Top-N-Rank: A Scalable List-wise Ranking Method for Recommender Systems

Junjie Liang† , Jinlong Hu‡*, Shoubin Dong‡ and Vasant Honavar†
†Artificial Intelligence Research Laboratory, Center for Big Data Analytics and Discovery Informatics
College of Information Sciences and Technology, Pennsylvania State University, University Park, PA 16802, United States
Email: {jul672, vhonavar}@ist.psu.edu
‡Guangdong Key Laboratory of Communication & Computer Network, School of Computer Science and Engineering
South China University of Technology, Guangzhou 510006, China
Email: {jlhu, sbdong}@scut.edu.cn

***Abstract*—We propose Top-N-Rank, a novel family of list-wise Learning-to-Rank models for reliably recommending the N top-ranked items. The proposed models optimize a variant of the widely used cumulative discounted gain (DCG) objective function which differs from DCG in two important aspects: (i) It limits the evaluation of DCG only on the top N items in the ranked lists, thereby eliminating the impact of low-ranked items on the learned ranking function; and (ii) it incorporates weights that allow the model to leverage multiple types of implicit feedback with differing levels of reliability or trustworthiness. Because the resulting objective function is non-smooth and hence challenging to optimize, we consider two smooth approximations of the objective function, using the traditional sigmoid function and the rectified linear unit (ReLU). We propose a family of learning-to-rank algorithms (Top-N-Rank) that work with any smooth objective function. Then, a more efficient variant, Top-N-Rank.ReLU, is introduced, which effectively exploits the properties of ReLU function to reduce the computational complexity of Top-N-Rank from quadratic to linear in the average number of items rated by users. The results of our experiments using two widely used benchmarks, namely, the MovieLens data set and the Amazon Video Games data set demonstrate that: (i) The "top-N truncation" of the objective function substantially improves the ranking quality of the top N recommendations; (ii) using the ReLU for smoothing the objective function yields significant improvement in both ranking quality as well as runtime as compared to using the sigmoid; and (iii) Top-N-Rank.ReLU substantially outperforms the well-performing list-wise ranking methods in terms of ranking quality.**

***Index Terms*—list-wise ranking; top n recommendation; learning to rank; latent factor model; rectifier function;**

I. Introduction

Collaborative filtering (CF) is one of the most widely used methods in recommender systems. CF systems recommend similar items to users who share similar traits or similar tastes [1]. Learning to Rank (LTR) methods, which directly learn to accurately rank items based on the user's ratings, rankings, or implicit feedback over a set of items, are widely used to learn the perfect rankings for top-n recommendation scenarios [2], [3].

Part of this work was conducted when the first author was at the South China University of Technology.

*Corresponding author (jlhu@scut.edu.cn).

A. Related Work

Learning to Rank (LTR) methods can be categorized into point-wise, pair-wise and list-wise methods. Point-wise methods learn ranking models from the scores assigned by users to individual items (see e.g., [4]). Pair-wise methods (e.g., BPR [5]), learn binary classifiers that compare ordered pairs of items to decide whether the first item is preferred to the second. The applicability of such methods is limited by the high computational cost of pair-wise comparisons of user-rated items in generating the training samples for the binary classifiers [6]. List-wise methods leverage the entire list of items consumed by the users to optimize a list-wise ranking loss function or the probability of permutations that map items to ranks [7], [8]. Typically, such methods optimize a smooth approximation of a loss function that measures the distance between the reference lists of ranked items in the training data and the ranked list of items produced by the ranking model. For example, CLiMF [3], which optimizes a smooth lower bound of mean-reciprocal rank (MRR), aims at ranking a small set of most-preferred items at the top of the list; TFMAP [9] optimizes the mean average precision (MAP) of top-ranked items for each user in a given context. Other methods optimize discounted cumulative gain (DCG) or normalized DCG (NDCG) can be found in [10], [11]. Examples of list-wise methods that optimize the probability of permutations that map items to ranks include: ListPMF, which represents each user as a probability distribution of the permutations over rated items based on the Plackett-Luce model [12]; ListRank [13], which aims to identify a ranking permutation that minimizes the cross-entropy between the distribution of the observed ranking of items based on user ratings and the predicted rankings with respect to the top-ranked item; or methods that optimize the log-posterior over the predicted preference order with the observed preference orders [12]; and methods that leverage deep neural nets (e.g., [4]) to learn the non-linear interaction between user-item pairs (See [14] for a survey of such methods).

Existing LTR approaches suffer from several limitations. Although, in practical applications, only the top (say N) items in the ranked list are of interest, and the lower-ranked

ratings in the list are less reliable, most existing LTR methods are optimized on the ranks of the entire lists, which, could potentially reduce the ranking quality of the top-ranked items. Furthermore, the computational complexity of straightforward approaches to optimizing ranking measures (e.g., DCG [15], MRR [3], AUC [11] or MAP [9]), scale quadratically with $\tilde{m}$ (the average number of observed items across all users), which renders such methods impractical in large-scale real-world settings.

B. Overview and Contributions

To address the limitations of existing LTR systems, we propose Top-N-Rank, a novel latent factor based list-wise ranking model for top-N recommendation problem which directly optimizes a novel weighted "top-heavy" truncated variant of the DCG ranking measure, namely, wDCG@N. Since in many situations, the users only refer to the top-ranked items in the list, the higher positions often have more impact on the ranking score than the lower ones. Our proposed measure, wDCG@N, differs from the conventional DCG in two important aspects: (i) It considers only on the top N items in the ranked lists, thereby eliminating the impact of low-ranked items; and (ii) it incorporates weights that allow the model to learn from multiple kinds of implicit feedback.

Because wDCG@N is non-smooth, we introduce the rectified linear unit (ReLU) as a smoothing function, which is more suited to top N ranking problems than the traditional sigmoid function. ReLU not only eliminates the contribution of the low-ranked items on our loss function, but also allows us to obtain a significantly faster variant of the wDCG@N-based LTR approach (Top-N-Rank.ReLU), yielding a substantial reduction in computational complexity from $O(kn'\tilde{m}^2)$ to $O(kn'\tilde{m})$, where k denotes the dimension of latent factors, n' denotes the batch size of users for the stochastic gradient descent algorithm and $\tilde{m}$ is the average number of (observed) items, making Top-N-Rank.ReLU scalable to large-scale real-world settings.

The main contributions of this paper can be summarized as follows:

1) We have introduced a novel list-wise ranking model for top-N recommendation, which directly optimizes a weighted top-heavy truncated ranking objective function, wDCG@N. Our model improves the quality of the top-N item lists by mitigating the impact of the lower-ranked items, and is capable of handling multiple types of implicit feedback (if available).
2) We have introduced the rectified linear unit (ReLU) to smooth our objective function. We have demonstrated that ReLU could (1) eliminate the impact of the lower-ranked items and (2) substantially speed up the calculations by careful algorithm design.
3) We have proposed a fast learning algorithm (Top-N-Rank) for generic smoothing functions, and a substantially more efficient variant (Top-N-Rank.ReLU) for the ReLU smoothing function which reduce the computational complexity from $O(kn'\tilde{m}^2)$ to $O(kn'\tilde{m})$.

We compared the performance of Top-N-Rank and Top-N-Rank.ReLU with several state-of-the-art list-wise LTR methods [3], [10], [12], [13], [16] using the MovieLens (20M) data set [17] and the Amazon video games data set [18]. All experiments were performed on Apache Spark cluster [19] and the raw data are stored on Hadoop Distribute File System (HDFS).

II. Preliminaries

Let $\mathcal{U} = \{u_1, u_2, \ldots, u_n\}$ be the set of n users, $\mathcal{I} = \{i_1, i_2, \ldots, i_m\}$ be the set of m items and $\mathcal{P} = \{p_1, p_2, \ldots, p_t\}$ be the set of t types of implicit feedback. The interactions of users with items and the associated implicit feedback are represented by $\mathcal{X} = \mathcal{U} \times \mathcal{I} \times \mathcal{P}$, where the entry $(u, i, p_{ui}) \in \mathcal{X}$ denotes the interaction of user u with item i and the associated implicit feedback p_{ui}. We further denote by $\mathcal{I}_u^+$ the subset of items actually observed by or presented to u. For each $i \in \mathcal{I}_u^+$, we denote the rating of i by f_{ui} and the position of i based on u's rank ordering of items by R_{ui}^+. We reserve the indexing letters u to indicate arbitrary user in $\mathcal{U}$ and i to represent arbitrary item in $\mathcal{I}$.

A. Latent Factor Model

Latent factor models (LFMs) are state-of-the-art in terms of both the quality of recommendations as well as scalability [20]. LFMs represent both users and items using low-dimensional vectors of latent factors. Let θ be the set of latent factors such that θ^{user} is an $n \times k$ matrix with the u-th row θ_u^{user} denoting the latent factors of u and θ^{item} is an $m \times k$ matrix with the i-th row θ_i^{item} denoting the latent factors of i. The rank k, of the latent factor matrices is much smaller than n or m. The rating for u to i is predicted by the dot product of θ_u^{user} and θ_i^{item}.

B. Discounted Cumulative Gain

The discounted Cumulative Gain (DCG) [21] is a widely used measure of quality of recommendations, which measures the degree to which higher ranked items are placed ahead of the lower ranked ones, with the contribution of lower ranked items discounted by a logarithmic factor. Let y_{ui} be a binary indicator to represent whether i is relevant to u, then DCG of u is computed by:

$$\mathrm{DCG}_u = \sum_{i \in \mathcal{I}} \frac{2^{y_{ui}} - 1}{\log\left(R_{ui} + 2\right)} \tag{1}$$

Notice that the ranked position (start from zero) of item i can be computed by:

$$R_{ui} = \sum_{j \in \mathcal{I}} 1(f_{ui} < f_{uj}) \tag{2}$$

where $1(x)$ is an indicator function with $1(x) = 1$ if x is true and otherwise $1(x) = 0$. Given our emphasis on getting the top rated items ranked correctly in the list of recommended items, DCG appears to be good criterion to optimize on. However, as evident from (1), DCG suffers from two important limitations:

(i) Although DCG de-emphasizes the contribution of the lower ranked items, it does not eliminate the collective effect of a large number of lower ranked items, even if the ranking of such lower ranked items are less reliable. If the goal is to optimize the ranking of the N top rated items, it makes sense to tailor objective function to focus explicitly on the ranking of the N top-rated items and ignore the rest. (ii) Because DCG assigns equal weights to all implicit user feedback, it fails to account for differences in their trustworthiness.

III. Top-N-Rank

We proceed to introduce wDCG@N, a variant of DCG that overcomes its drawbacks. We then describe two smoothing functions (sigmoid and rectified linear unit (ReLU)) that can convert wDCG@N to a smoothed function that is amenable to being optimized using the standard optimization techniques. Finally, we show how to use the ReLU approximation of wDCG@N to obtain a scalable LTR algorithm.

A. Top-N-Rank Training Objective

To address the limitations of DCG, we introduce wDCG@N, which is defined as follows:

$$\mathrm{wDCG}_u@\mathrm{N} = \sum_{i \in \mathcal{I}} 1\left(R_{ui} < N\right) \cdot \frac{w_{p_{ui}}\left(2^{y_{ui}} - 1\right)}{\log\left(R_{ui} + 2\right)} \quad (3)$$

The first term in (3), $1\left(R_{ui} < N\right)$ is an indicator function that selects only the N top-rated items and ignores the rest. The coefficient $w_{p_{ui}}$ in the second term denotes the weight of the implicit feedback p_{ui}, which can model the reliability or importance of the feedback. The choice of $w_{p_{ui}}$ is application and data-dependent. For example, one can set $w_{p_{ui}}$ to the number of items rated by (or presented to) the user [22] or the conversion rate (the proportion between buyers and users who conducted the implicit feedback). The resulting ranking objective can be formulated as:

$$L(\theta) = \max_{\theta} \sum_{u \in \mathcal{U}} \mathrm{wDCG}_u@\mathrm{N} - \lambda \left|\left|\theta\right|\right|_2^2 \quad (4)$$

where $||\cdot||_2^2$ denotes the L^2-norm and λ is the regularization coefficient that controls over-fitting.

B. Smooth Approximations of Top-N-Rank Training Objective

A non-smooth training objective such as the one in (4) is challenging to optimize. Hence, we replace the non-smooth training objective in (4) by its smooth approximation. Specifically, we approximate the indicator function in (2) by a smooth function h such that $1\left(f_{ui} < f_{uj}\right) \approx h\left(\Delta_{uji}\right)$ with $\Delta_{uji} = f_{uj} - f_{ui}$. In what follows, we will consider two different smooth functions that accomplish this goal.

Sigmoid function. The sigmoid function is widely used in existing list-wise LTR-based recommendation models (e.g., [3], [9]) for its appealing performance in practice. Instead of adopting the sigmoid function directly, we introduce a scaling constant C $(C \geq 1)$ to provides more accurate estimation, such that the indicator function is approximated by $g\left(C\Delta_{uji}\right)$ where $g\left(x\right) = 1/(1 + \exp\left(-x\right))$.

Rectifier function. The rectified linear units (ReLU) [23], is a nonlinear smooth function with several properties that make it attractive in our setting. The one-sided nature of ReLU $(relu\left(x\right) = \max\left\{0, x\right\})$ eliminates the contribution of of the lower-rated items to the objective function. Second, ReLU is computationally simper: only comparison and addition operations are required. Third, the form of ReLU permits an efficient algorithm (see Algorithm 2) with computational complexity that is linear in the average number of (observed) items across all users (see section III-C2). When ReLU is used, we have $1\left(f_{ui} < f_{uj}\right) \approx relu\left(\Delta_{uji}\right)$

1) Parameterization of the Smooth Functions: Recall that the "top-N term", $1\left(R_{ui} < N\right)$, was introduced to indicate whether item i ranks among the top N item list. However, a poor choice of the hyper-parameters in the smooth function could lead to gross under-estimation or over-estimation of R_{ui} and thus negate the utility of the "top-N term".

Here we examine how to choose the parameters of the sigmoid and the ReLU functions so that they behave as intended. In the case of the sigmoid function, we see that a choice of C matters, with proper values of C (e.g., $C = 7$) yielding the desired behavior. In the case of ReLU, we can ensure the desired behavior by controlling the initial distribution of latent factors θ. Suppose that $\theta \sim U\left(0, b\right)$, where b is the width of the uniform distribution. Then according to the Central Limit Theorem, for arbitrary u, i, f_{ui} approximately follows a Gaussian distribution, i.e., $f_{ui} \sim N\left(\mu, \sigma^2\right)$ with $\mu = \mathbb{E}\left[\sum_t \theta_{ut}^{user} \cdot \theta_{it}^{item}\right] = kb^2/4$ and $\sigma^2 = \mathbb{E}\left[\sum_t \theta_{ut}^{user} \cdot \theta_{it}^{item}\right]^2 - \mu^2 = 7kb^4/144$. In order to ensure that $|f_{ui} - \mu| \leq 1$, making use of the fact that $P\left(|f_{ui} - \mu| \leq 3\sigma\right) \approx 1$, we find $3\sigma = 1$, and hence $b = 2/\sqrt[4]{7k}$, which provides the basic setting for all of the Top-N models using the ReLU as the smoothing function.

C. Fast LTR Algorithms

1) Fast LTR Algorithm for Generic Smooth Function: To optimize the objective function reported in (4), we need to compute the predicted score of each item and then perform the pair-wise comparison to determine their positions in the rank-ordered list. Because in most cases, the number of items m far outnumbers the dimension of the latent factors k, the complexity of a single pass is $O\left(knm^2\right)$. One common practice is to exploit the sparsity of $\mathcal{X}$ by considering only the predicted scores of the observed items, yielding a smooth objective function such as:

$$\begin{aligned} L^+(\theta) = \min_{\theta} - \sum_{u \in \mathcal{U}} \sum_{i \in \mathcal{I}_u^+} h(N - R_{ui}^+) \\ \cdot \frac{w_{p_{ui}}(2^{y_{ui}} - 1)}{\log(R_{ui}^+ + 2)} + \lambda ||\theta||_2^2 \end{aligned} \quad (5)$$

The gradient of $L^+(\theta)$ w.r.t. θ is given by (6).

$$\frac{\partial L^+}{\partial \theta} = \sum_{u \in \mathcal{U}} \sum_{i \in \mathcal{I}_u^+} \mathrm{wDCG}_u^+ \cdot \Big(\sum_{j \in \mathcal{I}_u^+} h'(\Delta_{uji}) \frac{\partial \Delta_{uji}}{\partial \theta} \Big) \cdot \Big(h'(N - R_{ui}^+) + \frac{h(N - R_{ui}^+)}{(R_{ui}^+ + 2)\log(R_{ui}^+ + 2)} \Big) + 2\lambda\theta \tag{6}$$

Algorithm 1: Top-N-Rank

Input : User-item feedback $\mathcal{X} \subseteq \mathcal{U} \times \mathcal{I} \times \mathcal{P}$, the truncate coefficient N, smooth function h, dimension of latent factors k, learning rate α, regularization coefficient λ, batch size n'

Output: The learned latent factors θ

Initialize $\theta^{(0)}$; set $t = 0$
while *not converged* **do**
  $U^{(t)}$ = draw n' samples randomly from $\mathcal{U}$
  for $u \in U^{(t)}$ **do**
    Update $\theta_u^{user(t+1)} = \theta_u^{user(t)} - \alpha \left(\frac{\partial L^+}{\partial \theta_u^{user}} \right)^{(t)}$ based on (6)
    for $i \in \mathcal{I}_u^+$ **do**
      Update $\theta_i^{item(t+1)} = \theta_i^{item(t)} - \alpha \left(\frac{\partial L^+}{\partial \theta_i^{item}} \right)^{(t)}$ based on (6)
    end
  end
  $t = t + 1$
end
return $\theta^{(t)}$

The gradients for Δ_{uji} w.r.t. θ are $\frac{\partial \Delta_{uji}}{\partial \theta_u^{user}} = \left(\theta_j^{item} - \theta_i^{item}\right)$ and $\frac{\partial \Delta_{uji}}{\partial \theta_i^{item}} = -\theta_u^{user}$. h' is the derivative of smooth function which is presented in section III-B. The pseudo-code for Top-N-Rank (using stochastic gradient descent) is given in Algorithm 1.

Similar to [3], the computational complexity of Top-N-Rank for one iteration is $O(kn'\tilde{m}^2)$ ($\tilde{m}$ denotes the average number of observed items across all users).

2) Enhanced LTR Algorithm for Large-scale Top-N Recommendation: Algorithm 1 can be intractable in large-scale systems with massive number of items. The use of ReLU permits a more efficient version of Top-N-Rank (denoted as Top-N-Rank.ReLU) to further reduce the complexity to $O(kn'\tilde{m})$. The pseudo-code for Top-N-Rank.ReLU is given in Algorithm 2.

For a single user, step 5 and 12 is computed in $O(k\tilde{m} + \tilde{m}\log\tilde{m})$. Note that $R_{u\pi_i}^+$ (step 7 and step 14) and $\sum_{j \in \mathcal{I}_u^+} h'\left(\Delta_{uj\pi_i}\right) \frac{\partial \Delta_{uj\pi_i}}{\partial \theta_u^{user}}$ (step 8 and step 15) can be calculated in $O\left(1\right)$ and $O\left(k\right)$ respectively through step-by-step accumulation, the complexity of step 6-11 and step 13-17 are $O(k\tilde{m})$. Therefore, the overall computational complexity of Top-N-Rank.ReLU for one iteration is $O(n'\tilde{m}(k + \log\tilde{m}))$. In practice, $\log\widetilde{m}$ is usually very small (less than 20) even in large-scale systems, Thus, we can expect that k is of the same scale with $\log\widetilde{m}$, then the complexity is simplified to $O(kn'\tilde{m})$, making Top-N-Rank.ReLU suitable for large-scale settings with massive item sets.

Algorithm 2: Top-N-Rank.ReLU

Input : User-item feedback $\mathcal{X} \subseteq \mathcal{U} \times \mathcal{I} \times \mathcal{P}$, the truncate coefficient N, smooth function h, dimension of latent factors k, learning rate α, regularization coefficient λ, batch size n'

Output: The learned latent factors θ

1 Initialize $\theta^{(0)}$ randomly from $U(0, 2/\sqrt[4]{7k})$ and $t = 0$
2 **while** *not converged* **do**
3 $U^{(t)}$ = draw n' samples randomly from $\mathcal{U}$
4 **for** $u \in U^{(t)}$ **do**
5 π= the descending orders of items indicated by the predicted score f_u^+
6 **for** $i = 2, \ldots, |\pi|$ **do**
7 $R_{u\pi_i}^+ = \sum_{j<i} \left(f_{u\pi_j} - f_{u\pi_i}\right)$
8 $\sum_{j \in \mathcal{I}_u^+} h'\left(\Delta_{uj\pi_i}\right) \frac{\partial \Delta_{uj\pi_i}}{\partial \theta_u^{user}} = \sum_{j<i} \left(\theta_{\pi_j}^{item} - \theta_{\pi_i}^{item}\right)$
9 compute and add the current gradient to $\frac{\partial L^+}{\partial \theta_u^{user}}$ based on (6)
10 **end**
11 Update $\theta_u^{user(t+1)} = \theta_u^{user(t)} - \alpha \left(\frac{\partial L^+}{\partial \theta_u^{user}} \right)^{(t)}$ based on (6)
12 π= the descending orders of items indicated by the predicted score f_u^+
13 **for** $i = 2, \ldots, |\pi|$ **do**
14 $R_{u\pi_i}^+ = \sum_{j<i} \left(f_{u\pi_j} - f_{u\pi_i}\right)$
15 $\sum_{j \in \mathcal{I}_u^+} h'\left(\Delta_{uj\pi_i}\right) \frac{\partial \Delta_{uj\pi_i}}{\partial \theta_{\pi_i}^{item}} = (1 - i)\, \theta_u^{user}$
16 Update $\theta_{\pi_i}^{item(t+1)} = \theta_{\pi_i}^{item(t)} - \alpha \left(\frac{\partial L^+}{\partial \theta_{\pi_i}^{item}} \right)^{(t)}$ based on (6)
17 **end**
18 **end**
19 $t = t + 1$
20 **end**
21 **return** $\theta^{(t)}$

IV. Experiments and Results

We report results of two sets of experiments. The first set of experiments compare the performance of Top-N-Rank models using either the sigmoid and the ReLU functions for smoothing with or without the "top-N truncation". Our results show that Top-N-Rank.ReLU (using "top-N truncation" and the ReLU function, i.e., Algorithm 2) outperforms the other methods on both the benchmark data sets. The second set of experiments compare the performance of Top-N-Rank.ReLU with several state-of-the-art list-wise LTR CF approaches. Our results show that Top-N-Rank algorithms outperform the these methods on both the benchmark data sets.

All of our experiments were performed on an Apache Spark cluster [19] with four compute nodes (Intel Xeon 2.1 GHz CPU with 20G RAM per node) with the raw data stored on Hadoop Distributed File System (HDFS). The model parameters were tuned to optimize performance on the training data. We describe below the details of the experiments and the results.

A. Experimental Setup

1) Data Sets: We used two benchmark data sets in our experiments: (i) the Amazon video games data set [18], which contains a subset of video games product reviews (ratings, text, etc.) from Amazon. There are 7,077 users, 25,744 items and more than 1 million ratings in this data set. (ii) The MovieLens (20M) data set [17], which contains 138,493 users, 27,278 items and more than 20 millions of ratings. The ratings in both data sets are split to 1-5 stars, with more stars corresponding to higher ratings. We use only the user rating data to conduct the experiments.

2) Evaluation Procedure: We first remove users who rated fewer than 10 items. For the remaining users, we convert the ratings to implicit feedback based on the item ratings provided by each user. That is, for each u, we assign $w_{p_{ui}} = 1$ when $f_{ui} \geq 4$ and otherwise $w_{p_{ui}} = -1$ [10]. We randomly select half of the ratings provided by each user for training, and use the rest for evaluation. On each test run, we average the performance over all of the users. We repeat this process 5 times and report the performance averaged across the 5 independent experiments.

We measure the performance based only on the rated items as in [10]. Because we focus on the placement of the top-rated items in the rank-ordered list, it is natural to use the Normalized Discounted Cumulative Gain (NDCG) [24] as the performance measure. In this paper, we report the average of NDCG@1 through NCDG@N across all users.

The definition of NDCG at the top-N positions for a user u is given by:

$$\mathrm{NDCG}_u@\mathrm{N} = \frac{\mathrm{DCG}_u@\mathrm{N}}{\mathrm{IDCG}_u@\mathrm{N}} \tag{7}$$

where DCG_u@N is the DCG value for the top-N ranked items as described in (1). IDCG_u@N is the perfect ranking score which is obtained when the ranked list is created by sorting the items in descending order of their implicit feedback values (ratings).

B. Comparison of Variants of Top-N-Rank

We compare the performance of LTR models trained with the smoothed and regularized wDCG@N objective using either the sigmoid and the ReLU functions for smoothing, with or without the "top-N truncation": (i) **Top-N-Rank.ReLU**: our proposed Top-N-Rank model trained to optimize wDCG@N smoothed using the ReLU function (Algorithm 2); (ii) **non-Top-N.ReLU**: The LTR model trained to optimize wDCG smoothed using the ReLU; (iii) **Top-N-Rank.sgm**: our proposed top-N model trained to optimize wDCG@N smoothed using the sigmoid function (Algorithm 1); and (iv) **non-Top-N.sgm**: The LTR model trained to optimize wDCG smoothed using the sigmoid function.

In these experiments, we set the number of latent factors $k = 10$ and the number of items ranked, $N = 20$. For the sigmoid function, $C = 7$ and for the ReLU function, $b = 2/\sqrt[4]{7k}$ (see section III-B1). The regularization coefficient λ is set to 0.1; and the batch size n' is set to 10% of the users in the training data. All methods are run until either maximum iteration $maxR = 30$ is reached or sum-of-square distance between parameters of two consecutive runs falls below the threshold $\epsilon = 0.1$.

The results of our experiments are summarized in Table I. Our results clearly show that the Top-N-Rank models with the "top-N truncation" term in the objective function consistently and statistically significantly (based on paired Student's t-test) outperform the non top-N counterparts. This confirms our intuition that Top-N-Rank models focus on correctly ordering the top-rated items, and hence are resistant to the cumulative effect (often unreliable) of lower-rated items. The results in Table I also show that Top-N-Rank.ReLU substantially outperforms Top-N-Rank.sgm. Moreover, the performance of Top-N-Rank.sgm is comparable to that of Non-Top-N.ReLU. We conclude that the ReLU function, with an appropriate choice of b is better able to more accurately rank the top-rated items. The runtime for Top-N-Rank.ReLU is significantly lower than that of Top-N-Rank.sgm (results not shown), proving the appealing efficiency of Top-N-Rank.ReLU.

C. Top-N-Rank.ReLU Compared with the State-of-the-Art List-wise LTR Models

We compare Top-N-Rank.ReLU with several state-of-the-art list-wise LTR CF approaches: (i) **MF-ADG**: An algorithm that optimizes the Averaged Discounted Gain (ADG), which is obtained by averaging the DCG across all users [10]. Similar to our work, MF-ADG is designed to work with implicit feedback data sets. The sampling parameter γ is fixed at 100; (ii) **CLiMF**: A MF model that is designed to work with binarized implicit feedback data sets, which optimizes mean-reciprocal rank (MRR) [3]. Instead of directly optimizing MRR, CLiMF learns the latent factors by maximizing the smoothed lower bound of MRR; (iii) **xCLiMF**: An extension of CLiMF that optimizes the expected reciprocal rank (ERR), which is designed to work with graded user ratings [16]; (iv) **ListRank**: A MF model that optimizes the cross-entropy between the distribution of the observed and predicted ratings using top-one probability, which is obtained using the softmax function [13]; and (v) **ListPMF-PL**: A list-wise probabilistic matrix factorization method that maximizes the log posterior over the predicted rank order with the observed preference order, using the Plackett-Luce model based permutation probability [12].

The results of our experiments are summarized in Table II. Top-N-Rank.ReLU consistently outperforms the baseline models on both the Amazon Video Game and MovieLens data sets, regardless of the length of recommended item lists. Student's t test further demonstrate the significance of our results (not

TABLE I: Comparison of Variants of Top-N-Rank

Data sets	Algorithms	NDCG@1	NDCG@3	NDCG@5	NDCG@10	NDCG@20
Amazon Video Games	Top-N-Rank.ReLU	**0.8186**	**0.8009**	**0.8079**	**0.8334**	**0.8455**
	non-Top-N.ReLU	0.8033	0.7866	0.7976	0.8242	0.8350
	Top-N-Rank.sgm	0.7956	0.7747	0.7861	0.8167	0.8269
	non-Top-N.sgm	0.7871	0.7657	0.7780	0.8109	0.8196
MovieLens	Top-N-Rank.ReLU	**0.7811**	**0.7648**	**0.7532**	**0.7469**	**0.7521**
	non-Top-N.ReLU	0.7775	0.7593	0.7466	0.7389	0.7430
	Top-N-Rank.sgm	0.7784	0.7564	0.7415	0.7323	0.7346
	non-Top-N-Rank.sgm	0.7575	0.7315	0.7121	0.6968	0.6954

TABLE II: Top-N-Rank.ReLU compared with the state-of-the-art list-wise LTR models

Data sets	Algorithms	NDCG@1	NDCG@3	NDCG@5	NDCG@10	NDCG@20
Amazon Video Games	Top-N-Rank.ReLU	**0.8135**	**0.7964**	**0.8043**	**0.8325**	**0.8383**
	MF-ADG	0.7809	0.7646	0.7762	0.8103	0.8178
	CLiMF	0.7101	0.7137	0.7388	0.7779	0.7829
	xCLiMF	0.709	0.7131	0.7381	0.7776	0.7823
	ListRank	0.7045	0.7106	0.7367	0.7761	0.7809
	ListPMF-PL	0.7043	0.7123	0.7376	0.7762	0.78
Movielens	Top-N-Rank.ReLU	**0.7827**	**0.7665**	**0.7548**	**0.7483**	**0.7531**
	MF-ADG	0.7301	0.6993	0.6799	0.6681	0.6698
	CLiMF	0.7459	0.7187	0.7013	0.691	0.6943
	xCLiMF	0.7609	0.7406	0.7271	0.7193	0.7236
	ListRank	0.7657	0.7423	0.7284	0.7206	0.7232
	ListPMF-PL	0.6981	0.6715	0.659	0.6568	0.6638

shown). Although Top-N-Rank.ReLU maximize wDCG on the top-20 items, the results show that the model offers better quality of recommendations across the top 1-20 items relative to the baselines. This may be explained in part by the following limitations of the individual methods: CLiMF and xCLiMF are designed to optimize the smoothed reciprocal rank (RR), which does not fully exploit the user ratings, because of its emphasis on optimizing only a few of the relevant items for each user; MF-ADG maximizes an approximation of ADG, on a small set of sampled data which may limit the quality of the estimates; ListRank and ListPMF-PL are designed for rating data, but assign the same weight to all items with the same rating. Perhaps more importantly, all of the methods except Top-N-Rank.ReLU attempt to optimize the ranking over the entire set of the user-rated items, as opposed to only the N top-ranked items, which makes them susceptible to the noise in the ratings of low-ranked items.

V. Summary and Discussion

In this paper, we proposed Top-N-Rank, a novel family of list-wise Learning-to-Rank models for reliably recommending the N top-ranked items. The proposed models optimize wDCG@N, a variant of the widely used cumulative discounted gain (DCG) objective function which differs from DCG in two important aspects: (1) It limits the evaluation of DCG only on the top N items in the ranked lists, thereby eliminating the impact of low-ranked items on the learned ranking function; and (2) it incorporates weights that allow the model to learn from multiple kinds of implicit user feedback with differing levels of reliability or trustworthiness. Because wDCG@N is non-smooth, we considered two smooth approximations of wDCG@N, using the traditional sigmoid function and the rectified linear unit (ReLU). We proposed a family of learning-to-rank algorithms (Top-N-Rank) that work with any smooth objective function (e.g., smooth approximations of wDCG@N). We designed Top-N-Rank.ReLU, a more efficient version of Top-N-Rank that exploits the properties of ReLU function to reduce the computational complexity of Top-N-Rank from quadratic to linear in the average number of items rated by users. The results of our experiments using two widely used benchmarks, namely, the Amazon Video Games data set and the MovieLens data set demonstrate that: (i) The "top-N truncation" of the objective function substantially improves the ranking quality; (ii) using the ReLU for smoothing the wDCG@N objective function yields significant improvement in both ranking quality as well as runtime as compared to using the sigmoid function; and (iii) Top-N-Rank.ReLU substantially outperforms the state-of-the-art list-wise ranking CF methods (MF-ADG, CLiMF, xCLiMF, ListRank, and ListPMF-PL) in terms of ranking quality.

Some promising directions for further research include: (i) Fusing the proposed top-N truncation component and ReLU smoothing function with different list-wise LTR objectives (i.e., MAP, AUC or MRR); (ii) investigation of complex interaction structure of user-item pairs with the help of deep neural nets; (iii) extending the proposed model to tensor factorization or factorization machines to take in multiple types of features.

Acknowledgment

Dr. Jinlong Hu and Dr. Shoubin Dong were supported in part by the Scientific Research Joint Funds of Ministry of

Education of China and China Mobile [No. MCM20150512], and the Natural Science Foundation of Guangdong Province of China [No. 2018A030313309]; Junjie Liang was supported in part by a research assistantship funded by the National Science Foundation through the grant [No. CCF 1518732] to Dr. Vasant G. Honavar. Dr. Vasant Honavar was supported in part by the Edward Frymoyer Endowed Chair in Information Sciences and Technology at Pennsylvania State University, and in part by the Sudha Murty Distinguished Visiting Chair in Neurocomputing and Data Science at the Indian Institute of Science.

REFERENCES

[1] A. Gunawardana and G. Shani, "A survey of accuracy evaluation metrics of recommendation tasks," *Journal of Machine Learning Research*, vol. 10, no. Dec, pp. 2935–2962, 2009.

[2] T.-Y. Liu, "Learning to rank for information retrieval," *Foundations and Trends in Information Retrieval*, vol. 3, no. 3, pp. 225–331, 2009.

[3] Y. Shi, A. Karatzoglou, L. Baltrunas, M. Larson, N. Oliver, and A. Hanjalic, "CLiMF: learning to maximize reciprocal rank with collaborative less-is-more filtering," in *Proceedings of the sixth ACM conference on Recommender systems*. ACM, 2012, pp. 139–146.

[4] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T.-S. Chua, "Neural collaborative filtering," in *Proceedings of the 26th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 2017, pp. 173–182.

[5] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme, "BPR: Bayesian personalized ranking from implicit feedback," in *Proceedings of the twenty-fifth conference on uncertainty in artificial intelligence*. AUAI Press, 2009, pp. 452–461.

[6] S. Huang, S. Wang, T.-Y. Liu, J. Ma, Z. Chen, and J. Veijalainen, "Listwise collaborative filtering," in *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 2015, pp. 343–352.

[7] Z. C. T. Q. Tie-Yan and L. M.-F. T. H. Li, "Learning to Rank: From Pairwise Approach to Listwise Approach," 2014.

[8] F. Xia, T.-Y. Liu, J. Wang, W. Zhang, and H. Li, "Listwise approach to learning to rank: theory and algorithm," in *Proceedings of the 25th international conference on Machine learning*. ACM, 2008, pp. 1192–1199.

[9] Y. Shi, A. Karatzoglou, L. Baltrunas, M. Larson, A. Hanjalic, and N. Oliver, "TFMAP: optimizing MAP for top-n context-aware recommendation," in *Proceedings of the 35th international ACM SIGIR conference on Research and development in information retrieval*. ACM, 2012, pp. 155–164.

[10] D. Lim, J. McAuley, and G. Lanckriet, "Top-n recommendation with missing implicit feedback," in *Proceedings of the 9th ACM Conference on Recommender Systems*. ACM, 2015, pp. 309–312.

[11] H. Steck, "Gaussian ranking by matrix factorization," in *Proceedings of the 9th ACM Conference on Recommender Systems*. ACM, 2015, pp. 115–122.

[12] J. Liu, C. Wu, Y. Xiong, and W. Liu, "List-wise probabilistic matrix factorization for recommendation," *Information Sciences*, vol. 278, pp. 434–447, 2014.

[13] Y. Shi, M. Larson, and A. Hanjalic, "List-wise learning to rank with matrix factorization for collaborative filtering," in *Proceedings of the fourth ACM conference on Recommender systems*. ACM, 2010, pp. 269–272.

[14] S. Zhang, L. Yao, and A. Sun, "Deep learning based recommender system: A survey and new perspectives," *arXiv preprint arXiv:1707.07435*, 2017.

[15] N. Ifada and R. Nayak, "Do-Rank: DCG optimization for learning-to-rank in tag-based item recommendation systems," in *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer, 2015, pp. 510–521.

[16] Y. Shi, A. Karatzoglou, L. Baltrunas, M. Larson, and A. Hanjalic, "xCLiMF: optimizing expected reciprocal rank for data with multiple levels of relevance," in *Proceedings of the 7th ACM conference on Recommender systems*. ACM, 2013, pp. 431–434.

[17] F. M. Harper and J. A. Konstan, "The movielens datasets: History and context," *ACM Transactions on Interactive Intelligent Systems (TiiS)*, vol. 5, no. 4, p. 19, 2016.

[18] J. McAuley and A. Yang, "Addressing complex and subjective product-related queries with customer reviews," in *Proceedings of the 25th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 2016, pp. 625–635.

[19] A. G. Shoro and T. R. Soomro, "Big data analysis: Apache spark perspective," *Global Journal of Computer Science and Technology*, vol. 15, no. 1, 2015.

[20] C. C. Aggarwal, "Model-based collaborative filtering," in *Recommender Systems*. Springer, 2016, pp. 71–138.

[21] K. Järvelin and J. Kekäläinen, "Cumulated gain-based evaluation of ir techniques," *ACM Transactions on Information Systems (TOIS)*, vol. 20, no. 4, pp. 422–446, 2002.

[22] Y. Koren, "Factorization meets the neighborhood: a multifaceted collaborative filtering model," in *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2008, pp. 426–434.

[23] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[24] K. Jrvelin and J. Keklinen, "IR evaluation methods for retrieving highly relevant documents," in *Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, 2000, pp. 41–48.