 3.2.2 and 5 for a detailed comparison between our proposed loss and existing methods, as well as a discussion of their limitations.

3.1.2 Smooth Surrogate for NDCG@K. To tackle the *discontinuity* challenge, we proceed to relax the discontinuous NDCG@K into a smooth surrogate. Specifically, our approach focuses on deriving a smooth upper bound for $-\log \mathrm{DCG@}K$, since optimizing this upper bound is equivalent to lifting NDCG@K[1] [69, 71]. To ensure mathematical well-definedness, we make a simple assumption that DCG@K is non-zero, which is practical in optimization[2].

Upper bound derivation. While several successful examples (e.g., SL) of relaxing full-ranking metric DCG exist as references [66, 69, 71], special care must be taken to account for the differences in DCG@K introduced by the Top-K truncation. Based on the quantile technique and some specific relaxations, we can derive an upper bound for $-\log \mathrm{DCG@}K$ as follows:

$$\begin{aligned}
&-\log \mathrm{DCG@}K(u) \\
&\overset{(3.2)}{=} -\log\left(\sum_{i\in\mathcal{P}_u} \mathbb{I}(s_{ui} \ge \beta_u^K)\frac{1}{\log_2(\pi_{ui}+1)}\right) && (3.3a)\\
&\overset{①}{\le} -\log\left(\sum_{i\in\mathcal{P}_u} \mathbb{I}(s_{ui} \ge \beta_u^K)\frac{1}{\pi_{ui}}\right) && (3.3b)\\
&= -\log\left(\sum_{i\in\mathcal{P}_u} \frac{\mathbb{I}(s_{ui} \ge \beta_u^K)}{H_u^K}\frac{1}{\pi_{ui}}\right) - \log H_u^K && (3.3c)\\
&\overset{②}{\le} \sum_{i\in\mathcal{P}_u} \frac{\mathbb{I}(s_{ui} \ge \beta_u^K)}{H_u^K}\left(-\log\frac{1}{\pi_{ui}}\right) - \log H_u^K && (3.3d)\\
&\overset{③}{\le} \sum_{i\in\mathcal{P}_u} \mathbb{I}(s_{ui} \ge \beta_u^K)\log \pi_{ui}, && (3.3e)
\end{aligned}$$

where $H_u^K = \sum_{i\in\mathcal{P}_u} \mathbb{I}(s_{ui} \ge \beta_u^K)$ is the number of Top-K positive items (a.k.a. hits) for user u. Equation (3.3c) is well-defined since $H_u^K \ge 1$ due to our non-zero assumption[3]. Several important relaxations are applied in Equation (3.3): ① is due to $\log_2(\pi_{ui}+1) \le \pi_{ui}$; ② is due to Jensen's inequality [31]; ③ is due to $H_u^K \ge 1$.

The motivation behind the relaxations ① and ② is to simplify the DCG term $1/\log_2(\pi_{ui}+1)$, which includes the ranking position π_{ui} in the denominator. It is important to note that the ranking position π_{ui} is intricate and challenging to estimate accurately. Retaining π_{ui} in the denominator could exacerbate the optimization

[1]Note that optimizing DCG@K and NDCG@K is equivalent, as the normalization term IDCG@K is a constant.

[2]This assumption is conventional in RS [4, 69, 71]. Note that DCG@$K = 0$ suggests the worst result. During training, the scores of positive instances are rapidly elevated. As a result, there is almost always at least one positive item within the Top-K positions, ensuring DCG@$K > 0$. (cf. Appendix B.2 for empirical validation).

[3]Since DCG@$K > 0$, there is at least one Top-K hit i such that $s_{ui} \ge \beta_u^K$.

difficulty, potentially leading to high estimation errors and numerical instability. SONG@K [51] is a representative example. Although SONG@K utilizes a sophisticated compositional optimization technique [64], it still performs poorly in RS due to its highly skewed gradient distributions (cf. Figure 1c and Table 2). Therefore, we follow the successful paths of SL [69] and PSL [71], aiming to simplify this complex structure. This significantly facilitates gradient-based optimization and supports sampling-based estimation. Moreover, in relaxation ③, we drop the term H_u^K to reduce computational complexity. While retaining this term could potentially lead to improved performance, we empirically find that the gains are marginal, whereas the additional computational overhead is significant.

Furthermore, we can express the above upper bound in terms of the preference scores. Given the Heaviside step function $\delta(x) = \mathbb{I}(x \geq 0)$ [71], recall that $\pi_{ui} = \sum_{j \in \mathcal{I}} \mathbb{I}(s_{uj} \geq s_{ui}) = \sum_{j \in \mathcal{I}} \delta(d_{uij})$, where $d_{uij} = s_{uj} - s_{ui}$, we can rewrite the upper bound (3.3e) as:

$$(3.3e) = \sum_{i \in \mathcal{P}_u} \delta(s_{ui} - \beta_u^K) \cdot \log\left(\sum_{j \in \mathcal{I}} \delta(d_{uij})\right). \tag{3.4}$$

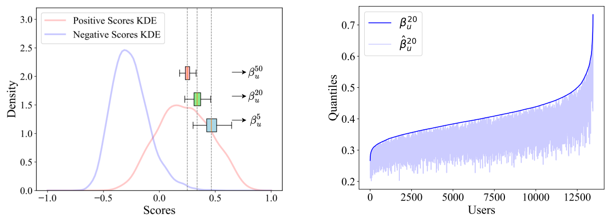
Smoothing Heaviside function. Note that Equation (3.4) is still discontinuous due to the Heaviside step function $\delta(\cdot)$. To address this, following the conventional approach, we approximate $\delta(\cdot)$ by two continuous activation functions $\sigma_w(\cdot)$ and $\sigma_d(\cdot)$, resulting in the following recommendation loss — **SoftmaxLoss@K (SL@K)**:

$$\mathcal{L}_{\mathrm{SL@}K}(u) = \sum_{i \in \mathcal{P}_u} \underbrace{\sigma_w(s_{ui} - \beta_u^K)}_{\text{weight term: } w_{ui}} \cdot \underbrace{\log\left(\sum_{j \in \mathcal{I}} \sigma_d(d_{uij})\right)}_{\text{SL term: } \mathcal{L}_{\mathrm{SL}}(u,i)}. \tag{3.5}$$

To approximate the Heaviside step function $\delta(\cdot)$, two conventional activation functions are widely adopted — the exponential function e^{x/τ_d} and the sigmoid function $1/(1 + e^{-x/\tau_w})$, where τ_d and τ_w are temperature hyperparameters. The exponential function serves as an upper bound of $\delta(\cdot)$ and has been employed in SL, while the sigmoid function provides a tighter approximation of $\delta(\cdot)$ and has been utilized in BPR [54]. Here we select σ_d as exponential and σ_w as sigmoid in Equation (3.5). This configuration guarantees that SL@K serves as a tight upper bound for $-\log$ DCG@K (cf. Theorem 3.2 in Section 3.2). In contrast, if both activations are chosen as sigmoid, the upper bound relation does not hold; if both are chosen as exponential, the bound is not as tight as in our setting. For detailed discussions, please refer to Appendix B.1.

As shown in Equation (3.5), SL@K can be interpreted as a specific *weighted Softmax Loss*, where each positive interaction (u, i) in SL (cf. Equation (2.2)) is assigned a *quantile-based weight* w_{ui}. Intuitively, w_{ui} serves to assign larger weights to positive instances with higher scores s_{ui}, emphasizing those ranked within the Top-K positions during optimization (i.e., those whose scores exceed the quantile). This aligns with the principle of Top-K ranking metrics.

3.1.3 Top-K Quantile Estimation. Now the question lies in how to estimate the Top-K quantile β_u^K efficiently and accurately. While quantile estimation [2, 20, 35] has been extensively studied in the field of statistics, these methods may not be appropriate in our scenarios. Given that the quantile evolves during training and the large scale of item set in RS, SL@K places high demands on estimation efficiency. To address this, our work develops a simple Monte Carlo-based estimation strategy. Specifically, we randomly sample a small set of N items for each user and estimate the Top-K quantile among these sampled items. The computational complexity of this method is $O(|\mathcal{U}|N \log N)$, as it only requires sorting the sampled items, which significantly reduces the computational overhead compared to sorting the entire item set (i.e., $O(|\mathcal{U}||\mathcal{I}| \log |\mathcal{I}|)$).



(a) Quantile distribution. **(b) Quantile estimation.**

Figure 2: (a) Quantile distribution. The distributions of ideal quantiles β_u^{20} and the positive/negative scores are illustrated using Kernel Density Estimation (KDE) [47]. (b) Quantile estimation. The estimated quantile $\hat{\beta}_u^{20}$ and ideal quantile β_u^{20} are illustrated. The estimation error is 0.06 ± 0.03.

Theoretical guarantees. Despite its simplicity, our quantile estimation strategy has theoretical guarantees. To ensure rigor and facilitate generalization to the continuous case, we follow the conventional definition of the *p-th quantile* [2]. In the context of RS, the p-th quantile is exactly the Top-$(1-p)|\mathcal{I}|$ quantile. We have:

Theorem 3.1 (Monte Carlo quantile estimation). *Given the cumulative distribution function (c.d.f.) $F_u(s)$ of the preference scores s_{ui} for user u, for any $p \in (0, 1)$, the p-th quantile is defined as $\theta_u^p \coloneqq F_u^{-1}(p) = \inf\{s : F_u(s) \geq p\}$. In Monte Carlo quantile estimation, we randomly sample N preference scores $\{s_{uj}\}_{j=1}^N \overset{i.i.d.}{\sim} F_u(s)$. The estimated p-th quantile is defined as $\hat{\theta}_u^p \coloneqq \hat{F}_u^{-1}(p)$, where $\hat{F}_u(s) = \frac{1}{N} \sum_{j=1}^N \mathbb{I}(s_{uj} \leq s)$ is the empirical c.d.f. of the sampled scores. Then, for any $\varepsilon > 0$, we have*

$$\Pr\left(\left|\hat{\theta}_u^p - \theta_u^p\right| > \varepsilon\right) \leq 4e^{-2N\delta_\varepsilon^2}, \tag{3.6}$$

where $\delta_\varepsilon = \min\{F_u(\theta_u^p + \varepsilon) - p, p - F_u(\theta_u^p - \varepsilon)\}$. Specifically, in the discrete RS scenarios, the Top-K quantile β_u^K is exactly $\theta_u^{1-K/|\mathcal{I}|}$.

The proof is provided in Appendix C.1. Theorem 3.1 provides the theoretical foundation for sampling-based quantile estimation — the error between the estimated and ideal quantile is bounded by a function that decreases exponentially with the sample size N. This implies that the Top-K quantile β_u^K can be estimated to arbitrary precision given a sufficiently large N.

Practical strategies. In practice, our Monte Carlo-based quantile estimation strategy can be further improved by leveraging the properties of RS. As shown in Figure 2a, the scores of positive items are typically much higher than those of negative items, and the Top-K quantile is often located within the range of positive item scores. Therefore, it is more effective to retain all positive instances and randomly sample a small set of negative instances for quantile estimation. This strategy, though simple, yields more accurate results. Figure 2b provides an example of estimated quantiles across users on the Electronic dataset, with a sample size of $N = 1000$. The estimated quantile $\hat{\beta}_u^{20}$ closely matches the ideal quantile β_u^{20}, with an average deviation of only 0.06. Further analyses and results can be found in Appendix C. The overall optimization process for SL@K is also summarized in Algorithm C.1.

3.2 Analyses of SL@K

3.2.1 Properties of SL@K. Our proposed SL@K offers several desirable properties (P), as summarized below:

(P1) Theoretical guarantees. We establish a theoretical connection between SL@K and NDCG@K as follows:

Theorem 3.2 (NDCG@K surrogate). *For any user u, if the Top-K hits $H_u^K > 1$*[4]*, SL@K serves as an upper bound of* $-\log \mathrm{DCG}@K$, *i.e.,*

$$-\log \mathrm{DCG}@K(u) \leq \mathcal{L}_{\mathrm{SL}@K}(u). \tag{3.7}$$

When the Top-K hits $H_u^K = 1$, a marginally looser yet effective bound holds, i.e., $-\frac{1}{2}\log \mathrm{DCG}@K(u) \leq \mathcal{L}_{\mathrm{SL}@K}(u)$.

The proof is provided in Appendix B.2. Theorem 3.2 reveals that minimizing SL@K leads to improved NDCG@K, ensuring the theoretical effectiveness of SL@K in Top-K recommendation.

(P2) Ease of implementation. Compared to SL, SL@K introduces only a quantile-based weight w_{ui}. Given the widespread adoption of SL in RS, SL@K can be seamlessly integrated into existing recommendation frameworks with minimal modifications.

(P3) Computational efficiency. The utilization of the Monte Carlo strategy for quantile estimation in SL@K (cf. Section 3.1.3) ensures computational efficiency. The conventional SL has a time complexity of $O(|\mathcal{U}|\bar{P}N)$, where $\bar{P}$ denotes the average number of positive items per user, and N denotes the sample size satisfying $N \ll |\mathcal{I}|$. Compared to SL, SL@K only introduces an additional complexity of $O(|\mathcal{U}|N \log N)$ for quantile estimation, which is typically negligible in practice (cf. Figure 1b).

(P4) Gradient stability. SL@K exhibits a moderate gradient distribution comparable to that of SL (cf. Figure 1c), which contributes to its training stability and data utilization effectiveness. This property is mainly attributed to the bounded weight $w_{ui} \in (0.1, 1)$ with sigmoid temperature $\tau_w \geq 1$, thus not significantly amplifying gradient variance. In contrast, other NDCG@K surrogate losses, including LambdaLoss@K [29] and SONG@K [51], are usually hindered by the excessively long-tailed gradients (cf. Figure 1c).

(P5) Noise robustness. *False positive noise* [12] is prevalent in RS, arising from various factors such as clickbait [65], item position bias [27], or accidental interactions [1]. Recent studies have shown that such noise can significantly mislead model training and degrade performance [67]. Interestingly, the introduction of weight w_{ui} in SL@K helps mitigate this issue. In fact, the false positives, which often resemble negative instances, tend to have lower preference scores s_{ui} than the true positives. As a result, these noisy instances typically receive smaller weights w_{ui} (which are positively correlated with s_{ui}) and contribute less in model training. This enhances the model's robustness against false positive noise, as demonstrated in the gradient analysis in Appendix B.3.

[4]The assumption $H_u^K > 1$ is commonly satisfied in practice, as the training process tends to increase the scores of positive items, making them typically larger than those of negative items. Appendix B.2 provides further empirical validation.

Table 1: Dataset statistics. Refer to Appendix D.1 for details.

Dataset	#Users	#Items	#Interactions	Density
Health	1,974	1,200	48,189	0.02034
Electronic	13,455	8,360	234,521	0.00208
Gowalla	29,858	40,988	1,027,464	0.00084
Book	135,109	115,172	4,042,382	0.00026

3.2.2 Comparison with Existing Losses. In this subsection, we delve into the connections and differences between SL@K and other closely related losses to provide further insights:

SL@K vs. Softmax Loss (SL). As discussed in Section 3.1.2, SL@K can be viewed as a specific weighted SL [69]. Although SL demonstrates theoretical advantages due to its close connection with NDCG, as well as practical benefits such as concise formulation and computational efficiency, it does not account for the Top-K truncation. Our SL@K bridges this gap by accompanying each term of SL with a quantile-based weight. As such, SL@K inherits the advantages of SL, while introducing additional merits, e.g., theoretical connections to NDCG@K and robustness to false positive noise.

SL@K vs. LambdaLoss@K and SONG@K. LambdaLoss@K [29] and SONG@K [51] take into account the Top-K truncation and have shown promising results in other fields like document retrieval [40]. However, we find that their effectiveness in RS is compromised, particularly given the large item space and sparse interactions. Specifically, both of them suffer from the issue of long-tailed gradients due to their inherent design. The gradients are dominated by a few instances, while the majority of instances have negligible contributions, which may lead to data utilization inefficiency and optimization instability. In contrast, SL@K exhibits moderate gradients by leveraging the quantile technique and appropriate relaxations, which addresses these issues and achieves superior performance (cf. Figure 1c).

Beyond the gradient instability, LambdaLoss@K also faces additional challenges on computational efficiency. Specifically, it requires calculating exact rankings, which is computationally impractical in RS. Even worse, the skewed gradient distribution hinders the sampling-based strategy to reduce computational overhead, since the gradients of sampled instances may be either vanishingly small or excessively large, leading to unstable optimization. In contrast, SL@K is both theoretically sound and computationally efficient, making it a more suitable choice for RS.

Notably, while SONG@K also employs a threshold to tackle Top-K truncation similar to SL@K's quantile, SL@K differs significantly from SONG@K in two aspects: (i) we follow the successful approaches of SL [69] and PSL [71] to simplify and smooth NDCG@K, which facilitates sampling-based estimation and optimization, while SONG@K employs a compositional optimization technique, which may not be effective in RS. (ii) we employ a simple sampling-based strategy to estimate the threshold (quantile) with theoretical guarantees, as opposed to the complex bilevel optimization in SONG@K. These differences contribute to the significant superiority of SL@K over SONG@K in terms of both recommendation performance and practical applicability. Appendix A gives a detailed discussion on these two losses.

Table 2: Top-20 recommendation performance comparison of SL@K with existing losses. The best results are highlighted in bold, and the best baselines are underlined. "Imp." denotes the improvement of SL@K over the best baseline.

Backbone	Loss	Health		Electronic		Gowalla		Book	
		Recall@20	NDCG@20	Recall@20	NDCG@20	Recall@20	NDCG@20	Recall@20	NDCG@20
MF	BPR	0.1627	0.1234	0.0816	0.0527	0.1355	0.1111	0.0665	0.0453
	GuidedRec	0.1568	0.1093	0.0644	0.0385	0.1135	0.0863	0.0518	0.0361
	SONG@20	0.0874	0.0650	0.0708	0.0444	0.1237	0.0970	0.0747	0.0542
	LLPAUC	0.1644	0.1209	0.0821	0.0499	0.1610	0.1189	0.1150	0.0811
	SL	0.1719	0.1261	0.0821	0.0529	0.2064	0.1624	0.1559	0.1210
	AdvInfoNCE	0.1659	0.1237	0.0829	0.0527	0.2067	0.1627	0.1557	0.1172
	BSL	0.1719	0.1261	0.0834	0.0530	0.2071	0.1630	0.1563	0.1212
	PSL	0.1718	0.1268	0.0838	0.0541	0.2089	0.1647	0.1569	0.1227
	SL@20 (Ours)	**0.1823**	**0.1390**	**0.0901**	**0.0590**	**0.2121**	**0.1709**	**0.1612**	**0.1269**
	Imp. %	**+6.05%**	**+9.62%**	**+7.52%**	**+9.06%**	**+1.53%**	**+3.76%**	**+2.74%**	**+3.42%**
LightGCN	BPR	0.1618	0.1203	0.0813	0.0524	0.1745	0.1402	0.0984	0.0678
	GuidedRec	0.1550	0.1073	0.0657	0.0393	0.0921	0.0686	0.0468	0.0310
	SONG@20	0.1353	0.0960	0.0816	0.0511	0.1261	0.0968	0.0820	0.0573
	LLPAUC	0.1685	0.1207	0.0831	0.0507	0.1616	0.1192	0.1147	0.0810
	SL	0.1691	0.1235	0.0823	0.0526	0.2068	0.1628	0.1567	0.1220
	AdvInfoNCE	0.1706	0.1264	0.0823	0.0528	0.2066	0.1625	0.1568	0.1177
	BSL	0.1691	0.1236	0.0823	0.0526	0.2069	0.1628	0.1568	0.1220
	PSL	0.1701	0.1270	0.0830	0.0536	0.2086	0.1648	0.1575	0.1233
	SL@20 (Ours)	**0.1783**	**0.1371**	**0.0903**	**0.0591**	**0.2128**	**0.1729**	**0.1625**	**0.1280**
	Imp. %	**+4.51%**	**+7.95%**	**+8.66%**	**+10.26%**	**+2.01%**	**+4.92%**	**+3.17%**	**+3.81%**
XSimGCL	BPR	0.1496	0.1108	0.0777	0.0508	0.1966	0.1570	0.1269	0.0905
	GuidedRec	0.1539	0.1088	0.0760	0.0473	0.1685	0.1277	0.1275	0.0951
	SONG@20	0.1378	0.0948	0.0525	0.0320	0.1367	0.0985	0.1281	0.0964
	LLPAUC	0.1519	0.1083	0.0781	0.0481	0.1632	0.1200	0.1363	0.1008
	SL	0.1534	0.1113	0.0772	0.0490	0.2005	0.1570	0.1549	0.1207
	AdvInfoNCE	0.1499	0.1072	0.0776	0.0489	0.2010	0.1564	0.1568	0.1179
	BSL	0.1649	0.1201	0.0800	0.0507	0.2037	0.1597	0.1550	0.1207
	PSL	0.1579	0.1143	0.0801	0.0507	0.2037	0.1593	0.1571	0.1228
	SL@20 (Ours)	**0.1753**	**0.1332**	**0.0869**	**0.0571**	**0.2095**	**0.1717**	**0.1624**	**0.1277**
	Imp. %	**+6.31%**	**+10.91%**	**+8.49%**	**+12.40%**	**+2.85%**	**+7.51%**	**+3.37%**	**+3.99%**

4 Experiments

We aim to answer the following research questions (RQs):

- **RQ1**: How does SL@K perform compared with existing losses?
- **RQ2**: Does SL@K exhibit consistent improvements across different NDCG@K metrics with varying K?
- **RQ3**: Does SL@K exhibit robustness against false positive noise?
- **RQ4**: Can SL@K be effectively applied to other information retrieval (IR) tasks?

4.1 Experimental Setup

Datasets. To ensure fair comparisons, our experimental setup closely follows the prior work of Wu et al. [68] and Yang et al. [71]. We conduct experiments on four widely-used datasets: Health [22, 44], Electronic [22, 44], Gowalla [14], and Book [22, 44]. Additionally, given the inefficiency of LambdaLoss@K [29] in handling these large datasets, we further evaluate its performance on two relatively smaller datasets, i.e., MovieLens [21] and Food [43]. Refer to Table 1 and Appendix D.1 for detailed dataset descriptions.

Recommendation backbones. Following the settings in Yang et al. [71], we evaluate the proposed losses on three backbones: MF [36] (classic Matrix Factorization model), LightGCN [24] (SOTA graph-based model), and XSimGCL [72] (SOTA contrastive-based model). The implementation details can be found in Appendix D.3.

Baseline losses. We compare SL@K with the following baselines: (i) Pairwise loss (BPR [54]); (ii) NDCG surrogate losses (GuidedRec [52] and Softmax Loss (SL) [69]); (iii) NDCG@K surrogate losses (LambdaLoss@K [29] and SONG@K [51]); (iv) Partial AUC surrogate loss (LLPAUC [59]); (v) Advanced SL-based losses (AdvInfoNCE [73], BSL [68], and PSL [71]). Refer to Appendix D.4 for details.

Hyperparameter settings. For fair comparisons, SL@K adopts the same temperature parameter τ_d as the optimal τ in SL. SL@K also uses the same negative sampling strategy as SL for both training and quantile estimation with sample size $N = 1000$. For all baselines, we follow the hyperparameter settings provided in original papers and further tune them to achieve the best performance. We provide the details in Appendix D.4, optimal hyperparameters in Appendix D.5, and supplementary results in Appendix E.

Information Retrieval Tasks. To extend SL@K to other fields, we adapt it to three different IR tasks: (i) **Learning to rank (LTR)**, aiming to order a list of candidate items according to their relevance to a given query; (ii) **Sequential recommendation (SeqRec)**,

Table 3: NDCG@K (D@K) comparisons with varying K on Health and Electronic datasets and MF backbone. The best results are highlighted in bold, and the best baselines are underlined. "Imp." denotes the improvement of SL@K over the best baseline.

Method	Health						Electronic					
	D@5	**D@10**	**D@20**	**D@50**	**D@75**	**D@100**	**D@5**	**D@10**	**D@20**	**D@50**	**D@75**	**D@100**
BPR	0.0940	0.1037	0.1234	0.1621	0.1804	0.1925	0.0345	0.0419	0.0527	0.0690	0.0777	0.0845
GuidedRec	0.0769	0.0881	0.1093	0.1484	0.1671	0.1811	0.0228	0.0294	0.0385	0.0551	0.0635	0.0703
SONG	0.0353	0.0392	0.0488	0.0709	0.0834	0.0930	0.0316	0.0393	0.0493	0.0661	0.0744	0.0803
SONG@K	0.0503	0.0535	0.0650	0.0896	0.1037	0.1135	0.0276	0.0349	0.0444	0.0581	0.0651	0.0706
LLPAUC	0.0887	0.0996	0.1209	0.1592	0.1765	0.1892	0.0305	0.0388	0.0499	0.0686	0.0778	0.0848
SL	0.0922	0.1037	0.1261	0.1620	0.1791	0.1924	0.0353	0.0430	0.0529	0.0696	0.0783	0.0845
AdvInfoNCE	0.0926	0.1038	0.1237	0.1608	0.1789	0.1920	0.0341	0.0423	0.0527	0.0697	0.0782	0.0843
BSL	0.0922	0.1037	0.1261	0.1620	0.1791	0.1924	0.0344	0.0425	0.0530	0.0691	0.0776	0.0843
PSL	0.0940	0.1048	0.1268	0.1613	0.1789	0.1912	0.0356	0.0434	0.0541	0.0700	0.0784	0.0845
SL@K (Ours)	**0.1080**	**0.1190**	**0.1390**	**0.1736**	**0.1916**	**0.2035**	**0.0402**	**0.0484**	**0.0590**	**0.0760**	**0.0844**	**0.0908**
Imp. %	**+14.89%**	**+13.55%**	**+9.62%**	**+7.09%**	**+6.21%**	**+5.71%**	**+12.92%**	**+11.52%**	**+9.06%**	**+8.57%**	**+7.65%**	**+7.08%**

Table 4: Performance exploration of SL@K on NDCG@K' with varying K and K'. The best results are highlighted in bold.

SL@K	Health						Electronic					
	D@5	**D@10**	**D@20**	**D@50**	**D@75**	**D@100**	**D@5**	**D@10**	**D@20**	**D@50**	**D@75**	**D@100**
SL@5	**0.1080**	0.1180	0.1379	0.1724	0.1906	0.2032	**0.0402**	0.0480	0.0583	0.0753	0.0839	0.0900
SL@10	0.1077	**0.1190**	0.1377	0.1734	0.1909	0.2028	0.0400	**0.0484**	0.0583	0.0755	0.0839	0.0901
SL@20	0.1076	0.1188	**0.1390**	0.1733	0.1909	0.2029	0.0400	0.0483	**0.0590**	0.0759	0.0837	0.0900
SL@50	0.1062	0.1167	0.1364	**0.1736**	0.1901	0.2020	0.0398	0.0481	0.0587	**0.0760**	0.0842	0.0907
SL@75	0.1073	0.1179	0.1387	0.1734	**0.1916**	0.2031	0.0397	0.0481	0.0587	0.0759	**0.0844**	0.0907
SL@100	0.1071	0.1177	0.1375	0.1727	0.1904	**0.2035**	0.0399	0.0481	0.0587	0.0759	0.0843	**0.0908**
SL (@∞)	0.0922	0.1037	0.1261	0.1620	0.1791	0.1924	0.0353	0.0430	0.0529	0.0696	0.0783	0.0845

focusing on next item prediction in a user's interaction sequence; and (iii) **Link prediction (LP)**, predicting links between two nodes in a graph. We closely follow the experimental settings in prior work [33, 38, 49] and provide the details in Appendix D.6.

4.2 Performance Comparison

SL@K vs. Baselines (RQ1). Table 2 presents the performance comparison of SL@K against existing losses. As shown, SL@K consistently outperforms all competing losses across various datasets and backbones. The improvements are substantial, with an average increase of 6.03% over the best baselines. This improvement can be attributed to the closer alignment of SL@K with NDCG@K, highlighting the importance of explicitly modeling Top-K truncation during optimization, as opposed to NDCG surrogate losses. Notably, SL@K also demonstrates strong performance on Recall@K. This is because optimizing NDCG@K naturally increases the positive hits in Top-K positions, thereby enhancing Recall@K performance.

SL@K vs. NDCG@K surrogate losses (RQ1). We further compare SL@K with existing NDCG@K surrogate losses, i.e., SONG@K and LambdaLoss@K, in Tables 2 and 17. Although these losses are also designed to optimize NDCG@K, our experiments show that SL@K consistently outperforms them, with significant improvements of over 70% and 13% in NDCG@20 compared to SONG@K and LambdaLoss@K, respectively. The unsatisfactory performance of these surrogate losses can be attributed to their unstable and ineffective optimization process, as discussed in Section 3.2.2. Moreover, LambdaLoss@K incurs significantly higher computational costs compared to SL@K. While sampling strategies could be employed to accelerate LambdaLoss@K (i.e., LambdaLoss@K-S in Table 17), they lead to substantial performance degradation (over 30%).

NDCG@K performance with varying K (RQ2). Table 3 illustrates the NDCG@K performance across different values of K. Experimental results show that SL@K consistently outperforms the baseline methods in all NDCG@K metrics. We also observe that as K increases, the magnitude of the improvements decreases, which aligns with our intuition. Specifically, the Top-K truncation has a greater impact when K is small. As K increases, NDCG@K degrades to the full-ranking metric NDCG. Consequently, the advantage of optimizing for NDCG@K diminishes as K grows.

Top-K recommendation consistency (RQ2). Table 4 presents the performance of NDCG@K' for SL@K with varying values of K, K' in $\{5, 10, 20, 50, 75, 100\}$. We observe that the best NDCG@K' performance is always achieved when $K' = K$ in SL@K. This consistency aligns with our theoretical analysis in Section 3.2.1, i.e., SL@K is oriented towards optimizing NDCG@K rather than other NDCG@K' when $K \neq K'$. For instance, SL@20 achieves the best NDCG@20 performance, but its performance on NDCG@50 is lower compared to SL@50. Nonetheless, SL@K always outperforms SL(@∞), emphasizing the effectiveness of SL@K in real-world RS.

Noise Robustness (RQ3). In Figure 3, we assess the robustness of SL@K to false positive instances. Following Wu et al. [68], we manually introduce a certain ratio of negative instances as noisy positive instances during training. As the noise ratio increases, SL@K demonstrates greater improvements over SL (up to 24%),

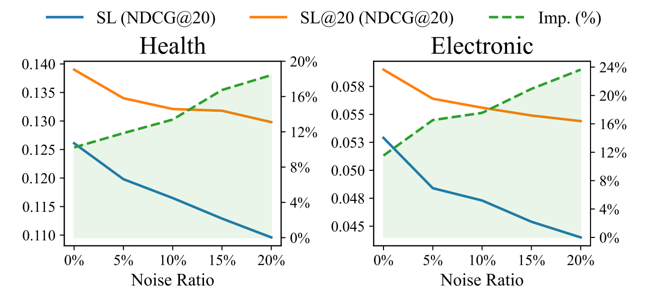


Figure 3: NDCG@20 performance of SL@K compared with SL under varying ratios of imposed false positive instances.

Table 5: LTR results on WEB10K, WEB30K [50], and Istella [15] datasets (metrics: NDCG@5).

Loss	WEB10K	WEB30K	Istella
ListMLE [70]	0.4145	0.4433	0.5671
ListNet [8]	0.4225	<u>0.4594</u>	0.6290
RankNet [6]	0.4253	0.4426	0.6189
LambdaLoss@5 [29]	0.4320	0.4496	0.5860
NeuralNDCG [49]	<u>0.4338</u>	0.4524	0.5823
SL [69]	0.4310	0.4552	<u>0.6327</u>
SL@5 (Ours)	**0.4633**	**0.4895**	**0.6412**
Imp. %	**+6.80%**	**+6.55%**	**+1.34%**

indicating superior robustness to false positive noise. This finding is consistent with our analysis in Section 3.2.1.

Application to other IR tasks (RQ4). We adapt SL@K to three different IR tasks: LTR (Table 5), SeqRec (Table 6), and LP (Table 7). Results show that SL@K consistently outperforms baseline ranking losses (e.g., LambdaLoss@K [29] and NeuralNDCG [49]) and classification losses (e.g., BCE [33] and SL [69]) across all tasks, demonstrating its versatility for general IR applications.

5 Related Work

Recommendation losses. Recommendation losses play a vital role in recommendation models optimization. The earliest works treat recommendation as a simple regression or binary classification problem, utilizing pointwise losses such as MSE [23] and BCE [26]. However, due to neglecting the ranking essence in RS, these pointwise losses usually result in inferior performance. To address this, pairwise losses such as BPR [39, 54] have been proposed. BPR aims to learn a partial order among items and serves as a surrogate for AUC. Following BPR, Softmax Loss (SL) [69] extends the pairwise ranking to listwise by introducing the Plackett-Luce models [42, 48] or contrastive learning principles [13, 46]. SL has been proven to be an NDCG surrogate and achieves SOTA performance [4, 71].

Recent works have further improved ranking losses from various approaches. For example, robustness enhancements to SL have been explored via Distributionally Robust Optimization (DRO) [57], as seen in AdvInfoNCE [73], BSL [68] and PSL [71]. Other approaches directly optimize NDCG, including LambdaRank [5], LambdaLoss [66], SONG [51], and PSL [71]. There are also works focusing on alternative surrogate approaches for NDCG, including ranking-based [10], Gumbel-based [18], and neural-based [52] methods.

Table 6: SeqRec results on Beauty and Games [22, 44] datasets.

Loss	Beauty		Games	
	Hit@20	NDCG@20	Hit@20	NDCG@20
BCE [33]	0.1130	0.0484	0.1577	0.0671
SL [69]	<u>0.1578</u>	<u>0.0766</u>	<u>0.2243</u>	<u>0.1024</u>
SL@20 (Ours)	**0.1586**	**0.0780**	**0.2283**	**0.1045**
Imp. %	**+0.51%**	**+1.82%**	**+1.78%**	**+2.05%**

Table 7: LP results on Cora and Citeseer [56] datasets.

Loss	Cora		Citeseer	
	Hit@20	MRR	Hit@20	MRR
BCE [33]	0.3643	0.1482	0.3560	0.1424
SL [69]	<u>0.4668</u>	<u>0.1772</u>	<u>0.4989</u>	<u>0.1942</u>
SL@20 (Ours)	**0.4839**	**0.1812**	**0.5099**	**0.1963**
Imp. %	**+3.65%**	**+2.25%**	**+2.20%**	**+1.08%**

Despite recent advancements, most ranking losses struggle in practical Top-K recommendation, where only the top-ranked items are retrieved. Losses ignoring Top-K truncation may face performance bottlenecks. To address this, LambdaLoss@K and SONG@K optimize NDCG@K using elegant lambda weights and compositional optimization, respectively, but their performance in RS remains unsatisfactory, as discussed in Section 4.2. Other methods, such as AATP [3], LLPAUC [59], and OPAUC [58], target metrics like Precision@K and Recall@K, yet their theoretical connections to NDCG@K remain unclear. While AATP employs a quantile technique, it lacks a theoretical foundation and suffers from inefficiency issues, making it impractical for RS. LLPAUC and OPAUC rely on complex adversarial training, potentially limiting their effectiveness and applicability.

6 Conclusion and Future Directions

This work introduces SoftmaxLoss@K (SL@K), a novel recommendation loss tailored for optimizing NDCG@K. SL@K employs a quantile-based technique to address the Top-K truncation challenge and derives a smooth approximation to tackle the discontinuity issue. We establish a tight bound between SL@K and NDCG@K, demonstrating its theoretical effectiveness. Beyond its theoretical soundness, SL@K offers a concise form, introducing only quantile-based weights atop the conventional Softmax Loss, making it both easy to implement and computationally efficient.

Looking ahead, a promising direction is to develop incremental quantile estimation methods to further improve the efficiency of SL@K and enable incremental learning in RS. Additionally, since Top-K metrics are widely used, further exploring the application of SL@K in other domains, such as multimedia retrieval, question answering, and anomaly detection, is valuable.

Acknowledgments

This work is supported by the Zhejiang Province "JianBingLingYan+X" Research and Development Plan (2025C02020).

References

[1] Panagiotis Adamopoulos and Alexander Tuzhilin. 2014. On unexpectedness in recommender systems: Or how to better expect the unexpected. *ACM Transactions on Intelligent Systems and Technology (TIST)* 5, 4 (2014), 1–32.

[2] Peter J Bickel and Kjell A Doksum. 2015. *Mathematical statistics: basic ideas and selected topics, volumes I-II package.* Chapman and Hall/CRC.

[3] Stephen Boyd, Corinna Cortes, Mehryar Mohri, and Ana Radovanovic. 2012. Accuracy at the top. *Advances in neural information processing systems* 25 (2012).

[4] Sebastian Bruch, Xuanhui Wang, Michael Bendersky, and Marc Najork. 2019. An analysis of the softmax cross entropy loss for learning-to-rank with binary relevance. In *Proceedings of the 2019 ACM SIGIR international conference on theory of information retrieval.* 75–78.

[5] Christopher Burges, Robert Ragno, and Quoc Le. 2006. Learning to rank with nonsmooth cost functions. *Advances in neural information processing systems* 19 (2006).

[6] Chris Burges, Tal Shaked, Erin Renshaw, Ari Lazier, Matt Deeds, Nicole Hamilton, and Greg Hullender. 2005. Learning to rank using gradient descent. In *Proceedings of the 22nd international conference on Machine learning.* 89–96.

[7] 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.

[8] 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.

[9] George Casella and Roger Berger. 2024. *Statistical inference.* CRC Press.

[10] Olivier Chapelle and Mingrui Wu. 2010. Gradient descent optimization of smoothed information retrieval metrics. *Information retrieval* 13 (2010), 216–235.

[11] Jiawei Chen, Hande Dong, Yang Qiu, Xiangnan He, Xin Xin, Liang Chen, Guli Lin, and Keping Yang. 2021. AutoDebias: Learning to debias for recommendation. In *Proceedings of the 44th international ACM SIGIR conference on research and development in information retrieval.* 21–30.

[12] Jiawei Chen, Hande Dong, Xiang Wang, Fuli Feng, Meng Wang, and Xiangnan He. 2023. Bias and debias in recommender system: A survey and future directions. *ACM Transactions on Information Systems* 41, 3 (2023), 1–39.

[13] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. 2020. A simple framework for contrastive learning of visual representations. In *International conference on machine learning.* PMLR, 1597–1607.

[14] Eunjoon Cho, Seth A Myers, and Jure Leskovec. 2011. Friendship and mobility: user movement in location-based social networks. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining.* 1082–1090.

[15] Domenico Dato, Claudio Lucchese, Franco Maria Nardini, Salvatore Orlando, Raffaele Perego, Nicola Tonellotto, and Rossano Venturini. 2016. Fast ranking with additive ensembles of oblivious and non-oblivious regression trees. *ACM Transactions on Information Systems (TOIS)* 35, 2 (2016), 1–31.

[16] Chongming Gao, Kexin Huang, Jiawei Chen, Yuan Zhang, Biao Li, Peng Jiang, Shiqi Wang, Zhong Zhang, and Xiangnan He. 2023. Alleviating matthew effect of offline reinforcement learning in interactive recommendation. In *Proceedings of the 46th international ACM SIGIR conference on research and development in information retrieval.* 238–248.

[17] Chongming Gao, Shiqi Wang, Shijun Li, Jiawei Chen, Xiangnan He, Wenqiang Lei, Biao Li, Yuan Zhang, and Peng Jiang. 2023. CIRS: Bursting filter bubbles by counterfactual interactive recommender system. *ACM Transactions on Information Systems* 42, 1 (2023), 1–27.

[18] Aditya Grover, Eric Wang, Aaron Zweig, and Stefano Ermon. 2019. Stochastic optimization of sorting networks via continuous relaxations. *arXiv preprint arXiv:1903.08850* (2019).

[19] Michael U Gutmann and Aapo Hyvärinen. 2012. Noise-contrastive estimation of unnormalized statistical models, with applications to natural image statistics. *Journal of machine learning research* 13, 2 (2012).

[20] Lingxin Hao and Daniel Q Naiman. 2007. *Quantile regression.* Number 149. Sage.

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

[22] Ruining He and Julian McAuley. 2016. Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In *proceedings of the 25th international conference on world wide web.* 507–517.

[23] 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.* 355–364.

[24] Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yongdong Zhang, and Meng Wang. 2020. Lightgcn: Simplifying and powering graph convolution network for recommendation. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval.* 639–648.

[25] 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.

[26] 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.

[27] Katja Hofmann, Anne Schuth, Alejandro Bellogin, and Maarten De Rijke. 2014. Effects of position bias on click-based recommender evaluation. In *Advances in Information Retrieval: 36th European Conference on IR Research, ECIR 2014, Amsterdam, The Netherlands, April 13-16, 2014. Proceedings 36.* Springer, 624–630.

[28] Neil Hurley and Mi Zhang. 2011. Novelty and diversity in top-n recommendation–analysis and evaluation. *ACM Transactions on Internet Technology (TOIT)* 10, 4 (2011), 1–30.

[29] Rolf Jagerman, Zhen Qin, Xuanhui Wang, Michael Bendersky, and Marc Najork. 2022. On optimizing top-k metrics for neural ranking models. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval.* 2303–2307.

[30] Kalervo Järvelin and Jaana Kekäläinen. 2017. IR evaluation methods for retrieving highly relevant documents. In *ACM SIGIR Forum*, Vol. 51. ACM New York, NY, USA, 243–250.

[31] Johan Ludwig William Valdemar Jensen. 1906. Sur les fonctions convexes et les inégalités entre les valeurs moyennes. *Acta mathematica* 30, 1 (1906), 175–193.

[32] Xu Ji, Joao F Henriques, and Andrea Vedaldi. 2019. Invariant information clustering for unsupervised image classification and segmentation. In *Proceedings of the IEEE/CVF international conference on computer vision.* 9865–9874.

[33] Wang-Cheng Kang and Julian McAuley. 2018. Self-attentive sequential recommendation. In *2018 IEEE international conference on data mining (ICDM).* IEEE, 197–206.

[34] Hyeyoung Ko, Suyeon Lee, Yoonseo Park, and Anna Choi. 2022. A survey of recommendation systems: recommendation models, techniques, and application fields. *Electronics* 11, 1 (2022), 141.

[35] R Koenker. 2005. Quantile Regression Cambridge, UK: Cambridge Univ.

[36] Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix factorization techniques for recommender systems. *Computer* 42, 8 (2009), 30–37.

[37] Dong Li, Ruoming Jin, Jing Gao, and Zhi Liu. 2020. On sampling top-k recommendation evaluation. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining.* 2114–2124.

[38] Juanhui Li, Harry Shomer, Haitao Mao, Shenglai Zeng, Yao Ma, Neil Shah, Jiliang Tang, and Dawei Yin. 2023. Evaluating graph neural networks for link prediction: Current pitfalls and new benchmarking. *Advances in Neural Information Processing Systems* 36 (2023), 3853–3866.

[39] Siyi Lin, Chongming Gao, Jiawei Chen, Sheng Zhou, Binbin Hu, Yan Feng, Chun Chen, and Can Wang. 2025. How do recommendation models amplify popularity bias? An analysis from the spectral perspective. In *Proceedings of the Eighteenth ACM International Conference on Web Search and Data Mining.* 659–668.

[40] Tie-Yan Liu et al. 2009. Learning to rank for information retrieval. *Foundations and Trends® in Information Retrieval* 3, 3 (2009), 225–331.

[41] Xiao Liu, Fanjin Zhang, Zhenyu Hou, Li Mian, Zhaoyu Wang, Jing Zhang, and Jie Tang. 2021. Self-supervised learning: Generative or contrastive. *IEEE transactions on knowledge and data engineering* 35, 1 (2021), 857–876.

[42] R Duncan Luce. 1959. *Individual choice behavior.* Vol. 4. Wiley New York.

[43] Bodhisattwa Prasad Majumder, Shuyang Li, Jianmo Ni, and Julian McAuley. 2019. Generating personalized recipes from historical user preferences. *arXiv preprint arXiv:1909.00105* (2019).

[44] 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.

[45] Liqiang Nie, Wenjie Wang, Richang Hong, Meng Wang, and Qi Tian. 2019. Multimodal dialog system: Generating responses via adaptive decoders. In *Proceedings of the 27th ACM international conference on multimedia.* 1098–1106.

[46] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. 2018. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748* (2018).

[47] Emanuel Parzen. 1962. On estimation of a probability density function and mode. *The annals of mathematical statistics* 33, 3 (1962), 1065–1076.

[48] Robin L Plackett. 1975. The analysis of permutations. *Journal of the Royal Statistical Society Series C: Applied Statistics* 24, 2 (1975), 193–202.

[49] Przemysław Pobrotyn and Radosław Białobrzeski. 2021. Neuralndcg: Direct optimisation of a ranking metric via differentiable relaxation of sorting. *arXiv preprint arXiv:2102.07831* (2021).

[50] Tao Qin and Tie-Yan Liu. 2013. Introducing LETOR 4.0 Datasets. *CoRR* abs/1306.2597 (2013). http://arxiv.org/abs/1306.2597

[51] Zi-Hao Qiu, Quanqi Hu, Yongjian Zhong, Lijun Zhang, and Tianbao Yang. 2022. Large-scale stochastic optimization of NDCG surrogates for deep learning with provable convergence. *arXiv preprint arXiv:2202.12183* (2022).

[52] 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.* 605–613.

[53] Zhaochun Ren, Shangsong Liang, Piji Li, Shuaiqiang Wang, and Maarten de Rijke. 2017. Social collaborative viewpoint regression with explainable recommendations. In *Proceedings of the tenth ACM international conference on web search and*

data mining. 485–494.

[54] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian personalized ranking from implicit feedback. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*. 452–461.

[55] Herbert Robbins and Sutton Monro. 1951. A stochastic approximation method. *The annals of mathematical statistics* (1951), 400–407.

[56] Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Galligher, and Tina Eliassi-Rad. 2008. Collective classification in network data. *AI magazine* 29, 3 (2008), 93–93.

[57] Alexander Shapiro. 2017. Distributionally robust stochastic programming. *SIAM Journal on Optimization* 27, 4 (2017), 2258–2275.

[58] Wentao Shi, Jiawei Chen, Fuli Feng, Jizhi Zhang, Junkang Wu, Chongming Gao, and Xiangnan He. 2023. On the theories behind hard negative sampling for recommendation. In *Proceedings of the ACM Web Conference 2023*. 812–822.

[59] Wentao Shi, Chenxu Wang, Fuli Feng, Yang Zhang, Wenjie Wang, Junkang Wu, and Xiangnan He. 2024. Lower-Left Partial AUC: An Effective and Efficient Optimization Metric for Recommendation. In *Proceedings of the ACM on Web Conference 2024*. 3253–3264.

[60] Xiaoyuan Su. 2009. A Survey of Collaborative Filtering Techniques. (2009).

[61] Bohao Wang, Jiawei Chen, Changdong Li, Sheng Zhou, Qihao Shi, Yang Gao, Yan Feng, Chun Chen, and Can Wang. 2024. Distributionally robust graph-based recommendation system. In *Proceedings of the ACM Web Conference 2024*. 3777–3788.

[62] Bohao Wang, Feng Liu, Jiawei Chen, Xingyu Lou, Changwang Zhang, Jun Wang, Yuegang Sun, Yan Feng, Chun Chen, and Can Wang. 2025. MSL: Not All Tokens Are What You Need for Tuning LLM as a Recommender. *arXiv preprint arXiv:2504.04178* (2025).

[63] Bohao Wang, Feng Liu, Changwang Zhang, Jiawei Chen, Yudi Wu, Sheng Zhou, Xingyu Lou, Jun Wang, Yan Feng, Chun Chen, et al. 2024. Llm4dsr: Leveraing large language model for denoising sequential recommendation. *arXiv preprint arXiv:2408.08208* (2024).

[64] Mengdi Wang, Ethan X Fang, and Han Liu. 2017. Stochastic compositional gradient descent: algorithms for minimizing compositions of expected-value functions. *Mathematical Programming* 161 (2017), 419–449.

[65] Wenjie Wang, Fuli Feng, Xiangnan He, Hanwang Zhang, and Tat-Seng Chua. 2021. Clicks can be cheating: Counterfactual recommendation for mitigating clickbait issue. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1288–1297.

[66] Xuanhui Wang, Cheng Li, Nadav Golbandi, Michael Bendersky, and Marc Najork. 2018. The lambdaloss framework for ranking metric optimization. In *Proceedings of the 27th ACM international conference on information and knowledge management*. 1313–1322.

[67] Hongyi Wen, Longqi Yang, and Deborah Estrin. 2019. Leveraging post-click feedback for content recommendations. In *Proceedings of the 13th ACM Conference on Recommender Systems*. 278–286.

[68] Junkang Wu, Jiawei Chen, Jiancan Wu, Wentao Shi, Jizhi Zhang, and Xiang Wang. 2024. Bsl: Understanding and improving softmax loss for recommendation. In *2024 IEEE 40th International Conference on Data Engineering (ICDE)*. IEEE, 816–830.

[69] Jiancan Wu, Xiang Wang, Xingyu Gao, Jiawei Chen, Hongcheng Fu, and Tianyu Qiu. 2024. On the effectiveness of sampled softmax loss for item recommendation. *ACM Transactions on Information Systems* 42, 4 (2024), 1–26.

[70] 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*. 1192–1199.

[71] Weiqin Yang, Jiawei Chen, Xin Xin, Sheng Zhou, Binbin Hu, Yan Feng, Chun Chen, and Can Wang. 2024. PSL: Rethinking and Improving Softmax Loss from Pairwise Perspective for Recommendation. *arXiv preprint arXiv:2411.00163* (2024).

[72] Junliang Yu, Xin Xia, Tong Chen, Lizhen Cui, Nguyen Quoc Viet Hung, and Hongzhi Yin. 2023. XSimGCL: Towards extremely simple graph contrastive learning for recommendation. *IEEE Transactions on Knowledge and Data Engineering* (2023).

[73] An Zhang, Leheng Sheng, Zhibo Cai, Xiang Wang, and Tat-Seng Chua. 2024. Empowering Collaborative Filtering with Principled Adversarial Contrastive Loss. *Advances in Neural Information Processing Systems* 36 (2024).

[74] Ziwei Zhu, Jianling Wang, and James Caverlee. 2019. Improving top-k recommendation via jointcollaborative autoencoders. In *The World Wide Web Conference*. 3483–3482.

A Appendix

Due to space limitations, the complete appendix is provided online and can be accessed at https://github.com/Tiny-Snow/IR-Benchmark/blob/main/paper/SLatK-KDD-2025.pdf.