# Do-Rank: DCG Optimization for Learning-to-Rank in Tag-based Item Recommendation Systems

Noor Ifada[1,2], Richi Nayak[1]

[1]School of Electrical Engineering and Computer Science, Queensland University of Technology, Australia
[2]Informatics Department, University of Trunojoyo Madura, Indonesia

noor.ifada@{qut.edu.au,if.trunojoyo.ac.id},r.nayak@qut.edu.au

**Abstract.** Discounted Cumulative Gain (DCG) is a well-known ranking evaluation measure for models built with multiple relevance graded data. By handling tagging data used in recommendation systems as an ordinal relevance set of $\{negative, null, positive\}$, we propose to build a DCG based recommendation model. We present an efficient and novel learning-to-rank method by optimizing DCG for a recommendation model using the tagging data interpretation scheme. Evaluating the proposed method on real-world datasets, we demonstrate that the method is scalable and outperforms the benchmarking methods by generating a quality top-$N$ item recommendation list.

***Keywords*:** tagging data, tag-based item recommendation, discounted cumulative gain, top-$N$ recommendation

## 1. INTRODUCTION

In a tag-based recommendation system, users annotate items of their interest with freely-defined tags, hence the ternary relation of $\langle user, item, tag\rangle$ is naturally formed. This system must interpret the observed and non-observed entries in tagging data efficiently in order to generate quality recommendations [1]. The observed, or positive, entries reveal the user interest by indicating that the user has annotated an item using certain tags. The non-observed entries can reveal two types of information: (1) negative entries that indicate users are not interested with the items; or (2) null values that indicate users might be interested in them in the future and they need to be predicted [2]. Accordingly, tagging data can be labelled using the ordinal relevance set of $\{negative, null, positive\}$ for a tuple of $\langle user, item, tag\rangle$.

The task of a tag-based recommendation system is to generate the list of items, which may be of interest to a user, by learning the users past tagging behavior. The list of recommended items is ordered in descending order based on the predicted preference score. Users usually show more interest to the fewer items at the top of the list than the ones further down the list [3]. The order of items in the recommendation list is crucial and, therefore, the recommendation task can be considered as a ranking problem in which the item preference score needs to be determined and sorted for generating the list. A "learning-to-rank" approach optimizes the recommendation model with respect to the evaluation measure such that it can generate a quality top-$N$ recommendation list [3-5].

The measures widely used to evaluate the performance of a ranking model are Discounted Cumulative Gain (DCG), Mean Average Precision (MAP), and Mean Reciprocal Rank (MRR) [6]. Compared to other two measures, DCG is more widely used especially for models that include multiple relevance graded data [4]. Therefore, for the tagging data that is labelled using ordinal relevance set, DCG becomes the most suitable evaluation measure for measuring the ranking quality [4]. DCG emphasizes on getting the correct order of higher ranked items than that of lower ranked items. Since the users only refer to the top-$N$ items in the list, the higher positions have more influence on the score than the lower positions. DCG consists of two functions: (1) a discount function which makes items lower down in the ranked list contribute less to the DCG score; and (2) a gain function which gives the significance of the known relevance grade [4, 7]. These two functions make DCG significantly useful for solving the recommendation task which generates a quality top-$N$ recommendation list.

In this paper, we propose an efficient and novel learning-to-rank method, called as *Do-Rank*, by optimizing DCG for a recommendation model as the ranking measure. This method generates an optimal list of recommended items from the DCG perspective for all users. However, directly optimizing DCG across all users in the recommendation model is computationally expensive. To deal with this, we propose a fast learning algorithm with an efficient tagging data interpretation scheme [1] for the learning-to-rank model. This interpretation scheme has shown good performance when implemented on a *pair-wise ranking model* [1, 2, 8], i.e. the objective function of the model is formed based on *the pairs of items* on each user-tag set. In this paper, we design this scheme on a *list-wise ranking model*. This means that we must form the objective function only based on the list of *all items* on each user-tag set. Experimental results on real-world datasets show that *Do-Rank* is scalable and outperforms state-of-the-art recommendation methods in recommendation quality. It ascertains that optimizing DCG while building the learning-to-rank recommendation model improves the recommendation performance.

Recent works have proposed the optimized recommendation models based on MAP [9] and MRR [10], dealing with binary relevance data. Since the tag-based recommendation systems use the ordinal relevance data, these models are not suitable. Weimer et.al [11] have proposed a recommendation method by optimizing the Normalized DCG (NDCG) using rating data as *explicit* feedback. Our problem is quite different and difficult in comparison to this work as the tag-based recommendation systems use tags as *implicit* feedback. A recommendation system with explicit rating data builds its model by collecting the ratings which represent the preference level of each $\langle user, item\rangle$ binary relation. The list of recommendations is then generated by ranking the predicted preference scores inferred from the unobserved $\langle user, item\rangle$ relations [7, 11]. In contrast, a tag-based recommendation system builds its model by using the user tagging history as data entries. The key challenges it faces are modeling the $\langle user, item, tag\rangle$ ternary input data, inferring the latent relationships, and predicting each entry with a score which indicates its relevance degree [1, 2]. The recommendation list needs to be generated by ranking the predicted preference scores of list of items under all tags which may be interest to a user.

Recently researchers have used tensor models to represent and analyze the latent relationships inherent in a three-dimensional tagging data model [12, 13]. A tensor model can be used as a predictor function which maps the $\langle user, item, tag \rangle$ latent relationship inherent in tagging data to a predicted preference score, and enables optimization of the evaluation measure [14] such as DCG.

The proposed item recommendation method *Do-Rank* comes closest to the tag recommendation method PITF [8] that implements tensor approach and applies the ordinal relevance set labelling to build a learning-to-rank model. *Do-Rank* has four significant differences compared to PITF. Firstly, *Do-Rank* employs a *list-wise ranking model*, whereas PITF employs a *pair-wise ranking model*. PITF focuses on getting the ranking order within each item pair correctly, instead of getting the correct order in the entire items recommendation list as in *Do-Rank*. Secondly, the objective function of PITF is an AUC-based optimization. In contrast to DCG as utilized in *Do-Rank* that assigns higher penalty to lower ranked items, AUC assigns equal penalty to the mistakes done at the top or bottom positions in the recommendation list [9]. Thirdly, PITF infers the negative values of non-observed tagging data from all items that have not been annotated by the user using a tag, whereas, *Do-Rank* implements an efficient scheme which interprets the negative values from items that have never been annotated by the user using any other tags. Lastly, PITF is a tag recommendation method and *Do-Rank* is an item recommendation method. The tag and item recommendations are two distinct tasks. Tag recommendations are generated with two specified dimensions, i.e. user and item, while the item recommendations are made with specified users only and, therefore, the item predicted preference scores must be calculated for the whole available tags before being sorted as a list of top-$N$ recommendation.

To the best of our knowledge, this is the first work on tag-based item recommendation system that directly optimizes the ranking evaluation measure. Our contributions can be summarized as follows: (1) We propose a novel tag-based item recommendation method that directly optimizes a (smoothed) DCG for building the learning-to-rank model, (2) *Do-Rank* is the first tensor-based item recommendation ranking method with ordinal relevance set tagging data, and (3) We propose a fast learning-to-rank algorithm that implements an efficient tagging data interpretation scheme.

The remainder of this paper is organized as follows. Section 2 details the *Do-Rank* learning method. Section 3 presents the experimental results based on real-world datasets. Section 4 concludes the paper.

## 2. *DO-RANK*: DCG OPTIMIZATION FOR LEARNING TO RANK

Let $U = \{u_1, u_2, u_3, \dots, u_Q\}$ be the set of $Q$ users, $I = \{i_1, i_2, i_3, \dots, i_R\}$ be the set of $R$ items, and $T = \{t_1, t_2, t_3, \dots, t_S\}$ be the set of $S$ tags. The observed tagging data can be denoted as $A \subseteq U \times I \times T$, where a vector of $(u, i, t) \in A$ represents the observed tagging activity of user $u$ using tag $t$ to annotate item $i$.

### 2.1. Tensor based recommendation prediction model

The user tagging data can be naturally modelled as a three-dimensional tensor of $\mathcal{Y} \in \mathbb{R}^{Q \times R \times S}$. Figure 1 illustrates a tensor model representing a toy example of the observed tagging data, $\mathcal{Y} \in \mathbb{R}^{3 \times 4 \times 5}$ where $U = \{u_1, u_2, u_3\}$, $I = \{i_1, i_2, i_3, i_4\}$, and $T = \{t_1, t_2, t_3, t_4, t_5\}$. Each slice of the tensor represents a user matrix which contains the user tag usage for an item. The latent relationship between users, items, and tags can be inferred after decomposition. We use CP [15] as the predictor function in our model since Tucker, the other well-known decomposition technique, is more expensive in both memory and time [15]. As illustrated in Figure 2, CP factorizes a third-order tensor $\mathcal{Y} \in \mathbb{R}^{Q \times R \times S}$ into three factor matrices $\hat{U} \in \mathbb{R}^{Q \times F}$, $\hat{I} \in \mathbb{R}^{R \times F}$, $\hat{T} \in \mathbb{R}^{S \times F}$, and a diagonal core tensor $\mathcal{C} \in \mathbb{R}^{F \times F \times F}$, where $F$ is the column size of the corresponding reduced factor matrices. The predicted preference score is calculated as:

$$\hat{y}_{u,i,t} := \sum_{f=1}^{F} \hat{u}_{u,f} \cdot \hat{\imath}_{i,f} \cdot \hat{t}_{t,f} = [\![\hat{U}_u, \hat{I}_i, \hat{T}_t]\!] \qquad (1)$$

A predicted score reflects the preference level of a user in choosing an item for a tag. Research has shown that simply ranking the scored entries of tensor model does not produce quality recommendation [13]. Assuming that users show interest to a few top recommended items only [3], a recommendation model can be optimized with respect to the evaluation measure during the learning procedure [4] in order to generate quality recommendation by ranking the tensor entries most effectively.

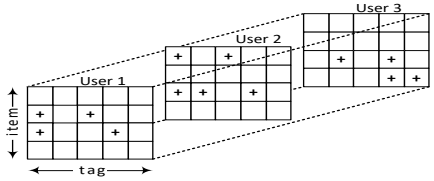



**Fig. 1.** A Tensor model showing a sample tagging data with observed (i.e. positive) entries

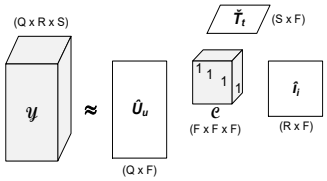



**Fig. 2.** CP factorization model for third-order tensor

The task now becomes to recommend an optimal (from the DCG perspective) item list to users using the latent factor matrices after decomposing input tensor model. In DCG, the correct order of higher ranked items is more important than that of the lower ranked items and, therefore, the higher positions have more influence on the score. The DCG score consists of discount and gain functions formulated as denominator and numerator in Equation (2) [4, 7]. The discount function makes items lower down in the ranked list contribute less to DCG score while the gain function gives weight to the items based on their relevance label. The DCG score for a user $u$ across all items $i$ under tag $t$ can be defined as:

$$\mathrm{DCG}_{u,t} := \sum_{i \in I} \frac{2^{y_{u,i,t}} - 1}{\log_2(1 + r_{u,i,t})} \qquad (2)$$

where $y_{u,i,t}$ is the relevance label and, is assigned as one of element in the ordinal relevance set of $\{negative, null, positive\}$

(or $\{-1, 0, 1\}$) from the initial tensor model. The $r_{u,i,t}$ is the ranking position of item $i$ for user $u$ with tag $t$ and, is approximated using $\hat{y}_{u,i,t}$ (which is calculated from the factor matrices using Equation (1)). The DCG score of all users over all items under all tags can be defined as:

$$\mathrm{DCG} := \frac{1}{\mathrm{QS}} \sum_{\mathrm{u}\in\mathrm{U}} \sum_{\mathrm{t}\in\mathrm{T}} \sum_{\mathrm{i}\in\mathrm{I}} \frac{2^{\mathrm{y_{u,i,t}}-1}}{\log_2(1+\mathrm{r_{u,i,t}})} \quad (3)$$

The item recommended list is generated for user $u$ by ranking all the items in descending order of the computed preference scores over all items under all tags.

### 2.2. Smoothed DCG and Optimization

It can be seen from Equation (3) that DCG is dependent on the ranking positions. The rankings change in a non-smooth way with respect to predicted relevance scores calculated based on the model parameters (i.e. factor matrices). The non-smooth function of DCG makes the application of standard optimization methods difficult [4, 16] which require smoothness in the objective function such as the gradient based approaches [4]. In this paper, we solve this problem by approximating the ranking position $r_{u,i,t}$ by a smoothed function with respect to the model parameters. Inspired by the Information Retrieval approach of "learning-to-rank" [4, 16], we approximate $r_{u,i,t}$ by the following smoothing function:

$$\mathrm{r_{u,i,t}} \approx 1 + \sum_{\mathrm{j}\neq\mathrm{i}} \sigma(\Delta\hat{y}) \quad (4)$$

where $\sigma(x)$ is the logistic function $\frac{1}{1+e^{-x}}$, and $\Delta\hat{y} = \hat{y}_{u,i,t} - \hat{y}_{u,j,t}$ is the predicted relevance scores difference for two items calculated from the decomposed tensor model (i.e. using Equation (1)). Substituting Equation (4) to Equation (3), we obtain the smoothed approximation of DCG:

$$\mathrm{sDCG} := \frac{1}{\mathrm{QS}} \sum_{\mathrm{u}\in\mathrm{U}} \sum_{\mathrm{t}\in\mathrm{T}} \sum_{\mathrm{i}\in\mathrm{I}} \frac{2^{\mathrm{y_{u,i,t}}-1}}{1+\log_2\left(\sum_{\mathrm{j}\neq\mathrm{i}} \sigma(\Delta\hat{y})\right)} \quad (5)$$

The resulted objective function can now be formulated as:

$$\mathrm{L}(\Theta) := \sum_{\mathrm{u}\in\mathrm{U}} \sum_{\mathrm{t}\in\mathrm{T}} \sum_{\mathrm{i}\in\mathrm{I}} \frac{2^{\mathrm{y_{u,i,t}}-1}}{1+\log_2\left(\sum_{\mathrm{j}\neq\mathrm{i}} \sigma(\Delta\hat{y})\right)} - \lambda_\Theta \|\Theta\|_{\mathrm{F}}^2 \quad (6)$$

where $\lambda_\Theta$ is the regularization coefficient corresponding to $\sigma_\Theta$ as model parameters that controls overfitting. Note that the constant coefficient ($\frac{1}{QS}$) in $sDCG$ can be neglected since it has no influence on the optimization. We can now perform gradient descent to optimize the objective function in Equation (6). Given a case $(u, i, t)$ with respect to the model parameters $\{\hat{U}_u, \hat{I}_i, \hat{T}_t\}$, the gradient of $sDCG$ can be computed as:

$$\frac{\partial L}{\partial \theta} = \sum_{\mathrm{u}\in\mathrm{U}} \sum_{\mathrm{t}\in\mathrm{T}} \sum_{\mathrm{i}\in\mathrm{I}} \frac{-\left(2^{\mathrm{y_{u,i,t}}-1}\right)\left[\frac{1}{\left(\left(\sum_{\mathrm{j}\neq\mathrm{i}} \sigma(\Delta\hat{y})\right)\ln 2\right)}\left(\frac{\partial}{\partial\theta}\left(\sum_{\mathrm{j}\neq\mathrm{i}} \sigma(\Delta\hat{y})\right)\right)\right]}{\left[1+\log_2\left(\left(\sum_{\mathrm{j}\neq\mathrm{i}} \sigma(\Delta\hat{y})\right)\right)\right]^2} - \lambda_\theta\theta \quad (7)$$

$$\frac{\partial \mathrm{L}}{\partial \theta} = \sum_{\mathrm{u}\in\mathrm{U}} \sum_{\mathrm{t}\in\mathrm{T}} \sum_{\mathrm{i}\in\mathrm{I}} \frac{-\left(2^{\mathrm{y_{u,i,t}}-1}\right)\left[\frac{1}{\left(\left(\sum_{\mathrm{j}\neq\mathrm{i}} \sigma(\Delta\hat{y})\right)\ln 2\right)}\left(\left(\sum_{\mathrm{j}\neq\mathrm{i}}\left(-\sigma(\Delta\hat{y})+\left(\sigma(\Delta\hat{y})\right)^2\right)\right)\frac{\partial}{\partial\theta}\Delta\hat{y}\right)\right]}{\left[1+\log_2\left(\sum_{\mathrm{j}\neq\mathrm{i}} \sigma(\Delta\hat{y})\right)\right]^2} - \lambda_\theta\theta \quad (8)$$

By substituting $\sigma(\Delta\hat{y})$ with $\delta$ we get:

$$\frac{\partial \mathrm{L}}{\partial \theta} = \sum_{\mathrm{u}\in\mathrm{U}} \sum_{\mathrm{t}\in\mathrm{T}} \sum_{\mathrm{i}\in\mathrm{I}} \frac{-\left(2^{\mathrm{y_{u,i,t}}-1}\right)\left[\frac{1}{\left(\ln 2 \sum_{\mathrm{j}\neq\mathrm{i}} \delta\right)}\left(\sum_{\mathrm{j}\neq\mathrm{i}}\left(-\delta+\delta^2\right)\frac{\partial}{\partial\theta}\Delta\hat{y}\right)\right]}{\left[1+\log_2\left(\sum_{\mathrm{j}\neq\mathrm{i}} \delta\right)\right]^2} - \lambda_\theta\theta \quad (9)$$

Based on Equation (9), we can see that we only have to compute $\frac{\partial}{\partial\theta}\Delta\hat{y}$ with respect to the model parameters to implement the $sDCG$ optimization. However, we can also see that directly optimizing $sDCG$ across all users is an expensive task as we need to compute the predicted relevance score difference $\Delta\hat{y}$ for all items pair-wise across all tags. Therefore, in the next subsection, we propose a fast learning algorithm that implements the efficient tagging data interpretation scheme so that the computation of $\Delta\hat{y}$ of all users is only required on observed or positive items across all tags.

### 2.3. Fast Learning

As per Equation (5), in order to optimize $sDCG$, we need to calculate the predicted relevance score difference $\Delta\hat{y}$ between each item with all other items in the system. We propose to solve this expensive process by employing a fast learning algorithm. The basic idea of the fast learning algorithm is to optimize $sDCG$, on each $(u, t)$ set, by calculating only the predicted relevance score difference $\Delta\hat{y}$ between items which have observed to show user interest and items which are not of user interest, i.e. positive and negative items, respectively. The key challenge here is to efficiently infer the positive and negative items from the tagging data, on each $(u, t)$ set. Our previous work [1] has demonstrated that the *User-Tag Set* ($UTS$) scheme efficiently interprets the observed and non-observed tagging data and labels each entry as one of element in the ordinal relevance set of $\{negative, null, positive\}$ when implemented on a pair-wise ranking model. In this paper, we propose to approximate $\Delta\hat{y}$ in Equation (5) using the $UTS$ scheme that is implemented on a *list-wise ranking model*.

***User-Tag Set* (*UTS*) scheme.** The *UTS* scheme, based on $(u, t) \in A$, interprets the positive and negative entries amongst items. The items of positive entries are derived from the observed data, while the items of negative entries are interpreted from items that have not been annotated by user $u$ using any other tags [1]. As illustrated in Figure 3(a) and (b), the positive entries show that $u_1$ has used $t_1$ to reveal his interest for items $\{i_2, i_3\}$, $t_3$ for $\{i_2\}$, and $t_4$ for $\{i_3\}$. This means that, on $(u_1, t_1)$ set, the positive and negative entries are $\{i_2, i_3\}$ and $\{i_1, i_4\}$, respectively, given $I = \{i_1, i_2, i_3, i_4\}$. Using the scheme, the input set consists of a list of tag assignment $A$ labelled with the corresponding relevance score $y_{u,i,t}$ using the following rules:

$$y_{u,i,t} := \begin{cases} 1 & if\ (u,i,t) \in A \\ -1 & if\ (u,i,t) \notin A\ and\ i \in I \backslash \{i | (u,i,*) \in A\} \\ 0 & otherwise \end{cases}$$


(a) Observed or positive entries (b) The *UTS* scheme

**Fig. 3.** Toy example for $u_1$

The fast learning algorithm based on the $UTS$ scheme infers the user's positive and negative items on each $(u, t) \in A$ and defines them as: (1) $ZP = \{i | y_{u,i,t} = 1\}$, positive items derived from the observed data, and (2) $ZN = \{i | y_{u,i,t} = -1\}$, negative items derived from the items that have not been tagged by $u$ using any other tags. The resultant objective function can be formulated by:

$$L(\Theta) := \sum_{u\in U}\sum_{t\in T}\sum_{i\in ZP}\frac{2^{y_{u,i,t}}-1}{1+\log_2\left(\sum_{j\in ZN}\sigma(\Delta\hat{y})\right)} - \lambda_\Theta\|\Theta\|_F^2 \quad (10)$$

where $\Delta\hat{y} = \hat{y}_{u,i,t} - \hat{y}_{u,j,t}$. The gradient of $sDCG$ given a case $(u,i,j,t)$ with respect to the model parameter $= \{\hat{U}_u, \hat{I}_i, \hat{I}_j, \hat{T}_t\}$ is given by Equation (11).

$$\frac{\partial L}{\partial\theta} = \sum_{u\in U}\sum_{t\in T}\sum_{i\in ZP}\frac{-\left(2^{y_{u,i,t}}-1\right)\left[\frac{1}{\left(\ln 2\sum_{j\in ZN}\delta\right)}\left(\sum_{j\in ZN}\left(-\delta+\delta^2\right)\frac{\partial}{\partial\theta}\Delta\hat{y}\right)\right]}{\left[1+\log_2\left(\sum_{j\in ZN}\delta\right)\right]^2} - \lambda_\theta\theta \quad (11)$$

where $\delta = \sigma(\Delta\hat{y})$. To apply the $sDCG$ optimization, we only have to compute the gradient of $\frac{\partial}{\partial\theta}\Delta\hat{y}$. The gradients for the model based on its parameters are: $\frac{\partial\Delta\hat{y}}{\partial\hat{U}_u} = \left(\hat{I}_i\odot\hat{T}_t - \hat{I}_j\odot\hat{T}_t\right)$, $\frac{\partial\Delta\hat{y}}{\partial\hat{I}_i} = \left(\hat{U}_u\odot\hat{T}_t\right)$, $\frac{\partial\Delta\hat{y}}{\partial\hat{I}_j} = -\left(\hat{U}_u\odot\hat{T}_t\right)$, $\frac{\partial\Delta\hat{y}}{\partial\hat{T}_t} = \left(\hat{U}_u\odot\hat{I}_i - \hat{U}_u\odot\hat{I}_j\right)$, where $\odot$ denotes element-wise product. From Equation (11), we can see that for optimizing $sDCG$ across all users and under all tags, we only need to compute $\Delta\hat{y}$ for each $ZP$ that is less computationally expensive than computing $\Delta\hat{y}$ for each $R$, since $|ZP| \ll R$. The *Do-Rank* learning algorithm is outlined in Figure 4.

**Algorithm: *Do-Rank* Learning**
**Input :** Training set $D_{train} \subseteq U \times I \times T$, learning rate $\alpha$, factor matrix column size $F$, regularization $\lambda$, maximal iteration $iterMax$
**Output:** Learned factor matrices $\hat{U}$, $\hat{I}$, $\hat{T}$
$Q = |U|$, $R = |I|$, $S = |T|$, $\mathcal{Y} \in \mathbb{R}^{Q\times R\times S}$
$ZP = \{i|y_{u,i,t} = 1\}$, $ZN = \{i|y_{u,i,t} = -1\}$
Initialize $\hat{U}^{(0)} \in \mathbb{R}^{Q\times F}$, $\hat{I}^{(0)} \in \mathbb{R}^{R\times F}$, $\hat{T}^{(0)} \in \mathbb{R}^{S\times F}$, $h = 0$
$g_0 = DCG$ based $\mathcal{Y}$ and $\hat{U}^{(0)}, \hat{I}^{(0)}, \hat{T}^{(0)}$
**repeat**
  **for** $u \in U$ **do**
    $\hat{U}_u \leftarrow \hat{U}_u + \alpha\frac{\partial L}{\partial\hat{U}_u}$ based on Equation (11)
  **for** $t \in T$ **do**
    $\hat{T}_t \leftarrow \hat{T}_t + \alpha\frac{\partial L}{\partial\hat{T}_t}$ based on Equation (11)
  **for** $u \in U$ **do**
    **for** $t \in T$ **do**
      **for** $i \in ZP$ **do**
        **for** $j \in ZN$ **do**
          $\hat{I}_i \leftarrow \hat{I}_i + \alpha\frac{\partial L}{\partial\hat{I}_i}$ based on Equation (11)
          $\hat{I}_j \leftarrow \hat{I}_j + \alpha\frac{\partial L}{\partial\hat{I}_j}$ based on Equation (11)
  $++h$
  $g = DCG$ based $\mathcal{Y}$ and $\hat{U}^{(h)}, \hat{I}^{(h)}, \hat{T}^{(h)}$
  **if** $g - g_0 \le 0$
    break
**until** $h \ge iterMax$

**Fig. 4.** *Do-Rank* Learning Algorithm

### 2.4. Complexity Analysis and Convergence

We analyze the complexity of learning process for a single iteration. The initial *Do-Rank* complexity is $O\left(F|A|(Q+S+QSR^2)\right)$. When the fast learning approach is implemented, the *Do-Rank* complexity (illustrated in Figure 4), becomes $O\left(F|A|(Q+S+QS\tilde{p}\tilde{n})\right)$, where $\tilde{p}$ and $\tilde{n}$ denote the average number of $ZP$ and $ZN$ per $(u,t)$ set. Since $\tilde{p},\tilde{n} \ll R$, the *Do-Rank* complexity now becomes $O\left(F|A|(Q+S+QSR)\right)$.

The objective function of *Do-Rank* is optimizing $sDCG$ as given in Equation (5) that is the smoothed approximation of DCG as given in Equation (3). We use the iterated DCG scores during the optimization process as the termination criterion [9], instead of using the conventional criteria such as the number of iterations [10] and the convergence rate [8]. Optimization process is terminated when DCG scores start declining.

## 3. EMPIRICAL ANALYSIS

We used two real-world tagging datasets, detailed in Table 1, to implement the proposed tag-based item recommendation method *Do-Rank*. Adapting standard practice of eliminating noise and decreasing data sparsity [12], datasets are refined by using $p$-core technique, i.e. selecting users, items, and tags that have occurred in at least $p$ number posts. Post is the set of distinct user-item sets in the observed tagging data.

**Table 1.** The details of datasets

| Dataset | #Users ($Q$) | #Items ($R$) | #Tags ($S$) | #Observed Data | $p$-core |
|---|---|---|---|---|---|
| LastFM | 867 | 1,715 | 1,423 | 99,211 | 10 |
| MovieLens | 571 | 1,684 | 1,559 | 25,103 | 5 |

We implemented the 5-fold cross-validation. For each fold, we randomly divide into a training set $D_{train}$ (80%) and a test set $D_{test}$ (20%) based on the number of posts data. $D_{train}$ and $D_{test}$ do not overlap in posts, i.e., there exist no triplets for a user-item combination in the training set if a triplet $(u,i,*)$ is present in the test set. The recommendation task is to predict and rank the Top-$N$ items for the users present in $D_{test}$ according to DCG. The evaluation metrics [4] used to measure the recommendation performance are (1) NDCG, Normalized DCG score, presented at various top-$N$ positions, and (2) MAP, Mean Average Precision (AP).

Following the initialization approach proposed for a ranking model [7], we randomly initialized the factor matrices of our model. In order to empirically tune the parameters in *Do-Rank*, we randomly selected 25% of all the observed data available in $D_{train}$. The best performances were achieved using the following values: learning rate parameter $\alpha = 0.01$ and regularization parameter $\lambda = 1e^{-05}$. We used $F = 128$ as the size of latent factor matrix.

### 3.1. Scalability of *Do-Rank*: Impact of Fast Learning Approach

We first investigated the impact of implementing the fast learning approach on optimizing $sDCG$ as defined in Equation (10), by measuring the running time for learning the model on a single iteration at different scales, i.e. 10% to 100% of $D_{train}$. Figures 5(a) and 5(b) show that the learning time of "fast learning" approach, on the LastFM and Movielens datasets respectively, is linear to the size of data, i.e. determined by the size of items $R$. The "regular" approach, i.e. optimizing $sDCG$ without implementing fast learning approach requires more learning time as the computational complexity is determined by $R^2$, as previously described in Section 2.4. These results confirm that the *User-Tag Set* ($UTS$) scheme employed in this algorithm efficiently interprets the tagging data and, the pair-wise difference between the positive and negative item entries is sufficient for determining effective ranking instead of calculating difference between all pair of items in the dataset.

### 3.2. Effectiveness of Convergence Criterion

Figures 6(a) and 6(b) show the evolution of DCG@10 across iterations on the training and test sets respectively. DCG increases through early iterations on both sets, before the performance is declined. It ascertains that *Do-Rank* is able to effectively optimize DCG. We can notice that the DCG measure drops after a few iterations (less than 15) which indicates that using a measure score as termination criterion is a useful approach in order to avoid the model to overfit [9].

### 3.3. Performance Comparison

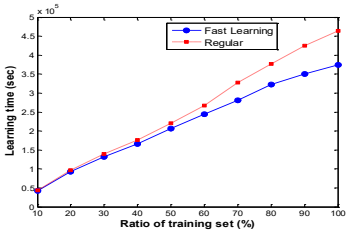
The performance of *Do-Rank* is compared with the following methods.

- **PITF [8]**. We have adapted PITF [8] to generate item recommendations based on the user-tag sets. PITF is the pair-wise tensor factorization model for generating tag recommendations that implements the ordinal relevance set labelling to build the *pair-wise ranking model* on each user-item set. The tuned parameters are $\alpha = 0.01$, $\lambda = 5e^{-05}$, and $F = 128$.
- **CP-TRPR** [13]. A probabilistic ranking tensor-based method that ranks the predicted preference scores, calculated from the decomposed models, by utilizing the users past collaborative tagging data. The tuned parameters for this method are $tolerance = 1e^{-04}$, $\lambda = 0$, $voc\_size = 20$, and $F = 128$.
- **CTS** [17]. The state-of-the-art matrix-based method that ranks the recommendations by using the users past tagging activities in forming users' likelihood. The tuned parameters are $neigborhood\_size = 50$ and $model\_size = 20$.

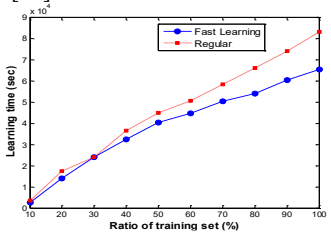
The recommendation performance comparisons of the proposed *Do-Rank* and the benchmarking methods on LastFM and MovieLens datasets are listed in Table 2 and Table 3, respectively. We can observe that *Do-Rank* outperforms the benchmarking methods in terms of NDCG (at any top-*N* positions) on both datasets. It can be noted that the higher the top-*N* position, the less the NDCG score is. In terms of MAP, *Do-Rank* is still able to outperform other methods on the LastFM dataset.

Compared to PITF, an AUC-based optimization approach which gives equal penalty to the mistakes at the top and bottom list of recommendations [9], *Do-Rank* enhances the top-*N* recommendation performance by optimizing the top-biased measure DCG. This confirms that optimizing top-*N* recommendation evaluation measure for building the learning model will improve the recommendation performance. Additionally, PITF is *pair-wise ranking model* that aims to get the ranking order within each pair correctly, while a DCG measure objective is to get the correct order of all lists, i.e. list-wise measure.

Both *Do-Rank* and CP-TRPR utilize the users past tagging history to correctly rank the order of items that might interest users. However, CP-TRPR interprets the tagging data as a binary relevance set to build the model. In other word, the method simply considers the observed entries as "1" and overfits the negative and null values inferred from the non-observed entries as "0". This inappropriate tagging data interpretation impacts the recommendation performance [2]. Moreover, the model was designed to solve the classification problem by minimizing the Mean Square Error (MSE) [13]. Finally, *Do-Rank* outperformance towards CTS is again proving that three-dimensional characteristic of tagging data must be captured so that the many-to-many relationships that exist among the dimensions can be kept rather than projecting the three-dimension into two-dimensions [12].

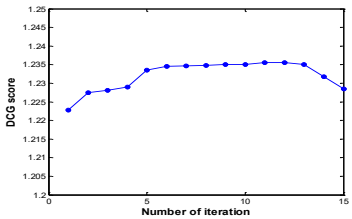


(a) LastFM dataset

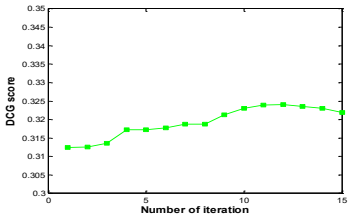


(b) MovieLens dataset

**Fig. 5.** Impact of fast learning approach



(a) Training set



(b) Test set

**Fig. 6.** Effectiveness of convergence criterion in the learning procedure

**Table 2.** NDCG at top-*N* position and MAP on the LastFM dataset

| Methods | Score (%) | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | NDCG@1 | NDCG@2 | NDCG@3 | NDCG@4 | NDCG@5 | NDCG@10 | NDCG@20 | MAP |
| PITF | 9.06 | 9.01 | 8.17 | 7.90 | 7.56 | 6.45 | 5.45 | 5.96 |
| CP-TRPR | 7.12 | 7.67 | 7.08 | 6.67 | 6.56 | 5.68 | 4.79 | 5.20 |
| CTS | 6.60 | 5.65 | 5.11 | 4.90 | 4.87 | 4.17 | 3.50 | 3.87 |
| *Do-Rank* | **9.57** | **9.37** | **8.66** | **8.38** | **8.15** | **7.05** | **5.98** | **6.50** |

**Table 3.** NDCG at top-$N$ position and MAP on the MovieLens dataset

| Methods | Score (%) | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | NDCG@1 | NDCG@2 | NDCG@3 | NDCG@4 | NDCG@5 | NDCG@10 | NDCG@20 | MAP |
| PITF | 3.02 | 2.77 | 2.63 | 2.70 | 2.60 | 2.42 | 2.26 | 2.70 |
| CP-TRPR | 5.18 | 4.85 | 4.27 | 4.02 | 3.72 | 3.17 | 2.60 | **3.54** |
| CTS | 4.10 | 3.44 | 3.14 | 3.19 | 3.25 | 2.74 | 2.43 | 3.31 |
| *Do-Rank* | **5.40** | **4.86** | **4.29** | **4.04** | **3.97** | **3.26** | **2.68** | **3.21** |

## 4. CONCLUSION AND FUTURE WORK

We have presented *Do-Rank*, a top-*N* tag-based item recommendation method that directly optimizes the (smoothed) DCG in building the learning model for generating an ordered list of items that might interest the user. Entries in the tensor-based model are generated from the ordinal relevance set tagging data. We presented a fast learning approach that implements the *User-Tag Set* (*UTS*) tagging data interpretation scheme, and enables efficient execution of *Do-Rank*. The experimental results on real datasets show that *Do-Rank* is scalable and outperforms all benchmarking methods on the NDCG measure. This ascertains that optimizing DCG for building the learning model improves the recommendation performance. For the future work, we are planning to investigate the implementation of DCG optimization on other ranking models and the potential of optimizing other measures for tag-based item recommendation as different measures possibly yield different recommendation performances.

## REFERENCE


1. Ifada, N.,Nayak, R.: An Efficient Tagging Data Interpretation and Representation Scheme for Item Recommendation. In: The 12th Australasian Data Mining Conference. Brisbane, Australia: ACS. (2014).
2. Rendle, S., Balby Marinho, L., Nanopoulos, A., Schmidt-Thieme, L.: Learning Optimal Ranking with Tensor Factorization for Tag Recommendation. In: The 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. pp. 727-736. Paris, France: ACM. (2009).
3. Cremonesi, P., Koren, Y., Turrin, R.: Performance of Recommender Algorithms on Top-N Recommendation Tasks. In: The 4th ACM Conference on Recommender Systems. pp. 39-46. Barcelona, Spain: ACM. (2010).
4. Chapelle, O.,Wu, M.: Gradient Descent Optimization of Smoothed Information Retrieval Metrics. Information Retrieval, 13(3): pp. 216-235. (2010).
5. Xu, J.,Li, H.: AdaRank: A Boosting Algorithm for Information Retrieval. In: The 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval. pp. 391-398. Amsterdam, The Netherlands: ACM. (2007).
6. Liu, T.-Y.: Learning to Rank for Information Retrieval. Foundations and Trends in Information Retrieval, 3(3): pp. 225-331. (2009).
7. Balakrishnan, S.,Chopra, S.: Collaborative Ranking. In: The 5th ACM International Conference on Web Search and Data Mining. pp. 143-152. Seattle, Washington, USA: ACM. (2012).
8. Rendle, S.,Schmidt-Thieme, L.: Pairwise Interaction Tensor Factorization for Personalized Tag Recommendation. In: The 3rd ACM International Conference on Web Search and Data Mining. pp. 81-90. New York, USA: ACM. (2010).
9. Shi, Y., Karatzoglou, A., Baltrunas, L., Larson, M., Hanjalic, A., Oliver, N.: TFMAP: Optimizing MAP for Top-N Context-Aware Recommendation. In: The 35th International ACM SIGIR Conference on Research and Development in Information Retrieval. pp. 155-164. Portland, Oregon, USA: ACM. (2012).
10. Shi, Y., Karatzoglou, A., Baltrunas, L., Larson, M., Oliver, N., Hanjalic, A.: CLiMF: Learning to Maximize Reciprocal Rank with Collaborative Less-is-More Filtering. In: The 6th ACM Conference on Recommender Systems. pp. 139-146. Dublin, Ireland: ACM. (2012).
11. Weimer, M., Karatzoglou, A., Le, Q.V., Smola, A.: Maximum Margin Matrix Factorization for Collaborative Ranking. Advances in Neural Information Processing Systems. (2007).
12. Symeonidis, P., Nanopoulos, A., Manolopoulos, Y.: A Unified Framework for Providing Recommendations in Social Tagging Systems Based on Ternary Semantic Analysis. IEEE Transactions on Knowledge and Data Engineering, 22(2): pp. 179-192. (2010).
13. Ifada, N.,Nayak, R.: A Two-Stage Item Recommendation Method Using Probabilistic Ranking with Reconstructed Tensor Model, in User Modeling, Adaptation, and Personalization. Springer. p. 98-110. (2014).
14. Karatzoglou, A., Amatriain, X., Baltrunas, L., Oliver, N.: Multiverse Recommendation: n-Dimensional Tensor Factorization for Context-Aware Collaborative Filtering. In: The 4th ACM conference on Recommender Systems. pp. 79-86. Barcelona, Spain: ACM. (2010).
15. Kolda, T.,Bader, B.: Tensor Decompositions and Applications. SIAM Review, 51(3): pp. 455-500. (2009).
16. Wu, M., Chang, Y., Zheng, Z., Zha, H.: Smoothing DCG for Learning to Rank: A Novel Approach using Smoothed Hinge Functions. In: The 18th ACM Conference on Information and Knowledge Management. pp. 1923-1926. Hong Kong, China: ACM. (2009).
17. Kim, H.-N., Ji, A.-T., Ha, I., Jo, G.-S.: Collaborative Filtering based on Collaborative Tagging for Enhancing the Quality of Recommendation. Electronic Commerce Research and Applications, 9(1): pp. 73-83. (2010).