A Weighted Meta-graph Based Approach for Mobile Application Recommendation on Heterogeneous Information Networks

Fenfang Xie, Liang Chen(✉), Yongjian Ye, Yang Liu, Zibin Zheng, and Xiaola Lin

School of Data and Computer Science,
National Engineering Research Center of Digital Life, Sun Yat-sen University,
Guangzhou 510006, China
{xieff5,yeyj7,liuy296}@mail2.sysu.edu.cn,
{chenliang6,zhzibin,linxl}@mail.sysu.edu.cn

Abstract. Explosive growth in the number of mobile applications (apps) makes it difficult for users to find relevant apps. Therefore, it is an urgent task to recommend desired apps for users. Traditional approaches focus on exploiting the context information, user's interest, privacy, security and other features for app recommendation. Most of them do not consider heterogeneous information network (HIN) in the scenario of mobile app recommendation. HIN contains rich structure and semantic information, and it can satisfy various requirements of users and generate better recommendation results. In this paper, we propose a **W**eighted **M**eta-**G**raph based approach for app **Rec**ommendation, called WMGRec, on HIN. Specifically, we firstly introduce the concept of weighted meta-graph, which not only distinguishes different rating scores to depict the subtle semantics but also utilizes meta-graph to capture complex semantics. And then, we apply weighted meta-graph to measure the semantic similarity between users and apps. Furthermore, we leverage non-negative matrix factorization on user-app similarity matrix to obtain user latent features and app latent features. Finally, the concatenated user and app latent features are fed into the factorization machine & deep neural network model to learn the higher-order interactions and get the final prediction score. Extensive experiments conducted on two real-world datasets validate the effectiveness of the proposed approach compared to state-of-the-art recommendation algorithms.

Keywords: Mobile app recommendation · Meta-graph
Heterogeneous information network · Factorization
Deep neural network

1 Introduction

Mobile devices (e.g., smart phones, tablet computers) have been becoming increasingly popular in recent years. They are gradually becoming a part of our daily life for study, entertainment, social intercourse, browsing news and businesses. Mobile devices promote the explosive growth of mobile apps. It becomes considerably difficult and time-consuming for users to find a relevant mobile app from a huge number of mobile apps. Therefore, it is essential to help users to locate their desired apps. Mobile app recommendation is a good choice to solve this problem and enhance user experience.

Previous studies concentrate on considering the context information, user's interest, privacy, security and other features for mobile app recommendation [2,5,8,18,21]. They usually employ collaborative filtering (CF) (e.g., item-based CF, user-based CF and matrix factorization) methods to recommend apps to users. Most of them do not consider the rich structure and semantic information on HIN. HIN has been widely exploited to data mining tasks, such as similarity measure, clustering, classification, link prediction and recommendation [10]. There exist multiple object types (e.g., users, mobile apps, reviews, and rating scores) and rich relations among object types (e.g., use and used by relation between users and apps, write and written by relation between users and reviews) in the scenario of app recommendation, which naturally constitutes an HIN. Exploiting the rich structure and semantic information can reveal subtle relations among objects. Therefore, it will satisfy different kinds of requirements of users and improve recommendation performance. As an important characteristic of HIN, meta-path is usually applied to model the multiple semantic information. HIN accomplishes traditional CF methods by building various meta-paths. For example, if we want to recommend apps to users, we can build a simple meta-path "User-App-User" or "User-APP-APP" and learn from this meta-path to make generalizations. These two kinds of meta-paths achieve the user-based CF and item-based CF methods, respectively. From HIN's schema, we can define more complicated meta-paths like "User-Review-Topic-Review-App". This meta-path defines a similarity to measure whether a user tends to like an app if his/her reviews are similar to those written by other users for the same app. In addition, users only use and rate a limited number of apps in the real-world. Therefore, the user-app rating matrix is considerably sparse. As a result, the recommendation results obtained by matrix factorization may be inaccurate. The rich structure and semantic information on HIN can alleviate this sparsity problem.

Inspired by the significance of HIN, we prefer to apply HIN to solve our app recommendation problem. When utilizing meta-path based similarities to app recommendation, there exist the following two major challenges: (1) Traditional HIN and meta-path do not consider the attribute values (the rating scores in app recommendation) on links. Ignoring the rating scores between users and apps may result in bad similarity discovery and cannot reveal the subtle difference. Figure 1 shows an example, given a meta-path "User-APP-User", if both *U1* and *U2* provide a rating score as 5 to Facebook, then *U1* and *U2* are quite similar due to the same preference. If *U1* gives a 5 score to Facebook and *U2* gives a 1 score to Facebook, then *U1* and *U2* may not be so similar due to the totally different taste. But the conventional meta-path regards these two cases as the same. Therefore, it will result in inaccurate similarity and cannot recognize the

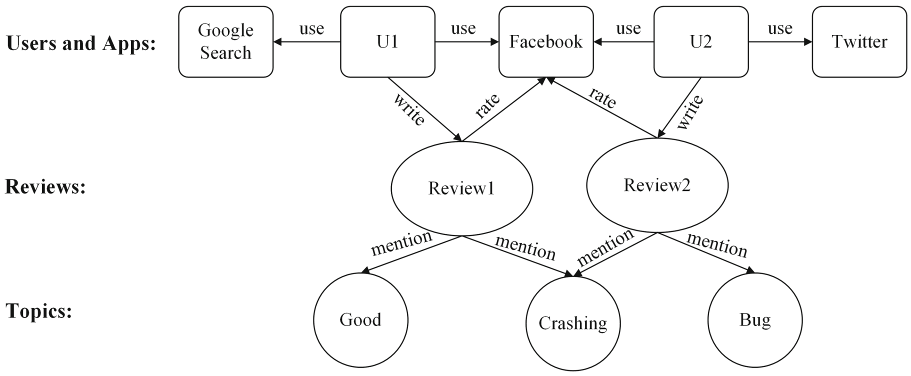


Fig. 1. Example of HIN built by mobile app dataset

subtle difference. However, in the scenario of app recommendation, this difference is very important because it can help to more accurately capture interactions between users and apps. It is essential to extend the conventional meta-path for considering attribute values on links. (2) Meta-path may not be very suitable to describe the rich structure and capture complicated semantics. Figure 1 presents a concrete example of this kind of case. Given a meta-path "User - Review - Topic - Review - User", which can be used to capture users' similarity since both of them write reviews and mention the same topic. However, if we intend to capture the semantic relation that *U1* and *U2* provide rating scores to the same app (e.g., Facebook), and at the same time, they mention the same topic (e.g., crashing), the meta-path fails. As a result, it is essential to find a more suitable way to capture such complex semantics. Meta-graph [3,6,19] is a good way to represent the above mentioned semantic relation. Meta-graph computes similarity between homogeneous type of entities over HINs, which can capture more complex semantics that meta-path cannot.

To alleviate the above challenges, this paper proposes a weighted meta-graph based approach for mobile apps recommendation on HIN. Specifically, we first describe the concept of weighted meta-graph rather than meta-path to capture the complicated semantics. Moreover, weighted meta-graph is used to measure the semantic similarity between users and apps through distinguishing different rating scores. And then, we use all latent features of all meta-graphs. For each meta-graph, we first compute the user-app similarity matrix under the guidance of the weighted meta-graph, and then exploit non-negative matrix factorization to obtain user latent features and app latent features. Finally, by concatenating different user latent feature and app latent feature vectors computed from different weighted meta-graphs, we use the factorization machine & deep neural network (DeepFM) model to learn the low- and high-order interactions between users and apps, and to further learn the rating matrix. Experimental results on two real-world datasets demonstrate the proposed approach outperforms other state-of-the-art recommendation approaches.

In summary, the major contributions of this paper are three folds:

- To the best of our knowledge, we are the first to introduce the concept of weighted meta-graph. The weighted meta-graph considers rating scores on links and capture more complex semantic similarity between users and apps.
- We propose a weighted meta-graph based approach, which not only effectively integrates rich structure and semantic information contained in app recommendation, but also captures the higher order interactions between users and apps. Furthermore, WMGRec can satisfy different kinds of requirement of users and improve recommendation performance.
- Empirical studies on two real-world datasets verify the effectiveness of WMGRec. WMGRec achieves better performance than other recommendation algorithms with the help of weighted meta-graph, non-negative matrix factorization and DeepFM.

The rest of this paper is organized as follows. Section 2 presents the proposed approach in detail. Section 3 analyzes and discusses the experimental results and impact of parameters. Section 4 introduces some related works. Section 5 concludes the paper and gives some future directions.

2 Weighted Meta-graph Based Mobile App Recommendation

In this section, the details of our proposed approach are described. For simplicity, the same definitions of HIN and the corresponding network schema in [14] are adopted in this paper. An illustration of the HIN network schema and meta-graphs are shown in Fig. 2(a) and (b), respectively.

2.1 Basic Concepts

Given an HIN $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ and its network schema $\mathcal{T}_{\mathcal{G}} = (\mathcal{L}, \mathcal{Q})$, where $\mathcal{V}$ is the node set, $\mathcal{E}$ is the link set, $\mathcal{L}$ is the node type set, $\mathcal{Q}$ is the link type set. We propose a novel concept, named weighted meta-graph, to capture complex relationship between two HIN objects.

Definition 1 Weighted Meta-graph. A meta-graph $\mathcal{D}$ is a directed acyclic graph with a single source node $\mathcal{N}_s$ (i.e., with in-degree 0) and a single target node $\mathcal{N}_t$ (i.e., with out-degree 0). Formally, $\mathcal{D} = (\mathcal{N}, \mathcal{M}, \mathcal{N}_s, \mathcal{N}_t)$, where $\mathcal{N} \subseteq \mathcal{V}$ is a set of nodes and $\mathcal{M} \subseteq \mathcal{E}$ is a set of links. For any node $x \in \mathcal{N}$, $x \in \mathcal{L}$; for any link $(x, y) \in \mathcal{M}, (x, y) \in \mathcal{Q}$. If any link (x, y) has an attribute value c on it, where $c \in (0, \mathcal{C})$, $\mathcal{C} \in \mathbb{R}^+$, the meta-graph is called a *weighted meta-graph*; if any link (x, y) has no attribute value on it, the meta-graph is called an *unweighted meta-graph.*

Figure 2(b) illustrates a meta-graph $\mathcal{D}_3$, which depicts that two users provide rating scores to the same app, and have mentioned the same topic at the same

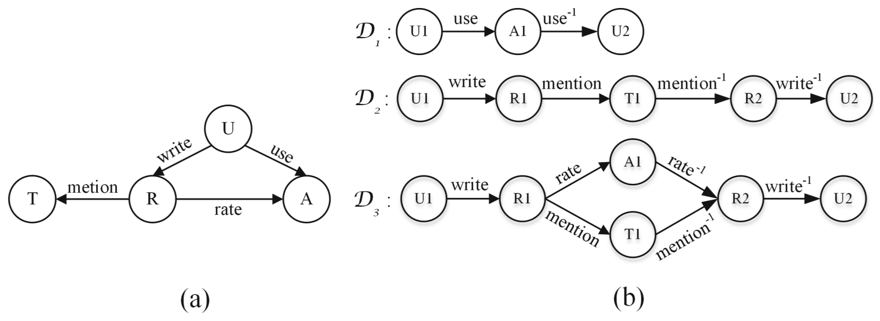

U
write
use
T
metion
R
rate
A
(a)
$\mathcal{D}_1$: U1 use A1 use^{-1} U2
$\mathcal{D}_2$: U1 write R1 mention T1 mention^{-1} R2 write^{-1} U2
$\mathcal{D}_3$: U1 write R1 rate A1 rate^{-1} mention T1 mention^{-1} R2 write^{-1} U2
(b)

Fig. 2. Example of HIN network schema and meta-graphs used for mobile app dataset. T: topics extracted from reviews; R: reviews; U: users; A: apps.

time. However, a meta-path fails to capture such complex relationship. A meta-graph is convenient to express this kind of relationship. A meta-path (e.g., $\mathcal{D}_1$ or $\mathcal{D}_2$ in Fig. 2(b)) is a special case of a meta-graph. Therefore, we call it meta-graph uniformly in the following. In addition, an unweighted meta-graph cannot capture the subtle deference of users' preference due to lack of considering rating scores on links.

Definition 2 Atomic Meta-graph. If all attribute values in a weighted meta-graph take a specific value, the meta-graph is called an atomic meta-graph. A weighted meta-graph is a group of atomic meta-graphs which contains all atomic meta-graphs that satisfy the constraint $\mathcal{C}$.

The attribute value c in our datasets is an integer which is in the range of $[1, 5]$. A toy example is shown in Fig. 3, "$\mathrm{U}\overset{1}{-}\mathrm{A}\overset{1}{-}\mathrm{U}$" and "$\mathrm{U}\overset{4}{-}\mathrm{A}\overset{4}{-}\mathrm{U}$" are two different atomic meta-graphs. The weighted meta-graph $\mathrm{U}\overset{i}{-}\mathrm{A}\overset{j}{-}\mathrm{U}|i = j$ is a group of five atomic meta-graphs (e.g., $\mathrm{U}\overset{2}{-}\mathrm{A}\overset{2}{-}\mathrm{U}$, $\mathrm{U}\overset{3}{-}\mathrm{A}\overset{3}{-}\mathrm{U}$ and $\mathrm{U}\overset{5}{-}\mathrm{A}\overset{5}{-}\mathrm{U}$).

Definition 3 Commuting Matrix. A commuting matrix M for a meta-graph $\mathcal{D} = (\mathcal{L}_1\mathcal{L}_2...\mathcal{L}_l)$ is defined as $M = W_{\mathcal{L}_1\mathcal{L}_2} \circ W_{\mathcal{L}_2\mathcal{L}_3} \circ ... \circ W_{\mathcal{L}_{l-1}\mathcal{L}_l}$, where $W_{\mathcal{L}_i\mathcal{L}_j}$ is the adjacency matrix between type $\mathcal{L}_i$ and type $\mathcal{L}_j$. $\circ$ can be two operations (Hadamard product "$\odot$" and multiplication "$\cdot$").

To better comprehend the computation of commuting matrix, we take the meta-graphs in Fig. 2(b) for example. For meta-graph $\mathcal{D}_1$, the commuting matrix $M_{\mathcal{D}_1}$ is computed as $M_{\mathcal{D}_1} = W_{UA} \cdot W_{AU}$. For meta-graph $\mathcal{D}_3$, the computation of the commuting matrix $M_{\mathcal{D}_3}$ should be divided into four steps:

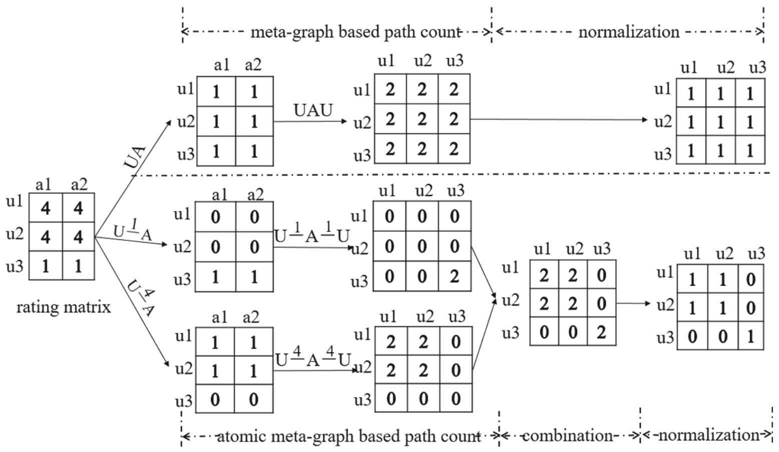
- Compute t_1: $t_1 = W_{RA} \cdot W_{AR}$,
- Compute t_2: $t_2 = W_{RT} \cdot W_{TR}$,
- Compute t_3: $t_3 = t_1 \odot t_2$,
- Compute $M_{\mathcal{D}_3}$: $M_{\mathcal{D}_3} = W_{UR} \cdot t_3 \cdot W_{RU}$.

2.2 Meta-graph Based Similarity

There are plenty of similarity measurements [16] to compute the similarity between two objects. Herein, we exploit the most widely used similarity measurement in HIN. Given a meta-graph $\mathcal{D}$, PathSim [14] between two objects i and j from the same type can be calculated as:

$$S_{i,j} = \frac{2M_{i,j}}{M_{i,i} + M_{j,j}} \tag{1}$$

where M is the commuting matrix for the meta-graph $\mathcal{D}$, $M_{i,i}$ and $M_{j,j}$ are the visibility for i and j in the network, namely, the number of meta-graphs between themselves.



meta-graph based path count
normalization
rating matrix
UA
UAU
$U\xrightarrow{1}A$
$U\xrightarrow{1}A\xrightarrow{1}U$
$U\xrightarrow{4}A$
$U\xrightarrow{4}A\xrightarrow{4}U$
atomic meta-graph based path count
combination
normalization

Fig. 3. A toy example for pathsim based similarity measurement on meta-graph

A toy example for PathSim based similarity measurement on unweighted meta-graph (the upper part) and weighted meta-graph (the lower part) is presented in Fig. 3. After the process of path count (the number of path instances between two objects), atomic meta-graph combination and normalization, the traditional unweighted meta-graph considers that $u1, u2, u3$ are all similar with each other. The weighted meta-graph draws a conclusion that only $u1$ and $u2$ are similar due to their same taste in apps. Obviously, the result found by weighted meta-graph is more accurate than that of unweighted meta-graph.

After obtaining the meta-graph based similarity of users, we can find the similar users of a target user under a given meta-graph. And then, the similarity between the target user and the app can be inferred according to his/her similar users.

$$\begin{aligned} H^{(l)}_{u,i,r} &= \sum_{v} S^{(l)}_{u,v} \times B_{u,i,r}, \\ B_{u,i,r} &= \{ \begin{matrix} 1 & \mathcal{R}_{u,i} = r \\ 0 & otherwise, \end{matrix} \end{aligned} \tag{2}$$

where $S_{u,v}^{(l)}$ is the PathSim based similarity between user u and user v under the meta-graph $\mathcal{D}_l$. $B_{u,i,r}$ is an indication function. If user u provides a rating score r to app i, $B_{u,i,r}$ is equal to 1; otherwise, it is equal to 0. $H_{u,i,r}^{(l)}$ is the intensity of user u rating app i with score r.

The similarity $\hat{\mathcal{R}}_{u,i}^{(l)}$ between user u and app i along a given meta-graph $\mathcal{D}_l$ can be computed as follows:

$$\hat{\mathcal{R}}_{u,i}^{(l)} = \sum_{r=1}^{N} r \times \frac{H_{u,i,r}^{(l)}}{\sum_{k=1}^{N} H_{u,i,k}^{(l)}} \tag{3}$$

2.3 Meta-graph Based Latent Features

By repeating the above similarity computation process to all meta-graphs, we can obtain L different kinds of similarity matrices. We denote these similarity matrices as $\mathcal{R}^{(1)}, \dots, \mathcal{R}^{(L)}$. For each similarity matrix under a given meta-graph, a non-negative matrix factorization [7] technique is applied to get the low-rank matrix representation for users and apps. The idea of matrix factorization is that a matrix is approximately equal to the multiplication of two low-rank matrices. Mathematically,

$$\begin{aligned} \hat{\mathcal{R}} \approx & \;\; \mathcal{U} \cdot \mathcal{A} \\ s.t. & \;\; \mathcal{U} \geq 0, \mathcal{A} \geq 0 \end{aligned} \tag{4}$$

By minimizing the following equation, the low-rank representation of users and apps can be obtained:

$$\min_{\mathcal{U},\mathcal{A}} \frac{1}{2} \| I(\mathcal{R} - \hat{\mathcal{R}}) \|_F^2 + \frac{\lambda_1}{2} \|\mathcal{U}\|_F^2 + \frac{\lambda_2}{2} \|\mathcal{A}\|_F^2 \tag{5}$$

where I is the indicator matrix. If the rating of user i on app j is observed, I_{ij} is 1; otherwise, I_{ij} is 0. λ_1 and λ_2 are the hyper-parameters that control the influence of the regularization term to avoid overfitting. $\|\cdot\|_F$ is the Frobenius norm.

After factorizing all similarity matrices $\mathcal{R}^{(1)}, \dots, \mathcal{R}^{(L)}$, we can obtain L types of low-rank representation matrix pairs $(\mathcal{U}^{(1)}, \mathcal{A}^{(1)}), \dots, (\mathcal{U}^{(L)}, \mathcal{A}^{(L)})$. For each pair of low-rank representation matrix $\mathcal{U}^{(l)}$ and $\mathcal{A}^{(l)}$, we concatenate the ith row of $\mathcal{U}^{(l)}$ (e.g., $u_i^{(l)}$) and the jth row of $\mathcal{A}^{(l)}$ (e.g., $a_j^{(l)}$). Taking all meta-graphs into consideration, we can finally get a sample or a feature vector x^n like this:

$$x^n = \underbrace{u_i^{(1)}, \dots, u_i^{(l)}, \dots, u_i^{(L)}}_{L \times K}, \underbrace{a_j^{(1)}, \dots, a_j^{(l)}, \dots, a_j^{(L)}}_{L \times K} \tag{6}$$

where the value of i is from 1 to the number of users, the value of j is from 1 to the number of apps. K is the number of latent factors. It could be different for different matrices, but we keep it the same for simplicity.

2.4 WMGRec Model

After the above calculation, we can get the concatenated feature vectors of samples. These features vectors are fed into DeepFM model to learn the low- and high-order interactions between users and apps.

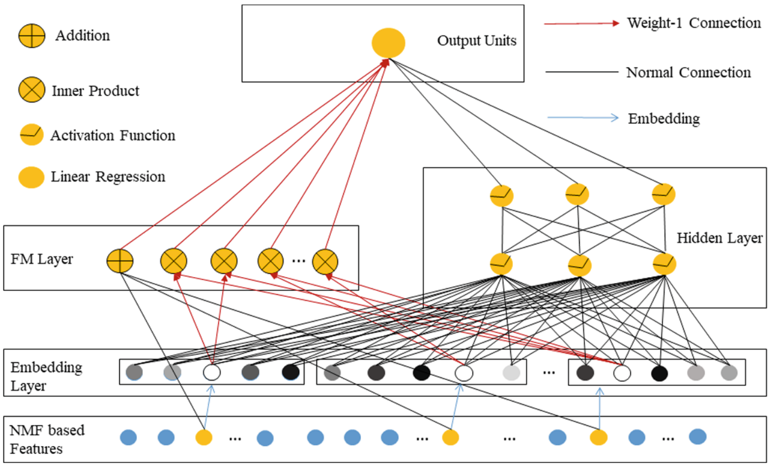


Fig. 4. The architecture of DeepFM

The architecture of DeepFM [4] is introduced in Fig. 4. The DeepFM model is divided into two components: FM (factorization machine) component and DNN (deep neural network) component. The two components share the same input feature vectors. The FM component is an improved factorization machine [9]. As aforementioned, we have obtained the feature vectors. These feature vectors are firstly fed into the embedding layer. The embedding techniques in neural network have the advantage of reducing the dimensionality of feature vectors. Secondly, the embedding vectors are put into the FM layer to learn the low-order interactions between users and apps. FM is a good way to deal with sparsity data and can be calculated in linear time. The output of FM component is formulated as follows:

$$\hat{y}(X) = w_0 + \sum_{i=1}^{f} w_i x_i + \sum_{i=1}^{f-1} \sum_{j=i+1}^{f} < V_i, V_j > x_i x_j \qquad (7)$$

where, f is the length of the feature vector. w_0 is the global bias, w_i is the weight of the ith variable. A row V_i within $V \in \mathbb{R}^{f \times d}$ describes the ith variable with d factors. X is the feature vector, which is concatenated by user latent features and app latent features.

The DNN component is a traditional feed-forward neural network in which data flows from the input layer to the output layer without looping back. DNN can model complex non-linear relationships. For DNN component, embedding vectors are put into one or more hidden layer to learn the high-order interactions between users and apps. The output of the embedding layer can be formulated as:

$$z^{(0)} = [e_1, e_2, ..., e_m] \tag{8}$$

where e_i is the embedding of the ith field and m is the number of fields. Then, $z^{(0)}$ is fed into the DNN, and the forward process is:

$$z^{(q+1)} = \sigma(\mathcal{W}^{(q)} z^{(q)} + b^{(q)}) \tag{9}$$

where q is the layer depth and σ is an activation function. $z^{(q)}, \mathcal{W}^{(q)}, b^{(q)}$ are the output, model weight, and bias of the qth layer, respectively.

Through a series of hidden layers, the output of the final hidden layer is fed into the output units. An activation function is used to acquire the output of the DNN component.

$$y_{DNN} = \sigma(\mathcal{W}^{|Q|+1} z^{Q} + b^{|Q|+1}) \tag{10}$$

where $|Q|$ is the number of hidden layers.

By combing the output of FM component and DNN component, all parameters (e.g., including w_i, V_i, and the network parameters $\mathcal{W}^{(q)}, b^{(q)}$) can be trained jointly. The predicted rating score $\hat{y}$ is calculated as follows:

$$\hat{y} = h(y_{FM} + y_{DNN}) \tag{11}$$

where y_{FM} is the output of FM component, and y_{DNN} is the output of deep component, and h is a linear regression. Note that we modify the original DeepFM model (i.e., a classification task model) to fit our regression task (i.e., rating prediction) by leveraging the linear regression in the output units.

3 Empirical Study

In this section, we first describe the two mobile app datasets. Then, we introduce the evaluation metrics for performance comparison. Moreover, we compare our WMGRec model with other state-of-the-art models. Finally, we study impact of parameters on performance.

3.1 Dataset Description

We conduct experiments on two datasets. One dataset is crawled from a famous app market Apple App Store. This dataset includes 12688 users and 10556 apps with 335744 app ratings ranging from 1 to 5. The other dataset is crawled from Google Play, provided by [8]. This dataset includes 14379 users and 25515 apps with 330212 app ratings ranging from 1 to 5. The density ($\frac{\#Ratings}{\#Users \times \#Apps}$) of Apple App Store and Google Play dataset are 0.2507% and 0.09%, respectively. Both of the datasets also collect the reviews of users on apps. The detailed statistics of two datasets are given in Table 1.

Table 1. Statistics of the datasets

Dataset	Relation of (A-B)	Number of A	Number of B	Number of (A-B)
Apple App Store	User-app	12688	10556	335744
	User-review	12688	335744	335744
	Review-app	335744	10556	335744
	Review-topic	335744	4	973499
Google Play	User-app	14379	25515	330212
	User-review	14379	330212	330212
	Review-app	330212	25515	330212
	Review-topic	330212	4	288927

3.2 Evaluation Metrics

In the following experiments, we use Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE) to evaluate the errors between our predicted results and the reality outcomes [15]. MAE is given by:

$$MAE = \frac{\sum_{(i,j)\in \Gamma_{test}} |\hat{r}_{ij} - r_{ij}|}{|\Gamma_{test}|}, \tag{12}$$

and RMSE is denoted as:

$$RMSE = \sqrt{\frac{\sum_{(i,j)\in \Gamma_{test}} (\hat{r}_{ij} - r_{ij})^2}{|\Gamma_{test}|}}. \tag{13}$$

where, Γ_{test} represents the set of all user-app pairs (i, j) in the testing set, $\hat{r}_{ij}$ represents the predicted rating score of user i on app j, while r_{ij} is the observed rating score of user i on app j in the testing set.

3.3 Performance Comparison

To validate the effectiveness of the proposed approach, we compare WMGRec model with the following models.

- NMF [7]. This approach employs matrix factorization to user-app rating matrix with a constraint that the factorized matrix is positive.
- FM [9]. This approach is the traditional factorization machine. It concatenates user id and app id as sparsity features, and learns the interactions between users and apps to complete the user-app rating matrix.
- SemRec [12]. This approach applies weighted meta-path based similarity, and designs different kinds of strategies to predict the rating scores of users on apps.
- FMG [19]. This approach utilizes unweighted meta-graph based similarity and standard matrix factorization to obtain user and app latent features. And then, it uses factorization machine to predict the rating scores of users on apps.

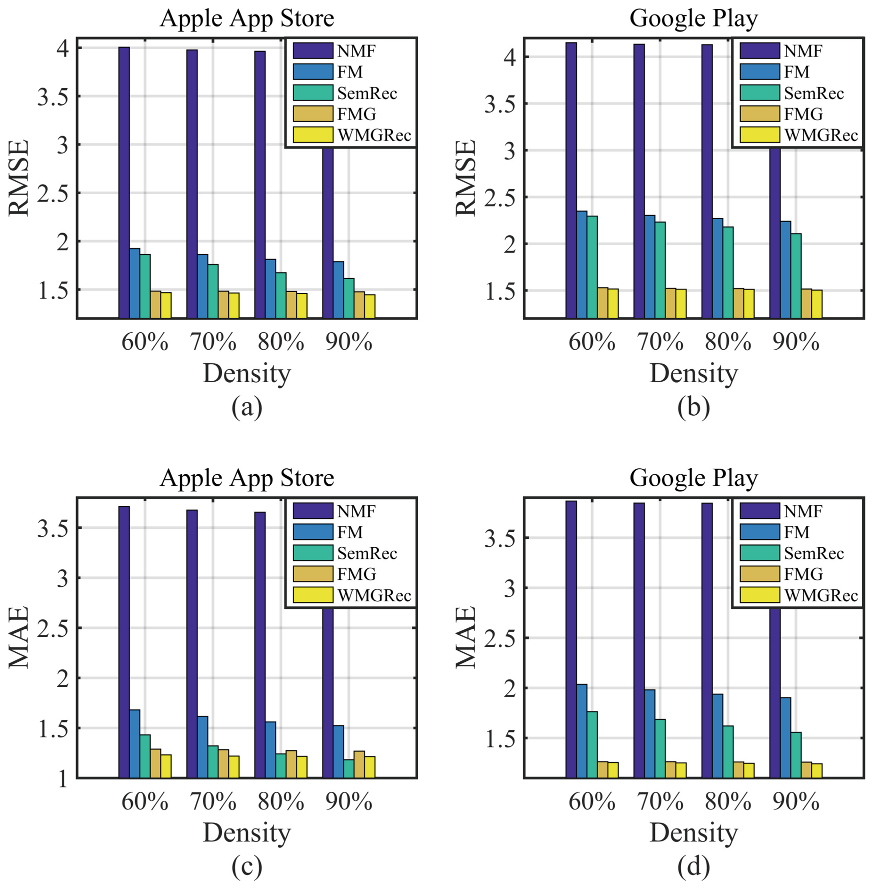


Fig. 5. Experimental results of all comparison methods

We employ the meta-graphs in Fig. 2(b) whose length are not longer than 4, since the longer meta paths are not meaningful and they fail to produce good similarity measure [14]. We repeat the experiments five times and use the average RMSE and MAE of five rounds as the final result. The topics are extracted from review texts by applying LDA (Latent Dirichlet Allocation) [1] model. The number of hidden layers is 2. The dropout in WMGRec is 0.5 and learning rate is set as 0.001. The latent factors and embedding size of all methods are fixed as 10 for fair. The other parameters in those comparison methods are set with the best performances.

Experiment results of all comparison methods are shown in Fig. 5. It can be found that the denser training data brings better performance. The reason lies in that with more observations in the training set, more information of the whole matrix can be obtained. This leads to more accurate predicted rating scores of users on apps. From Fig. 5, we can see that our proposed method beats the other state-of-the-art methods under all training data density (i.e., 60%, 70%, 80% and 90%) and evaluation metrics (i.e., RMSE and MAE). Specifically, the performance of FM is better than that of NMF due to capturing interactions between users and apps. HIN based approaches (e.g., SemRec, FMG, WMGRec) are better than FM due to taking advantage of rich structure and abundant semantics. The comparison between SemRec and FMG validates the effectiveness of meta-graph. By comparing WMGRec with FMG, it indicates considering the rating scores on links and applying DeepFM to learn high-order interactions are indeed helpful to improve the recommendation accuracy. Concretely, compared with FMG, the improvement of WMGRec on Apple App Store dataset is 1.15% to 2.05% for RMSE and 4.25% to 4.99% for MAE. In addition, the improvement of WMGRec on Google Play datset is 0.46% to 0.88% for RMSE and 0.62% to 1.31% for MAE. The enhancement of Google Play datset is smaller than that of Apple App Store dataset, because the density of the former is considerably sparse. Note that SemRec achieves the best MAE when the training data density is 90%. That's because it's not always possible to get the best performance of both evaluation metrics simultaneously, when optimizing the loss function. Moreover, we mainly optimize the RMSE evaluation metric in this paper.

3.4 Study on Parameter Impacts

In this section, we discuss the impact of some major parameters on Apple App Store dataset. In the following experiments, we fix 80% of the whole data as the training set and the remaining 20% as the testing set.

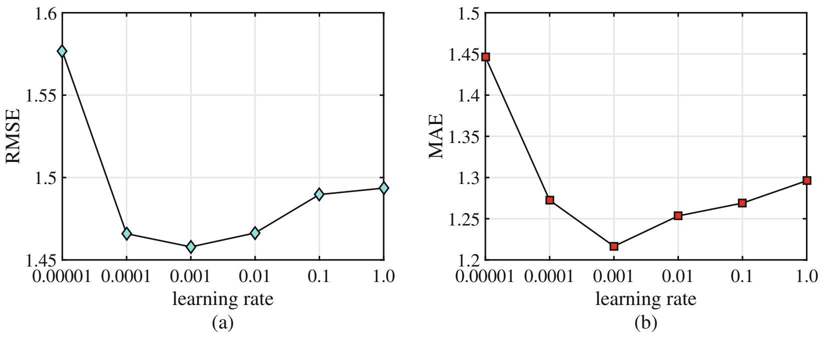


Fig. 6. Impact of learning rate

Impact of Learning Rate. Learning rate is an important factor in machine learning and deep learning. How to adjust the learning rate is one of the key steps to train a model. When solving the minimum problem with gradient descent, the gradient can be neither too large nor too small. If the learning rate is too large, it will hinder the convergence. That is, the loss shocks near the extreme point. If the learning rate is too small, it will lead to the inability to quickly find a good downward direction. Namely, when the number of iterations increases, the loss keeps unchanged. In this experiment, we set learning rate from 0.00001 to 1.0. As shown in Fig. 6, the RMSE and MAE firstly present a downward trend, and then an upward trend. The best performance achieves when the learning rate is set as 0.001. The result indicates that a relatively larger learning rate is helpful to improve the recommendation accuracy.

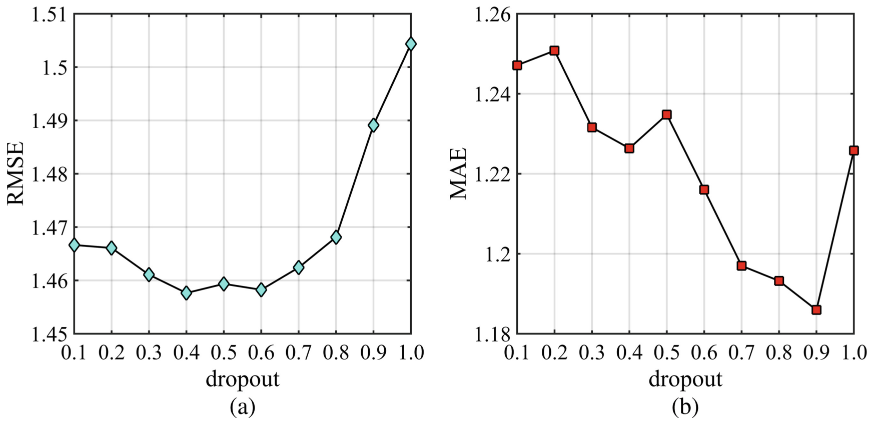


Fig. 7. Impact of dropout

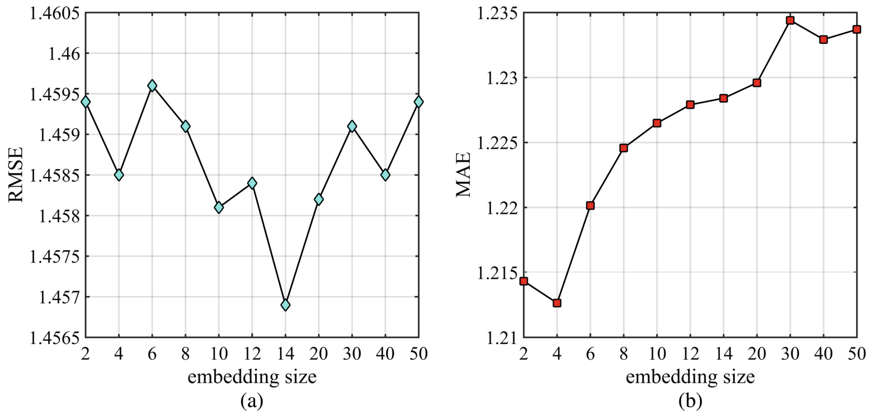


Fig. 8. Impact of embedding size

Impact of Dropout. Dropout [13] refers to a probability that each neural network unit may be discarded from the network in the process of DNN training.

Dropout is a regularization technique for reducing overfitting in neural networks. We set the dropout from 0.1 to 1.0 with a step value of 0.1. As shown in Fig. 7, the RMSE and MAE increase after the first decline. Our model can reach the best performance when the dropout is set as 0.4 for RMSE and 0.9 for MAE. The observation demonstrates that adding reasonable randomness to the model can enhance the effectiveness of the model.

Impact of Embedding Size. Embedding size d determines how many embedding features are extracted from feature vectors. Namely, the dimensionality of embedding vector. To study the impact of embedding size, we vary it from 2 to 50. It can be observed from Fig. 8, when increasing the embedding size from 2 to 50 the RMSE and MAE value decline firstly, and then grow up. The lowest RMSE can be obtained when the embedding size is 14. For MAE, the best performance can be acquired when the embedding size is 4. The result shows that a relatively smaller embedding size is better to achieve a good model. It is reasonable because larger embedding size means higher computational cost.

4 Related Work

To the best of our knowledge, this paper is the first work to exploit weighted meta-graph and HIN for mobile app recommendation. The work of this paper is mainly related to two aspects: mobile app recommendation and HIN for recommendation. Several studies referred to these two areas will be introduced in the following.

Mobile App Recommendation. Most existing investigations usually consider the context information, user's interest, privacy, security and other characteristics for mobile app recommendation. Liang et al. [8] proposed a broad learning approach for context-aware app recommendation with tensor analysis. Yin et al. [18] proposed a mobile sparse additive generative model to recommend apps by considering both user interests and category-aware user privacy preferences. Huang et al. [5] presented a skewness-based framework for mobile app permission recommendation and risk evaluation. Zhu et al. [21] proposed a novel location-based probabilistic factor analysis mechanism to help people get an appropriate mobile app. Cao et al. [2] proposed a novel version-sensitive mobile app recommendation framework by jointly exploring the version progression and dual-heterogeneous data.

HIN for Recommendation. HIN is widely applied to movie recommendation, research collaborator recommendation, product recommendation and social recommendation. Shi et al. [12] proposed a weighted HIN and weighted meta-path based personalized recommendation method to predict the rating scores of users on items. Shi et al. [11] proposed a matrix factorization based dual regularization framework to integrate different types of information. Yang et al. [17] generated high quality expert's profiles and proposed an approach based on the multiple heterogeneous network features. Zhao et al. [19] proposed a group lasso regularized FM to automatically learn from the observed ratings to effectively select useful meta-graph based features. Zheng et al. [20] proposed a new dual similarity regularization to impose the constraint on users and items with high and low similarities simultaneously. However, to the best of our knowledge, most of the existing studies do not explored weighted meta-graph and HIN in the scenario of mobile app recommendation. Weighted meta-graph takes into account the rating scores to capture the subtle semantics. HIN contains rich structure and semantic information, which can satisfy different kinds of requirements of users and generate better recommendation results.

5 Conclusion

This paper presents a weighted meta-graph based approach for mobile app recommendation. Firstly, we utilize weighted meta-graph, which considers the rating scores on links and captures more complex semantics, to measure the semantic similarity between users and apps. Then, we employ non-negative matrix factorization to obtain user latent features and app latent features. Finally, we exploit DeepFM model to predict the rating score of users on apps by leveraging the concatenated user and app latent features. Furthermore, we conduct a series of comprehensive experiments on two real-world datasets. First of all, we compare WMGRec model with other baseline approaches under different training data density, which indicates that our WMGRec model generates better recommendations and improves the recommendation accuracy. And then, we study how parameters (i.e., learning rate, dropout and embedding size) impact the recommendation results.

In the future, we attempt to collect more attribute information (e.g., permission and category) of apps and social relation information (e.g., friends relation and trust relation) of users to enrich features and semantics in the network.

Acknowledgments. The paper was supported by the National Key Research and Development Program (2017YFB0202200), the National Natural Science Foundation of China (61702568, U1711267), the Program for Guangdong Introducing Innovative and Enterpreneurial Teams (No.2017ZT07X355) and the Fundamental Research Funds for the Central Universities under Grant (17lgpy117).

References

1. Blei, D.M., Ng, A.Y., Jordan, M.I.: Latent Dirichlet allocation. J. Mach. Learn. Res. **3**(Jan), 993–1022 (2003)
2. Cao, D., et al.: Version-sensitive mobile app recommendation. Inf. Sci. **381**, 161–175 (2017)
3. Fang, Y., Lin, W., Zheng, V.W., Wu, M., Chang, K.C.C., Li, X.L.: Semantic proximity search on graphs with metagraph-based learning. In: 2016 IEEE 32nd International Conference on Data Engineering (ICDE), pp. 277–288. IEEE (2016)

4. Guo, H., Tang, R., Ye, Y., Li, Z., He, X.: DeepFM: a factorization-machine based neural network for CTR prediction. In: International Joint Conference on Artificial Intelligence (IJCAI), pp. 1725–1731 (2017)
5. Huang, K., Han, J., Chen, S., Feng, Z.: A skewness-based framework for mobile app permission recommendation and risk evaluation. In: Sheng, Q.Z., Stroulia, E., Tata, S., Bhiri, S. (eds.) ICSOC 2016. LNCS, vol. 9936, pp. 252–266. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46295-0_16
6. Huang, Z., Zheng, Y., Cheng, R., Sun, Y., Mamoulis, N., Li, X.: Meta structure: computing relevance in large heterogeneous information networks. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD), pp. 1595–1604. ACM (2016)
7. Lee, D.D., Seung, H.S.: Algorithms for non-negative matrix factorization. In: Conference on Neural Information Processing Systems (NIPS), pp. 556–562 (2001)
8. Liang, T., He, L., Lu, C.T., Chen, L., Yu, P.S., Wu, J.: A broad learning approach for context-aware mobile application recommendation. In: International Conference on Data Mining (ICDM), pp. 955–960. IEEE (2017)
9. Rendle, S.: Factorization machines. In: International Conference on Data Mining (ICDM), pp. 995–1000. IEEE (2010)
10. Shi, C., Li, Y., Zhang, J., Sun, Y., Philip, S.Y.: A survey of heterogeneous information network analysis. IEEE Trans. Knowl. Data Eng. **29**(1), 17–37 (2017)
11. Shi, C., Liu, J., Zhuang, F., Philip, S.Y., Wu, B.: Integrating heterogeneous information via flexible regularization framework for recommendation. Knowl. Inf. Syst. **49**(3), 835–859 (2016)
12. Shi, C., Zhang, Z., Luo, P., Yu, P.S., Yue, Y., Wu, B.: Semantic path based personalized recommendation on weighted heterogeneous information networks. In: Proceedings of the 24th ACM International on Conference on Information and Knowledge Management (CIKM), pp. 453–462. ACM (2015)
13. Srivastava, N., Hinton, G.E., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Dropout: a simple way to prevent neural networks from overfitting. J. Mach. Learn. Res. **15**(1), 1929–1958 (2014)
14. Sun, Y., Han, J., Yan, X., Yu, P.S., Wu, T.: PathSim: meta path-based top-k similarity search in heterogeneous information networks. Proc. VLDB Endow. **4**(11), 992–1003 (2011)
15. Willmott, C.J., Matsuura, K.: Advantages of the mean absolute error (MAE) over the root mean square error (RMSE) in assessing average model performance. Clim. Res. **30**(1), 79–82 (2005)
16. Wu, J., Chen, L., Xie, Y., Zheng, Z.: Titan: a system for effective web service discovery. In: Proceedings of the 21st International Conference on World Wide Web, pp. 441–444 (2012)
17. Yang, C., Sun, J., Ma, J., Zhang, S., Wang, G., Hua, Z.: Scientific collaborator recommendation in heterogeneous bibliographic networks. In: Hawaii International Conference on System Sciences (HICSS), pp. 552–561. IEEE (2015)
18. Yin, H., Chen, L., Wang, W., Du, X., Nguyen, Q.V.H., Zhou, X.: Mobi-SAGE: a sparse additive generative model for mobile app recommendation. In: International Conference on Data Engineering (ICDE), pp. 75–78. IEEE (2017)
19. Zhao, H., Yao, Q., Li, J., Song, Y., Lee, D.L.: Meta-graph based recommendation fusion over heterogeneous information networks. In: Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD), pp. 635–644. ACM (2017)

20. Zheng, J., Liu, J., Shi, C., Zhuang, F., Li, J., Wu, B.: Dual similarity regularization for recommendation. In: Bailey, J., Khan, L., Washio, T., Dobbie, G., Huang, J.Z., Wang, R. (eds.) PAKDD 2016. LNCS (LNAI), vol. 9652, pp. 542–554. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-31750-2_43
21. Zhu, K., Zhang, L., Pattavina, A.: Learning geographical and mobility factors for mobile application recommendation. IEEE Intell. Syst. **32**(3), 36–44 (2017)