2016_WSDM

by Noor Ifada

Submission date: 09-Dec-2019 12:56PM (UTC+0700)
Submission ID: 1230369578
File name: tagging_data_for_a_learning-to-rank_model_accepted_version.pdf (1.65M)
Word count: 10107
Character count: 49447

How Relevant is the Irrelevant Data: Leveraging the Tagging Data for a Learning-to-Rank Model

Noor Ifada[1,2], Richi Nayak[1]
[1]Queensland University of Technology (QUT), Australia
[2]University of Trunojoyo Madura (UTM), Indonesia
noor.ifada@{qut.edu.au, if.trunojoyo.ac.id}, r.nayak@qut.edu.au

ABSTRACT

For the task of tag-based item recommendations, the underlying tensor model faces several challenges such as high data sparsity and inferring latent factors effectively. To overcome the inherent sparsity issue of tensor models, we propose the *graded-relevance* interpretation scheme that leverages the tagging data effectively. Unlike the existing schemes, the *graded-relevance* scheme interprets the tagging data richly, differentiates the non-observed tagging data insightfully, and annotates each entry as one of the "relevant", "likely relevant", "irrelevant", or "indecisive" labels. To infer the latent factors of tensor models correctly to produce the high quality recommendation, we develop a novel learning-to-rank method, *Go-Rank*, that optimizes Graded Average Precision (GAP). Evaluating the proposed method on real-world datasets, we show that the proposed interpretation scheme produces a denser tensor model by revealing "relevant" entries from the previously assumed "irrelevant" entries. Optimizing GAP as the ranking metric, the quality of the recommendations generated by *Go-Rank* is found superior against the benchmarking methods.

Keywords

Item recommendation, Tagging data, Graded-relevance scheme, Graded average precision

1. INTRODUCTION

In the websites that allow users to annotate items with tags, a ternary correlation is naturally formed. Over a period, the ternary data recording the users tagging activities can be accumulated that reflect the user profiles. This data includes complex relationships as users are allowed annotating the same item with different tags as well as annotating different items with the same tag. Analysis of this data enables discovering latent factors that govern the ternary correlations. A tag-based item recommendation system can utilize the captured correlation of tagging data and predict the list of items which may be of interest to a user. The success of such systems highly depends upon how those correlations are exploited [1].

The multi-dimensional tagging data can naturally be modeled with a tensor model, and the latent correlations inherent in the data can be inferred [2-4]. However, generating recommendations using a tensor model is not a trivial task. It imposes several challenges including the following two that are the focus of this paper.

Data sparsity. Each user uses and annotates only a small number of tags and items respectively. This means that only a few entries are observed per user and, consequently, the tensor model becomes highly sparse [5-7]. The interpretation of tagging data is crucial in solving the sparsity problem in tensor models [5, 8]. Given that tagging data holds records of existing entries, the observed entries can be interpreted in a straightforward manner from the data. In contrast, interpreting the non-observed data is a non-trivial task. Currently, there exist two interpretation schemes. According to the *boolean* scheme, non-observed entries are labelled as "0" and the observed entries are labelled as "1" [4]. According to the *set-based* scheme, non-observed entries are interpreted as a mixture of "irrelevant" and "indecisive" entries, whereas the observed or "relevant" entries are given higher values than that of the "irrelevant" ones [5]. The "irrelevant" entries are entries that users do not like, while the "indecisive" are entries that users might like in the future, i.e. entries to be predicted by the recommender system. The *boolean* scheme certainly faces the sparsity problem as the non-observed data dominate a tensor model. The *set-based* scheme has shown to solve this problem; however, it over-generalizes the "irrelevant" entries. The scheme interprets that, on each observed $(user, tag)$ set, any items other than those appearing in observed entries are "irrelevant" [5]. We have recently shown that the *set-based* interpretation leads inaccurate result and presented that the "irrelevant" entries should only be interpreted from items which have not been annotated by the user using any tags, on each observed $(user, tag)$ set [8]. It is of critical importance that how such non-observed data entries, which should not simply be regarded as "irrelevant" entries, be interpreted. Those entries are certainly not the "indecisive" entries as the items of the entries have already been selected by the user in the past and, for that reason, we are not interested to predict them in the future. An efficient interpretation scheme that can further differentiate the non-observed data and overcome the sparsity problem is in sighted.

Learning from the latent factors. 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' tagging history. The recommended items are ordered in descending order based on the predicted preference score which reflects the preference level of a user in selecting an item based on a tag. The score is calculated from the latent factors that govern the tagging data ternary correlations. In practice, users show more interest to the fewer items at the top of the list rather than the ones further down [9] and, therefore, the order of items in the list is crucial. In this case, the recommendation task can be considered as a ranking problem and can be approached as "learning-to-rank" for generating a quality top-N recommendation list [9-11]. For this reason, the latent factors of a tensor model can be learned so as to directly optimize ranking metric. The metrics such as

Mean Average Precision (MAP) and Mean Reciprocal Rank (MRR) have been formulated to handle the binary relevance data [12, 13]. Unfortunately, these metrics are unsuitable for learning the latent factors of tensor model if the tagging data is interpreted as ordinal relevance data [10]. Though the NDCG (Normalized Discount Cumulative Gain) metric can handle ordinal relevance data [10], it is unsuitable to handle graded data with "transitional" entries, as described in the previous challenge. An alternative metric that can handle the aforementioned graded data is required as the object of optimization for learning the latent factors of the tensor model.

In this paper we address the above two challenges by proposing a *graded-relevance* interpretation scheme to effectively leverage the tagging data, and a novel learning-to-rank method that directly optimizes a graded evaluation metric to learn the latent factors of the tensor model. The proposed *graded-relevance* scheme interprets the tagging data with four distinct entries, i.e. "relevant", "likely relevant", "irrelevant", and "indecisive". The scheme sets the entries of non-observed data, which are not "irrelevant", as the transitional entries between the "relevant" and "irrelevant" entries. Each tagging data entry can then be graded with one of the ordinal relevance values $\in \{2,1,0,-1\}$. The learning-to-rank method, called as *Go-Rank*, generates an optimal list of recommended items from the Graded Average Precision (GAP) [14] perspective for all users. GAP works as the generalization of Average Precision (AP) for ordinal relevance data. Using GAP as the optimized ranking metric allows the learning model to set up thresholds so that the "likely relevant" entries can be regarded as either "relevant" or "irrelevant" entries. Experimental results on real-world datasets show that *Go-Rank* outperforms state-of-the-art recommendation methods in recommendation quality. It ascertains that the efficient interpretation scheme solves the sparsity problem and assists the "learning-to-rank" method, which directly optimizes GAP as the appropriate ranking metric, to produce high quality recommendation.

To the best of our knowledge, this is the first work on tag-based item recommendation item that leverages the tagging data as graded relevance data. Our contributions can be summarized as follows: (1) a novel *graded-relevance* interpretation scheme to leverage the tagging data, (2) a novel tag-based item recommendation method, *Go-Rank*, that directly optimizes a (smoothed) GAP for learning the latent factors of tensor model built with data labelled using the *graded-relevance* scheme. This is the first time GAP has been proposed to optimize a learning-to-rank model for recommendation method using tagging data as implicit feedback.

The remainder of this paper is organized as follows. Section 2 presents the related works. Section 3 describes the proposed *graded-relevance* scheme. Section 4 details the *Go-Rank* learning method. Section 5 presents the experimental results based on real world datasets. Section 6 concludes the paper.

2. RELATED WORK

Tensor models have been successfully used to represent and analyze the latent relationships inherent in three-dimensional tagging data [3, 4, 6, 15]. The tensor model, used as a predictor function, maps the ternary latent correlations inherent in tagging data to a predicted preference score. These scores can then be used in ranking the recommendation items. The tensor based recommendation methods have shown improved results over the matrix based methods [4, 6]. The selection of tagging data interpretation scheme is crucial for tensor models as it determines how the user profiles are built based on the users past tagging histories. While the tagging data holds records of observed tagging entries, existing interpretation schemes vary in the way they interpret the non-observed entries.

The *boolean* scheme [4] simply labels the non-observed tagging entries as "0" and the observed ones as "1". The tensor model built with data labelled using this scheme results in highly sparse model as the non-observed data usually dominate the tensor entries. Moreover, this scheme overfits the mixed values that can be revealed from the non-observed entries, as using a one single value of "0" makes it difficult to predict the ranking list that might interest users in future [5]. The *set-based* scheme [5] was proposed to tackle the problems of *boolean* scheme and has shown more accurate interpretation of the tagging data. This scheme interprets non-observed data as a mixture of "irrelevant" and "indecisive" entries. The "irrelevant" entries are entries that users do not like, while the "indecisive" are entries that users might like in the future, i.e. entries to be predicted by the recommendation system. In this scheme, the rules of labelling are typically assigning higher values to the observed or "relevant" entries in comparison to "irrelevant" ones [5]. This scheme results in dense data. However, to generate quality recommendations, an efficient scheme is more important instead of just getting more entries [8].

Our recent study has shown that the *set-based* scheme over-generalizes the "irrelevant" entries [8] and can mislead the recommendation performance. According to this scheme, on each observed $(user, tag)$ set, any items other than those appearing in observed entries are labelled as "irrelevant" [5]. It disregards the fact that some of those items may have been annotated by the user using other tags. Principally, on each observed $(user, tag)$ set, only items, that have not been annotated by the user using any tags, should be interpreted as "irrelevant" [8, 16]. This opens up a new problem of how those entries of non-observed data, which should not simply be regarded as "irrelevant", are interpreted. Those entries cannot be regarded as "indecisive" entries since items of the entries have already been selected by the user in the past and therefore we are not interested to predict them in the future. In this paper, we propose a novel *graded-relevance* scheme that sets those entries as the transitional entries between the "relevant" and "irrelevant" entries, and label them as "likely relevant". As a result, our proposed interpretation scheme generates tagging data entries that are identical to explicit rating data.

Several Information Retrieval researches have listed that users show more interest to the fewer items at the top of the list than the ones further down the list [9]. This makes the "learning-to-rank" approach suitable for generating a quality top-N recommendation list [9-11]. In a tensor recommendation model, the latent factors of the ternary correlations can be learned so as to directly optimize ranking metric. In the last few years, researchers have proposed the recommendation models optimized based on MAP [12] and MRR [13], but these models can only deal with binary relevance data. Since the tag-based recommendation systems use the ordinal relevance data, these models are not suitable.

For ordinal relevance data, a well-known tag recommendation method is PITF [17]. The recommendation model is built by creating pairs of items as its focus is to get the ranking order within each items pair correctly, instead of getting the correct order of the entire items in recommendation list. The objective function is based on Area Under the ROC Curve (AUC) optimization that undesirably assigns equal penalty to all mistakes done in the recommendation list regardless of their positions, such as top or bottom [12]. PITF is based on the *set-based* scheme, consequently it overgeneralizes

the "irrelevant" entries of the non-observed data. As a tag recommendation method, PITF generated the recommendations with two specified dimensions, i.e. user and item. In contrast, in our work, we generate item recommendations that 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.

We have recently proposed a tag-based item recommendation method, Do-Rank [16], that implemented a more efficient version of the *set-based* scheme to interpret the tagging data and used Discount Cumulative Gain (DCG) as the learning model optimization criterion. However, given the characteristic of entries generated from the *graded-relevance* scheme proposed in this paper, a recommendation method will need an optimization criterion that allows the learning model to control when to regard the "likely relevant" entries as "relevant" or 'irrelevant". Consequently, Do-Rank or using (N)DCG as the optimization criterion for learning model will be inappropriate.

Researchers have proposed recommendation methods based on optimizing NDCG [18] and GAP [19] by using rating data as the explicit feedback. Our problem is considerably different and difficult in comparison to those works due to the nature of ordinal rating data generated in our model. A recommendation method with explicit rating data builds its model by collecting the rating values given on each $\langle user, item \rangle$ binary correlation. The list of recommendations is then generated from the ranked predicted preference scores inferred from the non-observed $\langle user, item \rangle$ correlations [18, 20]. In contrast, our model is built from the $\langle user, item, tag \rangle$ ternary correlations tagging data and the ordinal rating values are given on each $(user, tag)$ set. This means that there exist multiple rating values on each $(user, tag)$ set, instead of just one rating value like the one given for the explicit feedback data. Eventually, the recommendation list needs to be generated by ranking the predicted preference scores for all items, under all tags, which may be interest to a user.

In this paper, we propose a novel learning-to-rank method that optimizes GAP for the case of ordinal relevance data of implicit feedback, called as *Go-Rank*, and generates a list of recommended items from the GAP perspective to all users. Using GAP in our model allows us to set thresholds in the learning model so that the "likely relevant" entries can be regarded as either "relevant" or "irrelevant" entries.

3. LEVERAGING TAGGING DATA

Tagging data consists of observed entries which form the ternary correlations between users, items and tags. The observed entries can be labelled as "relevant" in which users have expressed their interest to items explicitly by annotating those items using tags. 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. Tagging data can be denoted as $A \subseteq U \times I \times T$, where a vector of $(u, i, t) \in A$ represents the activity of user u to annotate item i using tag t.

The ternary correlation within tagging data can be naturally modelled as a three-dimensional tensor of $\mathcal{Y} \in \mathbb{R}^{Q \times R \times S}$ where each tensor entry, $y_{u,i,t}$, is given a numerical value that represents the relevance grade of tagging activity. Each slice of the tensor represents a user matrix which contains the user tag usage for each item. The tensor $\mathcal{Y}$ is then used as the ranking learning model.

In this paper, we propose an extended set of possible tensor entries and derive a corresponding optimization metric for learning the tensor model. In the remainder of this section, we first explain why previously proposed entry choices are not optimal (Section 3.1) and present a new set of possible entries based on the idea that, on each observed (u, t) set, the entries of non-observed data cannot simply be regarded as "irrelevant" entries. The proposed *graded-relevance* interpretation scheme for generating the optimal tensor entries is one of the main contributions of this paper.

3.1 Data Interpretation: Problem

We present a tensor toy example of $\mathcal{Y} \in \mathbb{R}^{3 \times 4 \times 5}$ in Figure 1 where $U = \{u_1, u_2, u_3\}$, $I = \{i_1, i_2, i_3, i_4\}$ and $T = \{t_1, t_2, t_3, t_4, t_5\}$. The "+" symbols represent the observed entries of tagging data. For instance, the first slice in Figure 1 shows that user u_1 has tagged item i_2 with tag t_1 and t_3. A typical tag-based item recommendation system interprets tagging data A using the *boolean* scheme, as illustrated in Figure 2(a), to build the tensor learning model $\mathcal{Y}$. This scheme uses the following rules for interpreting the data:

$$y_{u,i,t} := \begin{cases} 1 & if\ (u,i,t) \in A \\ 0 & otherwise \end{cases} \quad (1)$$

Figure 2(a) reveals two shortcomings of *boolean* scheme: (1) the sparsity problem – 0 values dominate the data and (2) the overfitting problem – all non-observed entries are denoted as 0, however, other values can be revealed from them [5]. The *set-based* scheme [5] can overcome these drawbacks. Using the *set-based* scheme, the non-observed entries can be differentiated as "irrelevant" and "indecisive" according to each observed (u, t) set. The rules of labelling are as follows: higher ordinal values are assigned to the observed or "relevant" entries and labelled with "1" value, whereas "irrelevant" entries (i.e. users do not like) are labelled with "-1" value. The "0" value is used to label "indecisive" entries that can be predicted for generating recommendations. The *set-based* scheme uses the following rules:

$$y_{u,i,t} := \begin{cases} 1 & if\ (u,i,t) \in A \\ -1 & if\ (u,i,t) \notin A\ and\ (u,*,t) \in A \\ 0 & otherwise \end{cases} \quad (2)$$

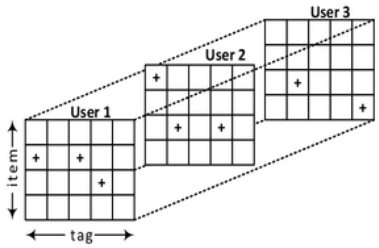


Figure 1. A tensor toy example of observed entries

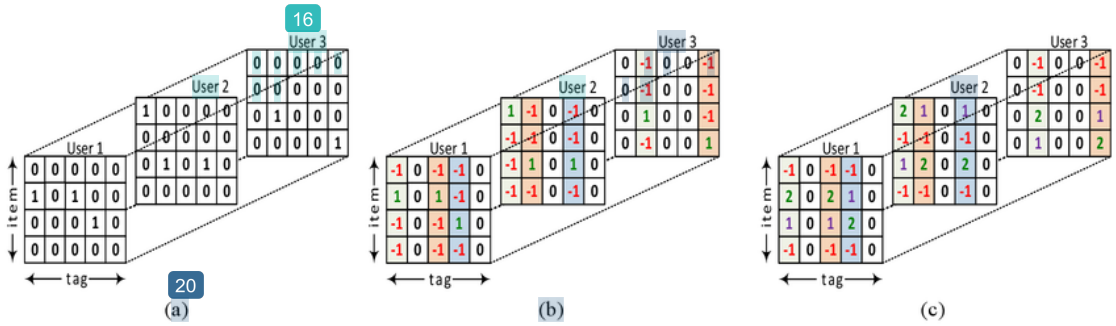


Figure 2: The entries of tensor $\mathcal{Y} \in \mathbb{R}^{3\times4\times5}$ generated from various interpretation schemes: (a) *boolean*, (b) *set-based*, and (c) *graded-relevance*

Our previous study [8] has shown that the *set-based* scheme is incorrect as it over-generalizes the "irrelevant" entries and results in inferior recommendation performance. On each observed (u, t) set, the scheme interprets any items other than those appearing in observed entries as "irrelevant" entries [5] and disregards the fact that some of those items have been annotated by the user using other tags. Actually, only the items, on each observed (u, t) set, that have not been annotated by user u using any tags should be interpreted as "irrelevant" entries [8]. Yet, how to interpret the entries of non-observed data that should not simply be regarded as "irrelevant" remains unsolved.

We use the entries of User 1 (u_1) in Figure 1 to better explain the issue. Using the *set-based* scheme as illustrated in Figure 2(b), we can list the observed and non-observed entries based on each (u, t) set of the user. The observed entries show that there exist (u_1, t_1), (u_1, t_3), and (u_1, t_4) sets that have been used to annotate $\{i_2\}$, $\{i_2\}$, and $\{i_3\}$, respectively. These observed entries are regarded as "relevant" entries while all non-observed entries of non-existed set, i.e. (u_1, t_2) and (u_1, t_5) on any items can easily be interpreted as "indecisive" entries. The problem becomes apparent when the "irrelevant" entries are to be interpreted. It can be seen that on all sets, both i_1 and i_4 have never been annotated by u_1 using any other tags, and therefore it makes sense to assume that the entries of (u_1, t_1), (u_1, t_3), and (u_1, t_4) with $\{i_1, i_4\}$ are "irrelevant". However, the entries of i_3 with (u_1, t_1) and (u_1, t_3) should not simply be interpreted as "irrelevant" as i_3 occurs as "relevant" on (u_1, t_4). Similarly, i_2 is "relevant" on (u_1, t_1) and (u_1, t_3), and therefore (u_1, t_4) with i_2 cannot be "irrelevant". Simply labelling those entries as "irrelevant" is improper and can result in inferior recommendation performance [8]. It can be noted that those entries definitely cannot be labelled "indecisive" as we do not want to predict them in the future.

3.2 Graded-Relevance Scheme

We propose the *graded-relevance* interpretation scheme to effectively leverage the tagging data for building the tensor ranking learning model $\mathcal{Y}$. Following the general rule, the observed entries are regarded as "relevant", indicating that users have shown their interest to the entries. From the observed entries, we extract the list of distinct items that have been annotated by user u using any tags. This list, denotes as C_u, is defined as follows:

$$C_u = \{i | (u, i, *) \in A\} \tag{3}$$

The item set C_u assists in distinguishing the non-observed data that do not belong to either the "irrelevant" or the "indecisive" category.

On each observed (u, t) set, the entries which items have been annotated using other tags are labelled as "likely relevant" entries. As a result, the *graded-relevance* scheme insightfully differentiates the non-observed data and exposes them as a mixture of three entries: (1) "likely relevant" entries – user is probably interested to the entries, yet it is not explicitly revealed, (2) "irrelevant" entries – user is not interested with the entries, and (3) "indecisive" entries – user might be interested to the entries in the future. The "likely relevant" entries are revealed as entries that, even though they do not occur in the observed set, the items of the entries have actually been annotated by the user using other tags.

Having the four possible distinct values for each entry, we assign the entries with an ordinal value which is graded from the highest to the lowest ones, i.e. "relevant", "likely relevant", "indecisive", and "irrelevant". As illustrated in Figure 2(c), the *graded-relevance* scheme can generate entries labelled with $\{2,1,0,-1\}$ for the tensor model and make it comparable to explicit rating data. The *graded-relevance* scheme uses the following rules:

$$y_{u,i,t} := \begin{cases} 2 & if\ (u,i,t) \in A \\ 1 & if\ (u,i,t) \notin A\ and\ i \in C_u\ and\ (u,*,t) \in A \\ -1 & if\ (u,i,t) \notin A\ and\ i \in I \backslash C_u and\ (u,*,t) \in A \\ 0 & otherwise \end{cases} \tag{4}$$

At the next stage, for learning the latent factors of the tensor model, we propose optimizing the ranking metric that can appropriately handle the ordinal relevance of tensor entries generated from the *graded-relevance* scheme.

4. LEARNING-TO-RANK WITH TENSOR MODEL

We first describe the tensor-based recommendation model in general and then present in detail how the tensor model can be learned by optimizing the GAP ranking metric.

4.1 Tensor-based Recommendation Model

The latent ternary correlation $\langle user, item, tag \rangle$ can be inferred after factorizing the tensor model in which entries are populated by implementing the proposed *graded-relevance* scheme to interpret the tagging data. In this paper, we use CP [2] as the factors model as well as the predictor function. CP is a well-known factorization technique that has shown to be less expensive in both memory and time consumption as compared to another popular factorization technique of Tucker [2].

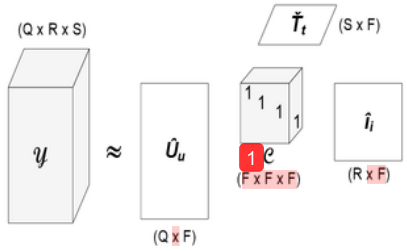


Figure 3. The CP factorization model for third-order tensor

As illustrated in Figure 3, CP factorizes a third-order tensor $\mathcal{Y} \in \mathbb{R}^{Q \times R \times S}$ into three latent 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 latent factor matrices. These latent factor matrices are used in calculating the predicted score that reflects the preference level of a user in choosing an item for a tag. The predicted preference score is calculated as:

$$\hat{y}_{u,i,t} := \sum_{f=1}^{F} \hat{u}_{u,f} \cdot \hat{\imath}_{i,f} \cdot \hat{t}_{t,f} = [\![\hat{U}_u, \hat{I}_i, \hat{T}_t]\!] \quad (5)$$

Assuming that users show interest to a few top recommended items only [9], the factorized tensor model can be optimized with respect to the ranking metric during the learning procedure in order to generate quality item recommendations to users by exploiting their tag usage.

4.2 Learning to Rank with Tensor Model

The development of *Go-Rank*, the second main contributions of this paper, is presented in this section. *Go-Rank* directly optimizes the appropriate ranking metric for learning the latent factors of tensor model built with entries labelled using the *graded-relevance* scheme.

4.2.1 Optimization Criterion

The unique characteristic of the generated tensor entries, detailed in Section 3.2, requires an optimization criterion that allows the learning model to control when to regard the "likely relevant" entries as "relevant" or 'irrelevant". We propose to use Graded Average Precision (GAP) as the ranking metric to optimize the latent factors of tensor model. GAP is the generalization of Average Precision (AP) metric for ordinal relevance data [14]. Using GAP as the optimized ranking metric allows us to set up thresholds in the learning model so that the "likely relevant" entries can be regarded as either "relevant" or "irrelevant" entries. The task of recommendation can now be formulated as recommend an optimal (from the GAP perspective) items list to users using the latent factor matrices of tensor model. Based on the original definition of GAP [14], the score for a user u under tag t is formulated as:

$$GAP_{u,t} := \frac{\sum_{l \in I} \sum_{k=1}^{c} \sum_{j \neq i} g_k \mathbb{I}(y_{u,i,t} \geq 1) \mathbb{I}(y_{u,j,t} \geq \mu_k) \frac{\mathbb{I}(r_{u,j,t} \leq r_{u,i,t})}{r_{u,i,t}}}{\sum_{l=1}^{c} n_{u,l,t} \sum_{k=1}^{l} g_k} \quad (6)$$

where $c = max(y_{u,i,t})$ and $y_{u,i,t}$ is the relevance label assigned from the ordinal relevance set of $\{2,1,0,-1\}$ obtained 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 latent factor matrices using Equation (5)). The g_k denotes the threshold probability that the user sets as a threshold of relevance at grade μ_k, i.e. regarding the entries with grades equal or larger than μ_k as "relevant" and the others as "irrelevant". In other words, g_k and μ_k are the parameters that control whether the "likely relevant" entries should be regarded as "relevant" or 'irrelevant". It is to be noted that the probability values must be exclusive and exhaustive probabilities [14]:

$$\sum_{k=1}^{c} g_k = 1 \quad (7)$$

The $\mathbb{I}(\cdot)$ is the indicator function which is equal to 1 if the condition is satisfied, and 0 otherwise. The $n_{u,l,t}$ is number of items labelled with grade l by user u using tag t. The GAP score for all users under all tags can be defined as:

$$GAP := \frac{1}{QS} \sum_{u \in U} \sum_{t \in T} \frac{\sum_{l \in I} \sum_{k=1}^{c} \sum_{j \neq i} \beta_{gy} \frac{\mathbb{I}(r_{u,j,t} \leq r_{u,i,t})}{r_{u,i,t}}}{\sum_{l=1}^{c} n_{u,l,t} \sum_{k=1}^{l} g_k} \quad (8)$$

Notice that for notational convenience, we perform the following substitution:

$$\beta_{gy} = g_k \mathbb{I}(y_{u,i,t} \geq 1) \mathbb{I}(y_{u,j,t} \geq \mu_k) \quad (9)$$

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

4.2.2 Ranking Smoothing

From Equation (8), we can see that GAP is dependent to the ranking positions of items in the recommendation list as it is reliant on the values of $\mathbb{I}(r_{u,j,t} \leq r_{u,i,t})$ and $r_{u,i,t}$. Given that the ranking positions are determined via the predicted relevance scores calculated based on the model parameters (i.e. latent factor matrices), the GAP function becomes unsmooth. This condition makes it difficult to apply the standard optimization methods which require smoothness in the objective function, such as the gradient based approaches, in the learning procedure [10, 21]. We tackle this problem by implementing the smoothing function [10] to the ranking position with respect to the model parameters as follows:

$$\mathbb{I}(r_{u,j,t} \leq r_{u,i,t}) \approx \sigma(\Delta\hat{y}) \quad (10)$$

$$r_{u,i,t} \approx 1 + \sum_{j \neq i} \sigma(\Delta\hat{y}) \quad (11)$$

where σ is the logistic function $\sigma(x) = \frac{1}{1+e^{-x}}$ and $\Delta\hat{y} = \hat{y}_{u,i,t} - \hat{y}_{u,j,t}$. Substituting Equations (10) and (11) to Equation (8), we obtain the smoothed approximation of GAP:

$$sGAP := \frac{1}{QS} \sum_{u \in U} \sum_{t \in T} \frac{\sum_{l \in I} \sum_{k=1}^{c} \sum_{j \neq i} \beta_{gy} \frac{\sigma(\Delta\hat{y})}{1+\sum_{j \neq i} \sigma(\Delta\hat{y})}}{\sum_{l=1}^{c} n_{u,l,t} \sum_{k=1}^{l} g_k} \quad (12)$$

The resultant objective function with λ_Θ as regularization coefficient corresponds to σ_Θ for avoiding overfitting is formulated as:

$$L(\Theta) := \frac{1}{QS} \sum_{u \in U} \sum_{t \in T} \frac{\sum_{l \in I} \sum_{k=1}^{c} \sum_{j \neq i} \beta_{gy} \frac{\sigma(\Delta\hat{y})}{1+\sum_{j \neq i} \sigma(\Delta\hat{y})}}{\sum_{l=1}^{c} n_{u,l,t} \sum_{k=1}^{l} g_k} - \lambda_\Theta \|\Theta\|_F^2 \quad (13)$$

4.2.3 Learning Algorithm

We present the algorithm to optimize model parameters Θ of $\Delta\hat{y}$ formulated in Equation (13). We use gradient descent to minimize the objective function. We do not consider QS and $\sum_{l=1}^{c} n_{u,l,t} \sum_{k=1}^{l} g_k$ coefficients as they have no influence on the optimization. Given a case (u, i, j, t) with respect to model parameters $\{\hat{U}_u, \hat{I}_i, \hat{I}_j, \hat{T}_t\}$, the gradient of $sGAP$ can be computed as:

$$\frac{\partial L}{\partial \theta} = \frac{\partial}{\partial \theta} \left(\sum_{u \in U} \sum_{t \in T} \frac{\sum_{l \in I} \sum_{k=1}^{c} \sum_{j \neq i} \beta_{gy} \sigma(\Delta\hat{y})}{1+\sum_{j \neq i} \sigma(\Delta\hat{y})} \right) - \lambda_\Theta \theta \quad (14)$$

By replacing $\frac{\partial}{\partial\theta}\Delta\hat{y}$ as δ, for notational convenience, the derivation formulation can be rewritten as:

$$\frac{\partial L}{\partial\theta} = \sum_{u\in U}\sum_{t\in T}\frac{\sum_{i\in I}\sum_{k=1}^{c}\sum_{j\neq i}\left[\left(\frac{\partial}{\partial\theta}(\beta_{gy}\delta)\right)(1+\sum_{j\neq i}\delta)\right]-\left[(\beta_{gy}\delta)\left(\frac{\partial}{\partial\theta}(1+\sum_{j\neq i}\delta)\right)\right]}{\left[1+\sum_{j\neq i}\delta\right]^2} - \lambda_\theta\theta \quad (15)$$

The resulted gradient of $sGAP$ is obtained as:

$$\frac{\partial L}{\partial\theta} = \sum_{u\in U}\sum_{t\in T}\frac{\sum_{i\in I}\sum_{k=1}^{c}\sum_{j\neq i}\beta_{gy}\left(\left[(-\delta+\delta^2)(1+\sum_{j\neq i}\delta)\frac{\partial}{\partial\theta}\Delta\hat{y}\right]-\left[\delta\left(\sum_{j\neq i}(-\delta+\delta^2)\frac{\partial}{\partial\theta}\Delta\hat{y}\right)\right]\right)}{\left[1+\sum_{j\neq i}\delta\right]^2} - \lambda_\theta\theta \quad (16)$$

Since β_{gy} is determined by $\mathbb{I}(y_{u,i,t}\geq 1)\mathbb{I}(y_{u,j,t}\geq\mu_k)$, it ensures that the entries must not be "irrelevant", the grade of entries must be at least equal to the threshold μ_k, and, most importantly, users are not using the whole available tags. Knowing this, we can implement a sparsity-aware approach to make the learning algorithm more efficient and faster. Instead of calculating $sGAP$ for the entire item set across all tags, we can optimize $sGAP$ only across tags that have been used by user u, $V_u = \{t|(u,*,t)\in A\}$, for items which entries are labelled as "relevant" or "likely relevant" for the user, i.e. $Z_u = \{i|y_{u,i,t}=2\cup y_{u,i,t}=1\}$. Other entries are not necessary to be included since their values are always less than any threshold graded values of μ, which means that $\mathbb{I}(y_{u,j,t}\geq\mu_k)$ will always be 0. As a result, the gradient of $sGAP$ given a case (u,i,j,t) with respect to the model parameter, $\{\hat{U}_u,\hat{I}_i,\hat{I}_j,\hat{T}_t\}$, becomes:

$$\frac{\partial L}{\partial\theta} = \sum_{u\in U}\sum_{t\in V_u}\frac{\sum_{i\in Z_u}\sum_{k=1}^{c}\sum_{j\neq i}\beta_{gy}\left(\left[(-\delta+\delta^2)(1+\sum_{j\neq i}\delta)\frac{\partial}{\partial\theta}\Delta\hat{y}\right]-\left[\delta\left(\sum_{j\neq i}(-\delta+\delta^2)\frac{\partial}{\partial\theta}\Delta\hat{y}\right)\right]\right)}{\left[1+\sum_{j\neq i}\delta\right]^2} - \lambda_\theta\theta \quad (17)$$

Algorithm: ***Go-Rank* Learning**
Input : Training set $D_{train}\subseteq U\times I\times T$, threshold probability $g\in\{g_1,g_2\}$, threshold grade $\mu\in\{\mu_1,\mu_2\}$, learning rate α, factor matrix column size F, regularization λ, maximal iteration $iterMax$
Output: Learned factor matrices $\hat{U}$, $\hat{I}$, $\hat{T}$
$Q=|U|$, $R=|I|$, $S=|T|$, $\mathcal{Y}\in\mathbb{R}^{Q\times R\times S}$, $y_{u,i,t}\in\{2,1,0,-1\}$
Populate $\mathcal{Y}$ using Equation (4)
$c=max(y_{u,i,t})$
$Z_u=\{i|y_{u,i,t}=2\cup y_{u,i,t}=1\}$
$V_u=\{t|(u,*,t)\in A\}$
Initialize $\hat{U}^{(0)}\in\mathbb{R}^{Q\times F}$, $\hat{I}^{(0)}\in\mathbb{R}^{R\times F}$, $\hat{T}^{(0)}\in\mathbb{R}^{S\times F}$, $h=0$
repeat
  for $u\in U$ **do**
    $\hat{U}_u\leftarrow\hat{U}_u+\alpha\frac{\partial L}{\partial\hat{U}_u}$ based on Equation (17)
  for $t\in T$ **do**
    $\hat{T}_t\leftarrow\hat{T}_t+\alpha\frac{\partial L}{\partial\hat{T}_t}$ based on Equation (17)
  for $u\in U$ **do**
    for $t\in V_u$ **do**
      for $i\in Z_u$ **do**
        for $k\leftarrow 1$ **to** c **do**
          for $j\neq i$ **do**
            $\hat{I}_i\leftarrow\hat{I}_i+\alpha\frac{\partial L}{\partial\hat{I}_i}$ based on Equation (17)
            $\hat{I}_j\leftarrow\hat{I}_j+\alpha\frac{\partial L}{\partial\hat{I}_j}$ based on Equation (17)
    $++h$
until $h\geq iterMax$

Figure 4. *Go-Rank* Learning Algorithm

To apply the $sGAP$ 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: $\frac{\partial\Delta\hat{y}}{\partial\hat{U}_u}=(\hat{I}_i\odot\hat{T}_t-\hat{I}_j\odot\hat{T}_t)$, $\frac{\partial\Delta\hat{y}}{\partial\hat{I}_i}=(\hat{U}_u\odot\hat{T}_t)$, $\frac{\partial\Delta\hat{y}}{\partial\hat{I}_j}=-(\hat{U}_u\odot\hat{T}_t)$, $\frac{\partial\Delta\hat{y}}{\partial\hat{T}_t}=(\hat{U}_u\odot\hat{I}_i-\hat{U}_u\odot\hat{I}_j)$, where $\odot$ denotes element-wise product. The *Go-Rank* learning algorithm is outlined in Figure 4.

4.2.4 *Complexity Analysis and Convergence*

We analyze the complexity of learning process for a single iteration. Complexity of the initial *Go-Rank* function (Equation 12) is $O(F(Q+S+QScR^2))$ where $c=max(y_{u,i,t})$. Given that $\mathcal{Y}\in\mathbb{R}^{Q\times R\times S}$, the total number of possible entries of the tensor model, i.e. the sum of "relevant", "likely relevant", irrelevant" and "indecisive" entries is calculated as $|Y|=QRS$. Since $|Y|\gg Q,S,R,F,c$, the overall complexity of *Go-Rank* in one iteration can be regarded as $|Y|$. In other words, it is linear to the total number of possible entries of tensor model.

After the implementation of the proposed sparsity-aware approach for making the learning process to run efficiently and faster, complexity of the *Go-Rank* function (as summarized in Figure 4), becomes $O(F(Q+S+Q\tilde{v}\,c\tilde{n}^2))$ where $\tilde{v}$ and $\tilde{n}$ denote the average number of V_u and Z_u respectively. Since $\tilde{v}\ll S$ and $\tilde{n}\ll R$, the *Go-Rank* complexity now becomes $|\tilde{Y}|$, i.e. the sum of "relevant" and "likely relevant" entries of the tensor model, where $|\tilde{Y}|\ll|Y|$.

5. EMPIRICAL ANALYSIS

We use the MovieLens and LastFM datasets to evalaute the proposed *graded-relevance* interpretation scheme of tagging data and the proposed tag-based item recommendation method *Go-Rank*. These two datasets have been commonly used in recommendation techniques evaluation. MovieLens is obtained from the GroupLens Research Group corpus (http://ww.grouplens.org) and is an extension of MovieLens10M dataset published, while the LastFM corpus is retrieved from the Last.fm online music system (http://www.last.fm/) [22]. The MovieLens dataset consists of 47,957 observed entries from 2,113 users, 5,908 movies/items, and 9,079 tags. The LastFM dataset consists of 186,479 observed entries from 1,892 users, 12,523 artists/items, and 9,749 tags.

In this paper, we implemented a variety of p-cores to eliminate the noise and decrease sparsity [4]. Moreover, it avoids the non-stable results that tend to occur when only one choice of core size is used experiments [23]. The p-core technique [24] is implemented by 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. In the experiments, we investigated the variety of p for each dataset, i.e. 15-core and 20-core sets for the MovieLens dataset, and 50-core and 55-core sets for the LastFM dataset. It is to be noted that, the bigger the p-core is, the less sparse the commonly resulted set. Table 1 shows that, for the MovieLens dataset, the density of "relevant" entries of the 15-core is more sparse than that of the 20-core, i.e. $0.0283<0.0284$. However, implementing p-core technique on the LastFM dataset results into conflicting behavior as Table 1 shows that the density of "relevant" entries of the 50-core is less sparse than that of the 55-core, i.e. $0.2465>0.1479$.

Each core of each dataset was randomly divided, using 5-fold cross validation, into a training set D_{train} (80%) and a test set D_{test} (20%) based on the number of posts data. The 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}. The performance is measured using AP and NDCG, and reported over the average values.

Following the initialization approach proposed for a ranking model [20], we randomly initialized the factor matrices of our model. For a tensor-based method, a latent factor matrix with size between 10-64 is usually acceptable for generating quality recommendation [5, 12, 13, 17, 19, 25]. Here, we set the size of latent factor matrix F 32 for *Go-Rank* and other tensor-based benchmarking methods. In order to empirically tune the other parameters in *Go-Rank*, we used all the observed entries available in D_{train} of each dataset. The best performances were achieved using the following values: learning rate $\alpha = 0.01$ and regularization $\lambda = 1e^{-05}$.

5.1 Impact of *Graded-Relevance* Scheme

We first examine the impact of *graded-relevance* scheme on tensor density. We compare the density of tensor entries resulted from the *graded-relevance* scheme with those of the *boolean* and *set-based* schemes as listed in Table 1. The statistics are reported, as the average density over all 5-fold D_{train}, on each dataset.

Table 1 shows that the proposed *graded-relevance* interpretation scheme is able to generate more dense data representation than the *boolean* scheme, concurring with previous outcome [5], as well as the *set-based* scheme. Results show that the density of "indecisive" entries on both the *graded-relevance* and *set-based* schemes are same. However, as previously explained in Section 3.2, the *graded-relevance* scheme breaks down the "irrelevant" entries of *set-based* scheme into "likely relevant" and "irrelevant" entries.

From Table 1, we can also see that, compared to the very low number of "relevant" entries, the *graded-relevance* scheme revealed significant number of "likely relevant" entries. In this case, by using the threshold probability [14], as shown in the following section, portion of these entries can then be possibly regarded as "relevant" entries. Additionally, the comparison between the numbers of "irrelevant" entries of the *graded-relevant* scheme to those of the *set-based* scheme is also worthwhile to observe. We can see that only a small amount of the "irrelevant" entries of *set-based* scheme are revealed as "likely relevant" entries using the *graded-relevance* scheme.

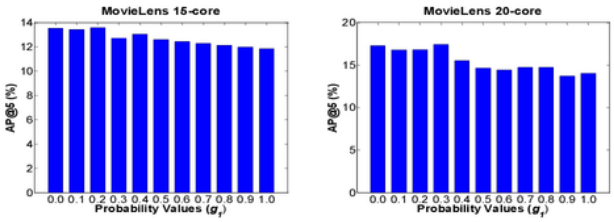
5.2 Impact of Probability Values

This experiment is conducted to examine how "relevant" is the "likely relevant" data. The probabilities values, $g \in \{g_1, g_2\}$ where $g_1 + g_2 = 1$, are the probabilities that regulate whether the "likely relevant" entries should be regarded as "relevant" or "irrelevant". We used g_1 and g_2 to determine the percentage of considering the "likely relevant" entries as "relevant" and 'irrelevant" respectively. For example, $g_1 = 0.1$ indicates that 10% of "likely relevant" entries will be considered as "relevant" and the other 90% of those will be considered as "irrelevant". We experimented with a total grid of probability values between 0 and 1 with an interval of 0.1, resulting $g_1 \in \{0.0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0\}$. Figure 5 and Figure 6 show the results as the average values over all 5-fold D_{train}, on each dataset. It is to be noted that the threshold of regarding the "likely relevant" entries as either "relevant" or 'irrelevant" entries is fixed as $\mu \in \{2,1\}$, following the grades of the "relevant" and "likely relevant" entries formulated in Equation (4). In this case, depending on which threshold value is used, $y_{u,j,t} \geq \mu$ will determine the entries as "relevant" and the others as "irrelevant".

For the MovieLens dataset, Figure 5(a) and Figure 5(b) present the impact of probability values on 15-core and 20-core sets, respectively. The results show that not all of the "likely relevant" entries are "irrelevant". For example, in Figure 5(a) and Figure 5(b), the highest AP@5 are achieved when $g_1 = 0.2$ and $g_1 = 0.3$ for the 15-core and 20-core sets, respectively. This means that, a total of 20% or 30% of the "likely relevant" entries are actually found "relevant".

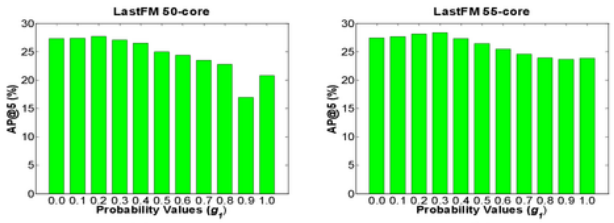
Interpretation Scheme	Entries	Density (%)			
		MovieLens		LastFM	
		15-core	20-core	50-core	55-core
boolean	Relevant	0.0283	0.0284	0.2465	0.1479
	Irrel./ Indecisive	99.9717	99.9716	99.7535	99.8521
set-based	Relevant	0.0283	0.0284	0.2465	0.1479
	Irrelevant	2.0969	2.4147	4.5894	4.8249
	Indecisive	97.8748	97.5569	95.1641	95.0272
graded-relevance	Relevant	0.0283	0.0284	0.2465	0.1479
	Likely Relevant	0.2493	0.3729	0.6984	0.7917
	Irrelevant	1.8477	2.0418	3.8911	4.0332
	Indecisive	97.8748	97.5569	95.1641	95.0272

Table 1. The comparison of tensor entries densities generated from various interpretation schemes: *boolean*, *set-based*, and *graded-relevance*



(a) (b)

Figure 5. Impact of probability values on the MovieLens dataset: (a) 15-core and (b) 20-core sets



(a) (b)

Figure 6. Impact of probability values on the LastFM dataset: (a) 50-core and (b) 55-core sets

For the LastFM dataset, Figure 6(a) and Figure 6(b) present the impact of probability values on 50-core and 55-core sets, respectively. Complementing the results on the MovieLens dataset, those of the LastFM dataset also show that not all of the "likely relevant" entries are "irrelevant". For example, in Figure 6(a) and Figure 6(b), the highest AP@5 is achieved when $g_1 = 0.2$ and $g_1 = 0.3$ for the 50-

core and 55-core sets, respectively. Hence, a total of 20% or 30% of the "likely relevant" entries of the sets can be regarded as "relevant".

All of these results establish that, on each observed (u, t) set, any items other than those appearing in the observed entries should not simply be regarded as "irrelevant" since some of them have been annotated by the user using other tags. In this case, the *graded-relevance* scheme is an efficient scheme as it sets the "transitional" entries as distinct entries positioned between the "relevant" and "irrelevant" entries.

5.3 Effectiveness of the Learning Algorithm

We examine the effectiveness of *Go-Rank* learning algorithm in terms of its learning time and convergence. We show the effectiveness of *Go-Rank* in Figure 7 using the MovieLens 20-core set. Implementation on other datasets gives similar results.

To investigate the impact of optimizing *sGAP* using the fast learning approach (Equation (17)), we measure the running time for learning the model on a single iteration at different scales, i.e. 10% to 100% of one of the $\mathcal{Y}_{train}$ resulted from the 5-fold cross validation split. Figure 7(a) shows that the fast learning approach time is extremely faster compared to that of the original learning model (Equation (16)). Figure 7(b) shows the evalution of AP@5 across iterations on the training and test sets. On both sets, AP score gradually increases along the iterations and converge after a few number of iterations. These results demonstrate that *Go-Rank* effectively optimizes GAP.

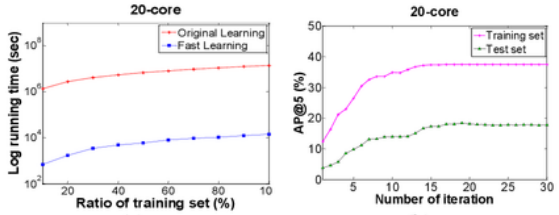


Figure 7. Effectiveness of *Go-Rank*: (a) Scalability and (b) Convergence

5.4 Performance Comparison

In this paper, we used the following benchmarking methods to be compared with the performance of the proposed *Go-Rank* method in terms of recommendation quality:

- **PITF** [17] is the pair-wise tensor model that was originally developed for tag recommendations. The method implements *set-based* scheme, to build *pair-wise ranking* on each user-item set, and AUC-based optimization. We have adapted the method to generate item recommendations based on the $(user, tag)$ sets. Note that the size of latent factor matrix is set as $F = 32$, the same as the setting for the proposed *Go-Rank* method. The other parameters are empirically tuned as $\alpha = 0.01$ and $\lambda = 5e^{-05}$
- **CP-TRPR** [3] is the probabilistic ranking tensor-based method that implements *boolean* scheme and ranks the predicted preference scores by utilizing users past collaborative tagging data via probabilistic ranking approach. The ranking process is a subsequent stage following the tensor model learning process. The size of latent factor matrix is set as $F = 32$, and the other parameters are empirically tuned as $tolerance = 1e^{-04}$, $\lambda = 0$ and $voc_size = 20$.
- **HOSVD-Max (MAX)** [4] is the conventional tensor-based method that implements *boolean* scheme and directly ranks the list of recommendations based on the maximum values of the predicted preferences scores. The size of latent factor matrix is set as $F = 32$, and the other parameters are empirically tuned as $tolerance = 1e^{-04}$ and $\lambda = 0$.
- **Graded-HOSVD-Max (Graded-MAX)** is the modification of the HOSVD-Max [4]. In this case, instead of using the *boolean* scheme, we implement our proposed *graded-relevance* scheme for building the tensor model.
- **CTS** [1] is the state-of-the-art matrix-based method that implements *boolean* scheme and ranks the recommendations by using the users past tagging activities in forming users' likelihood. The empirically tuned parameters are $neigborhood_size = 50$ and $model_size = 50$.

Table 2 and Table 3 show the performance comparison between the proposed *Go-Rank* with the benchmarking methods on the Movielens and LastFM datasets, respectively. Based on those results, we obtain the following three observations.

Methods	15-core (Score in %)						20-core (Score in %)					
	NDCG@1	NDCG@5	NDCG@10	AP@1	AP@5	AP@10	NDCG@1	NDCG@5	NDCG@10	AP@1	AP@5	AP@10
PITF	5.07	4.28	3.99	5.07	8.44	9.23	6.87	6.09	5.67	6.87	12.13	13.20
CP-TRPR	**9.74**	7.01	5.85	**9.74**	13.55	14.35	11.85	**8.91**	7.34	11.85	17.19	17.63
MAX	9.43	7.05	5.81	9.43	13.36	13.83	**12.77**	8.89	7.42	**12.77**	16.94	17.44
Graded-MAX	4.44	3.77	3.30	4.44	7.81	8.38	6.06	5.18	4.64	6.06	10.97	11.71
CTS	9.70	7.00	5.80	9.70	13.49	14.10	11.74	8.99	7.57	11.74	16.70	17.18
Go-Rank	**8.70**	**7.61**	**6.79**	**8.70**	**13.58**	**14.76**	**12.58**	**8.60**	**7.66**	**12.58**	**17.42**	**17.74**

Table 2. NDCG and AP at top-*N* position on MovieLens dataset

Methods	50-core (Score in %)						55-core (Score in %)					
	NDCG@1	NDCG@5	NDCG@10	AP@1	AP@5	AP@10	NDCG@1	NDCG@5	NDCG@10	AP@1	AP@5	AP@10
PITF	12.30	10.91	9.76	12.30	20.52	21.42	13.86	11.77	10.75	13.86	21.98	22.63
CP-TRPR	15.12	13.63	12.26	15.12	24.02	24.26	18.82	15.65	13.77	18.82	27.55	27.10
MAX	17.44	14.29	12.51	17.44	24.90	24.27	17.21	14.84	13.02	17.21	25.35	25.25
Graded-MAX	15.69	12.52	10.66	15.69	22.75	22.22	13.65	11.27	9.57	13.65	20.76	20.88
CTS	12.20	10.73	9.58	12.20	19.33	19.84	15.80	13.45	11.74	15.80	24.22	23.88
Go-Rank	**18.92**	**15.60**	**13.56**	**18.92**	**27.69**	**27.37**	**18.83**	**15.79**	**13.85**	**18.83**	**28.36**	**27.79**

Table 3. NDCG and AP at top-*N* position on LastFM dataset

First, the proposed *Go-Rank* outperforms all of benchmarking methods in terms of NDCG and AP (at most top-*N* positions) on both datasets. Note that, in contrast to AP, the higher the top-*N* position, the less the NDCG score is. The results confirm that *Go-Rank* is an effective approach for learning the tensor model built with data labelled using the proposed *graded-relevance* interpretation scheme. In this case, the GAP optimization of *Go-Rank* allows us to set up thresholds in the learning model so that the "likely relevant" entries can be regarded as either "relevant" or "irrelevant" entries. Previously described in Section 5.2, portion of "likely relevant" entries are found "relevant" that consequently assists *Go-Rank* to produce high quality recommendation. On the other hand, optimizing inappropriate objective function on a tensor model, built with data labelled using the *graded-relevance* scheme, results in inferior recommendation quality. This can be seen by comparing the results of Graded-MAX with those of MAX. It is to be noted that the objective function of both the Graded-MAX and MAX is minimizing the Mean Square Error (MSE), which is more suitable for solving a classification problem [3]. Their difference is in the interpretation scheme implemented for building the tensor model. Graded-MAX results a poorer recommendation quality compared to MAX, despite the fact that it implemented the *graded-relevance* scheme to build the tensor model, since it simply used the generated data as it is. In this case, Graded-MAX disregards the constraint that the "likely relevant" entries should be further regarded as either "relevant" or "irrelevant" in order to effectively learn the tensor model.

Second, we can see that the *Go-Rank* outperforms PITF. It is to be noted that both methods implements the non-*boolean* interpretation schemes to build the tensor models, i.e. *Go-Rank* uses the *graded-relevance* scheme while PITF uses the *set-based* schemes. *Go-Rank* attains great improvement over PITF, in both NDCG and AP, for two main reasons. *Go-Rank* build the learning model as a *list-wise* ranking model, i.e. aiming to get the order of lists correctly. Whereas PITF builds the learning model as a *pair-wise* ranking model, which means that it attempts to get the correct ranking order within each pair only. Moreover, *Go-Rank* enhances the top-*N* recommendation performance by optimizing the top-biased measure GAP, i.e. the generalization of AP for ordinal relevance data. While PITF implements the equal-penalty measure AUC [12]. On the other hand, we can also see the outperformance of CP-TRPR over PITF, despite the fact that the former implements the *boolean* scheme to generate the tensor entries. The reason for this is because CP-TRPR implements a subsequent stage, the probabilistic ranking approach, following the MSE-based optimization learning process [3], for generating the recommendations. In this case, the second stage of the method probabilistically re-ranks the list of candidate items for each user, generated from the learned latent factors, as a *list-wise* ranking model in order to get the correct order of recommendations. Additionally, we also notice that CTS underperforms *Go-Rank*, especially on the LastFM dataset. This result indicates that projecting the ternary correlations of tagging data into two-dimensional model is impacting recommendation quality [4].

Third, by relating Table 1 with Table 2 and Table 3, we can see that the "relevant" tensor entries density, resulted from variety of *p*-cores on the same dataset, is impacting the *Go-Rank* performance improvement over other benchmarking methods. In this case, the less sparse the density is, the higher the performance improvement achieved, and vice versa. For example, the improvement comparison of *Go-Rank* over CP-TRPR can be observed as follows. As on the MovieLens dataset, the density of 15-core (i.e. 0.0283) is more sparse than that of the 20-core (i.e. 0.0284), the improvement of the former (e.g. $AP@5 = 0.22\%$) is lower than that of the latter (e.g. $AP@5 = 1.34\%$). On the other hand, on the LastFM dataset, the density of 50-core (i.e. 0.2465) is less sparse than that of the 55-core (i.e. 0.1479), therefore the improvement of the former (e.g. $AP@5 = 15.28\%$) is higher than that of the latter (e.g. $AP@5 = 2.94\%$). This result indicates that the *graded-relevance* scheme succeed to optimally generate entries from the observed tagging data for building the tensor model and to accommodate the GAP-based learning algorithm.

6. CONCLUSION AND FUTURE WORK

We have presented *Go-Rank*, a novel top-*N* tag-based item recommendation method that implements the proposed tagging data *graded-relevance* interpretation scheme and directly optimizes the (smoothed) GAP for generating ordered list of items that might interest the users. For improving the scalability of *GoRank*, we developed and implemented a fast learning approach that apply optimization only across tags that have been used by the user, and items of "relevant" and "likely relevant" entries. Experimental results on real-world datasets confirm that the *graded-relevance* is an efficient scheme as it leverages the tagging data more effectively. It establishes that, on each observed (u, t) set, any items other than those appearing in observed entries should not simply be all regarded as "irrelevant" entries, as considered in previous schemes.

Empirical analysis shows that *Go-Rank* is scalable and outperforms all benchmarking methods on the AP and NDCG measures. All of these ascertain that implementing the *graded-relevance* scheme and optimizing GAP for building the learning model improve the recommendation performance. For the future work, we would like to investigate the potential of implementing the *graded-relevance* scheme for optimizing other ranking metrics to generate tag-based item recommendations.

7. ACKNOWLEDGEMENT

This work is supported by the Directorate General of Higher Education (DGHE) Indonesia. Computational resources and services are provided by the HPC and Research Support Group, Queensland University of Technology, Brisbane, Australia.

8. REFERENCES

[1] Kim, H.-N., Ji, A.-T., Ha, I., and Jo, G.-S., Collaborative Filtering based on Collaborative Tagging for Enhancing the Quality of Recommendation. *Electronic Commerce Research and Applications*, 9(1): 73-83, 2010.

[2] Kolda, T. and Bader, B., Tensor Decompositions and Applications. *SIAM Review*, 51(3): 455-500, 2009.

[3] Ifada, N. and Nayak, R., *A Two-Stage Item Recommendation Method Using Probabilistic Ranking with Reconstructed Tensor Model*. In *User Modeling, Adaptation, and Personalization*, pages 98-110, Springer, 2014.

[4] Symeonidis, P., Nanopoulos, A., and 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): 179-192, 2010.

[5] Rendle, S., Balby Marinho, L., Nanopoulos, A., and Schmidt-Thieme, L. Learning Optimal Ranking with Tensor Factorization for Tag Recommendation. In *Proceedings of The 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining,* pages 727-736, Paris, France, 2009.

[6] Rafailidis, D. and Daras, P., The TFC Model: Tensor Factorization and Tag Clustering for Item Recommendation in Social Tagging Systems. *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, 43(3): 673-688, 2013.

[7] Leginus, M., Dolog, P., and Žemaitis, V., *Improving Tensor Based Recommenders with Clustering*. In *User Modeling, Adaptation, and Personalization*, pages 151-163, Springer Berlin Heidelberg, 2012.

[8] Ifada, N. and Nayak, R. An Efficient Tagging Data Interpretation and Representation Scheme for Item Recommendation. In *Proceedings of The 12th Australasian Data Mining Conference*, Brisbane, Australia, 2014.

[9] Cremonesi, P., Koren, Y., and Turrin, R. Performance of Recommender Algorithms on Top-N Recommendation Tasks. In *Proceedings of The 4th ACM Conference on Recommender Systems,* pages 39-46, Barcelona, Spain, 2010.

[10] Chapelle, O. and Wu, M., Gradient Descent Optimization of Smoothed Information Retrieval Metrics. *Information Retrieval*, 13(3): 216-235, 2010.

[11] Xu, J. and Li, H. AdaRank: A Boosting Algorithm for Information Retrieval. In *Proceedings of The 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval,* pages 391-398, Amsterdam, The Netherlands, 2007.

[12] Shi, Y., Karatzoglou, A., Baltrunas, L., Larson, M., Hanjalic, A., and Oliver, N. 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,* pages 155-164, Portland, Oregon, USA, 2012.

[13] Shi, Y., Karatzoglou, A., Baltrunas, L., Larson, M., Oliver, N., and Hanjalic, A. CLiMF: Learning to Maximize Reciprocal Rank with Collaborative Less-is-More Filtering. In *Proceedings of The 6th ACM Conference on Recommender Systems,* pages 139-146, Dublin, Ireland, 2012.

[14] Robertson, S.E., Kanoulas, E., and Yilmaz, E. Extending Average Precision to Graded Relevance Judgments. In *Proceedings of The 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval,* pages 603-610, Geneva, Switzerland, 2010.

[15] Nanopoulos, A., Item Recommendation in Collaborative Tagging Systems. *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, 41(4): 760-771, 2011.

[16] Ifada, N. and Nayak, R., *Do-Rank: DCG Optimization for Learning-to-Rank in Tag-Based Item Recommendation Systems*. In *Advances in Knowledge Discovery and Data Mining*, pages 510-521, Springer, 2015.

[17] Rendle, S. and Schmidt-Thieme, L. Pairwise Interaction Tensor Factorization for Personalized Tag Recommendation. In *Proceedings of The 3rd ACM International Conference on Web Search and Data Mining,* pages 81-90, New York, USA, 2010.

[18] Weimer, M., Karatzoglou, A., Le, Q.V., and Smola, A., Maximum Margin Matrix Factorization for Collaborative Ranking. *Advances in Neural Information Processing Systems*, 2007.

[19] Shi, Y., Karatzoglou, A., Baltrunas, L., Larson, M., and Hanjalic, A. GAPfm: Optimal Top-N Recommendations for Graded Relevance Domains. In *Proceedings of The 22nd ACM International Conference on Information & Knowledge Management,* pages 2261-2266, San Francisco, California, USA, 2013.

[20] Balakrishnan, S. and Chopra, S. Collaborative Ranking. In *Proceedings of The 5th ACM International Conference on Web Search and Data Mining,* pages 143-152, Seattle, Washington, USA, 2012.

[21] Wu, M., Chang, Y., Zheng, Z., and Zha, H. Smoothing DCG for Learning to Rank: A Novel Approach using Smoothed Hinge Functions. In *Proceedings of The 18th ACM Conference on Information and Knowledge Management,* pages 1923-1926, Hong Kong, China, 2009.

[22] Cantador, I., Brusilovsky, P., and Kuflik, T. Second Workshop on Information Heterogeneity and Fusion in Recommender Systems (HetRec2011). In *Proceedings of The 5th ACM Conference on Recommender Systems,* pages 387-388, Chicago, Illinois, USA, 2011.

[23] Doerfel, S. and Jäschke, R. An Analysis of Tag-recommender Evaluation Procedures. In *Proceedings of The 7th ACM Conference on Recommender Systems,* pages 343-346, 2013.

[24] Batagelj, V. and Zaveršnik, M. *Generalized Cores*. arXiv preprint cs/0202039, 2002.

[25] Shi, Y., Karatzoglou, A., Baltrunas, L., Larson, M., and Hanjalic, A. xCLiMF: Optimizing Expected Reciprocal Rank for Data with Multiple Levels of Relevance. In *Proceedings of The 7th ACM conference on Recommender Systems,* pages 431-434, Hong Kong, China, 2013.

2016_WSDM

ORIGINALITY REPORT

19%	15%	17%	%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	link.springer.com Internet Source	9%
2	ausdm.org Internet Source	3%
3	"Advances in Knowledge Discovery and Data Mining", Springer Science and Business Media LLC, 2015 Publication	2%
4	doi.acm.org Internet Source	1%
5	www.casals.cz Internet Source	<1%
6	Steffen Rendle. "Learning optimal ranking with tensor factorization for tag recommendation", Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD 09 KDD 09, 2009 Publication	<1%

7	Yue Shi, Alexandros Karatzoglou, Linas Baltrunas, Martha Larson, Alan Hanjalic. "GAPfm", Proceedings of the 22nd ACM international conference on Conference on information & knowledge management - CIKM '13, 2013 Publication	<1%
8	repository.tudelft.nl Internet Source	<1%
9	"Web Information Systems Engineering – WISE 2019", Springer Science and Business Media LLC, 2019 Publication	<1%
10	Lecture Notes in Computer Science, 2015. Publication	<1%
11	Steffen Rendle. "Tag Recommendation", Studies in Computational Intelligence, 2010 Publication	<1%
12	delivery.acm.org Internet Source	<1%
13	Lecture Notes in Computer Science, 2014. Publication	<1%
14	Tai Keun Kwak. "Generalized Baer rings", International Journal of Mathematics and Mathematical Sciences, 2006 Publication	<1%

15	dx.doi.org Internet Source	<1%
16	ming.org.pk Internet Source	<1%
17	Wei-Ying Ma. "Directly optimizing evaluation measures in learning to rank", Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval - SIGIR 08 SIGIR 08, 2008 Publication	<1%
18	www.local-guru.net Internet Source	<1%
19	www2.warwick.ac.uk Internet Source	<1%
20	www.lancaster.ac.uk Internet Source	<1%
21	academy.2ch.net Internet Source	<1%
22	escholarship.org Internet Source	<1%
23	export.arxiv.org Internet Source	<1%
24	"Web Information Systems Engineering – WISE 2016", Springer Nature, 2016 Publication	<1%

25	www.cda.cn Internet Source	<1%
26	M. Brdys. "An extension to the modified two-step algorithm for steady-state system optimization and parameter estimation", International Journal of Systems Science, 8/1/1986 Publication	<1%
27	Recommender Systems, 2016. Publication	<1%
28	Lecture Notes in Computer Science, 2008. Publication	<1%
29	"Machine Learning and Knowledge Discovery in Databases", Springer Nature, 2016 Publication	<1%
30	"Web Information Systems Engineering – WISE 2013", Springer Science and Business Media LLC, 2013 Publication	<1%

Exclude quotes	On	Exclude matches	< 5 words
Exclude bibliography	On		