GTRM: A Top-N Recommendation Model for Smartphone Applications

Nengjun Zhu, Jian Cao*
Department of Computer Science and Engineering
Shanghai Jiao Tong University
Shanghai, China
Email:{zhu_nj, cao-jian}@sjtu.edu.cn

***Abstract*—With the number of smartphone applications (apps) growing explosively, it has a practical significance to provide personalized app recommendations. In this paper, we propose the GTRM, a new recommendation model which builds the top-N app list by optimizing the metric Group-oriented Mean Average Precision (GMAP). GMAP is an extension of the traditional metric Mean Average Precision (MAP) and it measures the precisions of top-N list in terms of the collective positions of related items rather than the position of individual item. Therefore, GTRM can recommend a more reasonable top-N app list by avoiding overfitting problem. The details of GMAP and GTRM are described. Extensive experiments on a real-world app dataset demonstrate the effectiveness of GTRM, and show that GTRM significantly outperforms the compared methods.**

***Keywords*-App Recommendation; Implicit Feedback; Collaborative Filtering; Learning to Rank; Metrics Optimization**

I. Introduction

With the number of smartphone applications (apps) growing explosively (e.g., more than 1 million on Google Play Store), it has been a serious problem for users to find apps that they prefer, and thus has a practical significance to provide personalized app recommendations. Currently, keywords based search still dominates app selection process. At the same time, many app markets have provided app recommendation functions. However, they recommend apps based on some simple rules, such as recommending the most popular apps or the newest apps, which can not satisfy individual's requirement.

In order to recommend apps to users, we need to know their preferences. The preference information can be learned from users ratings to apps. Unfortunately, this information is not available for most users. However, users' historical behaviors, (e.g., installation and downloading) can be easily collected nowadays by the smartphone manufacturers or telecom companies. These behaviors can reflect users' preference to apps to some degree. Therefore, we need to build an app recommender system based on the implicit feedback.

Rating prediction and ranking prediction are two common strategies for recommender systems. The former tries to fill the missing rating values of users to items (item and app are used interchangeably unless otherwise stated) while the latter explores preferential relations among multiple items. Currently, most real world recommender systems including app recommender systems provide the services of top-N recommendation, which essentially involves solving a ranking problem. Therefore, Learning to Rank (L2R) has been intensively investigated and applied to recommender systems. There are several measures (metrics) which are commonly used to judge how well an algorithm does. The typical metrics include MAP (mean average precision), MRR (mean reciprocal rank) and NDCG (normalized discounted cumulative gain). Often an L2R problem is reformulated as an optimization problem w.r.t one of these metrics. As a list-wise metric, MAP is the average of the precision value obtained for the top-N items. It has good discrimination and stability properties. Specifically, MAP has helpful feature which ensures that mistakes in recommended items at the top of the list carry a higher penalty than mistakes at the bottom of the list. For this reason, some recommendation models for implicit feedback domains are proposed by directly optimizing MAP [1], [2].

However, the MAP-based optimization method is overly concerned with obtaining high MAP values on history data, which leads to overfitting problems. For example, the apps that were not installed by many users tend to be ranked in lower positions although the user may potentially prefer these apps. On the contrary, some apps may be ranked in higher positions on the list because these apps have been ranked highly by quite a few people and are therefore popular, however, they have not been chosen based on the user's personal interests.

In order to overcome this problem, we propose an approach that tries to rank a user's potentially preferred apps in higher positions by modelling the latent relationships among apps. To achieve this goal, we optimize a more general metric, named GMAP (Group-oriented MAP) rather than MAP. GMAP inherits all the desirable properties of MAP, i.e., top-heavy bias, high informativeness, elegant probabilistic interpretation and a solid underlying theoretical basis [3]. However, GMAP maintains the "cascade" nature among item groups, rather than among individual

*Jian Cao is the corresponding author.

978-1-5386-0752-7/17 $31.00 © 2017 IEEE
DOI 10.1109/ICWS.2017.124

items. In order to optimize GMAP, we propose GTRM (the **G**MAP op**T**imization-based **R**ecommendation **M**odel, a latent model that combines the item-item recommendation approach and the L2R optimization method. GTRM improves the MAP-based methods since it tries to discover and make use of apps' relationships based on the user's inherent interests. In addition, GTRM can also accelerate the L2R learning process since less ranking tasks are required on app groups than on apps. The main contributions of this paper are as follows:

- In order to overcome the problems of top-N app recommendations by optimizing MAP, we propose GMAP, as an extension of MAP. GMAP is more fit to be an optimization metric in app recommender systems.
- We design GTRM to optimize GMAP and propose three strategies to produce top-N app recommendations.
- We perform extensive experiments which show that GTRM outperforms other baselines on various evaluation metrics. In addition, GTRM also has a faster learning rate.

The rest of this paper is organized as follows: the related work is reviewed in Section II. In Section III, the optimization metric GMAP is presented. In Section IV, GTRM is described in detail. Section V details experimental evaluation and finally, the conclusions and future work are presented in Section VI.

II. Related work

Collaborative filtering (CF) is a fundamental technology in recommender systems which can be divided into two categories: memory-based and model-based. Memory-based methods predict ratings by utilizing the relationships between items and users. Model-based methods try to learn the latent factors behind the raw rating data and rating predictions can be generated by applying learned factors. Matrix factorization (MF) is a widely used model-based CF algorithm [4]. In addition to the original version, different extensions have been proposed, including SLIM [5], SPMF [6] and HeteroMF [7]. Our GTRM also takes MF as an underlying model. However, it does not directly predict the relevance of an app to users, i.e., 1 or 0, but unfolds the partial order relation between apps. Furthermore, GTRM exploits CF similarity to cluster apps to groups.

L2R approaches learn user preferences through modelling partial or total ordering relation among items. They can be grouped into three approaches: pointwise, pairwise and listwise. The rating prediction methods are typically pointwise ones. They output the relevance degree of each individual item based on the features. Pairwise models, such as BPR [8], maximize AUC metrics by utilizing the partial order relations between items. Recently, approaches to directly optimize listwise metrics (e.g., MRR, MAP and NDCG) have been proposed, where the output is a ranked list (or permutation) which is analogous to the task of Top-N recommendation. For example, xCLiMF [9] is an optimizing model based on Expected Reciprocal Rank (ERR) criterion, aiming to guarantee the first item in the recommendation is a higher relevant item while SVM-MAP [2] and TFMAP [1], which are similar to ours, optimize MAP. However, our model finds the best list by adjusting the positions of app groups to maximize GMAP instead of MAP.

Some approaches have been proposed for smartphone application recommendation over the last decade. For example, Shi et al. [10] observed that the distribution of app dataset has higher kurtosis than common datasets such as movieLens, and thus used dimensionality reduction techniques to extract meaningful features from apps and then applied memory-based techniques to generate recommendations in this reduced space. Lin et al. [11] used a semi-supervised topic model to construct a representation of an apps version, and then discriminated the topics based on genre information to generate a version-sensitive ranked list. To relieve the cold-start problem in app recommendation, the work in [12] leverages the information on app followers available on Twitter and introduces Latent Dirichlet Allocation (LDA) to generate the latent user groups, and the probability that represents a users interest in an app is computed through the marginalization over these groups. However, these methods are content-based which need extra information besides user historical data. By contrast, KRPMF [13], which depends only on the information of users app installation as our GTRM does, incorporates the kernel information of users and apps w.r.t app-categorical information into conventional PMF, to capture the latent correlations among the latent factors. However, it is a pointwise approach and thus can not be used for top-N recommendation directly.

III. Group oriented MAP (GMAP)

MAP is an intuitive and popular metric to measure the average of the precision value obtained for the top-N items. However, to learn ranking latent factors for personalized recommendation by optimizing MAP leads to overfitting problems, as mentioned in Section I. Therefore, we propose a new optimization metric, GMAP.

A. Notations and Definitions

Suppose the implicit feedback data is from N users to M apps, and all apps are clustered to K groups according to some criteria, and ith group has k_i apps, $i = 1, 2, ..., K$. The vector $X_{ui} \in \mathcal{R}^{k_i}$ for user u records k_i entries which are denoted with x_{uil}: (1) $x_{uil} = 1$ indicates that user u has installed app l ($l = 1, 2, ..., k_i$) of group i. This shows that this user should have preferences for this group. (2) $x_{uil} = 0$ indicates the absence of an installation by user u to app l and thus app l makes no contribution to user u's preference for this group. However, the total preference of user u to app l can be estimated according to the ratio of nonzero entries in X_{ui}.

Then, the definition of Group-oriented Average Precision (GMAP) of a ranked list for user u is as follows:

$$GMAP_u = \frac{1}{\sum_{i=1}^{K}\mathbb{I}(\sum_{l=1}^{k_i} x_{uil})}\sum_{i=1}^{K}\frac{1}{R_{ui}}\mathbb{I}(\sum_{l=1}^{k_i} x_{uil}) \cdot \sum_{j=1}^{K}\frac{1}{k_j}\sum_{l=1}^{k_j} x_{ujl}\cdot\mathbb{I}(R_{uj}\leq R_{ui}) \quad (1)$$

where R_{ui} is the position of app group i in the current ranked list for user u, e.g., $R_{ui}=1$ denotes app group i is ranked in the highest position. $\mathbb{I}(x)$ is a binary indicator function, i.e., it is equal to 1 if x is true, otherwise 0. For notational convenience in the rest of this paper, we substitute some terms in Eq.(1) as shown as follows:

$$Z_u = \sum_{i=1}^{K}\mathbb{I}(\sum_{l=1}^{k_i} x_{uil}) \quad (2)$$

$$y_{uj} = \frac{1}{k_j}\sum_{l=1}^{k_j} x_{ujl} \quad (3)$$

$$I_{ui} = \mathbb{I}(\sum_{l=1}^{k_i} x_{uil}) = \mathbb{I}(k_i \cdot y_{ui}) \quad (4)$$

where Z_u is a substitution of the first two sum terms in Eq.(1). In fact, it is a constant normalizing coefficient counting the number of relevant app groups of user u, and y_{uj} represents the relevance degree of app group j to user u by calculating the ratio of relevant apps in that group. Different from y_{ui}, I_{ui} only indicates whether user u is interested in app group i, rather than how much the user likes it, i.e., $I_{ui}=1$ represents that this user is interested in the app group, otherwise he is not interested in.

GMAP is the average of $GMAP_u$ over all the users. Thus, it is defined based on $GMAP_u$:

$$GMAP = \frac{1}{N}\sum_{u=1}^{N} GMAP_u = \frac{1}{N}\sum_{u=1}^{N}\frac{1}{Z_u}\sum_{i=1}^{K}\frac{I_{ui}}{R_{ui}}\sum_{j=1}^{K} y_{uj}\mathbb{I}(R_{uj}\leq R_{ui}) \quad (5)$$

B. GMAP-based Method vs. MAP-based Method

The formulation of GMAP in Eq.(5) is similar to MAP, except in two aspects: (1) the ranking ontologies have been changed from app to app group, and (2) the relevant measurement is no longer binary , i.e., 1 or 0, but a ratio of relevant apps in a group, similar to graded relevance. However, this "graded relevance" varies from a traditional pseudo rating based on the frequency of the same app. It is more informative to represent the relevance of the group to users. Therefore, GMAP maintains the desirable qualities of MAP while overcoming the overfitting problem of MAP. GMAP-based methods reduce the concern about the precisions of the training data in order to pursue gains in precision on the test data, and thus it more likely measures average precision from an overall perspective to give more chance to rank unknown apps.

An interesting finding is, currently, some app markets provide different areas to show recommended apps. Different areas have different degrees of attractiveness but apps in the same area have the same attractiveness. The list provided by GMAP-based methods happens to be fit for this scenario. However, if we need a top-N recommendation, it can be derived based on the result from GMAP-based methods. Therefore, GMAP is an extension of MAP.

IV. A Top-N Recommendation Model for Smartphone Applications (GTRM)

In this section, we show in detail how to employ GMAP as an objective function to be optimized by GTRM. GTRM involves three steps: (1) app groups are discovered and organized, (2) app groups are ranked, and (3) apps are recommended based on the ranked app groups.

A. Smooth GMAP

GMAP is not smooth, which is caused by two terms in its definition, i.e., $1/R_{ui}$ and $\mathbb{I}(R_{uj}\leq R_{ui})$. These two terms show that GMAP depends on the ranking of the relevant app groups with respect to the user. This unsmoothness renders standard optimization methods unable to solve this optimization problem. Thus, we firstly smooth GMAP to an approximated version based on the core ideas borrowed from [1]. Then, the partial ordering relation between apps is represented by a real-value function as follows:

$$R_{uj}\leq R_{ui} \Rightarrow f_{uji}(\Theta) \quad (6)$$

where Θ represents the parameter vector specific to the arbitrary model. Here, we use the matrix factorization technique to learn two latent factor matrices, $U\in\mathcal{R}^{\mathcal{N}\times\mathcal{D}}$ and $V\in\mathcal{R}^{\mathcal{K}\times\mathcal{D}}$, which represent user and app groups with respect to D-dimension latent factors, respectively. Thus in the rest of the paper, $\Theta=(U,V)$. This pairwise relation can be depicted by a point-by-point computation as follows:

$$f_{uji}(\Theta) = f_{uj}(\Theta) - f_{ui}(\Theta) = \langle U_u, V_j\rangle - \langle U_u, V_i\rangle \quad (7)$$

Note that for notational convenience, we will omit Θ in the rest of this paper. The app groups are ranked in descending order, according to their predicted relevance degrees to a target user, e.g., f_{ui} for user u. Then, we can utilize the sigmoid function $g(x)=1/(1+e^{-k(x-c)})$ to approximate the term of the binary indicator as:

$$\mathbb{I}(R_{uj}\leq R_{ui}) \Rightarrow g(f_{uj}-f_{ui}) \quad (8)$$

Furthermore, the term of $1/R_{ui}$ can also be approximated by $g(x)$ as shown as follows:

$$\frac{1}{R_{ui}} \Rightarrow g(f_{ui}) \quad (9)$$

In order to simplify the analysis and computation process, suppose $c = 0$ and $k = 1$ in $g(x)$. Finally, we obtain a smoothed version of GMAP by substituting the approximations introduced in Eq.(8) and Eq.(9) into Eq.(5), as shown as follows:

$$GMAP \approx \frac{1}{N}\sum_{u=1}^{N}\frac{1}{Z_u}\sum_{i=1}^{K} I_{ui} g(f_{ui}) \sum_{j=1}^{K} y_{uj} g(f_{u(j-i)}) \tag{10}$$

where $f_{u(j-i)} := f_{uj} - f_{ui}$. The coefficient $\frac{1}{N}$ and $\frac{1}{Z_u}$ can be dropped, because it is independent of latent factors, and thus has no influence on the optimization of GMAP. Therefore, the final objective function of GTRM is shown as follows:

$$\mathcal{L}(U,V) = \sum_{u=1}^{N}\sum_{i=1}^{K} I_{ui} g(f_{ui}) \sum_{j=1}^{K} y_{uj} g(f_{u(j-i)}) - \frac{\lambda}{2}(||U||^2 + ||V||^2) \tag{11}$$

where $||U||$ and $||V||$ are Frobenius norms of U and V to avoid overfitting, and λ is the parameter controlling the magnitude of regularization.

B. Optimization Procedure

Now, we can use gradient ascent to optimize $\mathcal{L}(U,V)$ in Eq.(11) with respect to the model parameters U and V. Note that the gradient of the objective function for each user is derived independently of all other users. But for each app group, the gradient is derived based on one user in each iteration, since the relevance of the app group is dependent on the user-app group pair. Then, the gradient of $\mathcal{L}(U_u, V)$ and $\mathcal{L}(U, V_i)$ can be computed as follows:

$$\frac{\partial \mathcal{L}}{\partial U_u} = \sum_{i=1}^{K} I_{ui}\Big(g'(f_{ui}) \sum_{j=1}^{K} y_{uj} g(f_{u(j-i)}) \cdot V_i + g(f_{ui}) \sum_{j=1}^{K} y_{uj} g'(f_{u(j-i)}) \cdot (V_j - V_i)\Big) - \lambda U_u \tag{12}$$

$$\frac{\partial \mathcal{L}}{\partial V_i} = \Big(I_{ui} g'(f_{ui}) \sum_{j=1}^{K} y_{uj} g(f_{u(j-i)}) + \sum_{j=1}^{K}[I_{uj} y_{ui} g(f_{uj}) - I_{ui} y_{uj} g(f_{ui})] g'(f_{u(j-i)})\Big) U_u - \lambda V_i \tag{13}$$

The model parameters $\Theta = (U, V)$ are updated with a learning rate η based on the most common stochastic gradient ascent (SGA) algorithm:

$$\Theta \leftarrow \Theta + \eta \frac{\partial \mathcal{L}}{\partial \Theta} \tag{14}$$

The time complexity of updating the latent factors for each user based on Eq.(12) is $O(|I_u|^2 D)$, where $|I_u|$ denotes the number of relevant app groups of user u. Taking into account all the N users, the total complexity of updating matrix U is $O(N|\bar{I}|^2 D)$ in each iteration, and the $|\bar{I}|$ is the average number of relevant app groups across all users. Based on Eq.(13), the time complexity of updating the latent factors with respect to all relevant app groups for a given user u is also $O(|I_u|^2 D)$, since each relevant app group is computed to every relevant app group ranked higher than it. Similarly, in one iteration, the total complexity of updating matrix V across all users is the same as updating matrix U. Thus, our model finally takes $O(N|\bar{I}|^2 D)$ time to update the parameters in each iteration. Note that the term $(N|\bar{I}|)$ represents the number of all relevant app groups across all users. We substitute $|R|$ for $(N|\bar{I}|)$, and $|R| \gg |\bar{I}|, D$. This means the time complexity in each iteration can be regarded as $O(|R|)$. This is a linear complexity with respect to the number of all relevant ranking ontologies on the given dataset. As previously mentioned, the ranking ontology has been transferred from the app to a higher space, i.e., the app group. Then, the $|R|$ has been greatly decreased compared to individual app-based ranking methods, e.g., TFMAP.

C. Neighbour Selection

The key issue to integrate the item-item based recommendation idea to our model is the neighbour selection for apps. In fact, these unfamiliar apps are brought to higher positions by their relevant apps which have installed by target user. Thus, it is important to devise a better strategy for neighbour selection to improve the performance of GTRM. In this paper, we use two similarity functions to cluster apps to groups, which are attribute similarity and CF similarity respectively. The former is constructed using side information, for example, labels or classifications of apps. However, it is not always possible to have such information for the dataset, in which case we measure the distance between apps based on collaborative relation. Then, the similarity is calculated as follows:

$$S_{lk} = \frac{|N(l) \cap N(k)|}{\sqrt{|N(l)||N(k)|}} \tag{15}$$

where $|N(l)|$ denotes the number of users who once installed app l, or the number of attributes that app l has; $|N(l) \cap N(k)|$ denotes the number of users who both installed app l and k, or the number of attributes that both app l and k have, respectively in CF-based and attribute-based similarity.

Many clustering algorithms can be used in this framework, such as partitional algorithms and hierarchical algorithms. Note that the size of each cluster/app group should not be too large, since the size of the recommendation windows is limited and it is not appropriate to recommend apps to a target user only based on an individual app group. Thus, to simplify the analysis and control all groups with the same size by the variable S, we propose a one-iteration greedy clustering method. In this paper, to cluster a new app group, we select $S-1$ most similar apps to a ungrouped app over all

Algorithm 1 One-iteration cluster: NeighbourSelection

Input: All apps with their attributes or interacted user set, and the size of group S.
Output: Group list L
  Initialize $i = 1$
  repeat
    random select an ungrouped app l
    $L_i \leftarrow \{l \wedge S-1$ most similar apps to l in similarity network $\}$
    delete all apps $k \in L_i$ from similarity network
    $i \leftarrow i + 1$
  until All apps have been grouped.
  return L

other ungrouped apps in each step, then remove all these S apps from the similarity network until all apps are grouped. It will take a total of $O(nM^2/2S + nM/2) = O(KMn)$ times. Note that GMAP@S and GTRM@S denote GMAP and GTRM based on S-size app groups, respectively. The neighbour selection method is summarized in Algorithm 1, and the entire learning algorithm of GTRM is illustrated in Algorithm 2.

Algorithm 2 GTRM

Input: User-app pairs from training data, app group list L from Algo.1, app group size S, regularization term λ, learning rate η, and iteration times *itermax*.
Output: The learned latent factors U, V.
  Initialize $U^{(0)}, V^{(0)}$ with random values, and t=0;
  for each user u and each group i **do**
    for $l = 1$ to S **do**
      $X_{uil} = 1$ if app L_{il} has been installed by user u, otherwise $X_{uil} = 0$
    end for
  end for
  repeat
    for $u = 1$ to N **do**
      $U_u^{(t+1)} \leftarrow U_u^{(t)} + \eta \frac{\partial \mathcal{L}}{\partial U_u^{(t)}}$ based on Eq.(12)
      for $i \in \{L' |$the app groups interested by user $u\}$ **do**
        $V_i^{(t+1)} \leftarrow V_i^{(t)} + \eta \frac{\partial \mathcal{L}}{\partial V_i^{(t)}}$ based on Eq.(13)
      end for
    end for
    $t \leftarrow t + 1$
  until $t \geq$*itermax*
  return $U = U^{(t)}, V = V^{(t)}$

D. Top-N Recommendation List Construction

The outputs of GMAP-based methods contain the relevance degrees of all the app groups associated with a user. They can be applied to the scenario as previously discussed.

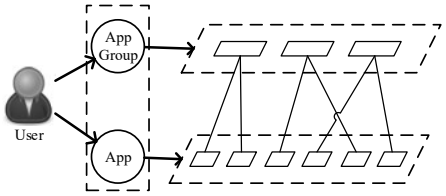


Figure 1. The user preferences are represented by two layers: the app group-wise and the app-wise layer

However, we can easily generate top-N recommendation app lists from these original outputs. Although the relevance degree of each app of the same group is equal, our model is able to make use of side information (e.g., the popularity) to rank these apps inside each app group. Moreover, we can produce the final top-N app lists through integrating two different rank frameworks as shown in Fig.1. The integration strategy can discover relevant apps from both the training set and the test set. At the same time, it can also reduce the influence of misoperation, which is a significant challenge for recommender systems using implicit feedback. In this paper, we propose three strategies for producing top-N app list, namely GTRM-All, GTRM-One and GTRM-Weight.

GTRM-All organizes all apps from top ranked app groups into a list in a cascading way. Then, for each app of the same app group, the ranks are based on the app popularity. GTRM-One gets the most popular app from each top ranked app group and organizes these into a list. In this case, the order of top-N apps inherits the order of app groups. GTRM-Weight returns the top-N recommendation list based on a synthetic utility function, which is defined based on the NDCG metric. The function is shown as follows:

$$r_{uil} = \alpha * \frac{1}{log(R_{ui}+1)} + (1-\alpha) * \frac{1}{log(P_{ul}+1)} \quad (16)$$

where r_{uil} represents the final relevance degree of app l of app group i for user u, R_{ui} and P_{ul} are the positions of app group i and of app l in two different rank frameworks, respectively, and $\alpha \in [0, 1]$ is the parameter controlling the weight between these two ranks. To ensure the unity of an experienced target, we still employ the app popularity ranks as the baseline individual app ranks.

V. Experimental Evaluation

We compare our methods with some state-of-the-art alternative approaches to investigate the performance from multiple perspectives. We also test the influence of different parameter settings followed by a discussion.

A. Experiment Setting

1) Dataset: Our original dataset contains 2,000,000 installations records from 19,859 users and 3,907 apps. For the purpose of our experiments, we sample users by the

number of their installation apps and users with more than 25 installed apps are retained in the dataset. Then the dataset for experiments contains 4670 users and 2,259 apps with 470,000 installations records in total. The sparsity of the user-app matrix in this dataset is 95.54%. Besides, we extract categorical labels of the apps from an online app store, *www.wandoujia.com*, and then the attribute similarity is based on 226 categories for mobile apps.

2) Experimental Approach: We separate the dataset into a training set and a test set by randomly removing a certain number of records from each user's historical data. For example, "Given 5" denotes that for each user, we randomly select five records from original historical records to form test set, and use the remaining data to form training set.

We use two evaluation metrics, MAP and precision, to measure the recommendation performance. Precision is a very popular metric for evaluating recommender systems. It reflects the ratio of relevant apps in the ranked list given a truncated position, e.g., P@5 denotes the ratio of relevant apps in the first five apps of the ranked list. MAP is the listwise metric which has been introduced above.

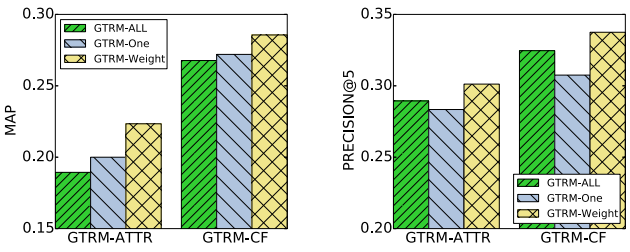
To investigate the performance of GTRM, we implement four comparative methods, namely iPOP, ICF, wALS-ITEM [14] and TFMAP along with three GTRM methods, GTRM-All, GTRM-One and GTRM-Weight. The iPOP recommends apps according to app popularity. ICF is a traditional item-based CF method using Pearson correlation-based similarity. wALS-ITEM is a weighted matrix factorization method using the alternative least square (ALS) learning method. The apps' weights are inversely proportional to the popularity of the apps. TFMAP is an L2R method using the MAP metric as the objective function.

All the parameters of each model are set to their optimal values in the experiments, i.e., the number of features in TFMAP, wALS-ITEM and GTRM is 10, and for ICF, we set the number of nearest neighbours to 30. The remaining parameters are also empirically tuned, such as the $\alpha = 0.9$ for GTRM-Weight, the size of the app group is set to 3, the regularization parameter $\lambda = 0.1$, and the learning rate $\eta = 0.002$. In addition, we utilize 5-fold cross validation on the datasets.

As Fig.2 shows, GTRM-CF (i.e., with CF similarity) performs better than GTRM-ATTR (i.e., with attribute similarity) no matter which method of top-N recommendation list construction we chose. This is reasonable since CF similarity possesses a higher dimension vector, depicting the latent properties of apps compared to the attribute dimension vector on these two datasets. Thus, without loss of generality, we select GTRM-CF to represent GTRMs in the following experiments unless otherwise stated.

B. Effectiveness of the Learning Process

We first validate the effectiveness of the GTRM learning process. We measure the GMAP values on both the training and the test sets across the iterations, and the results are shown in Fig.3a. Then, we can observe that both GMAP measures quickly improve towards an optimal value with only a few iterations, i.e., 10. This proves the effectiveness of our learning process. Furthermore, we present an experimental study that examines the change of the position of the relevant app groups, i.e., the maximal and minimal rank of relevant app groups in the ordered list. Both of them are the average values across all the users, and as Fig.3b shows, they decrease with each iteration as the model is gradually optimized. In addition, the average minimal rank of relevant app groups is close to one. These observations also provide empirical evidence that the learning process of GTRM is efficient.



(a) MAP (b) Precision

Figure 2. MAP and precision of three top-N lists based on different strategies

C. Performance Comparison

Now, we measure MAP and precision values on the test set, based on the different number of top apps in the recommendation list, such as MAP@5 and P@5. The results are shown in Table I, where all GTRM methods achieve better precision and MAP performance, In particular, the GTRM-Weight obtains the best results of all methods with respect to precision and MAP, then GTRM-All and GTRM-One, followed by TFMAP. TFMAP achieves higher values of precision and MAP than the remaining methods, i.e., iPOP, wALS-ITEM and ICF. This indicates that the L2R approach is effective for Top-N recommendation. In addition, an interesting observation is that GTRM-All is better than GTRM-One in relation to the the performance of precision. However, in relation to the performance of MAP, GTRM-One is better than GTRM-All. This is because GTRM-One can discard both relevant apps and irrelevant apps in the top app groups. The reduction of relevant apps on the test set may not reduce the value of MAP, but it will reduce the value of precision. The results also show that the performance of all methods becomes better when the number of relevant apps to be tested increases from 5 to 20. The reason for this is that more relevant apps in the test data can better cover the preference of users.

Table I
MAP AND PRECISION PERFORMANCE OF GTRM@3 COMPARED TO BASELINES

Metric	Methods	Given 5			Given 10			Given 20		
		@1	@3	@5	@1	@3	@5	@1	@3	@5
MAP	iPoP	0.2643	0.2701	0.2604	0.4359	0.4792	0.4749	0.5259	0.5892	0.5802
	ICF	0.3019	0.3289	0.3089	0.4989	0.5532	0.5572	0.6041	0.6203	0.6213
	wALS-ITEM	0.3054	0.4033	0.4034	0.5089	0.5732	0.5728	0.6816	0.7004	0.7212
	TFMAP	0.3447	0.4025	0.4172	0.5396	0.6122	0.6075	0.7023	0.7335	0.7575
	GTRM-One	0.3646	0.4582	0.4553	0.5443	0.6364	0.6325	0.7341	0.7352	0.7702
	GTRM-All	0.3554	0.4463	0.4524	0.5374	0.6303	0.6291	0.7253	0.7217	0.7675
	GTRM-Weight	**0.3694**	**0.4593**	**0.4564**	**0.5574**	**0.6423**	**0.6486**	**0.7348**	**0.7462**	**0.7726**
Precision	iPoP	0.2643	0.1902	0.1342	0.4359	0.2792	0.2056	0.5259	0.4692	0.3149
	ICF	0.3089	0.2032	0.1532	0.4389	0.2832	0.2073	0.5041	0.5003	0.3942
	wALS-ITEM	0.3108	0.2202	0.1701	0.4589	0.3132	0.2445	0.5216	0.5104	0.4442
	TFMAP	0.3404	0.2148	0.1653	0.5482	0.3769	0.2998	0.5681	0.5261	0.4501
	GTRM-One	0.3618	0.2441	0.1780	0.5489	0.3933	0.3075	0.5686	0.5401	0.4510
	GTRM-All	0.3684	0.2505	0.1833	0.5439	0.3940	0.3246	0.5686	0.5418	0.4519
	GTRM-Weight	**0.3690**	**0.2542**	**0.1875**	**0.5525**	**0.3983**	**0.3374**	**0.5875**	**0.5731**	**0.4723**

D. Comparison with TFMAP

Since TFMAP is also based on L2R, we compare GTRMs with TFMAP in more detail. That is, we compare changes in the MAP performance with the iterations of the learning process. Here, MAP is measured on both the training and the test sets across the iterations, and the size of the app group is still set to 3 for GTRMs. The results of GTRMs are shown in Fig.3c along with the results of TFMAP. It can be observed that the MAP measures gradually increase towards an optimal value with only a few iterations, e.g., 5 for TFMAP and GTRM-Weight. After the turning points, the MAP measures start to decrease. This indicates GTRMs and TFMAP are both effective as long as the number of iterations are controlled to avoid model overfitting. Note that the MAP measure of TFMAP increases sharply on the training data, but is almost flat on the test data. This property is also shown in the experiment from [1]. It can be found that the steady performance of GTRMs, including GTRM-All, GTRM-One and GTRM-Weight, is better than TFMAP on the test set, though the TFMAP obtains the best performance on the training set. This justifies our opinion that GMAP-based methods are more effective compared to MAP-based methods on the ranking recommendation problem. As shown in Fig.3d, GTRMs also learn faster than TFMAP and the time decreases when the size of the groups increases for GTRMs.

E. Performance on New Apps

GTRM can recommend new apps while all other comparative methods cannot. As this new app cannot be clustered in the right app group based on CF similarity, we employ attribute similarity in this experiment. We randomly eliminate one app from the training set. This means all actions on this app are removed. Thus, this app can be viewed as a new app to the user. After training the model, the performance of the model is evaluated on the test data. In order to calculate precision, we only evaluate the performance with users who have interaction records with this new app in the test set. Then, we examine GTRM-All and GTRM-Weight since both of them always retain the new app in the recommendation lists, while GTRM-One does not. The results are shown in Fig.3e, indicating GTRM is able to recommend the new apps.

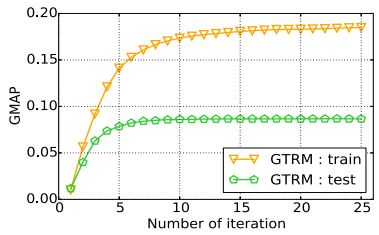
F. Influence of Group Size

The final experiment investigates the influence of the size of app groups. To simplify the analysis, we control all groups of the same size by the variable S, as discussed in Section IV-C. Then, we show the MAP measured on the test data across the S, as shown in Fig.3f. We observe that the MAPs of all GTRMs firstly increase towards the optimal value, then after the turning points, i.e., 3, the MAP lines decline linearly. This indicates that the size of both GTRMs should not be too large for the reason discussed in Section IV-C. It can also be seen that there is not much difference between our methods and TFMAP when $S = 1$. In fact, our model can be viewed as a generalized form of TFMAP as mentioned. Thus, when the individual app is considered as the group owning only one element, GTRM demonstrates a similar performance to TFMAP.

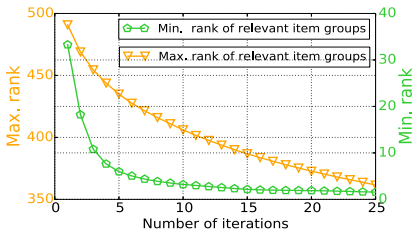
VI. CONCLUSION AND FUTURE WORK

In this paper, we firstly proposed a new optimization metric, GMAP, which is a general form of the MAP metric. Then, we optimized GMAP to learn the latent factors with respect to users and app groups. Furthermore, in order to construct app groups, we proposed a one-iteration greedy algorithm to cluster apps and then, methods to derive the top-N app recommendation list were suggested. Finally, the extensive experiments proved that our algorithm is effective and outperforms the others on various evaluation metrics.

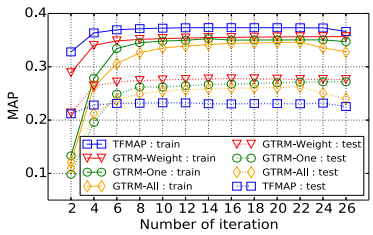
The model can still be improved in different aspects in the future, such as the apps can be clustered by applying more advanced methods. In particular, in this paper, in order to simplify the model, we have assumed that all groups share



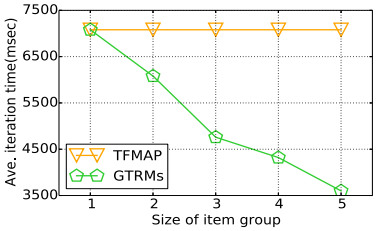
(a) The GMAP values



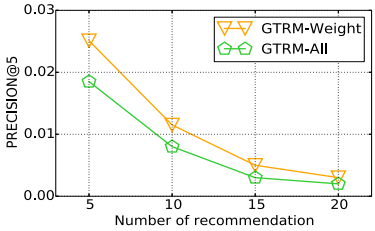
(b) The average maximal and minimal ranks



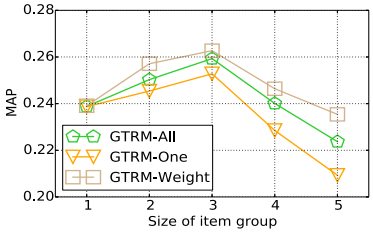
(c) The MAP values



(d) The average time across the size of app groups



(e) The precision for new app recommendation



(f) The MAP values across the size of app groups

Figure 3. The results of the experiments

the same size. It would be an interesting study to cluster apps to groups of different sizes.

ACKNOWLEDGMENT

This work is supported by China National Science Foundation (Granted Number 61472253).

REFERENCES

[1] Y. Shi, A. Karatzoglou, L. Baltrunas, M. Larson, A. Hanjalic, and N. Oliver, "Tfmap: optimizing map for top-n context-aware recommendation," in *Proceedings of the 35th International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 2012, pp. 155–164.

[2] Y. Yue, T. Finley, F. Radlinski, and T. Joachims, "A support vector method for optimizing average precision," in *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 2007, pp. 271–278.

[3] S. E. Robertson, E. Kanoulas, and E. Yilmaz, "Extending average precision to graded relevance judgments," in *Proceedings of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 2010, pp. 603–610.

[4] Y. Koren, R. Bell, C. Volinsky *et al.*, "Matrix factorization techniques for recommender systems," *Computer*, vol. 42, no. 8, pp. 30–37, 2009.

[5] X. Ning and G. Karypis, "Slim: Sparse linear methods for top-n recommender systems," in *Data Mining (ICDM), 2011 IEEE 11th International Conference on*. IEEE, 2011, pp. 497–506.

[6] H. Chen, D. Niu, K. Lai, Y. Xu, and M. Ardakani, "Separating-plane factorization models: Scalable recommendation from one-class implicit feedback," in *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*. ACM, 2016, pp. 669–678.

[7] M. Jamali and L. Lakshmanan, "Heteromf: recommendation in heterogeneous information networks using context dependent factor models," in *Proceedings of the 22nd International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 2013, pp. 643–654.

[8] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme, "Bpr: Bayesian personalized ranking from implicit feedback," in *Proceedings of the Twenty-fifth Conference on Uncertainty in Artificial Intelligence*. AUAI Press, 2009, pp. 452–461.

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

[10] K. Shi and K. Ali, "Getjar mobile application recommendations with very sparse datasets," in *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2012, pp. 204–212.

[11] J. Lin, K. Sugiyama, M.-Y. Kan, and T.-S. Chua, "New and improved: modeling versions to improve app recommendation," in *Proceedings of the 37th international ACM SIGIR conference on Research & development in information retrieval*. ACM, 2014, pp. 647–656.

[12] ——, "Addressing cold-start in app recommendation: latent user models constructed from twitter followers," in *Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval*. ACM, 2013, pp. 283–292.

[13] C. Liu, J. Cao, and J. He, "Leveraging kernel incorporated matrix factorization for smartphone application recommendation," in *International Conference on Database Systems for Advanced Applications*. Springer, 2017, pp. 459–474.

[14] R. Pan, Y. Zhou, B. Cao, N. N. Liu, R. Lukose, M. Scholz, and Q. Yang, "One-class collaborative filtering," in *Data Mining, 2008. ICDM'08. Eighth IEEE International Conference on*. IEEE, 2008, pp. 502–511.