Knowledge-Based Systems xxx (xxxx) xxx

Contents lists available at ScienceDirect

Knowledge-Based Systems

journal homepage: www.elsevier.com/locate/knosys

Multi-view factorization machines for mobile app recommendation based on hierarchical attention☆

Tingting Liang [a], Lei Zheng [b], Liang Chen [c], Yao Wan [a], Philip S. Yu [b,d], Jian Wu [a,*]

[a] College of Computer Science & Technology, Zhejiang University, Hangzhou 310027, China
[b] Computer Science Department, University of Illinois at Chicago, Chicago, IL 60607, USA
[c] School of Data and Computer Science, Sun Yat-Sen University, Guangzhou 510275, China
[d] Institute for Data Science, Tsinghua University, Beijing 100084, China

ARTICLE INFO

Article history:
Received 13 September 2018
Received in revised form 26 June 2019
Accepted 26 June 2019
Available online xxxx

Keywords:
Mobile application recommendation
Factorization machines
Attention network
Multi-view feature

ABSTRACT

Mobile app recommendation has been an effective solution to overcoming the information overload in mobile app markets. Recent studies have demonstrated the power of neural network in recommendation tasks which is however rarely exploited for mobile apps. As one of the development of neural network, attention-based models have shown promising results for recommendation because of its capability of filtering out uninformative features from raw inputs. In this paper, to effectively predict users' preferences for apps, we propose a hierarchical neural network model called MV-AFM for app recommendation which models the interactions of features from different views (view interactions for short) through the attention mechanism. Specifically, the novelty of MV-AFM is the introduction of view segmentation for feature interactions and the construction of two level attention networks: the feature-level attention, starting from the feature embeddings within each view, which intends to select the representative features for the view, and the view-level attention, which learns the importance of interactions between any two views. Extensive experiments on two real-world mobile app datasets demonstrate the effectiveness of MV-AFM.

1. Introduction

With the development of smart phone devices, the app stores are experiencing tremendous growth in the numbers of apps and users. Taking the two most representative app stores, Google Play and Apple's App Store, as examples, the number of available apps in the former reached 2.7 million in Feb. 2017.[1] and the latter has accumulated about 2.2 million downloadable apps until Jan. 2017.[2] The vast amount of apps leads to information load, making it difficult for users to find required apps.

Recommender system is a well-known solution for alleviating the information overload problem and assisting users in locating the target items. Recently, many recommendation algorithms for mobile apps have been proposed. Most of these methods can be divided into two types, collaborative filtering (CF) based approach [1,2] and feature-based approach [3–6]. CF-based approach merely depends on the historical interactions between users and apps without the consideration of additional auxiliary information. Feature-based approach exploits the app side information (*i.e.,* privacy information, description) as features to facilitate the prediction of users' preferences. However, all these methods do not consider the complex interactions between users and app features.

Factorization Machines (FM) [7] is one of the most effective feature-based recommendation models that can incorporate any side features to improve the performance of rating prediction task. FM model makes it possible to integrate any auxiliary information that can be encoded as a real-valued feature vector and model feature interactions. Table 1 shows several examples of app rating records. Each records involves a user, an app, and app features, which can be seen as heterogeneous data obtained in multiple views. Each view is represented by a set of features. The permission view contains a set of features like *precise location*, *take pictures & videos*, and the text description is a view including a set of word features. Different feature views can provide complementary information. For example, according to the category view, Instagram can be recognized as a tool for social interaction. Feature *take pictures & videos* in the permission view and the words in the text view can provide further information indicating its function of photo sharing.

☆ No author associated with this paper has disclosed any potential or pertinent conflicts which may be perceived to have impending conflict with this work. For full disclosure statements refer to https://doi.org/10.1016/j.knosys.2019.06.029.

* Corresponding author.
E-mail address: wujian2000@zju.edu.cn (J. Wu).

1 http://en.wikipedia.org/wiki/Google_Play

2 http://en.wikipedia.org/wiki/App_Store_(iOS)

https://doi.org/10.1016/j.knosys.2019.06.029
0950-7051/

Table 1
Feature examples of app rating records.

user_id	app_id	Category	Description text
SenTra IV	com.photolab.photoeditor	Photography	photo editor, stickers, skin, …
C Barauskas	gov.caltrans.quickmap	Maps&Navigation	location, real-time, traffic, …
Nizami Safi	com.instagram.android	Social	share, friends, post photos, …
Lammie New	com.topfreegame.sudoku	Puzzle	sudoku, puzzle game, brain, …

user_id	app_id	Permission				Rating
		Take pictures & videos	Precise location	Receive data from Internet	...	
SenTra IV	com.photolab.photoeditor	1	0	0	...	3
C Barauskas	gov.caltrans.quickmap	0	1	0	...	4
Nizami Safi	com.instagram.android	1	1	0	...	5
Lammie New	com.topfreegame.sudoku	0	0	1	...	5

This work aims to make accurate app rating prediction by capturing the interaction information among multi-view features based on FM model. Therefore, we propose to model the feature interactions considering view segmentation with deep neural network which has the potential to learn sophisticated feature interactions. Several FM-based neural network models have been proposed in recent years, e.g., DeepFM [8] and AFM [9], but most of them model pairwise interactions between all features without considering view segmentation. View segmentation here means classifying features into different views by feature types. With view segmentation, we can model the interactions of features from different views and neglect the feature interactions within the same view, which makes it possible to avoid the redundant correlations between features within the same view. For example, the interactions of features in permission view are unworthy as they cannot further provide insight information. The feature interactions within description view are easy to produce redundant information since the text features are arbitrary and sophisticated. But when model the interactions between features from permission and description view, we cannot only obtain the users' preference on app function and privacy information simultaneously but also judge the rationality of permission request related to the app function.

As Table 1 shows, a feature view can be represented by a set of features (e.g., permission and description view). However, there exists a case that some features in a view are insignificant. Taking the permission view of apps as an example, some permissions have minor influence on the view representation as they are commonly found in most apps, such as *full network access*, *view Wi-Fi connections*. Moreover, the features in a view contain complex information while different users may focus on different features in the view of the same app. To accurately represent a specific view, we should assign different weights on the set of features. It is also worth noting that a multi-view FM model treats all the view interactions equally. However in reality, it is common that not all feature views make the same contribution to the prediction of users' preferences for apps. For example, the determining factors for users' app downloads can be quite different from user to user. Some users may think more of the content, while others pay more attention to the privacy information. With these in mind, we consider to construct a hierarchical network that consists feature-level and view level attention. The key idea of attention is to assign attentive weights for a set of features: higher (lower) weights indicate that the corresponding features are more (less) informative. Feature-level attention is able to assign appropriate attentions for features to generate accurate view representations and view-level attention can provide different attentions to discriminate the importance of different view interactions.

In this paper, we propose a Multi-View Attentional Factorization Machines (MV-AFM) for app recommendation. MV-AFM is a neural network structured with hierarchical attention sub-networks. Specifically, a feature-level attention is exploited on the embedding vectors of features in each view, which aims at selecting the representative features for the view. Based on weighted embedding vectors, MV-AFM performs a view-level attention to differentiate the importance of different view interactions. To our best knowledge, this is the first work considering the differentiation of views with attention mechanism.

The main contributions of this paper are summarized as:

- We propose a novel method called MV-AFM to facilitate app recommendation by modeling feature interactions with view segmentation under neural network framework and constructing hierarchical attention sub-networks.
- Through the two attention sub-networks, i.e., feature-level and view-level, MV-AFM simultaneously learns the importance of features in the same view and the significance of different view interactions.
- Through extensive experiments conducted on two real-world datasets, we show that MV-AFM consistently outperforms several state-of-the-art prediction models.

The remainder of this paper is structured as follows. Section 2 presents the preliminaries about FM model and multi-view prediction. The proposed MV-AFM model is described in Section 3. Section 4 reports the experimental results. Section 5 presents the related work and we draw the conclusion in Section 6.

2. Preliminaries

In this section, we first review the factorization machines and briefly introduce the problem of multi-view prediction.

2.1. Factorization machines

As one of the state-of-the-art predictors, factorization machines (FM) models all possible interactions between values in the feature vector using factorized interactions rather than full parametrized ones [7]. Given a real-valued feature vector $\mathbf{x} \in \mathbb{R}^n$ where n denotes the number of features, the 2-way FM model is defined as follows:

$$\hat{y}(\mathbf{x}) = w_0 + \sum_{i=1}^{n} w_i x_i + \sum_{i=1}^{n} \sum_{j=i+1}^{n} \langle \mathbf{v}_i, \mathbf{v}_j \rangle x_i x_j, \tag{1}$$

where $w_0 \in \mathbb{R}$ is the global bias, w_i denotes the ith element of $\mathbf{w} \in \mathbb{R}^n$ and models the weight of the ith feature. The interaction between the ith and jth features is modeled by $\langle \mathbf{v}_i, \mathbf{v}_j \rangle = \sum_{f=1}^{k} v_{i,f} \cdot v_{j,f}$, where $\mathbf{v}_i \in \mathbb{R}^k$ denotes the latent vector for the ith feature and k denotes the dimensionality of latent vector.

2.2. Multi-view prediction

The most general prediction task is to learn a function $y : \mathbb{R}^n \rightarrow T$ from a real-valued feature vector $\mathbf{x} \in \mathbb{R}^n$ to a target

Please cite this article as: T. Liang, L. Zheng, L. Chen et al., Multi-view factorization machines for mobile app recommendation based on hierarchical attention, Knowledge-Based Systems (2019), https://doi.org/10.1016/j.knosys.2019.06.029.

domain T (e.g. $T = \mathbb{R}$ for regression). In multi-view settings, it is assumed that there is a training set with N labeled instances represented from m views: $\mathcal{D} = \{(\mathbf{x}_i, y_i)|i = 1, \ldots, N\}$, where $\mathbf{x}_i^{\mathrm{T}} = \left(\mathbf{x}_i^{(1)^{\mathrm{T}}}, \ldots, \mathbf{x}_i^{(m)^{\mathrm{T}}}\right)$. Here $\mathbf{x}_i^{(p)} \in \mathbb{R}^{I_p}$ is the pth view in the feature vector, I_p is the dimensionality of the pth view and $\sum_{p=1}^{m} I_p = n$.

It can be found that the pairwise interactions between all features in FM model does not consider the view segmentation. In the multi-view setting, the neglect of view segmentation would lead redundant correlations between features within the same view [10]. Therefore, in this work, we consider the multi-view variation of FM model which is defined as follows:

$$\begin{aligned}\hat{y}(\mathbf{x}) = w_0 + \sum_{p=1}^{m}\sum_{i_p=1}^{I_p} w_{i_p}^{(p)} x_{i_p}^{(p)} + \sum_{i_1=1}^{I_1}\sum_{i_2=1}^{I_2}\left\langle \mathbf{v}_{i_1}^{(1)}, \mathbf{v}_{i_2}^{(2)}\right\rangle x_{i_1}^{(1)} x_{i_2}^{(2)} \\ + \cdots + \sum_{i_{m-1}=1}^{I_{m-1}}\sum_{i_m=1}^{I_m}\left\langle \mathbf{v}_{i_{m-1}}^{(m-1)}, \mathbf{v}_{i_m}^{(m)}\right\rangle x_{i_{m-1}}^{(m-1)} x_{i_m}^{(m)},\end{aligned} \tag{2}$$

where $\mathbf{v}_{i_p}^{(p)}$ denotes the latent vector for the i_p-th feature in the pth view.

3. Multi-View Attentional Factorization Machines

3.1. General framework

MV-AFM is a hierarchical neural network that models multi-view feature interactions with feature-level and view-level attentions. Fig. 1 illustrates the architecture of the proposed MV-AFM model. We omit the linear regression part in the figure for clarity. MV-AFM starts from a sparse vector of input features with view segmentation. For each view, the embedding layer embeds each non-zero feature into a dense vector. It is worth noting that the embedding layer can model different types of features such as categorical and textual features, which means the proposed model is able to process these features together. The feature-level attention assign different attention values to the feature embedding vectors in each view to obtain a set of weighted embeddings. The next layer models the interactions between the weighted embeddings of features from different views to get the pair-wise interactions between views. View-level attention is applied to generate the attention values for different pair-wise view interactions to obtain the weighted interactions which can be used for the final prediction.

As the main contributions of this work, the embedding layer, multi-view pair-wise interaction layer, feature-level and view-level attention mechanism will be detailedly introduced as follows.

Embedding Layer. Each input vector (*i.e.,* rating record) consists of features from multiple views (*e.g.,* user id, genre, permission, and description) which need to be embedded to low-dimensional, dense real-value representations before being fed into the main module. The proposed embedding layer is able to turn the indexes into dense vectors of fixed size to transform the features to embedding vectors. The embedding layer can be simply regarded as a lookup table that read non-zero a feature as follows:

$$LT(i) = \mathbf{W}_i,$$

where $\mathbf{W} \in \mathbb{R}^{k\times n}$ is a matrix of parameters to be learned, $\mathbf{W}_i \in \mathbb{R}^k$ is the ith column of $\mathbf{W}$, and i is the index of the non-zero feature. Formally, let $\mathbf{v}_{i_p}^{(p)} \in \mathbb{R}^k$ be the embedding vector the i_p-th feature in the pth view. It is worth pointing out that (1) while the length of input vectors in different views can be different, the output embeddings are of the same size (k); (2) the embedding vector $\mathbf{v}_{i_p}^{(p)}$ plays the same role as the latent feature vector in FM and can be used to compress the input vector.

Pair-Wise Interactions between Views. Inspired by the pair-wise interaction layer proposed in [9], we propose a multi-view pair-wise interaction layer which models interactions between different views. It expands m views to $m(m-1)/2$ interacted views, where each interacted view is generated by the summation of the element-wise products of embedding vectors from two distinct views. Suppose the index set of non-zero features in the pth view feature vector $\mathbf{x}^{(p)}$ is $\mathcal{X}^{(p)}$ and the output of the embedding layer for the pth view is $\mathcal{E}^{(p)} = \{\mathbf{v}_{i_p}^{(p)} x_{i_p}^{(p)}\}_{i_p\in\mathcal{X}^{(p)}}$, then we can form the output of the multi-view pair-wise interaction layer as a set of vectors:

$$f_{VI}(\mathcal{E}) = \Big\{\sum_{i_p=1}^{I_p}\sum_{i_q=1}^{I_q}(\mathbf{v}_{i_p}^{(p)} * \mathbf{v}_{i_q}^{(q)}) x_{i_p}^{(p)} x_{i_q}^{(q)}\Big\}_{(p,q)\in\mathcal{R}_v},$$

where $*$ denotes the Hadamard (element-wise) product, $\mathcal{E} = \left\{(\mathcal{E}^{(p)}, \mathcal{E}^{(q)})|(p,q)\in\mathcal{R}_v\right\}$ denotes the set of input pairs, and $\mathcal{R}_v = \{(p,q)|p,q = 1,\ldots,m, q > p\}$ for short. Based on the newly proposed interaction layer, the multi-view variation of FM model can be represented under the neural network architecture. The final prediction result can be estimated by compressing f_{VI} with a sum pooling and using a fully connected layer as follows:

$$\hat{y} = \mathbf{p}^{\mathrm{T}} \sum_{(p,q)\in\mathcal{R}_v}\Big(\sum_{i_p=1}^{I_p}\sum_{i_q=1}^{I_q}(\mathbf{v}_{i_p}^{(p)} * \mathbf{v}_{i_q}^{(q)}) x_{i_p}^{(p)} x_{i_q}^{(q)}\Big) + b, \tag{3}$$

where $\mathbf{p} \in \mathbb{R}^k$ and $b \in \mathbb{R}$ respectively denote the weights and bias for the final prediction layer. Note that the multi-view FM model can be exactly recovered by fixing $\mathbf{p}$ to $\mathbf{1}$ and b to 0.

Feature-Level Attention. The multi-view pair-wise interaction layer introduced above models the features in a specific view with the same weights. However, not all features contribute equally to the representation of the view. Those irrelevant features can be considered as noises which would bring poor performance. The goal of feature-level attention is to assign features in the same view with attentive weights to represent the view. With the weighted feature vectors, the output of multi-view pair-wise interaction layer can be improved to:

$$f'_{VI}(\mathcal{E}) = \Big\{\sum_{i_p=1}^{I_p}\sum_{i_q=1}^{I_q}(\beta_{i_p,p}\mathbf{v}_{i_p}^{(p)} * \beta_{i_q,q}\mathbf{v}_{i_q}^{(q)}) x_{i_p}^{(p)} x_{i_q}^{(q)}\Big\}_{(p,q)\in\mathcal{R}_v}, \tag{4}$$

where $\beta_{i_p,p}$ can be interpreted as the importance of feature i_p in view p. There exist several effective ways to obtain the attention score and we introduce the multi-layer perceptron (MLP) to compute it as:

$$\begin{aligned}\beta'_{i_p,p} &= \mathbf{h}_f^{\mathrm{T}}\mathrm{relu}\left(\mathbf{W}_f\mathbf{v}_{i_p}^{(p)} + \mathbf{b}_f\right) \\ \beta_{i_p,p} &= \frac{\exp(\beta'_{i_p,p})}{\sum_{i_p=1}^{I_p}\exp(\beta'_{i_p,p})},\end{aligned} \tag{5}$$

where $\mathbf{W}_f \in \mathbb{R}^{d\times k}$, $\mathbf{b}_f \in \mathbb{R}^d$, and $\mathbf{h}_f \in \mathbb{R}^d$ are parameters of the feature-level attention network. And d denotes the hidden layer size of the feature-level attention network. The attention scores are normalized through the softmax function. Note that for different feature views, we can choose different methods to get the attention scores.

View-Level Attention. Assigning a uniform weight of 1 for all the view interactions in multi-view FM model may limit its generalization performance since different users may focus on features of different views. To address this problem, based on the attentional features in each view, we propose to implement

Please cite this article as: T. Liang, L. Zheng, L. Chen et al., Multi-view factorization machines for mobile app recommendation based on hierarchical attention, Knowledge-Based Systems (2019), https://doi.org/10.1016/j.knosys.2019.06.029.

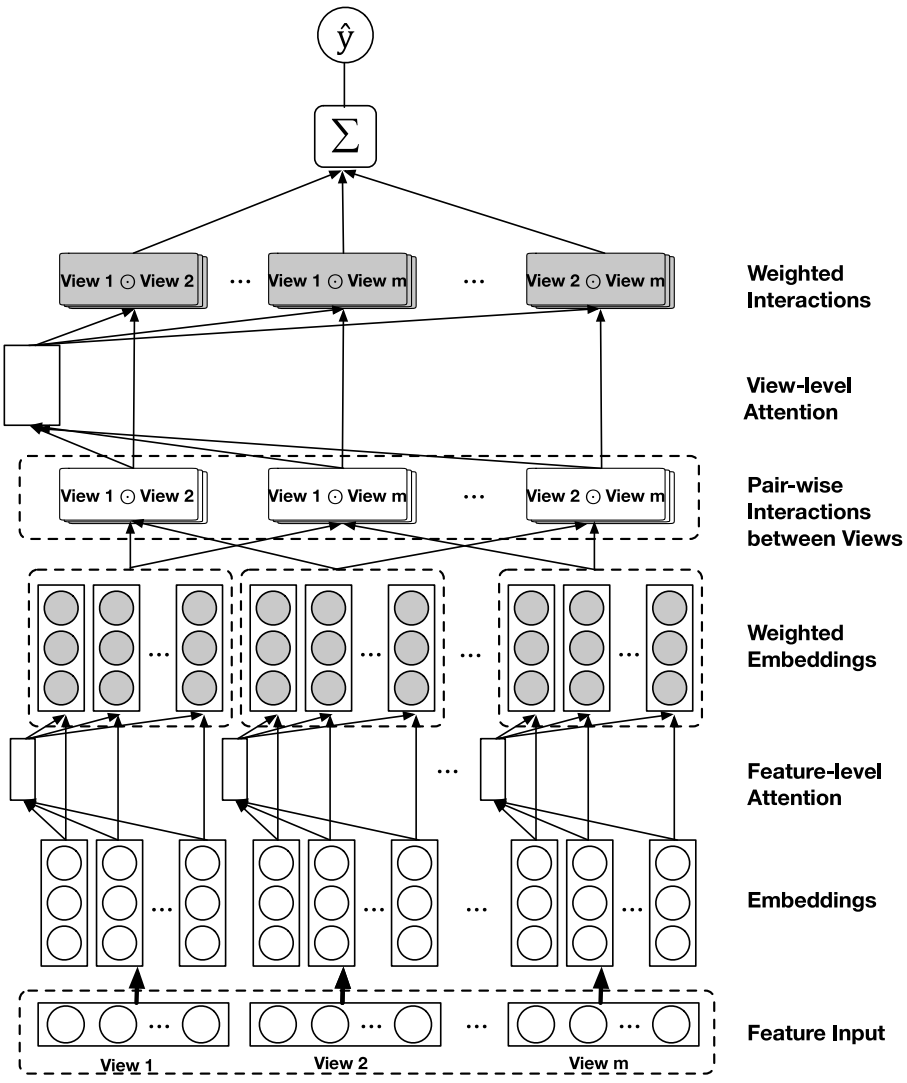


Fig. 1. The neural network architecture of the proposed MV-AFM model.

attention mechanism on view interactions (i.e., view-level attention) by performing a weighted summation on the interacted views:

$$f_{v\_att}\left(f'_{VI}(\mathcal{E})\right) = \sum_{(p,q)\in\mathcal{R}_v} \alpha_{p,q}\Big(\sum_{i_p=1}^{I_p}\sum_{i_q=1}^{I_q}(\beta_{i_p,p}\mathbf{v}_{i_p}^{(p)} * \beta_{i_q,q}\mathbf{v}_{i_q}^{(q)})x_{i_p}^{(p)}x_{i_q}^{(q)}\Big), \quad (6)$$

where $\alpha_{p,q}$ is the attention score that can be interpreted as the contribution made by the interaction between view p and q. We use the structure like feature-level attention network to estimate the attention score and take the summation of the interacted vectors from two views as input:

$$\begin{aligned}\alpha'_{p,q} &= \mathbf{h}_v^{\mathrm{T}}\mathrm{relu}\Big(\mathbf{W}_v\sum_{i_p=1}^{I_p}\sum_{i_q=1}^{I_q}(\beta_{i_p,p}\mathbf{v}_{i_p}^{(p)} * \beta_{i_q,q}\mathbf{v}_{i_q}^{(q)})x_{i_p}^{(p)}x_{i_q}^{(q)} + \mathbf{b}_v\Big)\\ \alpha_{p,q} &= \frac{\exp(\alpha'_{p,q})}{\sum_{(p,q)\in\mathcal{R}_v}\exp(\alpha'_{p,q})},\end{aligned} \quad (7)$$

where $\mathbf{W}_v \in \mathbb{R}^{t\times k}$, $\mathbf{b}_v \in \mathbb{R}^t$, and $\mathbf{h}_v \in \mathbb{R}^t$ are parameters of the view-level attention network. Here t denotes the hidden layer size of the attention network. We get the final view-level weights by normalizing the attention scores $\alpha'_{p,q}$ through the softmax function.

3.2. Rating prediction for apps

The output of the sum pooling layer with feature-level and view-level attentions is a k dimensional vector, which compresses all view interactions in the embedding space with distinct attentive scores. Given a feature vector $\mathbf{x}$ with multiple views that contains user and mobile app information, the final rating could be predicted by:

$$\begin{aligned}\hat{y}(\mathbf{x})_{MV-AFM} &= w_0 + \sum_{p=1}^{m}\sum_{i_p=1}^{I_p} w_{i_p}^{(p)}x_{i_p}^{(p)} + \mathbf{p}^{\mathrm{T}}f_{v\_att}\left(f'_{VI}(\mathcal{E})\right)\\ &= w_0 + \sum_{p=1}^{m}\sum_{i_p=1}^{I_p} w_{i_p}^{(p)}x_{i_p}^{(p)}\\ &\quad + \mathbf{p}^{\mathrm{T}}\sum_{(p,q)\in\mathcal{R}_v}\alpha_{p,q}\Big(\sum_{i_p=1}^{I_p}\sum_{i_q=1}^{I_q}(\beta_{i_p,p}\mathbf{v}_{i_p}^{(p)} * \beta_{i_q,q}\mathbf{v}_{i_q}^{(q)})x_{i_p}^{(p)}x_{i_q}^{(q)}\Big),\end{aligned} \quad (8)$$

where $\alpha_{p,q}$ and $\beta_{i_p,p}$ are defined in Eqs. (7) and (5), respectively.

3.3. Learning

To estimate model parameters of MV-AFM, we use the squared loss as objective function:

$$L = \sum_{(\mathbf{x},y)\in\mathcal{D}} (\hat{y}(\mathbf{x})_{MV-AFM} - y)^2, \tag{9}$$

where $\mathcal{D}$ denotes the training set and y is the target value. The regularization terms are optional and omitted here as some techniques such as dropout can effectively avoid overfitting in neural network modeling.

To optimize the objective function, we employ stochastic gradient descent (SGD) to learn model parameters $\Theta = \{\{w_{i_p}^{(p)}, \mathbf{v}_{i_p}^{(p)}\}, \mathbf{W}_f, \mathbf{W}_v, \mathbf{b}_f, \mathbf{b}_v, \mathbf{h}_f, \mathbf{h}_v, \mathbf{p}\}$. Algorithm 1 shows detailed learning process of MV-AFM, which can be divided into three steps: (1) Line 6 to Line 10 is the process of computing feature-level attention values and weighted feature embeddings within each view; (2) Line 11 to Line 14 is the process of computing view-level attention values and final rating prediction; (3) the optimizer runs back propagation with respect to Eq. (9) (see [11]). Note that Line 17 are the gradients of the model parameters updated using chain rules. To leverage the vectorization and parallelism speedup of modern computing platforms, we adopt mini-batch SGD that randomly samples a batch of training instances and updates parameters based on the batch.

Time Complexity. We analyze the time complexity of four main parts in whole neural network. For each example, the time complexity of the embedding layer (line 7) is $\mathcal{O}(n^2k)$, and the pair-wise view interaction layer (line 10) requires $\mathcal{O}(n^2k)$. The feature-level (line 8) and view-level (line 12) attention layers respectively require $\mathcal{O}(n + mdk)$ and $\mathcal{O}(m^2 + tk)$. The back propagation has the same complexity as the forward evaluation. As the number of view m usually is much smaller than the number of feature n, we omit m^2. When we set $d = t$, the total time complexity for Algorithm 1 is $\mathcal{O}((n^2 + md)k)$.

Space Complexity. The space complexity of the neural network is related to the number of model parameters. The embedding layer requires $\mathcal{O}(nk)$ and the linear regression part requires $\mathcal{O}(n)$. The space complexity of feature-level and view-level attention layer are $\mathcal{O}(d^2)$ and $\mathcal{O}(t^2)$, respectively. When we set $d = t$, the total space complexity of the proposed model is $\mathcal{O}(nk + d^2)$. It can be found that the most parameters need to be learned are in the embedding layer.

Algorithm 1: Learning Multi-View Attentional Factorization Machines

Input: Training data $\mathcal{D}$, embedding dimension k, attention dimension d and t, and learning rate η
Output: Model parameters Θ

1 Initialize Θ with xavier [11].
2 **repeat**
3 *//sample a mini-batch of size* N_b
4 $\mathcal{D}_{batch} \leftarrow sample(\mathcal{D}, N_b)$
5 **for** $\mathbf{x}_i \in \mathcal{D}_{batch}$ **do**
6 **for** $p = 1 : m$ **do**
7 Get embedding vector $\{\mathbf{v}_{i_p}^{(p)}\}_{i_p\in\mathcal{X}_p}$
8 Compute $\{\beta_{i_p,p}\}_{i_p\in I_p}$ according to Eq. (5)
9 **end**
10 Compute $f'_{VI}(\mathcal{E})$ according to Eq. (4)
11 **for** *each view pair* (p, q) **do**
12 Compute $\alpha_{p,q}$ according to Eq. (7)
13 **end**
14 Compute $\hat{y}(\mathbf{x}_i)$ according to Eq. (8)
15 **end**
16 **for** $\theta \in \Theta$ **do**
17 Update $\theta \leftarrow \theta - \eta \cdot \frac{1}{N_b}\sum_{i=1}^{N_b} 2(\hat{y}(\mathbf{x}_i) - y)\frac{\partial\hat{y}(\mathbf{x}_i)}{\partial\theta}$
18 **end**
19 **until** *convergence*;

4. Experiments

4.1. Data and setup

To evaluate the performance of our MV-AFM model, we conduct extensive experiments on two datasets, Google Play and Apple's App Store.

- **Google Play:** We crawled app's meta data (e.g., name, category, permissions, description) and user review ratings from its description page in Google Play. After filtering users and apps with less than 5 rating records, we obtain 8,875 users and 6,756 apps with 84,075 rating records. Each rating record is represented in four views, i.e., user, category, permission, and description text. The user view consists of binary feature vectors for user ids which means there is only one non-zero feature in the user view for each rating record, the same for the category view. The permission view is represented by a vector of permission indexes, and the text view is represented by the first 100 words indexes.
- **Apple's App Store:** The dataset is offered by [12,13] and consists of the apps in the "Top Free 300" and "Top Paid 300" leaderboards from Feb. 2010 to Sep. 2012, and the related user ratings and review information. After filtering users and apps with less than 10 rating records, we obtain 8,834 users and 4,105 apps with 150,797 rating records. Each rating record in this dataset has three views, *i.e.,* users, category, and text. The user and category views are constructed using the same way as in the Google Play dataset. The text view is represented by the first 200 words indexes.

The embedding vectors of the features in all the views can be obtained by the embedding layer turning the feature indexes into dense vectors of a fixed size.

We randomly split the dataset into three parts: $K\%$ ($K = 60, 70, 80$) for training, 10% for validation, and 10% for testing. Each experiment is repeated 10 times, and the mean value of each metric is reported.

4.2. Evaluation metric

We use Mean Absolute Error (MAE) and Mean Square Error (MSE) [14] to evaluate the performance of the proposed model and other baselines. The definition of MAE is:

$$MAE = \frac{1}{N}\sum_{i,j}|R_{ij} - \hat{R}_{ij}| \tag{10}$$

where R_{ij} represents the rating user i gave to app j, and $\hat{R}_{ij}$ denotes the rating predicted by a method. Moreover, N is the number of tested ratings. The metric MSE is defined as:

$$MSE = \frac{1}{N}\sum_{i,j}(R_{ij} - \hat{R}_{ij})^2 \tag{11}$$

A smaller MAE or MSE indicates the better performance.

Please cite this article as: T. Liang, L. Zheng, L. Chen et al., Multi-view factorization machines for mobile app recommendation based on hierarchical attention, Knowledge-Based Systems (2019), https://doi.org/10.1016/j.knosys.2019.06.029.

4.3. Compared models

We compare following models to demonstrate the effectiveness of the proposed model.

- **FM.** It is the Factorization Machine [7] that explores pairwise interactions between all features without view segmentation. We implement it under neural network structure.
- **Wide&Deep.** This model is proposed by Google to model low-order and high-order feature interactions simultaneously [15].
- **AFM.** Attentional Factorization Machines [9] improves FM model by utilizing the attention mechanism to discriminate the importance of feature interactions.
- **DeepFM.** It is a new neural network model that models low-order feature interactions like FM and models high-order feature interactions like deep neural network [8].
- **MV-FM.** It is a variation of the proposed MV-AFM without considering the hierarchical attention network and only conducting the interactions between features from different views (i.e., view segmentation).
- **MV-AFM.** It is the proposed multi-view factorization machines with feature-level and view-level attention networks.

All the models except DeepFM are trained by the mini-batch Adagrad with learning rate 0.01. The learning rate for DeepFM is 0.001. The batch size is set to 256 for all models. The embedding layers in the neural network based models are the same and the embedding size is set to 20. The hidden layer size of all the attention networks is set to 20, same as the embedding size. We use the early stopping strategy based on the performance on validation set.

4.4. Performance comparison

In this subsection, we compare the performance of the proposed MV-AFM model and the baselines with respect to two metrics, i.e., MAE and MSE. Tables 2 and 3 show the overall performance all the prediction methods on Google Play and Apple's App Store, from which we have the following observations.

- It can be found that the proposed MV-AFM achieves the best performance among all the models on both datasets. For Google Play, MV-AFM obtains the improvements of 2.68% and 4.80% in terms of MAE and MSE for 70% training data in the maximum extent. For the other dataset, the improvements are 2.99% and 2.39% in terms of MAE and MSE for 70% training data. The progress is powerful proof of the effectiveness of the proposed MV-AFM.
- MV-AFM outperforms its variation MV-FM, which indicates the superiority of the feature-level and view-level attention mechanism. The two-level attention network effectively discriminate the representative features in each view and the significant interactions between views.
- The models considering multiple views (i.e., MV-FM, MV-AFM) outperform FM and AFM. It demonstrates that the view segmentation can effectively improve the performance by reducing the redundant feature interactions within the same view.
- The models incorporating attention mechanism (i.e., AFM, MV-AFM) outperform the basic FM model. It proves the attention-based networks is capable of filtering out the uninformative features.
- The Wide&Deep and DeepFM, especially the latter, make some improvement compared to FM because of the deep components in the models. DeepFM achieves a better performance as it has a shared input and embedding vector for each part. Besides, compared to Wide&Deep model, DeepFM does not require tedious featuring engineering.
- Comparing the two datasets, it can be found that the superiority of the proposed MV-AFM is more significant for Google Play dataset. It is probably because of the fewer categories which might not sufficiently discriminate the important features in the specific category. Moreover, the two categories in Apple App Store, *i.e.,* "Free" and "Paid", do not have their own unique characteristics and are not easy to differentiate from each other. Nonetheless, even with very limited feature information, the proposed MV-AFM still outperform the baseline models.

4.5. Feature view analysis

As one significant contribution of MV-AFM is the construction of interactions between different feature views, we explore the impact of feature views on the regression models. Considering the feature views of Apple's App Store are not enough, we only investigate the view impact on Google Play. Each rating record consists of four views, *i.e.,* user, app auxiliary information (category, permission, and description text). We only compare the performance of models utilizing three views and four views rather than two views for two main reasons: two views (user and single app auxiliary feature) can only provide one interaction and it does not meet the setting of the proposed MV-AFM whose main idea is learning the attention weights of different view interactions; single category view or permission view is unable to exactly represent the corresponding app. Fig. 2 shows the prediction performance of MV-AFM and baseline models with three and four feature views. Note that U, C, P, T denote the views of user, category, permission, and description text, respectively. It can be found that for each model, the performance of view setting 'U+C+P' is the worst. We argue that neither of category view and permission are view is specific to apps. In other words, it is entirely possible that some apps belong to the same category and have the same permissions. The discriminative information provided by these two feature views is limited. When considering the description text, the models get impressive prediction results, which indicates that the feature view of text can offer more significant information. With all the settings of views except 'U+C+P', the proposed MV-AFM outperforms the other baselines. The obvious improvement obtained by MV-AFM proves the power of two level attention structure which can effectively learn the significant features and view interactions. Though the proposed MV-AFM get comparable results with the settings involving text view, the overall best performance is provided by the utilization of four views. It is probably because that more feature views bring more complementary information and MV-AFM is capable of training a proper combination of these information.

4.6. Parameter analysis

In this subsection, we analyze the impact of two important parameters in the proposed model, embedding size (number of latent factors) and the hidden layer size of the attention network (number of attention factors) based on Google Play dataset. For clarity, we compare the proposed MV-AFM and AFM as both of them are affected by the two parameters. We also implement FM model as the baseline for comparison.

Embedding Size. To analyze the impact of embedding size, we implement MV-AFM compared with FM and AFM by fixing the number of attention factors as 20. We can find that MV-AFM achieves the best performance overall. As shown in Fig. 3, increasing the embedding size does not always bring benefits. AFM keeps a stable performance while FM and MV-AFM even perform worse (especially for MSE) when the embedding size is increased from 30 to 50. It is mainly caused by the growing

Please cite this article as: T. Liang, L. Zheng, L. Chen et al., Multi-view factorization machines for mobile app recommendation based on hierarchical attention, Knowledge-Based Systems (2019), https://doi.org/10.1016/j.knosys.2019.06.029.

Table 2
Performance comparison on Google Play dataset.

Training	Metrics	FM	Wide&Deep	AFM	DeepFM	MV-FM	MV-AFM	Improvement
60%	MAE	0.9388	0.9319	0.9317	0.9383	0.9416	**0.9303**	1.19%
	MSE	1.6791	1.7168	1.6474	1.6403	1.6383	**1.6299**	5.06%
70%	MAE	0.9335	0.9259	0.9391	0.9387	0.9348	**0.9140**	2.68%
	MSE	1.6337	1.6671	1.6165	1.6027	1.5913	**1.5871**	4.80%
80%	MAE	0.9238	0.9125	0.9295	0.9268	0.9216	**0.9106**	2.03%
	MSE	1.6187	1.6368	1.6050	1.6051	1.5896	**1.5855**	3.14%

The best results are listed in bold.

Table 3
Performance comparison on Apple App Store dataset.

Training	Metrics	FM	Wide&Deep	AFM	DeepFM	MV-FM	MV-AFM	Improvement
60%	MAE	1.0556	1.0479	1.0408	1.0322	1.0345	**1.0271**	2.70%
	MSE	1.7370	1.7249	1.7115	1.7305	1.7231	**1.7078**	1.68%
70%	MAE	1.0428	1.0330	1.0297	1.0179	1.0205	**1.0117**	2.99%
	MSE	1.6982	1.6733	1.6654	1.6834	1.6769	**1.6576**	2.39%
80%	MAE	1.0385	1.0459	1.0283	1.0163	1.0176	**1.0130**	3.14%
	MSE	1.6836	1.6865	1.6616	1.6771	1.6645	**1.6543**	1.91%

The best results are listed in bold.

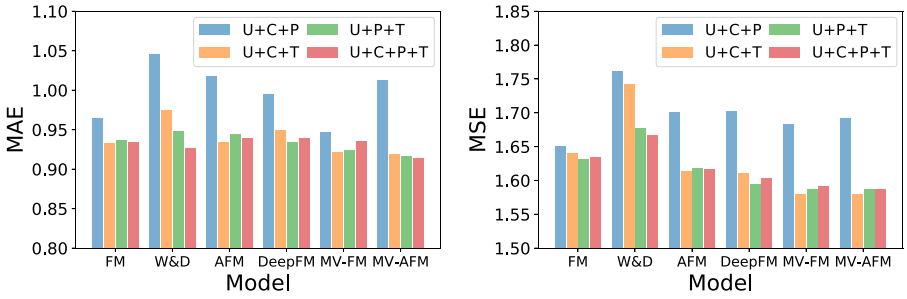


Fig. 2. Performance of MV-AFM and baseline models with different feature views.

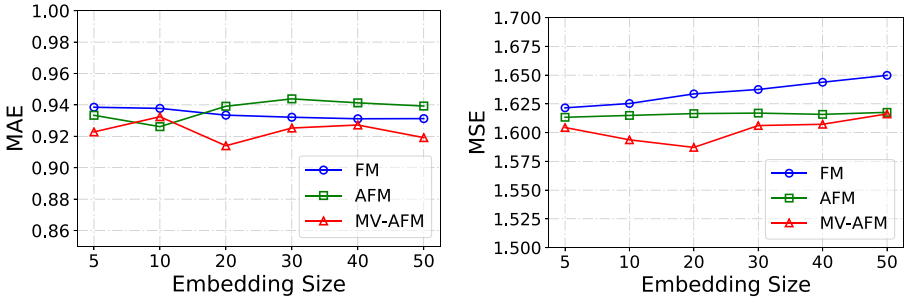
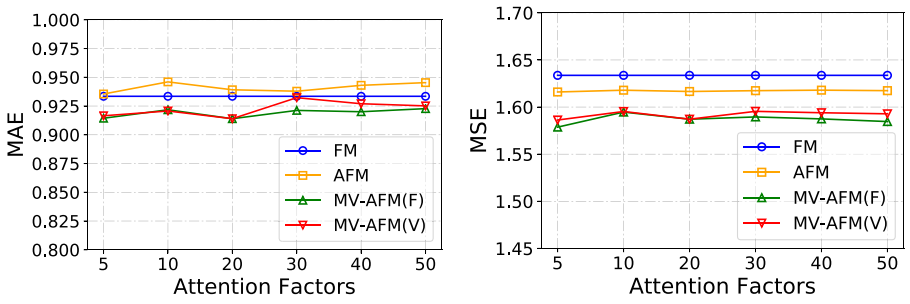


Fig. 3. Performance of MV-AFM *w.r.t.* different embedding size.

model complexity brought by the increasing embedding size, which makes the model tend to overfit. In our dataset, it is proper to set the embedding size to 20.

Attention Factor. We investigate the impact of the hidden layer size of attention networks in AFM and the proposed MV-AFM model. Here we set embedding size as 20. As MV-AFM has two levels of attention networks, we change the number of attention factors of one level with that of another level fixed by 20. Fig. 4 shows the performance in terms of MAE and MSE with different attention factors. MV-AFM(F) means changing the hidden layer size of feature-level attention network, and MV-AFM(V) denotes changing the number of view-level attention factors. It can be observed that MV-AFM model consistently outperforms the baseline models, even when compared with the optimal performance of FM and AFM, which proves the usefulness of the two-level attention network in learning the weights of features in a view and view interactions. We can also find that the performance of AFM is stable while MV-AFM has a larger fluctuation. Compared to MV-AFM(V), MV-AFM(F) provides the more stable results, which indicates that the MV-AFM model is more sensitive to the hidden layer size of view-level attention network. With a proper number of attention factors, MV-AFM can be significantly improved. In our dataset, 20 attention factors for both feature-level and view-level attention networks is a good choice.

Please cite this article as: T. Liang, L. Zheng, L. Chen et al., Multi-view factorization machines for mobile app recommendation based on hierarchical attention, Knowledge-Based Systems (2019), https://doi.org/10.1016/j.knosys.2019.06.029.



(a) (b)

Fig. 4. Performance of MV-AFM *w.r.t.* different attention factors.

Table 4
Efficiency on Google Play.

	Ave.Time (s)	#Para
FM	1.50	822,529
Wide&Deep	4.64	1,068,911
AFM	163.79	822,989
DeepFM	13.85	965,330
MV-FM	7.40	822,529
MV-AFM	68.65	825,731

Table 5
Efficiency on Apple's App Store.

	Ave.Time (s)	#Para
FM	3.00	815,410
Wide&Deep	10.59	1,032,017
AFM	266.41	815,870
DeepFM	30.70	946,032
MV-FM	2.64	815,410
MV-AFM	201.69	818,171

4.7. Efficiency analysis

Tables 4 and 5 record the number of parameters and running time of all the methods on two datasets. As we apply early stopping strategy in the evaluation, the number of epochs in different methods are different. So we average the running time as the time of each epoch.

It can be found that compared to the baseline models except AFM, the proposed MV-AFM has no advantage in running time. AFM and MV-AFM require more time for each epoch due to the computation of attention scores. Though MV-AFM has two layers of attention, its computational efficiency is higher than AFM, which is mainly caused by the view segmentation. The variation of MV-AFM (i.e., MV-FM) has relatively high efficiency, especially for the Apple' App Store. Wide&Deep and DeepFM need to train much more parameters than the other models as they consider both low-order and high-order feature interactions. The rest methods have the similar parameter numbers since most parameters come from the embedding layer which is almost the same.

In summary, the view segmentation approach is shown to be very effective. Both MV-AFM and MV-FM are different ways to implement view segmentation. MV-AFM is the most accurate but tends to be slow due to two layers of attention. If speed is the concerned, one can consider its variation MV-FM which is very fast, but less accurate.

4.8. Case study

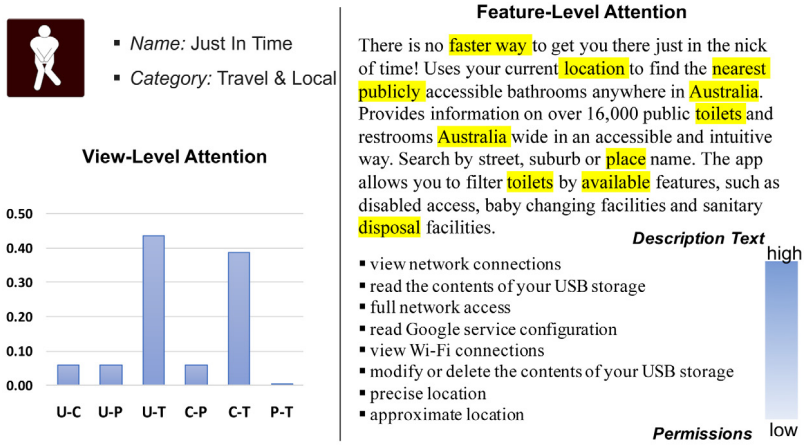
In this subsection, we provide two examples of Google Play in Fig. 5 for a clear understanding of the attention mechanism in MV-AFM. As shown in Fig. 5, for the feature level, we highlight the words assigned with higher attention values for the view of description text and use a heat map to represent the attention values of features in permission view. For the view-level, we show the attention values of all the six view interactions through bar graph in which U, C, P, T denote the views of user, category, permission, and description text, respectively. It is observed that the words in description text assigned the higher weights can substantially represent the details for the functions of apps. For example, terms like 'location', 'toilets' indicate the content of app *Just In Time* to some extent, and words like 'songs', 'audio', and 'recorder' in part reflect the function of app *J4T Multitrack Recorder*. The attention distributions of different instances are quite different. For the example in Fig. 5(a), the attention values of user-text interaction and category-text interaction are much higher than the others, which indicates the interactions between user and description, category and description provide useful complementary information. The attention values of interactions involving permission view are relatively small, which is probably because the permissions with high feature-level attention values are very common in many other apps. For the example in Fig. 5(b), the importance of description diminishes as the category and permission provide information directly related to the app function (i.e., *Music & Audio* and *record audio*). We can find that the weights of permission-text interaction in both examples are quite low. It is probably because of the complex feature components in these two views which easily brings noise. These observations suggest that MV-AFM is able to capture the significant features and view interactions via the hierarchical attentions.

5. Related work

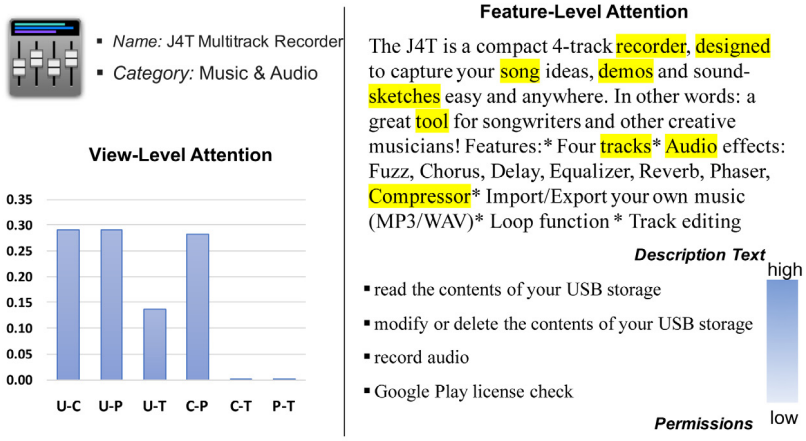
To the best of our knowledge, this is the first work considering the multiple interactions between views with attention-based neural network to facilitate mobile app recommendation. From the conceptual perspective, three topics is closely related to this work: mobile app recommendation, multi-view deep recommender systems, and attention mechanism. We give a short overview of these areas and distinguish our work from other existing approaches.

Mobile App Recommendation. As an effective solution for information overload, recommender system has been widely adopted in various domains [16–18]. A hybrid model for music item recommendation is proposed in [18], which leverages both collaborative information coming from the user's community and content information from the knowledge graph. Luo et al. [16, 19] propose a second-order latent factor based approach for preference prediction and adopt the principle of Hessian-free optimization to decrease the computational cost. Some work

Please cite this article as: T. Liang, L. Zheng, L. Chen et al., Multi-view factorization machines for mobile app recommendation based on hierarchical attention, Knowledge-Based Systems (2019), https://doi.org/10.1016/j.knosys.2019.06.029.



(a)



(b)

Fig. 5. Examples of the results on feature-level attention and view-level attention. For the feature-level, we highlight the words assigned higher attention values for the view of description text. We use a heat map to represent the attention values of features in permission view, in which the darker color indicates the higher attention value. For the view-level, we show the attention values of all the six view interactions. Here U, C, P, T represent the views of user, category, permission, and description text, respectively. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

focus on addressing the problem of high computational and storage complexity in traditional recommender systems [20–22]. Recently, mobile app recommendation has drawn an increasing number of attentions as an effective way to alleviate information overload in app market [4,5,23]. In [24], Yin et al. apply users' view/download sequences to mine the actual value and tempting value of apps, which are used to build a recommendation model considering the contest between apps. Zhu et al. [4] propose a flexible app recommendation approach combining apps' popularity and users' security preferences. Cao et al. [3] propose a hybrid solution for app recommendation that jointly models numerical ratings and textual content from multiple platforms. Liu et al. [25] propose a structural user choice model to learn fine-grained user preferences by leveraging the hierarchical taxonomy of apps as well as the competitive relationships among apps. [26] develops a sparse additive generative model for mobile app recommendation which jointly learns user interests and category-aware user privacy preferences in a unified way. Liang et al. propose a feature-oriented model to learn user preferences on apps which firstly predicts user ratings on features and uses them to generate the ratings on apps [5]. In this work, we proposed to model a neural network with attention mechanism considering multi-view features.

Multi-View Deep Recommender Systems. Multi-view learning has been applied for various tasks, such as clustering, classification, prediction [27–30]. Recently, multi-view based recommendation draws more and more attentions, which can be roughly classified into three types: multi-view tensor-based recommendation, multi-view latent factor based recommendation, multi-view deep learning based recommendation. Our work is related to multi-view deep learning based recommendation [17, 31–37]. In [31], Elkahky et al. develop a multi-view deep learning model for recommendation which uses a DNN to map high-dimensional sparse features from different domains into dense features in a joint semantic space. Based on this work, Song et al. [32] incorporate both long-term static and short-term temporal user preferences to improve the recommendation performance. In [35], each type of information source (review text, product image, numerical rating, etc.) is adopted to learn the corresponding user and item representations based on available (deep) representation learning architectures. The mentioned three works mainly apply the feedforward neural network, while some other works try more complex neural network for recommendation. [34] proposes a multi-view recurrent model for sequential recommendation in which the item representation

Please cite this article as: T. Liang, L. Zheng, L. Chen et al., Multi-view factorization machines for mobile app recommendation based on hierarchical attention, Knowledge-Based Systems (2019), https://doi.org/10.1016/j.knosys.2019.06.029.

is constructed with multi-view features (visual and textual features). [17] introduces two parallel convolutional networks to learn from the user view and the item view. Lu et al. [33] propose structural factorization machines (SFMs) to learn the common latent spaces shared by multi-view tensors and automatically adjust the importance of each view in the predictive model.

Attention Mechanism has been widely recognized as an effective technique in various tasks, such as machine translation [38, 39], speech recognition [40–42], caption generation [43–45] and so on. It works mainly because of the reasonable assumption that human recognitive attention is able to focus on certain parts of the entire perception space. Recently, attention-based method is successfully expanded into recommender systems [46–50] as it is able to filter out uninformative content and select the most representative information. Wu et al. [46] identify three social contextual aspects that affect users' preferences from heterogeneous data sources and design a hierarchical attention network to model the hierarchical structure of social contextual image recommendation. Chen et al. [49] present a CF framework that employs attention mechanism to address the implicit feedback in multimedia recommendation. The proposed model is a neural network consisting of a component-level attention module an item-level attention module. In [50], a neural network model named Neural Attentive Item Similarity model (NAIS) is proposed for item-based CF. The attention network in NAIS is able to distinguish which historical items in a user profile are more significant for a prediction. [47] proposes an attentive aspect-based recommendation model which effectively captures the interactions between aspects extracted from reviews for recommendation.

It is worth noting that the existing works for app recommendation have not tried deep learning model with multi-view features. Motivated from aforementioned works, this paper tries to offer accurate prediction of users' preference on apps by building a neural network that models the view interactions and estimating the importance of different interactions with attention network.

6. Conclusions

In this paper, we aim to make accurate rating prediction for mobile apps and propose an hierarchical neural network called MV-AFM that combines a multi-view variation of FM model with attention mechanism to learn sophisticated feature interactions. Compared with existing FM-based neural network models, the novelty of MV-AFM is the consideration of view segmentation, namely modeling the feature interactions from different views. The proposed feature-level attention and view-level attention in MV-AFM are capable of discriminating the weights of features within each view and interactions between views. We compare the proposed model with a set of state-of-the-art models on two real-world datasets. The experimental results demonstrate the effectiveness of MV-AFM model.

Acknowledgments

This work is partially supported in part by the NSFC, China through grants 61672453, 61672313, 61702568, and U1711267, the Subject of the Major Commissioned Project "Research on China's Image in the Big Data" of Zhejiang Province's Social Science Planning Advantage Discipline "Evaluation and Research on the Present Situation of China's Image" No. 16YSXK01ZD-2YB, Ministry of Education of China through grant 2017PT18, the Zhejiang University Education Foundation, China through grant K18-511120-004, K17-511120-017, and No. K17-518051-021, the National Key Research and Development Program, China through grant 2017YFB0202200, the NSF of China through grants IIS-1526499 and CNS-1626432, the Major Scientific Project of Zhejiang Lab, China under grant No. 2018DG0ZX01, the Program for Guangdong Introducing Innovative and Entrepreneurial Teams through grant 2017ZT07X355, and the Fundamental Research Funds for the Central Universities, China under grant 17lgpy117.

References

[1] E. Costa-Montenegro, A.B. Barragáns-Martínez, M. Rey-López, Which app? a recommender system of applications in markets: implementation of the service for monitoring users' interaction, Expert Syst. Appl. 39 (10) (2012) 9367–9375.
[2] C. Liu, J. Cao, S. Feng, Leveraging Kernel-incorporated matrix factorization for app recommendation, ACM Trans. Knowl. Discovery Data 13 (3) (2019) 31.
[3] D. Cao, X. He, L. Nie, X. Wei, X. Hu, S. Wu, T.-S. Chua, Cross-platform app recommendation by jointly modeling ratings and texts, ACM Trans. Inf. Syst. 35 (4) (2017) 37.
[4] H. Zhu, H. Xiong, Y. Ge, E. Chen, Mobile app recommendations with security and privacy awareness, in: Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2014, pp. 951–960.
[5] T. Liang, L. Chen, X. Ying, S.Y. Philip, J. Wu, Z. Zheng, Mobile application rating prediction via feature-oriented matrix factorization, in: International Conference on Web Services (ICWS), IEEE, 2017, pp. 261–268.
[6] Y. Yao, W.X. Zhao, Y. Wang, H. Tong, F. Xu, J. Lu, Version-aware rating prediction for mobile app recommendation, ACM Trans. Inf. Syst. 35 (4) (2017) 38.
[7] S. Rendle, Factorization machines, in: International Conference on Data Mining, IEEE, 2010, pp. 995–1000.
[8] H. Guo, R. Tang, Y. Ye, Z. Li, X. He, DeepFM: a factorization-machine based neural network for CTR prediction, in: Proceedings of the 26th International Joint Conference on Artificial Intelligence, AAAI Press, 2017, pp. 1725–1731.
[9] J. Xiao, H. Ye, X. He, H. Zhang, F. Wu, T.-S. Chua, Attentional factorization machines: learning the weight of feature interactions via attention networks, in: Proceedings of the 26th International Joint Conference on Artificial Intelligence, AAAI Press, 2017, pp. 3119–3125.
[10] B. Cao, H. Zhou, G. Li, P.S. Yu, Multi-view machines, in: Proceedings of the Ninth ACM International Conference on Web Search and Data Mining, ACM, 2016, pp. 427–436.
[11] X. Glorot, Y. Bengio, Understanding the difficulty of training deep feedforward neural networks, in: Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, 2010, pp. 249–256.
[12] H. Zhu, H. Xiong, Y. Ge, E. Chen, Discovery of ranking fraud for mobile apps, IEEE Trans. Knowl. Data Eng. 27 (1) (2015) 74–87.
[13] H. Zhu, C. Liu, Y. Ge, H. Xiong, E. Chen, Popularity modeling for mobile apps: A sequential approach, IEEE Trans. Cybern. 45 (7) (2015) 1303–1314.
[14] A. Gunawardana, G. Shani, A survey of accuracy evaluation metrics of recommendation tasks, J. Mach. Learn. Res. 10 (Dec) (2009) 2935–2962.
[15] H.-T. Cheng, L. Koc, J. Harmsen, T. Shaked, T. Chandra, H. Aradhye, G. Anderson, G. Corrado, W. Chai, M. Ispir, et al., Wide & deep learning for recommender systems, in: Proceedings of the 1st Workshop on Deep Learning for Recommender Systems, ACM, 2016, pp. 7–10.
[16] X. Luo, M. Zhou, S. Li, Y. Xia, Z.-H. You, Q. Zhu, H. Leung, Incorporation of efficient second-order solvers into latent factor models for accurate prediction of missing QoS data, IEEE Trans. Cybern. 48 (4) (2018) 1216–1228.
[17] L. Zheng, V. Noroozi, P.S. Yu, Joint deep modeling of users and items using reviews for recommendation, in: Proceedings of the Tenth ACM International Conference on Web Search and Data Mining, ACM, 2017, pp. 425–434.
[18] S. Oramas, V.C. Ostuni, T.D. Noia, X. Serra, E.D. Sciascio, Sound and music recommendation with knowledge graphs, ACM Trans. Intell. Syst. Technol. (TIST) 8 (2) (2017) 21.
[19] X. Luo, M. Zhou, S. Li, Y. Xia, Z. You, Q. Zhu, H. Leung, An efficient second-order approach to factorize sparse matrices in recommender systems, IEEE Trans. Ind. Inf. 11 (4) (2015) 946–956.
[20] X. Luo, M. Zhou, S. Li, Z. You, Y. Xia, Q. Zhu, A nonnegative latent factor model for large-scale sparse matrices in recommender systems via alternating direction method, IEEE Trans. Neural Netw. Learn. Syst. 27 (3) (2016) 579–592.
[21] X. Luo, M. Zhou, Y. Xia, Q. Zhu, An efficient non-negative matrix-factorization-based approach to collaborative filtering for recommender systems, IEEE Trans. Ind. Inf. 10 (2) (2014) 1273–1284.
[22] X. Luo, M. Zhou, S. Li, M. Shang, An inherently nonnegative latent factor model for high-dimensional and sparse matrices from industrial applications, IEEE Trans. Ind. Inf. 14 (5) (2018) 2011–2022.

[23] T. Liang, L. He, C.-T. Lu, L. Chen, S.Y. Philip, J. Wu, A broad learning approach for context-aware mobile application recommendation, in: 2017 IEEE International Conference on Data Mining (ICDM), IEEE, 2017, pp. 955–960.
[24] P. Yin, P. Luo, W.-C. Lee, M. Wang, App recommendation: a contest between satisfaction and temptation, in: Proceedings of the International Conference on Web Search and Data Mining, ACM, 2013, pp. 395–404.
[25] B. Liu, Y. Wu, N.Z. Gong, J. Wu, H. Xiong, M. Ester, Structural analysis of user choices for mobile app recommendation, ACM Trans. Knowl. Discovery Data 11 (2) (2016) 17.
[26] H. Yin, W. Wang, L. Chen, X. Du, Q.V.H. Nguyen, Z. Huang, Mobi-SAGE-RS: A sparse additive generative model-based mobile application recommender system, Knowl.-Based Syst. 157 (2018) 68–80.
[27] H. Wang, Y. Yang, B. Liu, H. Fujita, A study of graph-based system for multi-view clustering, Knowl.-Based Syst. 163 (2019) 1009–1019.
[28] F. Nie, G. Cai, J. Li, X. Li, Auto-weighted multi-view learning for image clustering and semi-supervised classification, IEEE Trans. Image Process. 27 (3) (2017) 1501–1511.
[29] C.-T. Lu, L. He, W. Shao, B. Cao, P.S. Yu, Multilinear factorization machines for multi-task multi-view learning, in: Proceedings of the Tenth ACM International Conference on Web Search and Data Mining, ACM, 2017, pp. 701–709.
[30] J. Bi, C. Zhang, An empirical comparison on state-of-the-art multi-class imbalance learning algorithms and a new diversified ensemble learning scheme, Knowl.-Based Syst. 158 (2018) 81–93.
[31] A.M. Elkahky, Y. Song, X. He, A Multi-View Deep Learning Approach for Cross Domain User Modeling in Recommendation Systems, International World Wide Web Conferences Steering Committee, 2015, pp. 278–288.
[32] Y. Song, A.M. Elkahky, X. He, Multi-rate deep learning for temporal recommendation, in: Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval, ACM, 2016, pp. 909–912.
[33] C.-T. Lu, L. He, H. Ding, B. Cao, S.Y. Philip, Learning from Multi-View Multi-Way Data via Structural Factorization Machines, International World Wide Web Conferences Steering Committee, 2018, pp. 1593–1602.
[34] Q. Cui, S. Wu, Q. Liu, W. Zhong, L. Wang, MV-RNN: A multi-view recurrent neural network for sequential recommendation, IEEE Trans. Knowl. Data Eng. (2018).
[35] Y. Zhang, Q. Ai, X. Chen, W.B. Croft, Joint representation learning for top-n recommendation with heterogeneous information sources, in: Proceedings of the 2017 ACM on Conference on Information and Knowledge Management, ACM, 2017, pp. 1449–1458.
[36] L. Zheng, C.-T. Lu, F. Jiang, J. Zhang, P.S. Yu, Spectral collaborative filtering, in: Proceedings of the 12th ACM Conference on Recommender Systems, ACM, 2018, pp. 311–319.
[37] Y. Guan, Q. Wei, G. Chen, Deep learning based personalized recommendation with multi-view information integration, Decis. Support Syst. 118 (2019) 58–69.
[38] J. Zhou, Y. Cao, X. Wang, P. Li, W. Xu, Deep recurrent models with fast-forward connections for neural machine translation, Trans. Assoc. Comput. Linguist. 4 (2016) 371–383.
[39] B. Zhang, D. Xiong, J. Su, Neural machine translation with deep attention, IEEE Trans. Pattern Anal. Mach. Intell. (2018).
[40] J.K. Chorowski, D. Bahdanau, D. Serdyuk, K. Cho, Y. Bengio, Attention-based models for speech recognition, in: Advances in Neural Information Processing Systems, 2015, pp. 577–585.
[41] S. Watanabe, T. Hori, S. Kim, J.R. Hershey, T. Hayashi, Hybrid CTC/attention architecture for end-to-end speech recognition, IEEE J. Sel. Top. Sign. Proces. 11 (8) (2017) 1240–1253.
[42] Y. Wu, H. Mao, Z. Yi, Audio classification using attention-augmented convolutional neural network, Knowl.-Based Syst. 161 (2018) 90–100.
[43] L. Gao, Z. Guo, H. Zhang, X. Xu, H.T. Shen, Video captioning with attention-based LSTM and semantic consistency, IEEE Trans. Multimed. 19 (9) (2017) 2045–2055.
[44] L. Chen, H. Zhang, J. Xiao, L. Nie, J. Shao, W. Liu, T.-S. Chua, Sca-cnn: Spatial and channel-wise attention in convolutional networks for image captioning, in: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), IEEE, 2017, pp. 6298–6306.
[45] L. Gao, X. Li, J. Song, H.T. Shen, Hierarchical LSTMs with adaptive attention for visual Captioning, IEEE Trans. Pattern Anal. Mach. Intell. (2019).
[46] L. Wu, L. Chen, R. Hong, Y. Fu, X. Xie, M. Wang, A hierarchical attention model for social contextual image recommendation, IEEE Trans. Knowl. Data Eng. (2019).
[47] X. Guan, Z. Cheng, X. He, Y. Zhang, Z. Zhu, Q. Peng, T.-S. Chua, Attentive aspect modeling for review-aware recommendation, ACM Trans. Inf. Syst. 37 (3) (2019) 28.
[48] F. Xue, X. He, X. Wang, J. Xu, K. Liu, R. Hong, Deep item-based collaborative filtering for top-n recommendation, ACM Trans. Inf. Syst. 37 (3) (2019) 33.
[49] J. Chen, H. Zhang, X. He, L. Nie, W. Liu, T.-S. Chua, Attentive collaborative filtering: Multimedia recommendation with item-and component-level attention, in: Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval, ACM, 2017, pp. 335–344.
[50] X. He, Z. He, J. Song, Z. Liu, Y.-G. Jiang, T.-S. Chua, NAIS: Neural attentive item similarity model for recommendation, IEEE Trans. Knowl. Data Eng. 30 (12) (2018) 2354–2366.

Please cite this article as: T. Liang, L. Zheng, L. Chen et al., Multi-view factorization machines for mobile app recommendation based on hierarchical attention, Knowledge-Based Systems (2019), https://doi.org/10.1016/j.knosys.2019.06.029.