IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received April 22, 2020, accepted May 11, 2020, date of publication May 25, 2020, date of current version June 30, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2997115

Attention-Based Adaptive Memory Network for Recommendation With Review and Rating

WEI LIU[1,4], (Member, IEEE), ZHIPING LIN[1,4], HUAIJIE ZHU[1,4], (Member, IEEE), JING WANG[2], AND ARUN KUMAR SANGAIAH[3], (Member, IEEE)

[1]School of Data and Computer Science, Sun Yat-sen University, Guangzhou 510006, China
[2]School of Artificial Intelligence, The Open University of Guangdong, Guangzhou 510091, China
[3]School of Computing Science and Engineering, Vellore Institute of Technology (VIT), Vellore 632014, India
[4]Guangdong Key Laboratory of Big Data Analysis and Processing, Guangzhou 510006, China

Corresponding author: Huaijie Zhu (zhuhuaijie@mail.sysu.edu.cn)

This work was supported in part by the Natural Science Foundation of Guangdong Province under Grant 2019A1515011159 and Grant 2019A1515011704, in part by the National Science Foundation for Post-Doctoral Scientists of China under Grant 2019M663237 and Grant 2018M643307, in part by the Young Teacher Training Project of Sun Yat-sen University under Grant 19lgpy223 and Grant 19lgpy214, in part by the Guangdong Key Laboratory of Big Data Analysis and Processing Open Fund project (2018-3), The Open University of Guangdong ''Innovation and Strong University'' Scientific Research Project under Grant 2019KDCQ04-03, and in part by The Open University of Guangdong Talent Project under Grant RC1916.

ABSTRACT Item recommendation has become a significant means to help people discover interesting items. Meanwhile, plenty of reviews and ratings in recommender system can be utilized to relieve data sparsity problem. However, existing review-based approaches ignore the influence of static preference of user and the static characteristics of item, which could reflect long-term and stationary property, and guide feature extraction from reviews. Moreover, adaptive property, i.e., the importance of the historical records to each user and item, is not fully exploited in previous works. In this paper, we propose an Attention-based Adaptive Memory Network (AAMN) model to leverage historical reviews and ratings systemically. Specifically, we propose an attention mechanism guided by the static features to learn the importance of different historical records, for modeling the adaptive features of users and items. Notably, this paper is the first to bring static features into adaptively extracting semantic information from reviews, which can not only characterize user and item from a global view, but also assist to distinguish the importance of different reviews. In addition to the attention mechanism, we propose a non-linear feature fusing layer and a deep interaction layer to combine the static features and adaptive features, which capture underlying interactions among these features. To further improve prediction accuracy and training efficiency, we propose a dynamic sampling strategy for model training. We conduct extensive experiments on 16 benchmark datasets from Amazon and Yelp. The results demonstrate that our model outperforms the state-of-the-art models.

INDEX TERMS Recommendation, review, attention, memory networks.

I. INTRODUCTION

Recommender system is everywhere in people's daily life [3], and it helps user discover more interesting products and useful services [2]. Merchants could also obtain more potential customers owing to recommender systems [4], [26]. In the early stage of recommender system, researchers mostly focused on user's interactions with items, *such as* rating, buying, and clicking. Collaborative filtering is the most popular method to uncover the potential features of users and items *via* user-item interactions. However, it easily corrupts due to data sparsity (*e.g.,* most of the users or items have few ratings).

Apart from the ratings, online reviews generated by users from social networks and e-commerce platforms often imply users' opinions to the products (or services). To understand, let's consider Table 1, in which several reviews and corresponding ratings are listed. One can easily see that the ratings only reflect whether the user likes the item, while the reviews imply more details on why a user likes the item. *For example*, user u expresses opinions to the appearance and practicability of item i_1, using the review sentence "the earrings are beautiful and sturdy, and I will order again." These details are useful to uncover semantic features of user preferences and item attributes.

The associate editor coordinating the review of this manuscript and approving it for publication was Rongbo Zhu.

TABLE 1. An example of user *u*'s review.

From	ID	Review	Rating
user u	r_{u,i_1}	I gave these Black Cubic Zirconia ***Earrings*** as a gift to my step-daughter after she received her job promotion. They are really beautiful and sturdy. I will order from this seller again.	5.0
	r_{u,i_2}	Lilyette uses an excellent design for their minimizer ***bras***. In addition, the effectiveness to minimize is fantastic. The straps are comfortable.	4.0
	r_{u,i_3}	I am so pleased to have purchased these 925 Sterling Silver ***Earrings***. I plan to order another pair for my daughter.	5.0
item i	$r_{u_1,i}$	... What a nice ***case*** to sort my collection. I love all the compartments and the layout too ... It will please even the most fussy, as I am. The price is fair ... Seems well made too ... This is for serious jewelry lover's. Buy it.....	5.0
	$r_{u_2,i}$	...It arrived with some of the leather scratched... This particular color pink also looked pretty bad in person. The quality of this ***box*** seemed lesser than the brown one I own. ...	3.0
	$r_{u_3,i}$	Really nice ***box***, a bit cheaply made. It is made of carboard with pleather ...I'm not so sure how it will stand up to my tween daughter's use. Every inch is useable space for jewelry & very compact, which is a nice feature.	4.0
u to i	$r_{u,i}$	My granddaughter likes the Shining Image Jewelry Box given to her for a birthday gift from Grandma.I wish the drawers were deeper for holding her many necklaces. Also, I wish the drawers had a stop, preventing the drawers from tipping contents entirely.	3.0

As such, much attention has been attracted to overcome the data sparsity problem using reviews. For example, McAuley and Leskovec [12] proposed to combine latent review topics with latent rating dimensions. Wang *et al.* [26] captured local context of words and integrated word embedding model with standard matrix factorization. Zheng *et al.* [22] utilized two parallel neural networks, which exploit reviews written by the user and the reviews for the item, respectively; then, the two networks were coupled into factorization machine for rating prediction. Moreover, Tay *et al.* [28] proposed a co-attention network that models adaptive semantic features of users and items, *namely*, they consider the historical reviews of a user are of different importance when the user faces different items. Although these recent review-based approaches relieve the data sparsity problem to some extent, they still have the following open issues.

A. STATIC FEATURES ARE IGNORED, WHEN EXTRACTING ADAPTIVE SEMANTIC FEATURES FROM REVIEWS

Recent works mainly focused on semantic features from a local view, e.g., sentences [22], [29] and words of reviews [28]. As the reviews of each user/item are limited, it is often difficult to predict the ratings of a user to an unmet item. For example in Table 1, the existing methods used the historical reviews with ratings by user u, and historical reviews with ratings for item i, to predict rating of u to i.[1] From the reviews we could guess user u probably likes earrings, and item i is likely a jewelry box. Maybe u needs a jewelry box to hold her earrings, but it is difficult to predict how much user u would like item i without static features (e.g., the correlation between "earrings" and "jewelry box"), because the semantic correlation between the reviews for u and i is weak given the limited number of reviews. Our paper alleviates this issue by introducing static features for both user and item, which depict user's (item's) long-term and stationary property and learn the potential relationship between users (items).

B. HISTORICAL INFORMATION OF USER (ITEM) IS NOT UTILIZED EFFICIENTLY

User's historical information includes reviews and ratings. Recent works [6], [28], [37] exploited part of the information, either they made prediction without history reviews of user and item [6], [37], or they utilized only history reviews of user and item, without behavior informa- tion such as click/rating history [28]. Few work has utilized all the information collectively and effectively. Specially, when facing different items, stationary features [23] could not depict user's property sufficiently. Moreover, ratings of history items are mostly utilized as only final prediction criterion [22], [37]. In fact, these ratings would be useful on predicting rating to other items [6]. For example, in Table 1, u gives i_2 a rating 4 and the other two items a higher rating 5. These ratings are good instructions for the importance of each item. In our paper, we leverage the memory network to store historical rated items of target user, and exploit the rating information with transformation function, which could evaluate each item's weight and form adaptive property feature of target user and target item.

C. FEATURE FUSION STILL REMAINS AT SHALLOW LEVEL

Since semantic feature is one of the main representation types of user (item) and potentially complicated, linear models [21], [28] (*e.g.,* matrix factorization) can only capture the first-order feature relationship (*i.e.,* the pointwise relationship between the dimensions of the embedding space). Although Multi-Layer Perception (MLP) [23] could be utilized to learn any continuous function according to the universal approximation theorem, there is no guarantee that the feature correlations (*i.e.,* the pairwise correlations between the dimensions of the embedding space) can be effectively captured with current optimization techniques [24]. In our paper, we address this issue by introducing outer product operation on features to obtain high-order features, and applying convolutional

[1] The last row of the table is u's real review and rating to item i for reference. When doing the prediction, it is not observable.

neural networks (CNN) to uncover the relationship of these high-order features.

D. SAMPLES ARE TREATED EQUALLY IN MOST OF THE RECENT WORKS, WHICH MAY LEAD TO SUB-OPTIMAL PERFORMANCE

Different samples should be of different importance for training because they would generate different prediction errors. For instance, the prediction error on sample A may be smaller than that of sample B. The contribution of sample B to the loss function would be more important than sample A. If the model considers them equally, the training process may waste time on unimportant samples (*e.g.,* sample A), and thus lead to a sub-optimal result. To address this issue, our paper employs a dynamic sampling strategy, which would give a sampling weight to each sample, considering their prediction error after each epoch of training.

Motivated by the above insights, we develop an Attention-based Adaptive Memory Network (AAMN) for item recommendations with reviews and ratings. Firstly to depict user's (item's) property from a global view, we learn the static features of users and items from all interaction data. Secondly to leverage historical records efficiently, we extract adaptive semantic and property features from historical reviews and interactions, respectively. Specifically, we design a memory network with attention mechanism to find the most relevant reviews/interactions for a user (item), based on the static features of the item (user). These adaptive features represent the specific state of the user (item) when facing different target items (users). Thirdly, we fuse the static and adaptive features by computing an outer product feature matrix and using a convolutional neural network. Fourthly to boost the effectiveness of model training, we propose a dynamic sampling method, based on the prediction errors of the samples.

To summarize, the contributions of this paper are as follows:

- We propose a model called *Attention-based Adaptive Memory Network*. It exploits static and adaptive features systematically. To the best of knowledge, this paper is the first to bring in static features, which represent the inherent property of user (item), to facilitate the extraction of adaptive features.
- We propose to use adaptive semantic and property features to characterize users (items) w.r.t. different target items (users). Meanwhile, we develop an attention mechanism that leverages static features to select most relevant historical interactions and reviews from a memory network, so as to construct the adaptive features efficiently.
- We suggest the deep convolutional layers to explore interactive relationship between features of users and items. Besides, we propose a dynamic sampling method for the training process, which can benefit to the convergence of model training and the improvement of recommendation performance.
- We conduct extensive experiments on 16 public datasets. The results demonstrate the competitiveness of our model, especially on sparse data. In addition, we also demonstrate the effectiveness of main modules *via* ablation study. (Our codes could be found at the open-source code repository, GitHub, https://github.com/wiio12/ADMN.)

The rest of the paper is organized as follows. Section II formulates the problem and some important concepts. Section III covers our model in detail. In Section IV, experiments and results analysis are presented. In Section V, we review recent works related to ours. Section VI concludes the paper.

II. PROBLEM DEFINITION

The mathematical notations used in this paper are summarized in Table 2. Italic bold uppercase letter denotes matrix, bold lowercase letter depicts vector, and normal lowercase letter denotes a scaler.

TABLE 2. Notations.

Symbols	Descriptions
$v, \mathbf{v}$	the word in reviews, the corresponding embedding
$\mathbf{v}$	the embedding of word in reviews
$\mathcal{V}$	dictionary of words
$w_{u,i}$	a review text written by user u for item i
$\mathbf{w}_{u,i}$	the embedding of review $w_{u,i}$
$\mathcal{W}_u$	the set of user u's reviews
$\mathcal{W}_i$	the set of reviews for item i
$\mathbf{s}_{u\|i}$	adaptive semantic feature of user u given item i
$\mathbf{s}_{i\|u}$	adaptive semantic feature for item i given user u
$\mathbf{x}_u$	static feature of user u
$\mathbf{y}_i$	static feature of item i
$\mathbf{x}'_{u\|i}$	adaptive property feature of user u given item i
$\mathbf{y}'_{i\|u}$	adaptive property feature of item i given user u
$\mathbf{x}''_u$	the combining feature of user u from multi-views
$\mathbf{y}''_i$	the combining feature for item i from multi-views
$\mathbf{O}$	matrix of outer product between $\mathbf{x}''_u$ and $\mathbf{y}''_i$
$\mathbf{o}$	feature extracted from deep interaction layer
$r_{u,i}$	the rating scored by user u to item i

Let $\mathcal{U}$ and $\mathcal{I}$ denote the set of users and items, respectively. Let $\mathcal{D}$ be a set of historical reviews, each review $w_{u,i}$ of a user u on item i corresponds a rating $r_{u,i}$. Let $\mathcal{W}_u$ be the set of historical reviews for item u, and $\mathcal{W}_i$ be the historical reviews for item i. The problem is to predict user u's rating on item i that u has not reviewed yet, utilizing historical reviews and ratings of user u and item i. Next, we proceed to formulate several main concepts used in this paper.

Definition 1 (Static Feature): The static feature is a latent vector that represents user's (or item's) stationary property. Specifically, for each user $u \in \mathcal{U}$, we define u's static feature as a latent vector $\mathbf{x}_u \in \mathbb{R}^d$. For each item $i \in \mathcal{I}$, we define its static feature as a latent vector $\mathbf{y}_i \in \mathbb{R}^d$. Though the static features are similar to latent factors in matrix factorization, few researchers have proposed them specially for assisting semantic features extraction from reviews.

Definition 2 (Review Embeddings): Given the word embeddings (e.g., word2vec) $\mathcal{V} \to \mathbb{R}^{|\mathcal{V}| \times d}$, which maps the vocabulary $\mathcal{V}$ to distributed vector representations, we define the embeddings of the review of user u on item i as the average

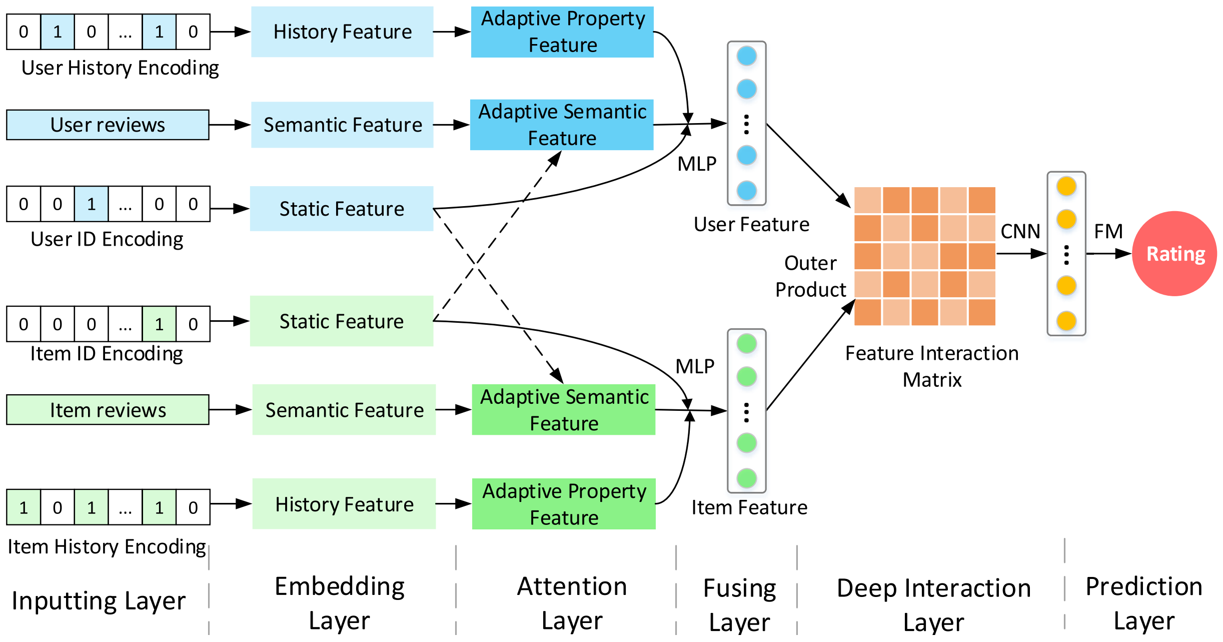


FIGURE 1. **Framework of AAMN.**

embeddings of the words in the review:

$$\mathbf{w}_{u,i} = \frac{1}{t}\sum_{l=1}^{t} \mathbf{v}_l^{u,i} \tag{1}$$

where $\mathbf{v}_l^{u,i}$ means the embedding of the l-th word in review $w_{u,i}$, t is the number of words in review $w_{u,i}$. Note that, any other method [22] that encodes a review into an embedding can be applied. Since this paper's con- tribution is not at this point, we leave the effect of different review embedding methods as future work.

Intuitively, the reviews imply the user's opinions on items, and thus they can be used as a reference to recommend new items to the users. Consider the fact that the historical reviews are not equally important when the user faces different new items. Typically, more relevant reviews should contribute more to construct the user's features on-the-fly. As such, given an item (user), the adaptive semantic feature of a user (item) can be defined as follows.

Definition 3 (Adatpive Semantic Feature): The adatpive semantic feature of a user u given a target item i is a latent vector $\mathbf{s}_{u|i} \in \mathbb{R}^d$ that encodes relevant information to item i from the historical reviews of u. Similarly, the adatpive semantic feature of an item i given a user u is defined as $\mathbf{s}_{i|u} \in \mathbb{R}^d$.

Similar to reviews, user-item interactions can also be considered adaptively, *i.e.,* given the target item (user), the historical interactions (*e.g.,* ratings, clicks) of the user (item) should be treated differently. As such, the adaptive property feature of a user (item) can be defined as follows.

Definition 4 (Adaptive Property Feature): The adaptive property feature of a user u given a target item i is a latent vector $\mathbf{x}'_{u|i} \in \mathbb{R}^d$ that encodes relevant information to item i from the historical user-item interactions of u. Similarly, the adaptive property feature of an item i given a user u is defined as $\mathbf{y}'_{i|u} \in \mathbb{R}^d$.

III. ATTENTION-BASED ADAPTIVE MEMORY NETWORK

In this part, we first provide an overview of our model, *i.e.,* AAMN (Section III-A). Then, we cover the key components of AAMN in detail (Sections III-B∼III-G). Finally, we introduce the training process of the model (Section III-H).

A. OVERVIEW

To support a systematic integration of static features and adaptive features, we design an Attention-based Adaptive Memory Network (AAMN). Fig. 1 shows the framework of our model. First of all, the **inputting layer** (*cf.,* the left most part) gets the raw input of reviews, user-item interactions, and the IDs of users/items; and it transforms the raw data to sparse encoding vectors. Next, the sparse encoding vectors are put into the **embedding layer**, which is responsible for converting the sparse encodings to dense embedding vectors, in order to uncover potential features (*e.g.,* static features) from the sparse encodings. With these potential features such as static features, the **attention layer** takes them as reference to select most relevant data (including reviews and user-item interactions) for constructing adaptive semantic and property features. Afterwards, the **fusing layer** combines static features, adaptive semantic features and adaptive property features of user (item) to construct a single dense vector. On top of the fusing layer, the **deep interaction layer** extracts the correlations between these features, and outputs a compact fused feature vector for prediction. Finally, the **prediction**

layer takes the fused feature vector as input to predict the rating of the user on the target item.

Comparing to existing review-based recommender systems, our proposed model, AAMN, captures static features of users (items), which is helpful to boost the quality of semantic features extracted from reviews. Moreover, static features make use of the collaborative relationship between user and user (or item and item), and thus provide an adaptive mechanism for efficiently utilizing reviews in semantic modeling. That said, when facing different items, the historical review and rated items of a user are to be considered differently. Note that, to store historical reviews and rated items, the *memory network* [25] is utilized in our model. Here the memory network can be interpreted as a memory store of the historical information. These core ideas make our model more effective than the state-of-the-arts, as demonstrated in Section IV. In what follows, we proceed to introduce the details of each component of our model.

B. INPUTTING LAYER

This layer takes the ID of target user u, historical rated items of u, reviews attached with ratings of u, the ID of target item i, historical rating users of i, and reviews attached with ratings of i as inputs. It transforms the IDs of user u and item i to one-hot encodings. The historical rated items of user u are transformed to a multi-hot encoding with $|\mathcal{I}|$ dimensions, where the i-th dimension takes 1 if u has rated on item i, and 0 otherwise. Similarly, the historical rating users of item i are transformed into a multi-hot encoding with $|\mathcal{U}|$ dimensions, where the u-th dimension takes 1 if i has been rated by user u, and 0 otherwise.

C. EMBEDDING LAYER

This layer transforms the sparse vectors in the input layer to low-dimension dense vectors to represent underlying features within the original sparse vectors in a compact way. Specifically, the one-hot encodings of user u and item i from the input layer are projected to a dense vector, i.e., the static features $\mathbf{x}_u$ and $\mathbf{y}_i$, respectively, as defined in Definition 1. Here, the vectors could be initialized randomly or with results of matrix factorization, then updated with model training. The static features capture the stationary property of each user/item. The review written by a user u on an item i is projected into a review encoding $\mathbf{w}_{u,i}$ defined in Definition 2. In addition, the historical rated items of user u are projected into a dense matrix where the i-th row is the static feature of item i, i.e., $\mathbf{y}_i$. Similarly, the historical rating users of item i are projected into a dense matrix where the u-th row is the static feature of user u, i.e., $\mathbf{x}_u$.

D. ATTENTION LAYER

Attention layer is the core of AAMN. It captures the adaptive semantic features and adaptive property features from the reviews and historical user-item interactions, respectively. To extract the adaptive semantic feature for a user u given a target item i, the idea is to find out the most relevant reviews to item i for constructing a feature vector for user u. Let $\{\mathbf{w}_{u,i_1}, \mathbf{w}_{u,i_2}, .., \mathbf{w}_{u,i_p}\}$ be the embeddings of u's reviews. Specifically, for a target item i, we assign an attention weight $a_{i_k,i}$ for each review embedding $\mathbf{w}_{u,i_k}$ by comparing its embedding and item i's embedding:

$$a_{i_k,i} = \frac{\exp(\mathbf{w}_{u,i_k}\mathbf{y}_i)}{\sum_{l=1}^{p}\exp(\mathbf{w}_{u,i_l}\mathbf{y}_{i_l})}, \tag{2}$$

where $\mathbf{y}_i$ is the static feature of target item i, the inner product between $\mathbf{w}_{u,i_k}$ and $\mathbf{y}_i$ is adopted to calculate their correlation. Then, the adaptive semantic feature of u given i, i.e., $\mathbf{s}_{u|i}$ is obtained by the weighted sum of the user's reviews:

$$\mathbf{s}_{u|i} = \sum_{k=1}^{p} a_{i_k,i} \cdot \mathbf{w}_{u,i_k} \tag{3}$$

The adaptive semantic feature for target item i given user u, i.e., $\mathbf{s}_{i|u}$, is computed in a similar way. To compute user's adaptive property feature, let $\{\mathbf{y}_{u,i_1}, \mathbf{y}_{u,i_2}, .., \mathbf{y}_{u,i_p}\}$ be the embeddings of u's historical rated items. For each historical rated item i_k, we assign a weight a'_{i_k} as the importance of the item by considering the rating r_{u,i_k}:

$$a'_{i_k} = \frac{\exp(f(r_{u,i_k}))}{\sum_{l=1}^{p}\exp(f(r_{u,i_l}))} \tag{4}$$

where $f(\cdot)$ is a transformation function applied to the ratings, we consider six types of transformation functions, shown in Table 3. Intuitively, *Base* directly use the percentage of each rating in sum of all ratings as the weight; *Softmax* transforms the rating into the exponential space to calculate the weight; *Unbias Softmax* transforms the rating by subtracting the average rating r_u^{avg} before passing into a softmax function; *Abs-unbias* is based on *Base*, it transforms the rating by calculating the absolute difference between the rating of i_k and the average rating; *Abs-unbias Softmax* extends *Abs-unbias* by applying a softmax function to the absolute difference between the rating of i_k and the average rating; *No Rating* considers the relationship between embeddings of historical rated items and the target item with a Multi-Layer Perception, where $\odot$ denotes the element-wise product.

TABLE 3. Six types of rating transformation function.

Name	Form
Base	$f(x) = \frac{r_{u,i_k}}{\sum_{i'_k \in \mathcal{I}_u} r_{u,i'_k}}$
Softmax	$f(x) = \frac{\exp(r_{u,i_k})}{\sum_{i'_k \in \mathcal{I}_u} \exp(r_{u,i'_k})}$
Unbias Softmax	$f(x) = \frac{\exp(r_{u,i_k} - r_u^{avg})}{\sum_{i'_k \in \mathcal{I}_u} \exp(r_{u,i'_k} - r_u^{avg})}$
Abs-unbias	$f(x) = \frac{\lvert r_{u,i_k} - r_u^{avg}\rvert}{\sum_{i'_k \in \mathcal{I}_u} \lvert r_{u,i'_k} - r_u^{avg}\rvert}$
Abs-unbias Softmax	$f(x) = \frac{\exp(\lvert r_{u,i_k} - r_u^{avg}\rvert)}{\sum_{i'_k \in \mathcal{I}_u} \exp(\lvert r_{u,i'_k} - r_u^{avg}\rvert)}$
No Rating	$f(x) = \frac{\exp(\sigma(W_t \cdot (\mathbf{y}_{i_k} \odot \mathbf{y_i}) + b_t))}{\sum_{i'_k \in \mathcal{I}_u} \exp(\sigma(W_t \cdot (\mathbf{y}_{i'_k} \odot \mathbf{y_i}) + b_t))}$

Given the weights of each rated item i_k, the adaptive property feature of a user u given item i is obtained by:

$$\mathbf{x}'_{u|i} = \sum_{k=1}^{p} a'_{i_k} \cdot \mathbf{y}_{i_k}, \tag{5}$$

where $\mathbf{y}_{i_k}$ means the history interactive item of u. The adaptive property feature of item i given user u, i.e., $\mathbf{y}'_{i|u}$ is computed in a similar way.

E. FUSING LAYER

This layer combines the three types of feature: static feature, adaptive semantic feature and adaptive property feature of user (item) to construct a single dense vector. We apply a fully connected network to condense the three types of features into a d-dimensional fused feature vector. The fused feature for the user u is calculated as follows.

$$\mathbf{x}''_u = f(W_f \cdot (\mathbf{x}_u \oplus \mathbf{s}_{u|i} \oplus \mathbf{x}'_{u|i}) + b_f) \tag{6}$$

where W_f and b_f are parameters in fully connected layer, $\oplus$ is the concatenation operation between vectors. Without ambiguity, we omit the target item i in the notation of $\mathbf{x}''_u$ for ease of presentation. The calculation of the fused feature of item i, i.e., $\mathbf{y}''_i$, is in a similar way.

F. DEEP INTERACTION LAYER

This layer captures the interactions between user and item features. On top of the fusing layer, we compute the outer product of $\mathbf{x}''_u$ and $\mathbf{y}''_i$ to obtain the interaction matrix M:

$$\mathbf{O} = \mathbf{x}''_u \otimes \mathbf{y}''_i = \mathbf{x}''_u \mathbf{y}''^T_i \tag{7}$$

where $\mathbf{O}$ is a $d \times d$ matrix, each element in O is evaluated as: $O_{d_1,d_2} = x''_{u,d_1} y''_{i,d_2}$. To capture the local relationship between neighboring embedding dimensions, a stack of hidden layers are applied over the interaction matrix M. We define the stack of hidden layers as a non-linear transformation function $\mathbf{o} = f_\Theta(\mathbf{O})$ with parameters Θ, and outputs $\mathbf{o}$. In this paper, we use convolution neural network to extract local correlations from the interaction matrix, because it requires much fewer parameters than other models (i.e., MLP) while has competitive performance. we adopt $log_2(d_k)$ layers of CNN, and set the stride of the CNN to 2 and the number of kernels to d_k, so that each time we reduce the size of the feature map by half. The output of the CNN is a matrix of size $1 \times 1 \times d_k$, which can be reduced to an output vector $\mathbf{o}$ of length d_k. The parameter selection of d_k could be found in Section IV-C.

The reason we use outer product to further model the interactions between the features is twofold: on one hand, the result of outer product could consider fine-grained embeddings' interaction between features, particularly for features containing complicate semantics. While, matrix factorization only considers corresponding-position embeddings' product in features, ignoring correlations between different position embeddings of features. Similarly, concatenation operation treats the embeddings of features independently, combining them without modeling their correlation. In fact, outer product could subsume MF and concatenation operation. On the other hand, it's beneficial for deep learning model to extract more potential features, since outer product explicitly models the interaction between elements of different features, especially for sparse data with high dimension.

G. PREDICTION LAYER

This layer accepts $\mathbf{o}$ as input. It adopts a factorization machine (FM) to predict the ratings. FM accepts a real-valued feature vector and models the pairwise interactions between features using factorized parameters. The FM function is defined as follows:

$$F(\mathbf{o}) = w_0 + \sum_{i=1}^{n} w_i x_i + \sum_{i=1}^{n} \sum_{j=i+1}^{n} \langle v_i, v_j \rangle o_i o_j \tag{8}$$

where $\mathbf{o} \in R^k$ is a real-valued input feature vector. $\langle ., . \rangle$ is the dot product. The parameters $\{v_1, \ldots, v_n\}$ are factorized parameters (vectors of $v \in \mathbb{R}^k$) used to model pairwise interactions (o_i, o_j). $w_0, \ldots, w_n$ are learnable weights, where w_0 is the global bias and w_i is linear coefficients of the i-th component of $\mathbf{o}$. The output of $F(\cdot)$ is a scalar, representing the strength of the user-item interaction. The network is trained end-to-end by minimizing the standard Mean Squared Error (MSE) loss following [21].

H. TRAINING AND TIME COMPLEXITY

In our model, the training method we adopt is stochastic gradient descent algorithm, a generic solver for neural network models. As most machine learning toolkits (e.g., Tensorflow, PyTorch, Theano etc.) provide the function of automatic differentiation, we omit the derivation of the update functions of our model. Specifically, in each epoch, all observed samples are first shuffled, and then sampled as a mini-batch in a sequential way. In training process, recent works mostly utilized random sampling, which treats each samples equally. However, in fact the importance of different samples are unequal [43], [45], [47]. Especially, the samples owning larger error between model's prediction values and ground truth, would play more important roles in performance evaluation. Maybe the training model is still under-fitting for these samples. Therefore, we utilize a dynamic sampling method. After each training epoch, we recalculate a weight for each sample. The weights depend on the prediction error of each sample in last epoch. The weight of a sample review written by user u on item i is calculated as:

$$\frac{\exp(|r_{u,i} - \hat{r}_{u,i}|)}{\sum_{(u',i') \in \mathcal{D}} \exp(|r_{u',i'} - \hat{r}_{u',i'}|)}. \tag{9}$$

The training process of AAMN is summarized in Algorithm 1. The time complexity for each dynamic sampling gradient step is $O(M \times d^2)$, where M denotes the number of historical rated items, which is often a constant and typically smaller than 100; d is the feature dimension, typically smaller than 100. We assume that our model needs N samples to reach convergence, thus the overall time complexity is

Algorithm 1 AAMN

```
Input: each sample $(u, i, \mathcal{W}_u, \mathcal{W}_i, w_{u,i}, r_{u,i})$
Output: $\Theta = \{\mathbf{x}, \mathbf{y}, \mathbf{v}, \Theta_{MLP}, \Theta_{CNN}\}$
procedure Train($D_{train}$)
Randomly initialize each $x_u, y_i$ with Gaussian distribution;
while not converged do
    for index in $|D_{train}|$ do
        Sampling $(\mathcal{W}_u, \mathcal{W}_i, w_{u,i})$ from $D_{train}$
        Compute embedding $\mathbf{w}_{u,i_k}$ for each review $w_{u,i_k}$ in $\mathcal{W}_u$
        Compute Adaptive semantic feature $\mathbf{s}_u$ for user $u$
        Compute embedding $\mathbf{w}_{u_k,i}$ for each review $w_{u_k,i}$ in $\mathcal{W}_i$
        Compute $\mathbf{s}_i, \mathbf{x}'_u, \mathbf{y}'_i$
        Construct historical feature $\{\mathbf{y}_{i_1}, \mathbf{y}_{i_2} \dots \mathbf{y}_{i_p}\}$ for user $u$, and $\{\mathbf{x}_{u_1}, \mathbf{x}_{u_2} \dots \mathbf{x}_{u_q}\}$ for item $i$
        Fusing $\mathbf{s}_u$, $\mathbf{x}'_u$ and $x_u$ into $x''_u$ with MLP, and fusing $\mathbf{s}_i$, $\mathbf{y}'_i$ and $y_i$ into $y''_i$ with MLP
        Construct interaction Matrix $O$ with $x''_u$ and $y''_i$
        Compute interaction feature $\mathbf{o}_{u,i}$ with CNN
        Prediction rating $\hat{r}_{u,i}$ with $\mathbf{o}_{u,i}$ by FM
        Compute loss between $\hat{r}_{u,i}$ and ground truth $r_{u,i}$
        Update $\Theta$ to minimize $loss_{train}$
    end for
    Recalculate sampling weight for each sample
end while
```

$O(N \times M \times d^2)$. Since GPU is usually utilized to accelerate the training by parallel computation and batch-size training, our model is efficient in training in practice.

IV. EXPERIMENTS

In this section, we study the performance of our proposed model. Our experiments are designed to answer the following research questions (RQs):

- **RQ1**: Does our model outperforms state-of-the-art models such as DeepCoNN, TransNet, D-ATT, MPCN and A^{3}NCF?
- **RQ2**: How much does each main part of our model affect the performance?
- **RQ3**: Whether our model can work well under different situations in terms of sparsity?
- **RQ4**: How about the convergence of our model?
- **RQ5**: Can our model learn meaningful attention weights for historical reviews and items (users)?

In what follows, we first describe the detailed experimental settings (Section IV-A), and then discuss and analyze the experimental results (Section IV-C∼IV-G).

A. EXPERIMENTAL SETUP

Datasets: We used 16 datasets, which are from two sources:

1) *Yelp dataset challenge.*[2] Yelp is an online review platform for business such as restaurants, bars, SPAs, etc. We used the Yelp dataset from the latest challenge; it contains 1M items, 2M users and more than 3M reviews with ratings.
2) *Amazon Product Reviews.*[3] Amazon is a well-known E-commerce platform. Users are able to write reviews for products they have purchased. We used 15 datasets from the Amazon product review corpus, including: Digital Music, Video Games, Clothing, Electronics and so on.

TABLE 4. The Statistics of Datasets.

Class	Users	Items	Ratings	Sparsity
Yelp	220,999	123,897	3,396,015	99.98%
Digital Music	5,541	3,568	64,706	99.67%
Clothing	39,387	23,033	278,677	99.96%
Video Games	24,303	10,672	231,780	99.91%
Electronics	192,403	63,001	1,689,188	99.98%
Gourmet Food	58,901	28,231	544,239	99.96%
Instant Video	4,902	1,683	36,486	99.55%
Kindle Store	68,223	61,934	982,619	99.97%
Patio / Lawn	1,672	962	13,077	99.18%
Pet Supplies	18,070	8,508	155,692	99.89%
Sports / Outdoors	31,176	18,355	293,306	99.94%
Toys / Games	17,692	11,924	166,180	99.92%
Movies / TV	2,088,428	200,915	4,606,671	99.99%
Baby	17,177	7,047	158,311	99.86%
Office Products	4,798	2,419	52,673	99.54%
Beauty	19,766	12,100	196,325	99.92%

The statistics of these datasets are summarized in Table 4. We divided each dataset shown in Table 4 into three parts: training set, validation set and test set. Following the setting in [12], [28]–[30], a time-based split is utilized, the last one of each user's history items is treated as the test set, the penultimate is set for validation set, and the residuals are considered as the training set.

Baselines: We compared our methods against the following competitive methods.

- *DeepCoNN* [22]. Deep Co-operative Neural Network is a review-based convolutional recommendation model. It trains convolutional representations of user and item, and passes the concatenated embedding into an FM model.
- *TransNet* [29]. It is an improved version of DeepCoNN. It incorporates transformation layers and additional training step that enforces the transformation representation to be similar with the embedding of the actual target review.
- *D-ATT* [31]. Dual Attention CNN model is a CNN-based model that uses reviews for recommendation. This model is characterized by its usage, which is related to two forms of attentions (i.e., local and global). A final user (item) representation is learned by concatenating representations learned from both local and global

[2] https://www.yelp.com/dataset/challenge

[3] http://jmcauley.ucsd.edu/data/amazon/

TABLE 5. Performance comparison (mean squared error) on 16 datasets. The best performance is in boldface. △DC, △TN, △DA, △MP, △AN are the relative improvements (%) of AAMN over DeepCoNN, TransNet, D-ATT and A^3NCF.

	Competitive model					Our model	Improvement(%)				
Dataset	DeepCoNN	Transnet	D-ATT	MPCN	A^3NCF	AAMN	△ DC	△ TN	△ DA	△ MP	△ AN
Yelp	1.385	1.363	1.428	1.349	1.327	**1.309**	5.4	3.8	8.2	2.9	1.28
Digital Music	1.202	1.004	1.000	0.970	0.985	**0.802**	32.4	19.1	18.7	16.2	17.5
Clothing	1.322	1.197	1.207	1.187	1.171	**1.108**	16.1	7.4	8.2	6.6	5.3
Video Games	1.307	1.276	1.269	1.257	1.238	**1.128**	13.1	11.0	10.5	9.7	8.3
Electronics	1.372	1.365	1.368	1.350	1.334	**1.209**	11.3	10.9	11.1	9.9	8.8
Gourmet Food	1.199	1.129	1.143	1.125	1.076	**0.965**	16.1	10.9	12.0	10.6	6.5
Instant Video	1.285	1.007	1.004	0.997	0.975	**0.948**	25.6	5.1	4.8	4.1	1.9
Kindle Store	0.823	0.797	0.813	0.775	0.756	**0.624**	23.6	21.1	22.6	18.8	16.8
Patio / Lawn	1.534	1.123	1.037	**1.005**	1.082	1.017	33.7	9.4	1.9	-0.6	6.0
Pet Supplies	1.447	1.346	1.337	1.328	1.334	**1.234**	14.1	7.6	7.0	6.4	6.8
Sports / Outdoors	1.039	0.994	0.990	0.980	0.884	**0.859**	16.4	12.6	12.2	11.3	1.7
Toys / Games	1.057	0.974	0.982	0.973	0.908	**0.813**	23.3	16.7	17.4	16.6	10.7
Movies / TV	1.960	1.176	1.187	1.144	1.123	**1.029**	47.6	12.8	13.6	10.3	8.6
Baby	1.440	1.338	1.325	1.304	1.176	**1.159**	19.6	13.5	12.6	11.2	1.5
Office Products	0.909	0.840	0.805	0.779	0.768	**0.726**	20.6	14.0	10.3	7.3	6.0
Beauty	1.453	1.404	1.409	1.387	1.327	**1.194**	17.3	14.5	14.8	13.4	9.5
Average △	–	–	–	–	–	–	21.0	11.9	11.6	9.6	7.2

attentions. The dot product between user and item representations is then used to estimate the rating score.

- *MPCN* [28]. Multi-Pointer Co-attention Networks is another model that uses reviews for recommendation. The model uses the co-attention mechanism to find the relevance between each pair of user comments and item reviews, and it extracts useful features from the selected reviews. It can filter useless information and improve recommendation.
- *A^3NCF* [23]. Adaptive Aspect Attention Neural Collaborative Filtering model is a topic model, which extracts user preferences and item characteristics from review texts. This model guides the process of representation learning, and captures a user's special attention on each aspect of the targeted item, using an attention network.

Evaluation Metric: The well-known evaluation metric on rating prediction is the Mean Squared Error (MSE), which measures the square error between the rating prediction and ground truth. Usually, a lower MSE indicates a better performance [22], [23], [29]. Following prior works [23], [28], [31], we also used the MSE to evaluation the performance of the proposed method. The MSE is computed as:

$$MSE = \frac{1}{|\mathcal{D}|} \sum_{(u,i)\in\mathcal{D}} (r_{u,i} - \hat{r}_{u,i})^2 \tag{10}$$

where $r_{u,i}$ is the ground truth rating of u to i, $\hat{r}_{u,i}$ is the predicted rating, and $\mathcal{D}$ is the corresponding dataset.

Other Settings: We summarize other settings in our experiments as follows. **(1)** In review pre-processing phase, all reviews were first passed through a Stanford Core NLP Tokenizer [28] to obtain the tokens, which were then lowercased. **(2)** Stop words (e.g., *the, is, and*) and punctuations were considered as separate tokens, and they were retained. **(3)** Words with less than 10 occurrences were filtered out; moreover, we set the maximum review length to 100 words, since few reviews are much longer than 100 words. **(4)** Skip-gram model [5] was utilized to generate a d-dimensional word2vec [32] embedding. **(5)** As for the prediction layer FM, the number of factors is set as 10. We set the size of word embedding layer as 64. For the outer product, the dimension of user feature and item feature are set as 64, which results in a 64 × 64 feature matrix. For dropout layers, the keep probability is 0.5. **(6)** In order to obtain interactive information of user products, we use 6 stacked CNN layers. The padding of CNN is set as *SAME*. The convolution kernel is set as 2 × 2; the step size is 2. The non-linearity activation function, α, is set as *ReLU*. **(7)** For compared models such as DeepCoNN, TransNet, D-ATT and MPCN, their parameters are set as *default* in their open source codes with the best performance. **(8)** Our model is trained with Adam, in which the initial learning rate is set as 10^{-3}. We trained all models for a maximum of 20 epochs with early stopping, and recorded the results with the average of five times running on the test set.

B. COMPARING WITH STATE-OF-THE-ARTS

Table 5 reports the experimental results comparing with the baselines. **Firstly**, we observe that our model, AAMN, achieves the best performance on 15 out of 16 datasets. For the Patio/Lawn dataset, even if AAMN cannot beat all these methods, it is still ranked in Top-2, and the performance gap is pretty small (about −0.6%). We observe that the number of items (users) on this dataset is relatively small, compared against other datasets. This could be the reason why our model cannot exhibit the best performance on this dataset, since few items (users) provide insufficient information of static features. Nevertheless, we can see that, on most of these datasets our model significantly outperforms DeepCoNN, TransNet, D-ATT, MPCN and A^3NCF, which are all recent competitive review-based methods for recommendation. For example, on the Digital Music dataset the relative improvement is very encouraging, it achieves gains of up to 32.4% (DeepCoNN), 19.1% (TransNet), 18.7% (D-ATT), 16.2% (MPCN) and 17.5% (A^3NCF). These results demonstrate the efficiency and competitiveness of our proposed model; meanwhile, it also answers **RQ1** positively.

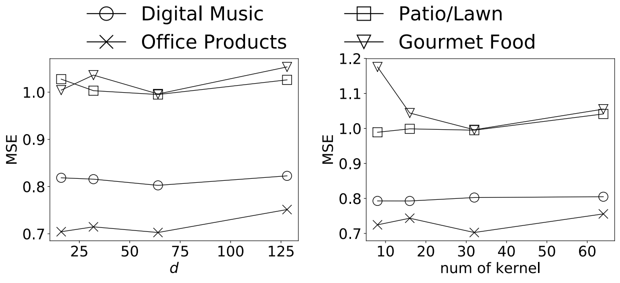
Secondly, although both MPCN and A^{3}NCF consider the attention module, our model beats them on almost all these datasets. The improvement of AAMN demonstrates the benefit of incorporating the static features to guide the attention module. Moreover, these two recent models only utilize the semantic features, but they ignore the features from a global view in the prediction phase. It is also the superiority of AAMN, which not only utilizes the adaptive semantic features with the attention mechanism and the static features, but also combines the adaptive property feature in prediction phase.

Thirdly, on the majority of these datasets, the performance has improved around 10% compared with the other models. Notably, the average improvement of AAMN over DeepCoNN, TransNet and D-ATT is over 10%. The average gains of AAMN over MPCN is nearly 10%. The average gains of AAMN over A^{3}NCF is more than 7.2%. These results show that our model, which incorporates static features, adaptive semantic features, adaptive property features and dynamic sampling, can improve performance effectively and significantly. Notably, we have conducted significant test on the improvement. Each model are run by 5 times independently, and we get averages of the results, which are statistically significant with p-values<0.05 using two-tailed paired test. So the statistical test could verify significance of the results.

C. IMPACT OF IMPORTANT PARAMETERS

This section studied the impact of several most important parameters in our model (including transformation function f, each feature embedding's dimension size d, and convolution kernel number d_k). We used four representative datasets, including Digital Music, Patio/Lawn, Office Products and Gourmet Food.

Impact of d: We studied the impact of feature embedding dimension size by varying d from 16 to 128. Fig. 2 shows the experimental results. We can find that, when dimension d increases, the overall trends on different datasets are similar. This implies that a too small dimension size may have a weak expressive ability, incurring a poor performance. On the other hand, when the dimension exceeds a threshold, the performance keeps decreasing. This is because when d is too large, the complicated parameters in our model cannot be well trained, incurring overfitting. Furthermore, we observed that, for all these datasets our model achieves the best performance when d approximates to 64. In the later experiments, we set $d = 64$, unless stated otherwise.



(a) MSE v.s. d (b) MSE v.s. d_k

FIGURE 2. Effect of parameters.

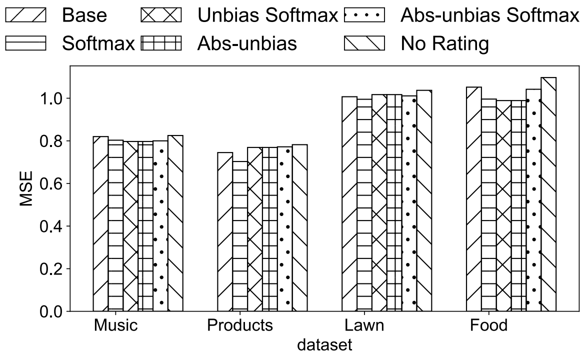


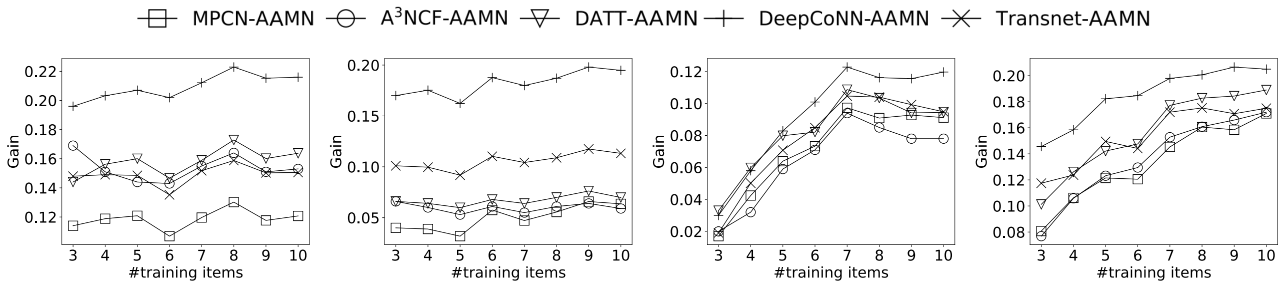
FIGURE 3. Effect of different transformation functions.

Impact of d_k: We studied the impact of the convolution kernel number by varying d_k from 4 to 64. Fig. 2(b) shows the results. The curves in this figure show different trends on different datasets. Especially, on the Patio/Lawn and Digital Music datasets, the curves change sharply when d_k changes. This indicates that parameter d_k is much more sensitive on these two datasets, compared with the other two datasets. Nevertheless, we observe that, when d_k approximates to 32, our model achieves the best performance for all these datasets. In the later experiments we set the kernel number of CNN to 32, unless stated otherwise.

Impact of f: To investigate the impact of different rating transformation functions, we tested a set of rating transformation functions including *Base*, *Softmax*, *Unbias Softmax*, *Abs-unbias*, *Abs-unbias Softmax*, *No Rating* (*cf.,* Table 3). Figure 3 shows the results of different transformation functions. Although the performance is not the same in different datasets, we find *No Rating* is the worst one in all these functions. It proves that the rating is beneficial to distinguish the importance of historical items. Besides, function *Softmax* almost achieves the best performance on all these datasets. In the later experiments, we adopted the *Softmax* function as the default transformation function, unless stated otherwise.

D. ABLATION STUDY

To answer **RQ2**, we study the impacts of each major module of AAMN on the performance. We report the results of several different variations. In the default setting, we use the standard model with all components: static features, adaptive semantic features, adaptive property features of user (item), deep interaction layer, and dynamic sampling. We set our model as case (1) for comparison. In case (2) we remove the deep interaction layer, which includes the outer product and CNN. In case (3) we remove the adaptive semantic features of user (item) by cancelling the attention module on semantic



(a) Digital Music (b) Office Products (c) Patio/Lawn (d) Gourmet Food

FIGURE 4. MSE in different sparsity values.

TABLE 6. Ablation analysis (MSE) on four datasets, the best performance is in boldface.

Architecture	Digital Music	Clothing	Video Game	Electronics	Avg. ▽
(1) Default	**0.802**	**1.108**	**1.128**	**1.209**	–
(2) Remove outer product and CNN	0.869	1.120	1.136	1.229	2.3%
(3) Remove adaptive semantic feature	0.872	1.166	1.207	1.302	6.4%
(4) Remove static feature	0.856	1.163	1.203	1.300	5.8%
(5) Remove adaptive property feature	0.816	1.145	1.158	1.217	1.47%
(6) Remove dynamic sampling	0.822	1.116	1.147	1.232	1.06%

features, but keep the static features and semantic features. In this variant, each review is treated equally. In case (4) we remove the static features of user (item) from the fusing layer, but retain adaptive semantic features of user (item). In case (5) we remove the adaptive property features of user (item), in which the model cannot use the historical records of user (item). In case (6) the dynamic sampling is removed.

Table 6 reports an ablation study conducted on four representative datasets: Digital Music, Clothing, Video Game and Electronics. **Firstly**, by removing the outer product with CNN, it leads to a performance degradation (about 2.3% on average). This implies that the deep interaction layer is feasible and effective. **Secondly**, removing the static feature of user (item) also hurts the performance (about 5.8% on average). This implies that the static feature of user (item) is helpful to improve the performance of the model. Notably, removing the adaptive semantic feature of user (item) hurts the performance more seriously (about 6.4% on average). It indicates that adaptive semantic feature is more effective for model performance. We may need to mention that, the adaptive semantic feature is obtained via static feature, this further implies that the static feature of user (item) plays an important role in extracting information from reviews. **Thirdly**, when removing the adaptive property feature, we find that the performance also degenerates (about 1.47% on average). Nevertheless, compared to the degeneration when removing adaptive semantic feature, the reduction is not obvious. This also proves the adaptive semantic feature is much more important than the adaptive property feature. **Fourthly**, when removing the dynamic sampling, one can find that the performance also degenerates on all these datasets, and the reduction is close to that of removing the adaptive property feature. This demonstrates the dynamic sampling is also feasible and effective in improving the performance, although it may not achieve a big improvement as the adaptive semantic feature does.

E. DATA SPARSITY

To answer **RQ3**, this part investigated the performance under different situations in terms of sparsity.[4] We varied the sparsity from 3 to 10. For ease of observing the performance gap, we utilized the baseline models' MSE to subtract AAMN's. This can be also called the performance gain in terms of MSE.

Fig. 4 shows the experimental results. One can find that, the performance gains are positive values on all these datasets. This shows us that our model can achieve a more outstanding performance on the sparsity situation. On the Digital Music and Office Products datasets, the model achieves a steady improvement on different sparsity values. Particularly, on the Patio/Lawn and Gourmet Food datasets, the performance of the model turns better and better, when the sparsity decreases. All these results demonstrate that, our model AAMN can work very well in the sparse situation. This nice performance of our model is mainly ascribed to two points: (i) the review text is incorporated to mitigate user's sparse behaviours, and (ii) user's static feature is used, which benefits to extracting key feature from reviews.

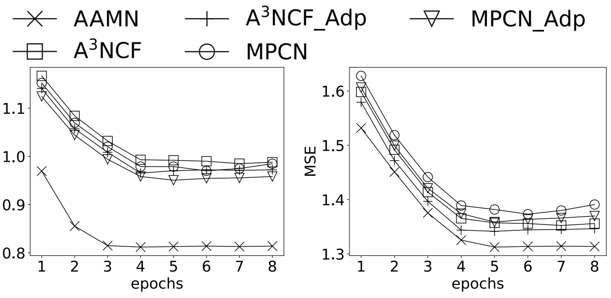
F. CONVERGENCE

To answer **RQ4**, we studied the the convergence of AAMN. We used two typical datasets: one is a large-scale dataset (Yelp), the other is a small-scale dataset from Amazon Product Reviews (i.e., Digital Music). With them, it allows us to

[4] Here, the sparsity means the number of user's rated items, the less of the number, the much sparser it is.

TABLE 7. Attention weights for user *u*'s reviews.

from	ID	review	rating	weight a	weight a'
user u	r_{u,i_1}	I gave these Black Cubic Zirconia ***Earrings*** as a gift to my step-daughter after she received her job promotion. They are really ***beautiful*** and ***sturdy***. I will order from this seller again.	5.0	0.8	0.42
	r_{u,i_2}	Lilyette uses an excellent design for their minimizer bras. In addition, the effectiveness to minimize is fantastic. The straps are comfortable.	4.0	0.05	0.16
	r_{u,i_3}	I am so pleased to have purchased these 925 Sterling Silver ***Earrings***. I plan to order another pair for my daughter.	5.0	0.15	0.42
item i	$r_{u_1,i}$	... What a nice case to sort my collection. I love all the compartments and the layout too ... It will please even the most fussy, as I am. The price is fair ... Seems well made too ... This is for serious ***jewelry lover's***. Buy it.....	5.0	0.61	0.67
	$r_{u_2,i}$	...It arrived with some of the leather scratched... This particular color pink also looked pretty bad in person. The quality of this box seemed lesser than the brown one I own...	3.0	0.11	0.09
	$r_{u_3,i}$	Really nice box, a bit cheaply made. It is made of carboard with pleather ...I'm not so sure how it will stand up to my tween daughter's use. Every inch is useable space for ***jewelry*** & very compact, which is a nice feature.	4.0	0.28	0.24



(a) Music (b) Yelp

FIGURE 5. Convergence comparison on different datasets.

easily observe model's convergence at different data scales. At the same time, to prove the advantage of introducing dynamic sampling methods on convergence, we also compared the convergence of each baseline in the case with the dynamic sampling and that without the dynamic sampling.

Fig. 5 shows the experimental results. We can find that each of these models reach convergence after several epochs, even though on a large-scale dataset. Also, we can see that the baselines containing the dynamic sampling converges at a lower MSE within a fewer epochs, compared with those baselines without the dynamic sampling. This further demonstrates the effectiveness of dynamic sampling. Furthermore, we can observe that our model AAMN converges faster than most of the models. This demonstrates the superiority of our model, from another perspective.

G. CASE STUDY OF ATTENTION WEIGHTS

This section revisits the running example in Table 1 and makes a deeper investigation on our AAMN model, in order to answer **RQ5**. We compute the weights of different reviews based on static features of item i and user u, respectively.

Table 7 shows the weights a of each review and weights a' of each historical item for user u or (each historical user for item i); these weights are calculated automatically by our model AAMN. We can see from Table 7 that the target item i's static feature is highly correlated with *the jewelry*, since they usually are bought together by users. Therefore, i's static feature is helpful to find highly related historical reviews of target user u. Since reviews r_{u,i_1} and r_{u,i_3} are related to *jewelry*, their weights are higher compared to r_{u,i_2}. Moreover, review r_{u,i_1} obtains a higher weight $a_{i_1,i} = 0.8$, since r_{u,i_1} mentions more words, i.e., *beautiful* and *sturdy*, which are related to target item i's static feature. In a similar way, the target user u's static feature is related to *jewelry lovers* who might have bought the target item, because they both like to buy *jewelry* or *having daughter*. Hence, user's static feature could help to obtain the appropriate weights for each items. Meanwhile, weight a' obtained by rating gives a different importance of historical rated items. As we can see, the weights a' for the rated items of user u are different from the weights a of u. Similar phenomenon can be observed for the weights of item i.

To summarize, AAMN can uncover the importance of different reviews effectively. In addition, the attention weights for reviews and user-item interactions capture different characteristics of users (items).

V. RELATED WORK

We review recent works highly related to our paper. These works can be divided into three categories (Sections V-A∼V-C).

A. REVIEW-BASED RECOMMENDATION

Recent works [6], [12], [14], [17], [18], [37], [42]–[46] begin to use user's reviews to items to predict rating for unrated items. Since the user-item interaction data (rating or click) is often sparse, it is challenging to accurately learn the representation of user preferences and item features. Nevertheless, with the abundant semantic information of reviews, data sparsity problem can be alleviated to some extent. At first, topic model is utilized to extract semantic information from reviews. For example, McAuley and Leskovec [12] used a defined transform function to learn latent topics together with user's (or item's) latent factors. Recently, neural network models were utilized to extract more complicated semantic feature [22], [29], [32]. For example, Zheng *et al.* [22]

utilized two parallel neural networks to learn user preferences and item features, respectively. However, their model ignores interactions between the user's and the item's reviews.

Some researchers utilized attention mechanism [7] to exploit the reviews further [21], [23], [28], [47]. For example, Chen *et al.* [47] utilized attention mechanism and focused on category of behaviors. Seo *et al.* [21] leveraged global and local attention to the whole review text and user's (item's) properties, respectively. Cheng *et al.* [23] deeply exploited user's varying aspect attention to different items. Tay *et al.* [28] operated on a multi-hierarchical paradigm to pay different attentions on reviews and words. Yuan *et al.* [49] mainly focused on exploiting inherent correlation among users/items to obtain more information with similar users/items. While, our work pays attention on user's static feature and semantic feature simultaneously, further considering the mutual influence between the two kinds of features. Besides, Chen *et al.* [50] exploited reviews' usefulness, Xia *et al.* [51] collected user's (item's) review together with Bi-directional Gate Recurrent Unit (GRU) and attention layer. But both of them ignored the mutual influence between user (user's review) and item (item's review)In these works, though semantic features of reviews are explored sufficiently with various attention-based methods, the static features of user and item are ignored. Static features reflect user's (item's) long-term and stationary property, which are able to guide the extraction of adaptive features, benefiting to better performance, as demonstrated in our experiments.

B. OTHER CONTENT-BASED RECOMMENDATION

Besides review information, other types of item content [18], [26] are used for recommendation [1], [9], [11], [38], [39]. For example, Grossetti *et al.* [48] presented a recommendation model based on a similarity graph which exploits homophily for propagation of probabilities. Tang *et al.* [1] modelled the long range dependence in recommender systems. Taniskidou *et al.* [3] discussed different representations of unstructure text that affect recommendation performance. Li *et al.* [18] proposed a Bayesian generative model called collaborative variational autoencoder for multimedia scenario, which can learn deep latent representation from content data in an unsupervised manner. Wang *et al.* [26] generalized recent advances in deep learning from i.i.d input to non-i.i.d. input, and proposed a hierarchical Bayesian model, which jointly performs deep representation learning for the content information and ratings. These methods focused on a lateral view of item, and extracted semantic features in an independent way. The interactions between user and item are not considered.

C. DEEP LEARNING FOR RECOMMENDATION

Deep learning models have been widely used in computer vision [30], natural language processing [15], [16], etc.

Recently, deep learning models were also applied to recommendation [24], [26], [34], [40], [41]. For example, He *et al.* [24], [34] generalized matrix factorization to deep feature extraction with multiple layer perceptron and Convolutional Neural Network (CNN), respectively, while the models just apply deep learning on user-item interaction data. On the other hand, different deep learning models such as Recurrent Neural Network (RNN) [35], CNN [22], Stacked Denoising Auto Encoder (SDAE) [26], were applied to distill review feature. For instance, Almahairi *et al.* [26], [35] utilized some sort of regularization terms to bridge two types of features. Zheng *et al.* [22] exploited review feature without considering user's (item's) static feature [10]. All these models could learn the independent features well, while ignoring the deep interactions between review features and static features. It requires a more well-designed model to learn the complicated interactions among features of users and items. Difference from the aforementioned existing approaches, we propose to extract the adaptive features of users (items) according to the static features. In addition, our model exploits the deep interactions between user and item features.

In this branch, Multi-Pointer Co-attention Networks (MPCN) [28] and Adaptive Aspect Attention Neural Collaborative Filtering (A^{3}NCF) [23] could be the most related to our work. Nevertheless, our work is different from them in several aspects. (i) In MPCN, user's (item's) global property, which depicts user's (item's) long-term and stationary characteristics, is ignored. The method directly utilizes semantic feature extracted from reviews to represent user (item), without considering the difference between global property learned from user-item rating matrix and semantic feature learned from reviews. In contrast, our method captures the difference by learning static feature and semantic feature of user (item), respectively. (ii) In A^{3}NCF, user's (item's) semantic feature is extracted from reviews with a topic model, which is fixed for all items (users) and can not reflect the change when facing different target items (users). In contrast, our work captures such adaptive semantic feature based on the static feature of the target item. (iii) When depicting user's (item's) feature, both MPCN and A^{3}NCF ignore the importance of historical rated items's feature and ratings. Instead, we consider all the information collectively by constructing adaptive property feature of user (item). (iv) Unlike the two existing models, which directly concatenate features of user and item, our feature fusing method learns high-order interactions of feature dimensions, and thus make full use of user's and item's features. Meanwhile, we introduce a dynamic sampling strategy to boost the training process further, while both MPCN and A^{3}NCF treat each training sample as of the same importance.

VI. CONCLUSION

Data sparsity is a serious problem in recommender systems. Recent works mostly focus on extracting semantic features from reviews in a local view. In this paper, we consider static features learned from user-item interactions, in addition to the semantic features learned from reviews. Additionally, based on the static features, our model is able to extract adaptive semantic and property features to better model a user when facing different items. We also propose to extract deep interactions among the features and train the model with a dynamic sampling method to improve the recommendation performance. We conduct experiments on 16 public datasets. Experimental results demonstrate that our model can achieve better performance than state-of-the-art models.

ACKNOWLEDGMENT

The authors thank the anonymous reviewers very much for their efforts in evaluating their paper.

REFERENCES

[1] J. Tang, F. Belletti, S. Jain, M. Chen, A. Beutel, C. Xu, and E. H. Chi, "Towards neural mixture recommender for long range dependent user sequences," in *Proc. World Wide Web Conf. (WWW)*, 2019, pp. 1782–1793.

[2] K. Mirylenka, P. Scotton, C. Miksovic, and J. Dillon, "Hidden layer models for company representations and product recommendations," in *Proc. EDBT*, 2019, pp. 468–476.

[3] E. Taniskidou, G. Papadakis, G. Giannakopoulos, and M. Koubarakis, "Comparative analysis of content-based personalized microblog recommendations," in *Proc. EDBT*, 2019, pp. 193–204.

[4] A. Pourali, F. Zarrinkalam, and E. Bagheri, "Point-of-interest recommendation using heterogeneous link prediction," in *Proc. EDBT*, 2018, pp. 481–484.

[5] Q. V. Le and T. Mikolov, "Distributed representations of sentences and documents," in *Proc. ICML*, 2014, pp. 1188–1196.

[6] Y. Bao, H. Fang, and J. Zhang, "TopicMF: Simultaneously exploiting ratings and reviews for recommendation," in *Proc. AAAI*, 2014, pp. 2–8.

[7] T. Luong, H. Pham, and C. D. Manning, "Effective approaches to attention-based neural machine translation," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2015, pp. 1412–1421.

[8] Y. Tan, M. Zhang, Y. Liu, and S. Ma, "Rating-boosted latent topics: Understanding users and items with ratings and reviews," in *Proc. IJCAI*, 2016, pp. 2640–2646.

[9] Y. Li, D. Zhang, Z. Lan, and K.-L. Tan, "Context-aware advertisement recommendation for high-speed social news feeding," in *Proc. IEEE 32nd Int. Conf. Data Eng. (ICDE)*, May 2016, pp. 505–516.

[10] H. Li, Y. Ge, D. Lian, and H. Liu, "Learning user's intrinsic and extrinsic interests for point-of-interest recommendation: A unified approach," in *Proc. IJCAI*, 2017, pp. 2117–2123.

[11] D. Lian, Y. Ge, F. Zhang, N. J. Yuan, X. Xie, T. Zhou, and Y. Rui, "Scalable content-aware collaborative filtering for location recommendation," *IEEE Trans. Knowl. Data Eng.*, vol. 30, no. 6, pp. 1122–1135, Jun. 2018.

[12] J. McAuley and J. Leskovec, "Hidden factors and hidden topics: Understanding rating dimensions with review text," in *Proc. 7th ACM Conf. Recommender Syst. (RecSys)*, 2013, pp. 165–172.

[13] J. Y. Chin, K. Zhao, S. Joty, and G. Cong, "ANR: Aspect-based neural recommender," in *Proc. 27th ACM Int. Conf. Inf. Knowl. Manage.*, Oct. 2018, pp. 147–156.

[14] S. Sedhain, A. K. Menon, S. Sanner, and L. Xie, "AutoRec: Autoencoders meet collaborative filtering," in *Proc. 24th Int. Conf. World Wide Web-WWW Companion*, 2015, pp. 111–112.

[15] C. Xiong, V. Zhong, and R. Socher, "Dynamic coattention networks for question answering," in *Proc. ICLR*, 2017, pp. 1–14.

[16] J. Lu, J. Yang, D. Batra, and D. Parikh, "Hierarchical question-image co-attention for visual question answering," in *Proc. NIPS*, 2016, pp. 289–297.

[17] D. Kim, C. Park, J. Oh, S. Lee, and H. Yu, "Convolutional matrix factorization for document context-aware recommendation," in *Proc. 10th ACM Conf. Recommender Syst.*, Sep. 2016, pp. 233–240.

[18] X. Li and J. She, "Collaborative variational autoencoder for recommender systems," in *Proc. KDD*, 2017, pp. 305–314.

[19] D. Liang, R. G. Krishnan, M. D. Hoffman, and T. Jebara, "Variational autoencoders for collaborative filtering," in *Proc. World Wide Web Conf. World Wide Web (WWW)*, 2018, pp. 689–698.

[20] P. Li, Z. Wang, Z. Ren, L. Bing, and W. Lam, "Neural rating regression with abstractive tips generation for recommendation," in *Proc. 40th Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.*, Aug. 2017, pp. 345–354.

[21] S. Seo, J. Huang, H. Yang, and Y. Liu, "Interpretable convolutional neural networks with dual local and global attention for review rating prediction," in *Proc. 11th ACM Conf. Recommender Syst.*, Aug. 2017, pp. 297–305.

[22] L. Zheng, V. Noroozi, and P. S. Yu, "Joint deep modeling of users and items using reviews for recommendation," in *Proc. 10th ACM Int. Conf. Web Search Data Mining (WSDM)*, 2017, pp. 425–434.

[23] Z. Cheng, Y. Ding, X. He, L. Zhu, X. Song, and M. Kankanhalli, "A^3NCF: An adaptive aspect attention model for rating prediction," in *Proc. IJCAI*, 2018, pp. 3748–3754.

[24] X. He, X. Du, X. Wang, F. Tian, J. Tang, and T. Chua, "Outer product-based neural collaborative filtering," in *Proc. IJCAI*, 2018, pp. 2227–2233.

[25] Y. Tay, L. Anh Tuan, and S. C. Hui, "Latent relational metric learning via memory-based attention for collaborative ranking," in *Proc. World Wide Web Conf. World Wide Web (WWW)*, 2018, pp. 729–739.

[26] H. Wang, N. Wang, and D.-Y. Yeung, "Collaborative deep learning for recommender systems," in *Proc. KDD*, 2015, pp. 1235–1244.

[27] J. Ding, G. Yu, X. He, Y. Quan, Y. Li, T. Chua, D. Jin, and J. Yu, "Improving implicit recommender systems with view data," in *Proc. IJCAI*, 2018, pp. 3343–3349.

[28] Y. Tay, A. T. Luu, and S. C. Hui, "Multi-pointer co-attention networks for recommendation," in *Proc. 24th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Jul. 2018, pp. 2309–2318.

[29] R. Catherine and W. Cohen, "TransNets: Learning to transform for recommendation," in *Proc. 11th ACM Conf. Recommender Syst.*, Aug. 2017, pp. 288–296.

[30] G. Ling, M. R. Lyu, and I. King, "Ratings meet reviews, a combined approach to recommend," in *Proc. 8th ACM Conf. Recommender Syst. (RecSys)*, 2014, pp. 105–112.

[31] W. Zhang and J. Wang, "Integrating topic and latent factors for scalable personalized review-based rating prediction," *IEEE Trans. Knowl. Data Eng.*, vol. 28, no. 11, pp. 3013–3027, Aug. 2016.

[32] Y. Zhang, Q. Ai, X. Chen, and W. B. Croft, "Joint representation learning for top-N recommendation with heterogeneous information sources," in *Proc. ACM Conf. Inf. Knowl. Manage.*, Nov. 2017, pp. 1449–1458.

[33] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, p. 436, 2015.

[34] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T. Chua, "Neural collaborative filtering," in *Proc. 26th Int. Conf. World Wide Web*, 2017, pp. 173–182.

[35] A. Almahairi, K. Kastner, K. Cho, and A. Courville, "Learning distributed representations from reviews for collaborative filtering," in *Proc. 9th ACM Conf. Recommender Syst. (RecSys)*, 2015, pp. 147–154.

[36] W. Zhang, Q. Yuan, J. Han, and J. Wang, "Collaborative multi-level embedding learning from reviews for rating prediction," in *Proc. IJCAI*, Jul. 2016, pp. 2986–2992.

[37] Z. Cheng, Y. Ding, L. Zhu, and M. Kankanhalli, "Aspect-aware latent factor model: Rating prediction with ratings and reviews," in *Proc. World Wide Web Conf. World Wide Web (WWW)*, 2018, pp. 639–648.

[38] Q. Diao, M. Qiu, C.-Y. Wu, A. J. Smola, J. Jiang, and C. Wang, "Jointly modeling aspects, ratings and sentiments for movie recommendation (JMARS)," in *Proc. 20th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining (KDD)*, 2014, pp. 193–202.

[39] X. He, T. Chen, M.-Y. Kan, and X. Chen, "TriRank: Review-aware explainable recommendation by modeling aspects," in *Proc. 24th ACM Int. Conf. Inf. Knowl. Manage. (CIKM)*, 2015, pp. 1661–1670.

[40] X. He and T.-S. Chua, "Neural factorization machines for sparse predictive analytics," in *Proc. 40th Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.*, Aug. 2017, pp. 355–364.

[41] H. Zhang, F. Shen, W. Liu, X. He, H. Luan, and T. Chua, "Discrete collaborative filtering," in *Proc. 39th Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.*, 2016, pp. 325–334.

[42] Z. Cheng, X. Chang, L. Zhu, R. C. Kanjirathinkal, and M. Kankanhalli, "MMALFM: Explainable recommendation by leveraging reviews and images," *ACM Trans. Inf. Syst.*, vol. 37, no. 2, pp. 16:1–16:28, Mar. 2019.

[43] Y. Hou, N. Yang, Y. Wu, and P. S. Yu, "Explainable recommendation with fusion of aspect information," *World Wide Web*, vol. 22, no. 1, pp. 221–240, Jan. 2019.

[44] C. Ma, P. Kang, B. Wu, Q. Wang, and X. Liu, "Gated attentive-autoencoder for content-aware recommendation," in *Proc. 12th ACM Int. Conf. Web Search Data Mining*, Jan. 2019, pp. 519–527.

[45] Y. Zhang, Q. Yao, Y. Shao, and L. Chen, "NSCaching: Simple and efficient negative sampling for knowledge graph embedding," in *Proc. IEEE 35th Int. Conf. Data Eng. (ICDE)*, Apr. 2019, pp. 614–625.

[46] J. He, J. Qi, and K. Ramamohanarao, "A joint context-aware embedding for trip recommendations," in *Proc. IEEE 35th Int. Conf. Data Eng. (ICDE)*, Apr. 2019, pp. 292–303.

[47] T. Chen, H. Yin, H. Chen, R. Yan, Q. V. H. Nguyen, and X. Li, "AIR: Attentional intention-aware recommender systems," in *Proc. IEEE 35th Int. Conf. Data Eng. (ICDE)*, Apr. 2019, pp. 304–315.

[48] Q. Grossetti, C. Constantin, C. Mouza, and N. Travers, "An homophily-based approach for fast post. Recommendation on Twitter," in *Proc. EDBT*, 2018, pp. 229–240.

[49] Z. Yuan, F. Wu, J. Liu, C. Wu, Y. Huang, and X. Xie, "Neural review rating prediction with user and product memory," in *Proc. 28th ACM Int. Conf. Inf. Knowl. Manage.*, Nov. 2019, pp. 2341–2344.

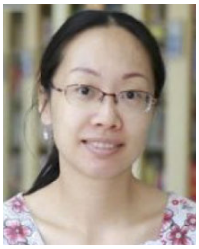
[50] C. Chen, M. Zhang, Y. Liu, and S. Ma, "Neural attentional rating regression with review-level explanations," in *Proc. World Wide Web Conf. World Wide Web (WWW)*, 2018, pp. 1583–1592.

[51] H. Xia, Z. Wang, B. Du, L. Zhang, S. Chen, and G. Chun, "Leveraging ratings and reviews with gating mechanism for recommendation," in *Proc. 28th ACM Int. Conf. Inf. Knowl. Manage.*, Nov. 2019, pp. 1573–1582.

WEI LIU (Member, IEEE) received the B.S. degree from Shanghai University, in 2010, the M.S. degree from the South China University of Technology, in 2013, and the Ph.D. degree from Sun Yat-Sen University, China, in 2018, all in computer science. He has been a Postdoctoral Fellow with Sun Yat-sen University, since 2018. His current research interests include recommendation systems, user behavior learning in location-based social networks, spatio-temporal data mining, and deep learning.

ZHIPING LIN is currently pursuing the M.S. degree with the State Key Laboratory of the Big Data Analytics and Processing, Guangzhou, China. His research interest includes recommendation systems.

HUAIJIE ZHU (Member, IEEE) received the M.Sc. and Ph.D. degrees from Computer Science Department, Northeastern University, China. He is currently an Associate Researcher with Sun Yat-sen University, China. His research interests include spatial database, social networks, and data privacy. He is a member of the CCF.



JING WANG received the B.S. and M.S. degrees from the Nanjing University of Science and Technology, Jiangsu, China, in 2002 and 2004, respectively, and the Ph.D. degree from Sun Yat-sen University, Guangdong, China, in 2013. From 2004 to 2018, she served as the Director of the Software Technology Teaching and Research Section, Guangdong Neusoft University. Since 2015, she has been an Associate Professor of computer software in Guangdong. She is currently the Director of Artificial Intelligence Technology Service Major of The Open University of Guangdong (Guangdong Polytechnic Institute). She is also a Big Data Analyst (Senior) of the Ministry of Industry and Information Technology of China, a Senior Trainer of Baidu artificial intelligence, and a Java Certification Trainer of Oracle. She is the author of 30 articles, and holds five patents and two software copyrights. Her research interests include deep learning, personalized recommendation, and education data mining. She is a member of the Association for Computing Machinery (ACM) and China Computer Federation (CCF).

ARUN KUMAR SANGAIAH (Member, IEEE) received the M.E. degree from Anna University and the Ph.D. degree from VIT University, Vellore, India. He is currently a Professor with the School of Computing Science and Engineering, VIT University. In 2016, he was a Visiting Professor with the School of Computer Engineering, Neusoft Institute, Guangdong, China, for six months. In addition, he has been appointed as a Visiting Professor with Southwest Jiaotong University, Chengdu, the Changsha University of Science and Technology, China, the Dongguan University of Technology, Guangdong, and the Hwa-Hsia University of Technology, Taiwan. Furthermore, he has visited many research centers and universities in China, Japan, Singapore, and South Korea, for joined collaboration toward research projects and publications. His research interests include e-learning, machine learning, software engineering, computational intelligence, and the IoT.