Attentive Aspect Modeling for Review-Aware Recommendation

XINYU GUAN, Xi'an Jiaotong University, P. R. China
ZHIYONG CHENG, Qilu University of Technology (Shandong Academy of Sciences), China
XIANGNAN HE, University of Science and Technology of China, China
YONGFENG ZHANG, Rutgers University, USA
ZHIBO ZHU and QINKE PENG, Xi'an Jiaotong University, P. R. China
TAT-SENG CHUA, National University of Singapore, Singapore

In recent years, many studies extract aspects from user reviews and integrate them with ratings for improving the recommendation performance. The common aspects mentioned in a user's reviews and a product's reviews indicate indirect connections between the user and product. However, these aspect-based methods suffer from two problems. First, the common aspects are usually very sparse, which is caused by the sparsity of user-product interactions and the diversity of individual users' vocabularies. Second, a user's interests on aspects could be different with respect to different products, which are usually assumed to be static in existing methods. In this article, we propose an Attentive Aspect-based Recommendation Model (AARM) to tackle these challenges. For the first problem, to enrich the aspect connections between user and product, besides common aspects, AARM also models the interactions between synonymous and similar aspects. For the second problem, a neural attention network which simultaneously considers user, product, and aspect information is constructed to capture a user's attention toward aspects when examining different products. Extensive quantitative and qualitative experiments show that AARM can effectively alleviate the two aforementioned problems and significantly outperforms several state-of-the-art recommendation methods on the top-N recommendation task.

CCS Concepts: • **Information systems** → **Collaborative filtering**; **Recommender systems**;

Additional Key Words and Phrases: Top-N recommendation, neural network, attention mechanism, aspects

This work was finished when Xinyu Guan was a visiting student at the National University of Singapore. The support provided by China Scholarship Council (CSC) during the visit of Xinyu Guan to National University of Singapore is acknowledged. The first author claims that this work is under the supervision of Dr. Zhiyong Cheng and Dr. Xiangnan He. This work is supported by the National Natural Science Foundation of China (grant number 61872288). This work is also supported by the NExT research centre, which is supported by the National Research Foundation, Prime Minister's Office, Singapore under its IRC@SG Funding Initiative.

Authors' addresses: X. Guan, Z. Zhu, and Q. Peng, Xi'an Jiaotong University, Systems Engineering Institute, 28, Xianning West Road, Xi'an, 710049, P. R. China; emails: xinyu_guan@foxmail.com, zdh96.zzb@stu.xjtu.edu.cn, qkpeng@mail.xjtu.edu.cn; Z. Cheng (corresponding author), Qilu University of Technology (Shandong Academy of Sciences), Shandong Computer Science Center (National Supercomputer Center in Jinan), Shandong Artificial Intelligence Institute, 19 Keyuan Road, Jinan, Shandong 250014, China; email: jason.zy.cheng@gmail.com; X. He, University of Science and Technology of China, School of Information Science and Technology, 443 Huangshan Road, Hefei, 230031, China; email: xiangnanhe@gmail.com; Y. Zhang, Rutgers University, Department of Computer Science, 110 Frelinghuysen Road, Piscataway 08854, USA; email: zhangyf07@gmail.com; T.-S. Chua, National University of Singapore, School of Computing, Singapore, 13 Computing Drive, 117417, Singapore; email: dcscts@nus.edu.sg.

1046-8188/2019/03-ART28 $15.00
https://doi.org/10.1145/3309546

ACM Reference format:
Xinyu Guan, Zhiyong Cheng, Xiangnan He, Yongfeng Zhang, Zhibo Zhu, Qinke Peng, and Tat-Seng Chua. 2019. Attentive Aspect Modeling for Review-Aware Recommendation. *ACM Trans. Inf. Syst.* 37, 3, Article 28 (March 2019), 27 pages.
https://doi.org/10.1145/3309546

1 INTRODUCTION

Recommender systems help users find their potentially interested products from an enormous list of products. Matrix Factorization (MF) methods [27] are widely adopted in recommendation systems because of their accuracy and scalability. MF methods usually rely on the explicit (e.g., user ratings) or implicit (e.g., click behaviors) interactions between users and products for recommendation. However, a rating or binary interaction can only reflect the user's overall attitude toward a product, which does not include information about the underlying reasons for the user behavior. As a result, it is difficult for MF methods to model user's fine-grained preferences on specific product features and provide an explanation to recommendations.

To tackle these limitations, researches have attempted to utilize reviews to alleviate the data sparsity problem and provide more explainable recommendations [3, 8, 10, 11, 21]. As accompanying information of ratings, the textual review expresses user's opinions on different product features, and thus contains more fine-grained information about the user preference. Different strategies have been applied to incorporate reviews into MF models, including sentiment analysis [35], representation learning [4, 48], and topic models [31, 41]. Although these methods have achieved some progress, the generated vector representations of users and products are still latent and thus cannot explicitly model user's preference on specific product features, which could impede their performance.

Another direction is to leverage the aspects mentioned in user reviews for recommendation. In this article, **aspect** is defined as the words or phrases used by users in their product reviews to describe product features. For example, "battery life" and "battery duration" are two different aspects while they refer to the same product feature. There are already some methods which detect aspects in user reviews and leverage them to model user's fine-grained preferences to specific product features [15, 49]. For example, EFM [49] conducted an aspect-level sentiment analysis to extract user's preference and product's quality on a specific product feature, then incorporated the results into an MF framework to provide a more accurate recommendation. SULM [1] and LRPPM [9] went beyond EFM [49] by using more effective methods to identify the impact of each aspect on the overall rating. However, these methods rely highly on the accuracy of external sentiment analysis tools.

Besides the above-mentioned limitations, these methods also suffer from the following two problems. First, for each user-product pair, they only consider the shared aspects in the user's reviews and the product's reviews. However, due to the sparsity of user-product interactions and users' diverse language usages, the number of common aspects mentioned in the reviews of both the targeted user and product is usually very limited. Second, a user's concerned aspects may be different for different products (even in the same category). For example, a user may be mostly concerned about "special effects" when watching a superhero movie, while paying more attention to the "plot" for a suspense movie.

Motivated by the above concerns, in this article, we propose an Attentive Aspect-based Recommendation Model (AARM), which can effectively tackle the above two problems. For the first problem of *aspect sparsity*, AARM models the interactions between synonymous and similar aspects to alleviate it, where *synonymous aspects* are the ones referring to the same product feature (e.g., "storyline" and "plot"); and *similar aspects* are those of different features that are closely related (e.g., "battery life" and "charging speed"). Intuitively, a user's attention to an unmentioned aspect can be inferred through its similar aspects. For instance, a user who cares about "battery life" of cell phones may also care about its "charging speed," although "charging speed" has never been mentioned in this user's reviews. In our model, an aspect extracted from reviews is first represented as an embedding vector. Then a user u's satisfaction about product v according to aspect a is estimated by calculating the interactions between a and all the aspects mentioned in v's reviews. And an attention module is designed to pick up the interactions between meaningful aspect pairs. In this way, we achieve the goal of capturing the interactions between synonymous and similar aspects.

For the second problem of *identifying user's varied interests on aspects*, AARM introduces another attention module which takes user, product, and aspect information into consideration. In this way, a user's varied interests on aspects can be captured by the product-dependent user attention. Instead of rating prediction, we target the top-N recommendation task with a pairwise learning-to-rank method, which is the most practically used recommendation scenario in real-world systems [14, 42]. To this end, our model estimates a user u's satisfaction toward a product v by (1) estimating v's performances on u's concerned aspects; and (2) identifying the impacts of these aspects on the overall satisfaction.

We evaluate our model on five product datasets from Amazon on the top-N recommendation task. Experimental results show that AARM outperforms several state-of-the-art methods. Comparative experiments have also been conducted to demonstrate the importance of modeling interactions between different aspects and the effectiveness of our attention module on capturing user's varied attentions toward aspects. Our main contributions are outlined as follows.

- We propose a novel recommendation method to model the interactions between both the same and the different aspects, which helps to alleviate the aspect sparsity problem in reviews. To the best of our knowledge, this is the first attempt to model the interactions between different aspects to model user preferences in recommendation. And the method to capture the similarity relation between different aspects can also be used in other recommendation scenes (e.g., recommendation with tags or item metadata).
- We design an attention mechanism in AARM to capture a user's varied attention on different aspects toward various products. The careful design of the inputs and structure of this attention module has been demonstrated to be very effective on improving the recommendation accuracy in the experiments.
- We conduct extensive experiments on real-world datasets to demonstrate the effectiveness of our model. Experimental results show that our method can achieve superior performance by a large margin.

The reminder of the article is organized as follows. We first discuss existing works related to our method in Section 2. In Section 3, we describe the details of AARM and describe how to train the model. In Section 4, we describe the experimental settings and report the results to verify our assumptions and compare our methods with some state-of-the-art baselines. Finally, in Section 5, we conclude the article.

2 RELATED WORK

In recent years, many researchers have paid more attention to users' product reviews in order to improve the recommendation accuracy and provide recommendation explanation. According to how these methods utilize user reviews, we broadly group them into three categories: *review-level*, *topic-level*, and *aspect-level* methods. In this section, we first review these three types of review-based methods, and then briefly discuss the recommendation methods with attention mechanism, which is an important component in our model.

2.1 Review-Level Methods

Review-level methods treat the review as a single piece of information and incorporate it with ratings. The opinion-driven MF model [35] calculates the overall opinion of a review by summing up the orientations of opinion words in the text, and then combines it with numerical ratings for rating prediction. Meng et al. [32] incorporated other users' emotions toward a review to calculate the importance of this review in the training of a MF model. Some methods concatenate all the reviews belonging to a user (or item) as a user (or item) document, and then employ deep learning methods to learn the continuous vector representation for the user (or item) [4, 19, 48, 51]. For example, Transnets [4] and DeepCoNN [51] process the user and item documents with convolutional neural network to generate the vector representation for users and items. JRL [48] adopts the PV-DBOW model [28], which is an unsupervised method to learn the continuous vector representations for documents, and the user and item vector representations from their reviews. In Transnets, DeepCoNN, and JRL, in order to estimate the matching degree between a user and an item, reviews of the user or item are compressed to a vector which is an overall representation of the reviews. In this way, these review-level methods neglect the user-item interactions at the review components (e.g., the user's opinions about the product's specific features) level, which can be used to connect the user with candidate products and provide more explainable recommendations.

2.2 Topic-Level Methods

Topic-level methods build a probabilistic graphical model to extract topics from reviews. HFT [31] combines topic vectors from reviews with latent factors from ratings to improve rating prediction accuracy. Subsequently, some studies employ different topic models and combination strategies for the review-based rating prediction task. For example, different from HFT, ITLFM [47] linearly combines the latent topics and the latent factors. CMR [45] is a probabilistic graphical model which simultaneously associates the review text, the hidden user communities, and item group relationship with numerical ratings. RBLT [41] also utilizes LDA to extract topics from review text. Then the preference distribution vector of each user and the recommendability distribution vector of each item are combined with a vanilla MF model for rating prediction. More recently, Cheng et al. [12] defined a high-level semantic concept "aspect" as a probability distribution of topics. They proposed the ATM model to extract topics from reviews and associated the topics with "aspects," and then proposed the ALFM model to associate latent factors with 'aspects'. In this way, topics are correlated with factors via the "aspects" indirectly. To estimate the overall rating score, they first calculated the item's scores on each aspect and then summed them up using aspect importance as weights. Similarly, MMALFM [10] follows the definition of "aspect" in [12] and jointly models the "aspects" in textual reviews and item images. These topic-level methods usually focus on rating prediction task, while we are targeting at top-N recommendation. Similar to review-level methods, when estimating the matching degree between a user and a product, these topic-level methods also neglect the interactions between the components of the user and the product's reviews. And it is difficult to associate a topic, which is a probabilistic distribution over words or phrases, with specific product features. Because of these limitations, these methods are incapable of capturing a user's preference toward product features in a finer-grained manner and thus provide more accurate and explainable recommendations.

2.3 Aspect-Level Methods

Aspect-level methods extract aspects from reviews and incorporate them with ratings for recommendation. The proposed method in this article falls into this category. Ganu et al. [17] manually defined six aspects and four sentiments for restaurant reviews and used a regression-based method for rating prediction. Zhang et al. [49] employed an unsupervised tool for aspect extraction and aspect-level sentiment analysis. Aspect and sentiment outputs from this step were integrated with MF methods for rating prediction. Chen et al. [9] proposed a tensor-MF method to select the most interesting product features for each user with a learning-to-rank method. The rating scores were then predicted as the weighted summation of the product's sentiment scores on the user's most cared product features. Bauman et al. [1] also extracted aspects and conducted aspect-level sentiment analysis with external tools. The results of aspect-level sentiment analysis were used in their model SULM as the ground-truth labels to train a latent factor model for every aspect. These aspect-level latent factor models were then used to predict user's sentiment scores toward each aspect of a product. The number of parameters in SULM is very large as a user or product usually has many aspects. As we can see, the above methods often rely on external sentiment tools for aspect-level analysis.

Specifically, there are also some papers which pay attention to users' varied interests. Chen et al. [9] proposed an aspect ranking method to capture a user's varied interests while they paid more attention to a user's interest variation over different categories. A^3NCF [11], ALFM [12], and ANR [13] also try to capture users' varied interests toward aspects. Specifically, A^3NCF and ANR also use neural attention layers to do it. But there are some important differences between them and our method. First, the "aspects" defined in A^3NCF, ALFM, and ANR are different from the one defined in our model. In A^3NCF, "aspect" is defined as a combination of topic vector and embedding vector. In ALFM, "aspect" is defined as a probability distribution of topics and thus ALFM is more like a topic-level model. In ANR, an "aspect" of a user is a weighted sum of all the words' embeddings in the user's reviews. Different from them, "aspects" in our model are words or phrases directly extracted from reviews, which is much more fine-grained concept. Second, A^3NCF, ALFM, and ANR have not considered interactions between different aspects. Different from them, our method models these interactions because intuitively these aspects are not independent of each other. Third, those three existing methods are originally designed for rating prediction, while our model is designed for top-N recommendation.

He et al. [21] did not conduct sentiment analysis but adopted the aspect frequency information in reviews to construct the user-item-aspect tripartite graph for recommendation. The improved performance in [21] from baselines verified that the aspect mention signals in reviews could have already been able to reflect the user's interests on aspects. Similarly, in AARM we do not conduct sentiment analysis on reviews explicitly, which helps to simplify the model design and implementation. Moreover, AARM considers both the interactions between different aspects and the user's varied preference toward aspects, which are neglected by previous studies.

2.4 Attention Mechanism

In recent years, many deep learning-based recommendation methods have been proposed and achieved good performance in many tasks [22, 24, 40, 46]. The attention mechanism which can assign adaptive weights for a set of features has also been employed in recommendation models [2, 6, 7, 16, 23]. For example, in the NARRE model [5] for review-based rating prediction, Chen et al. introduced an attention module to calculate the usefulness of reviews. In TEM [43], which utilizes user and item's side information for explainable recommendation, neural attention layer is used to assign weights to cross features and provide recommendation explanation. ACF [7], which focuses on multimedia recommendation, uses a component-level attention module to find informative components for multimedia items (images/videos), and an item-level attention module to select representative items to to represent users' preferences. AFM [44], which is an extension of FM machine [22, 36], uses an attention neural network to discriminate the importance of different

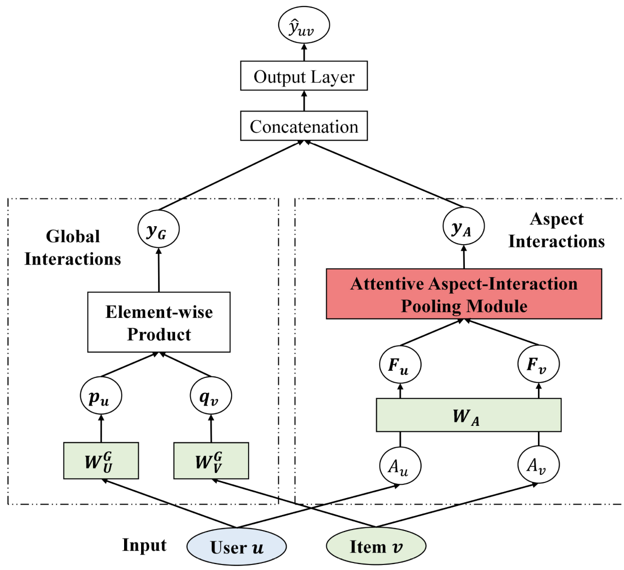

Fig. 1. Attentive aspect-based recommendation model.

feature interactions. A³NCF [11] and ANR [13] also have attention modules which have been discussed in the last section. Compared with these methods, we specially design two attention modules for the fine-grained modeling of product features extracted from user reviews. The user-level attention module in AARM is built to find out the user's most concerned product features for a candidate product, while the aspect-level attention module is constructed to select informative aspect interactions.

3 ATTENTIVE ASPECT-BASED RECOMMENDATION MODEL

In this section, we first provide an overview of our method and define some important notations, and then introduce how to extract aspects from user reviews. After that, we describe the structure and details of the proposed AARM model. In particular, we elaborate how AARM could model the interactions between different aspects and handle a user's varied interests in aspects. Finally, we discuss the parameter inference in AARM.

3.1 Preliminaries

Given a user set $U = \{u_1, u_2, \ldots u_{|U|}\}$ and a product set $V = \{v_1, v_2, \ldots v_{|V|}\}$, AARM estimates a satisfaction score $\hat{y}_{uv}$ for a user u toward a product v. The candidate products are then ranked in a descending order of $\hat{y}$ and the top-N products are recommended to u. In our method, aspects extracted from user reviews are used as the explicit features of users and products. We define $A = \{a_1, a_2, \ldots a_{|A|}\}$ as the aspect set of the dataset. The aspects that have been mentioned in the reviews of user u are represented as A_u, which is a subset of A. Similarly, product v's aspects that have been mentioned in v's reviews are represented as A_v. Product v's rating given by user u is denoted as $r_{uv} \in R$, where R is the collection of ratings.

The structure of AARM is shown in Figure 1. In the input layer, users and products are represented as binarized sparse vectors using the one-hot encoding method. Above the input layer, the Aspect Interactions part is used to model the interactions between the aspects from user u's aspect set A_u and the aspects from the product v's aspect set A_v. Because a user's review for a product may not cover all the factors which can influence the user's satisfaction toward the product, the aspects extracted from review text may not be able to fully explain the rating. Hence, the Global Interactions part is stacked above the input layer to model the implicit factors which influence a user's decision but have not been discussed in the reviews. Finally, the results of aforementioned two parts are concatenated as the input to the Output Layer.

3.2 Aspect Interactions Part

In the Aspect Interactions part, given a user u and a product v, aspects are first extracted from their reviews and used to construct their aspect sets A_u and A_v, respectively. To model the similarity between aspects, instead of one-hot encoding or bag-of-words model, embedding layers are used in AARM to represent aspects as continuous vectors. Specifically, aspect embedding matrix $\mathbf{W}_A \in \mathbb{R}^{d_a \times |A|}$ is defined to project aspects from A_u and A_v to $\mathbf{F}_u \in \mathbb{R}^{d_a \times M_u}$ and $\mathbf{F}_v \in \mathbb{R}^{d_a \times M_v}$, respectively, where d_a is the dimension of aspect embeddings, and M_u and M_v are, respectively, the number of aspects in A_u and A_v. The ith aspect in A_u is projected to $\mathbf{f}_{u,i}$ which is the ith column of $\mathbf{F}_u$. Similarly, aspects in A_v are projected to the embedding vectors in $\mathbf{F}_v$. Next, Attentive Aspect-Interaction Pooling Module is designed to model the bi-interactions between the aspect embeddings of $\mathbf{F}_u$ and that of $\mathbf{F}_v$, and outputs a vector $\mathbf{y}_A$ to represent the preference information in user reviews.

3.2.1 Aspect Extraction. Because the main contribution of this article focuses on how to leverage aspects for personalized recommendation, we refer to external tools for aspect extractions. In this article, we use the Sentires,[1] which has been successfully used in [49, 50] for aspect extraction. Other aspect extraction tools can also be applied. This toolkit extracts aspects via a hybrid of rule-based and machine learning algorithms. Given a dataset, it generates an aspect lexicon, which is used to build the aspect set A of the dataset in this article. With this toolkit, we could obtain user aspect set A_u for each user $u \in U$, and product aspect set A_v for each product $v \in V$ by extracting the mentioned aspects from their reviews. Some examples of the automatically extracted aspects are shown in Table 3.

Note that the size of aspect set varies for different users or products. To accelerate the training of AARM, we pad all the user aspect sets into the same length M_u and pad all the product aspect sets into the same length M_v. Taking a user aspect set as an example, we define a meaningless aspect $< PAD >$ and add it to the end of user aspect sets whose lengths are less than the pre-defined size M_u. For A_u whose length is larger than M_u, we calculate the *TF-IDF* score [38] of each $a \in A_u$, and truncate A_u to M_u aspects by dropping the aspects with low *TF-IDF* scores. The *TF-IDF* score is defined as

$$tfidf_u(a) = \frac{tf_u(a)}{\sum_{i \in A_u} tf_u(i)} \cdot ln \frac{|U|}{df(a) + 1}, \tag{1}$$

where $tf_u(a)$ is the frequency of a's occurrence in u's reviews, $|U|$ is the number of users, and $df(a)$ is the number of users who mentioned a. All the product aspect sets are padded into the same length M_v in a similar way.

3.2.2 Attentive Aspect-Interaction Pooling Module. As shown in Figure 2, given $\mathbf{F}_u$ and $\mathbf{F}_v$ as input, there are four parts in this module: ***aspect embedding transformation***, ***aspect interaction layer***, ***aspect-level attentive pooling layer***, and ***user-level attentive pooling layer***. The final output of this module is the vector $\mathbf{y}_A(u, v)$ which represents the overall satisfaction of a user u toward a product v estimated with review text. In this module, we hold the assumption that

[1] http://yongfeng.me/software/.

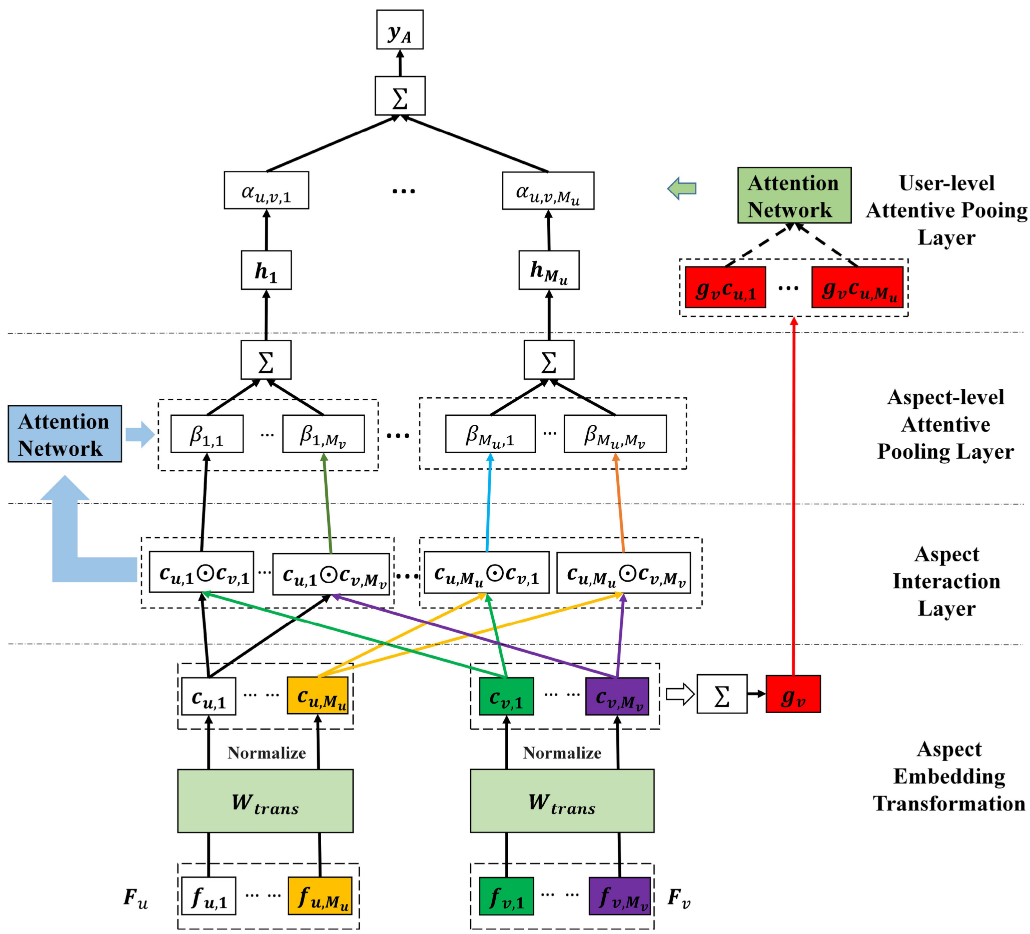

Fig. 2. The attentive aspect-interaction pooling module.

u's overall satisfaction for v is based on v's performances on u's concerned aspects (i.e., aspects from A_u). This module works as follows. First, for each aspect $a \in A_u$, the aspect interaction layer and aspect-level attentive pooling layer are employed to estimate v's performance on a, where the performance is represented as vector $\mathbf{h}_a(u, v)$. Then the user-level attentive pooling layer is used to estimate u's preference toward v by integrating $\mathbf{h}_a(u, v)$ for all the aspects $a \in A_u$ and represent the preference as a vector $\mathbf{y}_A(u, v)$. Finally, $\mathbf{y}_A(u, v)$ will be combined with the result of Global Interaction part and further input into the output layer to estimate the user u's satisfaction score toward the product v.

Aspect Embedding Transformation. To model the interactions between synonymous and related aspects, we expect the vector representation of aspect to encode the similarity relation between aspects. In this article, the Word2vec model [33], which is able to encode many linguistic regularities and patterns, is used to pre-train aspect embeddings with the review texts in each dataset. The aspect embedding matrix $\mathbf{W}_A$ is initialized with the pre-trained embeddings and its parameters would not be tuned during the training of AARM. Instead, a trainable matrix $\mathbf{W}_{trans} \in \mathbb{R}^{d_a \times d_a}$ is defined to customize the pre-trained aspect embedding $\mathbf{f}$ (column vector in $\mathbf{F}_u$ or $\mathbf{F}_v$) to make it oriented toward our recommendation task. Then these customized embeddings are normalized as

$$\mathbf{c} = \frac{\mathbf{W}_{trans}\mathbf{f}}{\|\mathbf{W}_{trans}\mathbf{f}\|}. \tag{2}$$

Here $\|\mathbf{x}\|$ is the Euclidean norm of $\mathbf{x}$. In this article, aspect interaction between two aspects is defined as the element-wise product of their embedding vectors. By normalizing the aspect embeddings with their corresponding Euclidean norms, the calculation of interaction between two aspects is similar to calculating their cosine similarity. As illustrated in [33], if two words have higher semantics and syntax similarities, their embeddings generated by Word2vec would have larger cosine similarity. In this way, the results of aspect interactions are associated with the semantics and syntax relations between aspects, which helps in identifying the synonymous and related aspects. Alternatively, we can also directly tune the aspect embedding matrix $\mathbf{W}_A$ during the training of AARM for top-N recommendation. We will compare the performances of these different settings in the experiment section.

Aspect Interaction Layer. This layer maps the vector representations of aspects in A_u and A_v to a set of d_a-dimensional interacted vectors. The aspect interaction between aspect $i \in A_u$ and $j \in A_v$ is defined as the element-wise product of their embedding vector $\mathbf{c}_i$ and $\mathbf{c}_j$. Hence, the output of the aspect interaction layer can be represented as a set of vectors:

$$f_{AI}(u,v) = \{\mathbf{c}_i \odot \mathbf{c}_j(x_i x_j)\}_{i \in A_u, j \in A_v}. \tag{3}$$

Here $x_i \in \{0,1\}$ is the masking indicator, where $x_i = 0$ if i is the meaningless aspect $\langle PAD \rangle$ (defined for padding). To implement the masking operation in AARM, we define an aspect masking vector $\mathbf{W}_{mask} \in \mathbb{R}^{d_a \times |A|}$, where the column of aspect $\langle PAD \rangle$ is a zero vector, and the columns of other aspects in A are vectors of ones. Before calculating the interactions between the aspect $i \in A_u$ and aspects in A_v, we first calculate the element-wise product between $\mathbf{c}_i$ and its corresponding column in $\mathbf{W}_{mask}$. After the masking operation, the embedding vector of aspect $\langle PAD \rangle$ is transformed into a zero vector. In this way, we make sure that the interactions between aspect $\langle PAD \rangle$ and other aspects are zero vectors. As shown in the following sections, these zero vectors would not influence AARM's final predictions.

As shown in Equation (3), besides the same aspects, the interactions between different aspects (when $i \neq j$) are also calculated. This is because we want to model the interactions between synonymous and similar aspects to alleviate the problem that *the same aspects shared in a user's reviews and a product's reviews are usually very sparse*. However, interactions between unrelated aspects are also considered in Equation (3). To emphasize on interactions between related aspects and filter out noisy interactions, the aspect-level attentive pooling layer is stacked above this layer.

Aspect-Level Attentive Pooling Layer. In the aspect interaction layer, for each aspect $a \in A_u$, we calculate its interaction with all the aspects in A_v. Intuitively, some aspect interactions should be given more attention than others. For example, the interactions between the same, synonymous, or similar aspects usually contain more information about the product's performance on the user's concerned aspects. Hence, an attention module is designed to focus on important aspect interactions. Word2vec embeddings of similar words would have higher cosine similarities [33]. Inspired by this, for aspect pair i and j, the input of the attention layer is defined as the element-wise product of their normalized embedding vectors $\mathbf{c}_i$ and $\mathbf{c}_j$ to mimic the cosine similarity between their embedding vectors. And the aspect-level attention layer is defined as

$$\begin{aligned} \hat{\beta}_{i,j} &= \mathbf{w}_{att_1}{}^T(\mathbf{c}_i \odot \mathbf{c}_j)(x_i x_j), \\ \beta_{i,j} &= \frac{\exp(\hat{\beta}_{i,j})}{\sum_{y \in A_v} \exp(\hat{\beta}_{i,y})}. \end{aligned} \tag{4}$$

Here $\mathbf{w}_{att_1} \in \mathbb{R}^{d_a}$ is a learnable vector, and $\beta_{i,j}$ is the attention value of the interaction between aspect i and j.

To estimate the product's performance on the user's aspect $a_i \in A_u$, we compress all the interactions between a_i and aspects in A_v with a weighted sum pooling where β is used as the weight:

$$\mathbf{h}_i = \sum_{j \in A_v} \beta_{i,j} (\mathbf{c}_i \odot \mathbf{c}_j)(x_i x_j). \tag{5}$$

The output of this layer is the vector set $\{\mathbf{h}_i \in \mathbb{R}^{d_a}\}_{a_i \in A_u}$.

User-Level Attentive Pooling Layer. We can integrate the vector set $\{\mathbf{h}_i\}_{a_i \in A_u}$, which represents how the product fits the user's requirements on each aspect, and thus to produce an estimation of the user's overall satisfaction on this product. Intuitively, different users may focus on different aspects even when purchasing the same products. For example, when purchasing a cell phone, some users are more concerned about battery duration while some other users are more concerned about the performance of CPU. Furthermore, when purchasing different products, a user's most concerned product features may be different. In other words, a user's attention toward an aspect when purchasing a specific product is influenced by the characteristics of the user, the aspect, and the product simultaneously.

To estimate user u's interest toward aspect $a \in A_u$ when purchasing a specific product v, a user-level attentive pooling layer is designed in AARM. The input of this attention layer should contain not only information of current aspect a, but also information of product v. Intuitively, if an aspect $a \in A_u$ is more important to product v, the user should pay more attention to the aspect a as compared with other unrelated aspects in A_u. The importance of the user's aspect a with respect to a product v can be measured by the similarities between a and the aspects that have been mentioned in v's reviews (i.e., aspects from A_v). To calculate the importance of aspect $a_i \in A_u$ with respect to product v, the interactions between a_i and all the aspects in A_v are calculated and summed up:

$$\begin{aligned} \mathbf{x}_{v,i} &= \mathbf{g}_v \odot \mathbf{c}_i, \\ \mathbf{g}_v &= \sum_{j \in A_v} \mathbf{c}_j. \end{aligned} \tag{6}$$

As the interaction between two aspects represents their similarity, $\mathbf{x}_{v,i}$ represents the overall similarity between the aspect a_i and the product v. To measure the importance of different aspect $a_i \in A_u$, $\mathbf{x}_{v,i}$ is used as aspect a_i's input to the user-level attention layer. The attention layer is defined as

$$\begin{aligned} \hat{\alpha}_{u,v,i} &= {\mathbf{w}_{att_2}}^T \mathbf{x}_{v,i}, \\ \alpha_{u,v,i} &= \frac{\exp\left(\hat{\alpha}_{u,v,i}\right)}{\sum_{j \in A_u} \exp\left(\hat{\alpha}_{u,v,j}\right)}. \end{aligned} \tag{7}$$

Here $\mathbf{w}_{att_2} \in \mathbb{R}^{d_a}$ is a learnable vector, and $\alpha_{u,v,i}$ represents the importance of aspect $a_i \in A_u$ in user u's preferences with regard to product v. This attention layer is different from the aspect-level attention layer defined in Equation (4) as $\mathbf{w}_{att_1}$ and $\mathbf{w}_{att_2}$ are two different vectors.

Finally, we compress the vector set $\{\mathbf{h}_i\}_{a_i \in A_u}$ with a weighted sum pooling to generate a vector which represents user u's overall satisfaction toward product v:

$$\mathbf{y}_A(u,v) = \sum_{i \in A_u} \alpha_{u,v,i} \mathbf{h}_i. \tag{8}$$

Here $\mathbf{y}_A(u,v) \in \mathbb{R}^{d_a}$ is the output of Aspect Interactions Module.

3.3 Global Interactions Part

To model the implicit factors which are not mentioned in review text but have influence over users' satisfaction, AARM assigns a latent factor for every user and product, respectively. In this module, embedding matrix $\mathbf{W}_U^G \in \mathbb{R}^{d_g \times |U|}$ is defined to project user u to $\mathbf{p}_u$, and the embedding matrix $\mathbf{W}_V^G \in \mathbb{R}^{d_g \times |V|}$ is defined to project product v to $\mathbf{q}_v$. These two embedding matrices are randomly initialized and tuned during the training for top-N recommendation. Then the global interaction between user u and product v is calculated in a way similar to that in vanilla latent factor models:

$$\mathbf{y}_G(u,v) = \mathbf{p}_u \odot \mathbf{q}_v. \tag{9}$$

Here $\mathbf{y}_G(u,v) \in \mathbb{R}^{d_g}$ is the output of this part.

3.4 Output Layer

To merge information from the aforementioned two modules, $\mathbf{y}_A(u,v)$ and $\mathbf{y}_G(u,v)$ are concatenated into one vector. And a regression layer without an activation function is stacked above it:

$$\hat{y}(u,v) = \mathbf{W}_{out} \begin{bmatrix} \mathbf{f}_G(u,v) \\ \mathbf{y}_A(u,v) \end{bmatrix}. \tag{10}$$

Here $\mathbf{W}_{out}$ belongs to $\mathbb{R}^{1\times(d_a+d_g)}$. $\hat{y}(u,v)$ represents user u's overall satisfaction score toward product v.

3.5 Learning

In this article, we binarize the ratings scores and train AARM with a learning-to-rank method. Ranking methods are widely used in information retrieval [25, 29, 30] and recommendation models [24, 34]. In AARM, we use Bayesian Personalized Ranking (BPR), which is a pairwise method. This makes AARM suitable for recommendation with implicit feedbacks. Given a user u, a triple (u, v^+, v^-) is constructed for pairwise training. Here, v^+ refers to the product that u has purchased, while v^- refers to an unpurchased one. During training, the positive user-product pair (u, v^+) is drawn from rating set R, which is accompanied with one negative pair (u, v^-), where v^- is randomly sampled from u's unpurchased products. Intuitively, AARM should give a higher satisfaction score to the positive pair (u, v^+) than the negative pair (u, v^-). Hence, the BPR optimization criterion is employed as the objective function of AARM:

$$L_{bpr} = \frac{-1}{|R|} \sum_{(u,v^+)\in R} \log(\sigma(\hat{y}(u,v^+) - \hat{y}(u,v^-))). \tag{11}$$

Here, σ refers to the sigmoid function, and $|R|$ is the number of positive pairs (u, v^+) in R.

To prevent the possible overfitting, L^2 regularization is used on the user and product embedding matrix and the kernel matrix of the output layer. As shown in Equation (12), to implement the L^2 regularization, we first calculate the mean values of the element-wise square of these three matrices. The results are then multiplied by the L^2 regularization coefficient λ and added to the loss function:

$$L = L_{bpr} + \lambda * \left(\frac{||\mathbf{W}_U^G||^2}{|\mathbf{W}_U^G|} + \frac{||\mathbf{W}_V^G||^2}{|\mathbf{W}_V^G|} + \frac{||\mathbf{W}_{out}||^2}{|\mathbf{W}_{out}|} \right). \tag{12}$$

Here λ controls the L^2 regularization strength, $||\mathbf{W}||$ refers to the L^2-norm of the matrix $\mathbf{W}$, and $|\mathbf{W}|$ refers to the number of elements in the matrix $\mathbf{W}$. We minimize the loss function L to fit AARM from data.

Besides L^2 regularization, we also use dropout [39] to reduce overfitting. Dropout can prevent complex co-adaptations on training data by randomly dropping some units during training [39].

Dropout is employed on the output of Global Interactions module and the output of Aspect Interactions module.

Aspect Embedding Pre-Training. In our experiments, *gensim*'s implementation[2] of Word2vec is used to train the aspect embeddings. Before training embeddings with Word2vec, we first construct a dictionary for every dataset and then segment the reviews of each dataset into lists of words or phrases according to this dictionary. All the aspects (in the form of words or phrases) of each dataset are added into the corresponding dictionary to make sure that the Word2vec tool can recognize all the aspects and train embedding vectors for them. For each dataset, all the reviews in the training set are used for the training of aspect embedding. These embedding vectors are used as the initial values of the aspect embedding matrix $\mathbf{W}_A$, which would not be tuned during the training for top-N recommendation.

4 EXPERIMENTS

In this section, we design experiments to study the following research questions:

- **RQ1** Can AARM outperform state-of-the-art methods on top-N recommendation task?
- **RQ2** Can the interactions between different aspects improve the performance of AARM?
- **RQ3** Can the modeling of varied user interests improve the performance of AARM?
- **RQ4** How does the initialization and tuning strategy of aspect embedding influence the performance of AARM?
- **RQ5** What are the contributions of the Global Interaction part and Aspect Interaction part in the overall performance of AARM?

In the rest of this section, we will first introduce experimental settings, and then successively answer the above research questions with not only quantitative experiments but also qualitative case studies.

4.1 Datasets

We use the "five-core" subsets from the publicly accessible "Amazon product dataset"[3] [20] for experiments. Here the "five-core" means that each user and product in the subset has at least five reviews. Each record in the dataset is composed of five variables including *user*, *product*, *rating*, *textual review*, and *helpfulness votes*. In AARM, we only use *user*, *product*, and *textual review*. To follow the setting of baseline methods, in our pairwise learning-to-rank framework, ratings are binarized to construct positive user-product pairs. We adopt five different product categories from the "Amazon product dataset," that is, "*Movies and TV*," "*CDs and Vinyl*," "*Clothing, Shoes, and Jewelry*," "*Cell Phones and Accessories*," and "*Beauty*." Some detailed statistics including the sparsity and the number of ratings (#Rating), users (#User), and products (#Product) of the five datasets are summarized in Table 1. Sparsity is defined as $\#Rating/(\#User \times \#Product)$. We can see that the five datasets are of different sizes and different levels of sparsity, which could cover different recommendation scenarios.

For each user, its 70% records are randomly selected as a training set, while the remaining 30% records are put into a test set. Particularly, we use the exact same splits and evaluation measures as the experimental settings in [48].[4] This is to guarantee that all the methods are evaluated on exactly the same settings for fair comparisons.

[2] https://radimrehurek.com/gensim/.
[3] http://jmcauley.ucsd.edu/data/amazon/.
[4] We would like to thank the authors for sharing with us the datasets and specific splits.

Table 1. Statistics of the Experimental Datasets

Dataset	**#Rating**	**#User**	**#Product**	**Sparsity**
Movies and TV	1,697,533	123,960	50,052	0.0274%
CDs and Vinyl	1,097,592	75,258	64,421	0.0226%
Clothing, Shoes, and Jewelry	278,677	39,387	23,033	0.0307%
Cell Phones and Accessories	194,439	27,879	10,429	0.0669%
Beauty	198,502	22,363	12,101	0.0734%

Table 2. Statistics of Aspects Extracted from Reviews

Dataset	**#Aspect**	**Ave. #Aspect/User**	**Ave. #Aspect//Product**
Movies and TV	2,865	14.72	32.24
CDs and Vinyl	4,033	31.04	41.31
Clothing, Shoes, and Jewelry	525	7.04	9.77
Cell Phones and Accessories	648	6.93	12.50
Beauty	691	9.72	13.13

Table 3. Some Examples of the Automatically Extracted Aspects

Dataset	**Aspects**
Movies and TV	3D movie, cast, halloween film, halloween movie, harden, melodrama, movie star, screen time, thrillers, zombie movie
CDs and Vinyl	1980s, band, crooners, crooning, country music, fingerwork, singers, rock fans, songwriters, composers
Clothing, Shoes, and Jewelry	color, cottony, diamonds, fit, price, presentation box, sleeve shirts, sleeve, traction, torso
Cell Phones and Accessories	usb, accessory, a little, car chargers, car speaker, charge cycle, charge cycles, looks, plastic, quality
Beauty	results, smell, chocolate smell, odor, ingredient, ingredients, face feeling, hair feeling, sheen, shampoos

4.2 Aspects from User Reviews

Some detailed statistics of the aspects extracted from user reviews by *Sentires* are shown in Table 2. We can see that the number of aspects (Aspect#), the average number of aspects per user (Ave. # Aspect/User), and the average number of aspects per product (Ave. # Aspect/Product) in the five datasets are varied, which makes our experiments more comprehensive.

Table 3 shows some examples of the aspects extracted from each dataset. We did not conduct any post-processing on the extracted aspects. Although there are some noise words in the aspect collection, *Sentires* is largely effective in extracting many meaningful aspects that correspond to important product features. And there are synonymous aspects like "songwriters" and "composers," and related aspects like "smell" and "chocolate smell," which would usually be treated as disparate product features in most existing aspect-level models.

4.3 Evaluation Protocols

To generate a top-N recommendation list for user u, a model first estimates the scores of u's candidate products, then ranks all the candidate products according to the scores and truncates the ranking list at N. In this article, u's candidate products include all the products in u's test set and those that have not been purchased by u. In the evaluation, products in u's test set would be used as ground truth. Following the settings in [48], we set $N = 10$. Four standard metrics are used in the evaluation: Recall, Precision, Normalized Discounted Cumulative Gain (NDCG), and Hit Ratio (HT).

Recall is the percentage of products that has been recommended to the user in the products that have been purchased by the user:

$$Recall = \frac{n_{tp}}{n_{gt}}, \tag{13}$$

where n_{tp} is the number of ground-truth products in the recommendation list, and n_{gt} is the number of ground-truth products. We average the measure across all testing users.

Precision is the percentage of products that has been purchased by the user in the top-N recommendation list:

$$Precision = \frac{n_{tp}}{N}. \tag{14}$$

The measure is averaged across all testing users.

NDCG is a measure when the positions of the purchased products in the recommendation list are considered. NDCG is based on the Discounted Cumulative Gain (DCG):

$$DCG = \sum_{i=1}^{N} \frac{2^{rel_i} - 1}{log_2(i+1)}. \tag{15}$$

Here, rel_i is the graded relevance of the product at position i of the recommendation list for a user. The NDCG of a user is then calculated as

$$NDCG = \frac{DCG}{IDCG}. \tag{16}$$

Here IDCG is the DCG of the ideal recommendation list where the user's ground-truth products are all ranked at the top. We average NDCG across all testing users.

HT is defined as in the following equation where n_{hit} is the number of users who have purchased at least one product in its recommendation list:

$$HT = \frac{n_{hit}}{|U|}. \tag{17}$$

4.4 Baselines and Parameter Settings

We compare our method AARM with the following baselines.

BPR-MF [37]. The MF based on BPR, which combines the MF model with a pairwise learning to rank loss function, is a solid baseline for top-N recommendation. Only user-product interaction data is used in this method.

BPR-HFT [31]. The HFT model associates topics extracted from reviews with latent factors learned from numerical ratings. It is one of the state-of-the-art review-based recommendation methods. The original HFT model is a rating prediction method. BPR-HFT [48] modifies HFT by adding a BPR loss on top of HFT to generate the top-N recommendation.

GMF [24]. Generalized Matrix Factorization (GMF) is one of the state-of-the-art neural network-based recommendation methods which only utilizes user-product interaction records. In experiments, we directly use the released code by the authors,[5]

[5] https://github.com/hexiangnan/neural_collaborative_filtering.

BPR-AFM [44]. Attentional Factorization Machine (AFM) is an improved variant of the famous factorization machine (FM) [36]. Similar to our method, AFM uses a neural attention network to discriminate the importance of different feature interactions. The original version of AFM is designed for regression task and optimizes the squared loss. We modified AFM by adding a BPR loss on top of AFM to generate the top-N recommendation. Given a user and an item as input, we use the user identity, the item identity, the user's aspects, and the item's aspects as features. Both the identity features and aspect features have corresponding embedding vectors in the model, which are randomly initialized and then fine-tuned during the training.

DeepCoNN [51]. The Deep Cooperative Neural Network is one of the state-of-the-art deep learning methods for recommendation which utilizes reviews to build user and product representations. It uses the review-based user and product representations for rating prediction.

JRL [48]. The Joint Representation Learning model is a state-of-the-art method which integrates different information sources with deep learning methods for top-N recommendation. Textual reviews, product images, and numerical ratings are jointly used in JRL.

JRL-Review [48]. JRL-Review is a single-view version of JRL which incorporates textual reviews for top-N recommendation. JRL-Review employs the PV-DBOW model [28] to learn the vector representations of users and products from their corresponding reviews. It is one of the state-of-the-art review-based recommendation methods.

eJRL [48]. eJRL is another variant of JRL which jointly utilizes textual reviews, product images, and numerical ratings for recommendation. The difference between them is that eJRL prevents information propagation among different information sources.

The hyper-parameters of baselines are tuned on a training set with fivefold cross-validation. In particular, the dimension of latent factors (or embeddings) for BPR-MF, BPR-HFT, and DeepCoNN is 100. For BPR-HFT, the number of topics is 10. For JRL, JRL-Review, and eJRL, the embedding size is set as 300. For GMF and BPR-AFM, the size of all the embedding vectors is set as 128.

Parameter Settings. We implemented our methods with Tensorflow.[6] When padding user aspect set to the same size, the maximum size M_u was defined as the 75% quantile of the sizes of all user aspect sets. Similarly, the maximum size M_v of product aspect set was defined as the 75% quantile of the sizes of all product aspect sets. For embedding layers, we set the dimension d_g of user and product embeddings in the global interactions module to 128; set the dimension d_a of aspect embeddings to 128. AARM was optimized with mini-batch Adam [26] because Adam uses adaptive learning rates for parameters with different update frequencies and converges faster than vanilla stochastic gradient descent. We tested the learning rate of [0.001, 0.003, 0.01]. For the coefficient of L^2 regularization, [0.0, 0.0001, 0.01, 0.1] was tested. To prevent overfitting, in dropout layers, the dropout rate was set to 0.5. When pre-training aspect embeddings with Word2Vec, the window size and the number of noise words for negative sampling are both 5.

The model was trained for a maximum of 300 epochs with early stopping. To build the validation set, 1,000 users are randomly selected from the users in the training set. For each user, one of his purchased products is randomly drawn from the training set as the ground-truth product in the validation set. And when evaluating the model on the validation set, for each user, all the products which are not paired with the user in the training set are added to the candidate set. Then to build a recommendation list for each user, products in the candidate set are ranked according to the estimated matching degrees between them and the user. The aforementioned four measures are used to evaluate the top-N recommendation lists and then averaged across all the validation users. For every 10 epochs, we will test the model's performance on the validation set. The training would be stopped if half of the four measures decreased for 40 successive epochs.

[6] https://www.tensorflow.org/.

Table 4. The NDCG and Hit Ratio (HT) Results of Baselines and the Proposed Method for RQ1

	Movies		CDs		Clothings		Cell Phones		Beauty	
Measures (%)	NDCG	HT	NDCG	HT	NDCG	HT	NDCG	HT	NDCG	HT
BPR-MF	1.267	4.421	2.009	8.554	0.601	1.767	1.998	5.273	2.753	8.241
GMF	3.519	10.897	4.530	14.266	1.144	2.795	3.623	8.230	4.079	11.112
BPR-HFT	2.092	6.378	2.661	9.926	1.067	2.872	3.151	8.125	2.934	8.268
DeepCoNN	3.800	10.522	4.218	13.857	1.310	3.286	3.636	9.913	3.359	9.807
BPR-AFM	3.649	11.578	4.716	15.278	1.354	3.511	3.627	9.229	4.103	11.899
JRL-Review	4.222	12.958	5.286	16.592	1.270	3.527	4.184	10.632	4.216	12.422
eJRL	4.405	13.292	5.023	16.081	1.523	4.182	4.185	10.531	3.896	11.090
JRL	4.334	13.245	5.378	16.774	1.735	4.634	4.364	10.940	4.396	12.776
AARM	**5.020**	**15.187**	**7.252**	**20.749**	**1.956**	**4.915**	**4.976**	**11.568**	**5.314**	**13.648**
Impr-JRL-Review	18.901	17.202	37.193	25.054	54.094	39.354	18.929	8.804	26.044	9.870
Impr-eJRL	13.961	14.257	44.376	29.028	27.742	17.527	18.901	9.847	36.396	23.066
Impr-JRL	15.828	14.662	34.846	23.697	12.795	6.064	14.024	5.740	20.883	6.825

Due to limitation of space, we present the name of dataset "Movies and TV" as "Movies," "CDs and Vinyl" as "CDs," "Clothing, Shoes, and Jewelry" as "Clothings," "Cell Phones and Accessories" as "Cell Phones" for short. The best results are highlighted in bold. The improvements (or decrements for negative values) achieved by AARM compared with the best review-based baseline (Impr-JRL-Review) and the best multi-modal baseline (Impr-JRL or Impr-eJRL) are shown in the last three rows.

Table 5. The Corresponding Recall and Precision Results of Baselines and the Proposed Method

	Movies		CDs		Clothings		Cell Phones		Beauty	
Measures (%)	Recall	Precision	Recall	Precision	Recall	Precision	Recall	Precision	Recall	Precision
BPR-MF	1.988	0.528	2.679	1.085	1.046	0.185	3.258	0.595	4.241	1.143
GMF	5.169	1.306	6.280	1.844	1.832	0.299	5.751	0.931	6.291	1.439
BPR-HFT	3.255	0.776	3.570	1.268	1.819	0.297	5.307	0.860	4.459	1.132
DeepCoNN	4.671	0.886	6.001	1.681	2.332	0.229	6.353	0.999	5.429	1.200
BPR-AFM	5.314	1.409	6.499	2.030	2.275	0.366	6.244	1.021	6.373	1.522
JRL-Review	6.145	1.465	7.454	2.079	2.211	0.336	7.275	1.062	6.766	1.467
eJRL	6.289	1.521	6.973	2.002	2.679	0.396	7.130	1.054	6.010	1.355
JRL	6.334	1.492	7.545	2.085	2.989	0.442	7.510	1.096	6.949	1.546
AARM	**7.140**	**1.834**	**9.965**	**2.716**	**3.292**	**0.511**	**8.014**	**1.259**	**7.947**	**1.818**
Impr-JRL-Review	16.192	25.188	33.687	30.640	48.892	52.083	10.158	18.550	17.455	24.777
Impr-eJRL	13.532	20.579	42.908	35.664	22.882	29.040	12.398	19.450	32.230	34.170
Impr-JRL	12.725	22.922	32.074	30.264	10.137	15.611	6.711	14.872	14.362	17.594

4.5 Model Comparison (RQ1)

Tables 4 and 5 show the performance of our method and baselines on top-N recommendation task. The performances of rating-based methods (BPR-MF and GMF), review-based methods (BPR-HFT, DeepCoNN, BPR-AFM, and JRL-Review), multi-modal methods (eJRL and JRL), and our method (AARM) are shown in the four blocks in each table from top to bottom. The last block of each table also presents the percentage of improvements (or decrements for negative values) achieved by AARM as compared with the best review-based baseline (Impr-JRL-Review) and the best multi-modal baseline (Impr-JRL or Impr-eJRL). The best results are highlighted in bold. As we use the same split as [48], we directly reproduce their results of BPR-MF, BPR-HFT, DeepCoNN, JRL-Review, eJRL, and JRL for fair comparisons. From Tables 4 and 5, we can see the following:

(1) In general, neural network-based methods outperform shallow models (e.g., BPR-MF and BPR-HFT). GMF, which only uses user-product interaction data, even largely outperforms BPR-HFT which incorporates reviews for recommendation. This might be attributed to the powerful representation learning capacity of neural models.
(2) Generally, review-based methods outperform rating-based methods. All the review-based methods outperform BPR-MF. Among neural network-based methods, BPR-AFM and JRL-Review also outperform GMF. This shows that review is an important information source to boost recommendation performance.
(3) Our proposed method AARM outperforms all the rating-based methods and review-based methods on all the datasets in terms of different metrics. Compared to these baselines, AARM makes better use of the user-product interaction records and review texts. This is because of AARM's finer-grained modeling of aspect interactions, which simultaneously considers the interactions between different aspects and user's varied attentions toward aspects. In the following sections, we further analyze how the specific designs of AARM boost its recommendation performance.
(4) AARM also outperforms both of the multi-modal deep learning methods on all the datasets and on all the measures. It is surprising that our method outperforms these multi-modal deep learning methods which not only utilize review data but also leverage product image and numerical rating data for recommendation. This further indicates that textual review is a very informative information source and AARM's finer-grained aspect modeling could effectively leverage reviews for recommendation. In the following sections, we will discuss the contribution of each part of AARM by comparing AARM with its variants.

4.6 Effect of Interactions Between Different Aspects (RQ2)

Previous aspect-based methods neglect the interactions between synonymous and similar aspects when making recommendations, and are limited by the sparsity of shared aspects in the reviews of users and products. AARM alleviates this problem by modeling the interactions between different aspects and using an attention module to capture the important aspect interactions. To verify the effect of this design, we compare AARM with its two variants, which are termed as "A_Inter" and "No-AspectAtt" in Figure 3, under the same experimental settings.

As variants of AARM, the differences between AARM, No-AspectAtt, and A_Inter are in the Aspect Interactions part. Given a user u and a product v, A_Inter only considers the interactions between shared aspects of u and v, that is, $a \in A_u \cap A_v$. Hence, in the Aspect Interactions part of A_Inter, we first calculate the intersection $A_{u,v}^{inter}$ of A_u and A_v. To estimate $\mathbf{h}_a$ which represents u's preference to v according to aspect $a \in A_u$, Equations (3), (4), and (5) of AARM are replaced with the following equation:

$$\mathbf{h}_a = \sum_{i \in A_{u,v}^{inter}} (\mathbf{c}_i \odot \mathbf{c}_i)(x_i). \tag{18}$$

Here $x_i \in \{0, 1\}$ is an indicator, where $x_i = 0$ if i is the meaningless aspect $\langle PAD \rangle$ defined for padding. As A_Inter only considers interactions between the same aspects, no aspect-level attention module is used here. In No-AspectAtt, the aspect-level attention layer is removed and the aspect interactions are directly summed up. Equations (4) and (5) of AARM are replaced with the following equation:

$$\mathbf{h}_i = \sum_{j \in A_v} (\mathbf{c}_i \odot \mathbf{c}_j)(x_i x_j). \tag{19}$$

We evaluate A-Inter and No-AspectAtt's performance on top-N recommendation task and compare them with AARM in Figure 3. All the experimental settings are kept the same to ensure

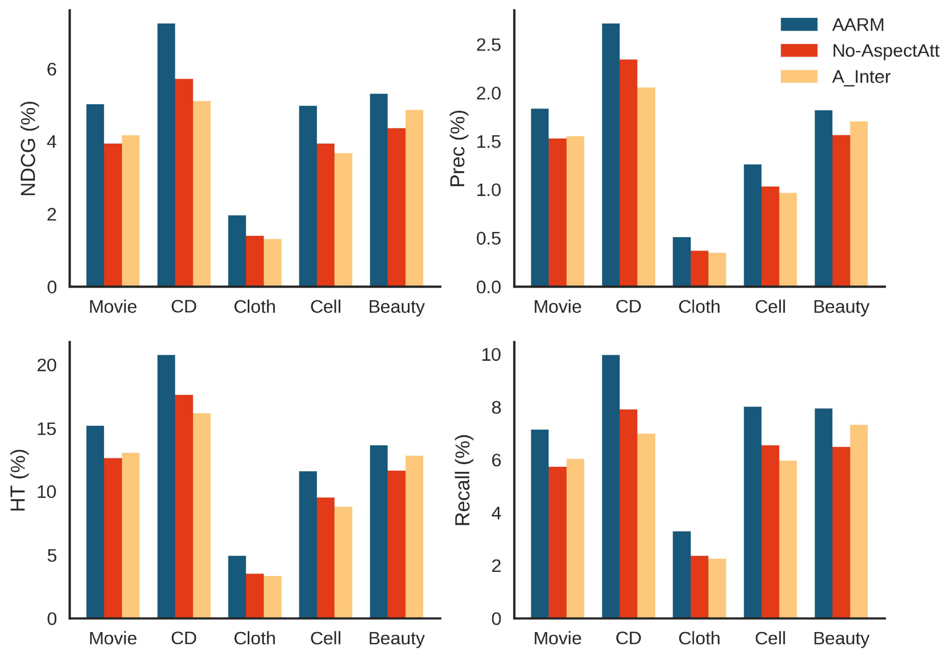

Fig. 3. Performance of AARM, No-AspectAtt, and A_Inter on five datasets for RQ2. Due to limitation of space, we present dataset "Movies and TV" as "Movie," "CDs and Vinyl" as "CD," "Clothing, Shoes, and Jewelry" as "Cloth," and "Cell Phones and Accessories" as "Cell" for short.

the reliability of results. As shown in Figure 3, AARM substantially outperforms A_Inter and No-AspectAtt on all datasets in terms of all measures. Compared to A-Inter, the average improvements achieved by AARM are 39.401% for NDCG, 37.427% for recall, 32.823% for HT, and 33.593% for precision. The results demonstrate the importance of modeling the interactions between different aspects and the effectiveness of our carefully designed aspect-level attentive layer. We will further perform a qualitative analysis of the aspect-level attention layer in Section 4.10.

4.7 Effect of Varied User Interest Modeling (RQ3)

In the design of AARM, we assume that a user's interests toward aspects are varied among different products. And a user-level attentive pooling layer (Equations (6), (7), and (8)), which simultaneously considers user, product, and aspect information, is designed to capture a user's different biases toward aspects when facing different products. To verify the effect of the user-level attention module, we design two variants of AARM, called A_Static and No-UserAtt, and compare them with AARM on the top-N recommendation task under the same settings.

The differences between AARM, A_Static, and No-UserAtt are in the design of the user-level attention module. A_Static also assumes that a user's interests toward different aspects are different. But different from AARM, A_Static assumes that a user's interests toward aspects are fixed when facing different products. Therefore, the inputs of the user-level attention layer in A_static do not consider the information of candidate products. When estimating user u's interests toward its aspects, different from AARM, the input of the aspect $a_i \in A_u$ is designed as

$$\begin{aligned} \mathbf{x}_{u,i} &= \mathbf{g}_u \odot \mathbf{c}_i, \\ \mathbf{g}_u &= \sum_{j \in A_u} \mathbf{c}_j. \end{aligned} \tag{20}$$

Table 6. The NDCG and Hit Ratio (HT) Results of AARM and Its Variants on Five Datasets for RQ3

Measures (%)	**Movies**		**CDs**		**Clothings**		**Cell Phones**		**Beauty**	
	NDCG	HT	NDCG	HT	NDCG	HT	NDCG	HT	NDCG	HT
AARM	**5.020**	**15.187**	**7.252**	**20.749**	**1.957**	**4.915**	**4.976**	**11.568**	**5.314**	**13.648**
A_Static	4.376	13.318	6.794	19.567	1.898	4.590	4.728	11.181	4.918	12.735
No-UserAtt	4.290	13.104	6.700	19.108	1.310	3.217	4.685	10.786	4.739	12.297
Impr A_static	14.717	14.034	6.741	6.041	3.109	7.081	5.245	3.461	8.052	7.169
Impr No-UserAtt	17.016	15.896	8.239	8.588	49.389	52.782	6.211	7.250	12.133	10.986

We follow the short form convention adopted in Table 4 to name the datasets. The best performance of each measure on each dataset is highlighted in bold. The last block shows the percentage of improvements (or decrements for negative values) achieved by AARM compared with A_static (Impr A_static) and No-UserAtt (Impr No-UserAtt).

Table 7. The Corresponding Precision and Recall Results of AARM and Its Variants on Five Datasets for RQ3

Measures (%)	**Movies**		**CDs**		**Clothings**		**Cell Phones**		**Beauty**	
	Recall	Precision	Recall	Precision	Recall	Precision	Recall	Precision	Recall	Precision
AARM	**7.140**	**1.834**	**9.965**	**2.716**	**3.292**	**0.511**	**8.014**	**1.259**	**7.947**	**1.818**
A_Static	6.275	1.588	9.075	2.470	3.131	0.476	7.776	1.219	7.337	1.699
No-UserAtt	6.076	1.561	8.953	2.403	2.193	0.337	7.583	1.176	7.046	1.648
Impr A_static	13.785	15.491	9.807	9.960	5.142	7.353	3.061	3.281	8.314	7.004
Impr No-UserAtt	17.512	17.489	11.303	13.025	50.114	51.632	5.684	7.058	12.787	10.316

Here $\mathbf{g}_u$ is the overall representation of aspects in A_u. And $\mathbf{x}_{u,i}$, which represents a summation of the similarities between aspect a_i and all the aspects in A_u, is aspect a_i's input to the user-level attention layer.

Similar to Equation (7), the attention layer is defined as

$$\begin{aligned} \hat{\alpha}_{u,i} &= {\mathbf{w}_{att_2}}^T \mathbf{x}_{u,i}, \\ \alpha_{u,i} &= \frac{\exp\left(\hat{\alpha}_{u,i}\right)}{\sum\limits_{j \in A_u} \exp\left(\hat{\alpha}_{u,j}\right)}. \end{aligned} \tag{21}$$

Here $\mathbf{w}_{att_2} \in \mathbb{R}^{d_a}$, and $\alpha_{u,i}$ represents the importance of aspect $a_i \in A_u$ with respect to the user u. From Equations (20) and (21), we can see that no product information is used in the user-level attention module.

Different from AARM, No-UserAtt assumes that a user would assign equal weights to its aspects when purchasing products. So instead of the user-level attentive pooling layer, No-UserAtt directly sums up the set of vectors $\{\mathbf{h}_i\}_{a_i \in A_u}$ which represents the candidate product's performances on the aspects of user u:

$$\mathbf{y}_A(u, v) = \sum_{j \in A_u} \mathbf{h}_j. \tag{22}$$

As shown in Tables 6 and 7, AARM outperforms A_static and No-UserAtt on all the datasets and on all the measures. We remind one that the only differences between AARM and A_static are the different assumptions about user attentions on aspects toward different products. From the results, we can see that AARM's varied user interests assumption is more reasonable as compared to the constant user interests assumption of A_static. In real-life scenarios, a user could be interested in many different kinds of products and each product can be described by a specific set of aspects. Obviously, the user will pay less attention to the aspects which are not related to the current product. As no two products are exactly alike, a user's interests on the diverse aspects can be

Table 8. The NDCG and Hit Ratio (HT) Results of AARM and Its Variants on Five Datasets for RQ4

Measures (%)	Movies		CDs		Clothings		Cell Phones		Beauty	
	NDCG	HT	NDCG	HT	NDCG	HT	NDCG	HT	NDCG	HT
AARM	**5.020**	**15.187**	**7.252**	**20.749**	**1.957**	**4.915**	**4.976**	**11.568**	**5.314**	**13.648**
Random+Tune	4.607	13.989	6.709	19.443	1.487	3.636	4.354	10.316	4.794	12.972
Pre-train+Tune	4.764	14.320	6.744	19.905	0.802	2.046	4.210	10.191	4.658	12.266
Random vs. Pre-train	−3.296	−2.311	−0.519	−2.321	85.411	77.713	3.420	1.227	2.920	5.756

We follow the short form convention adopted in Table 4 to name the datasets. The best performance of each measure on each dataset is highlighted in bold. The last block shows the percentage of improvements (or decrements for negative values) achieved by Random+Tune compared with Pre-train+Tune (Random vs. Pre-train).

Table 9. The Corresponding Precision and Recall Results of AARM and Its Variants on Five Datasets for RQ4

Measures (%)	Movies		CDs		Clothings		Cell Phones		Beauty	
	Recall	Precision	Recall	Precision	Recall	Precision	Recall	Precision	Recall	Precision
AARM	**7.140**	**1.834**	**9.965**	**2.716**	**3.292**	**0.511**	**8.014**	**1.259**	**7.947**	**1.818**
Random+Tune	6.495	1.667	8.957	2.428	2.476	0.382	7.161	1.135	7.288	1.706
Pre-train+Tune	6.744	1.719	9.270	2.616	1.346	0.216	7.012	1.110	6.969	1.647
Random vs. Pre-train	−3.692	−3.025	−3.376	−7.187	83.952	76.852	2.125	2.252	4.577	3.582

varied even for the products from the same category. We will further represent how the user-level attentive pooling works when facing different products in Section 4.10.

In Tables 6 and 7, A_static also outperforms No-UserAtt on all the datasets in general. As A_static can be viewed as an enhanced version of No-UserAtt, where a fixed user interests model is added, we can see that identifying the different importance of aspects can boost the recommendation performance. This result is reasonable because different users have different tastes, and they would put different attentions to different product features.

4.8 Effects of Initialization and Tuning Strategy of Aspect Embedding (RQ4)

In AARM, the embeddings of aspects are first initialized with the vectors which are pre-trained with Word2vec on each dataset, and then transformed by the matrix $\mathbf{W}_{trans}$. This is inspired by the findings in [33] that the word embeddings trained with Word2vec can retain the syntactic and semantic similarity relation between words. We keep the aspect embedding matrix $\mathbf{W}_A$ fixed during the training of AARM for top-N recommendation while the matrix $\mathbf{W}_{trans}$ are tunable during the training. We choose this tuning strategy because similar words will be shifted similarly as shown in [18].

There are also two other alternatives for the initialization and tuning strategies of aspect embedding matrix $\mathbf{W}_A$. The first one is to randomly initialize the aspect embedding matrix and then tune it during the training for top-N recommendation. We conducted experiments under this setting and presented the results in Tables 8 and 9 in the row of "Random+Tune." The second choice is to initialize the aspect embedding matrix with pre-trained embeddings and then tune it during the training for top-N recommendation. The experiment results of the second settings are presented in Tables 8 and 9 in the row of "Pre-train+Tune."

As shown in Tables 8 and 9, AARM with the "pre-training + trainable linear transformation" strategy outperforms Random+Tune and Pre-train+Tune on all the datasets and on all the measures. The results are reasonable because in the design of the attention layers in AARM, we assumed that the similarity between two aspects can be represented by the interaction between

Table 10. The NDCG and Hit Ratio (HT) Results of AARM and Its Variants on Five Datasets for RQ5

Measures (%)	Movies		CDs		Clothings		Cell Phones		Beauty	
	NDCG	HT	NDCG	HT	NDCG	HT	NDCG	HT	NDCG	HT
AARM	**5.020**	**15.187**	**7.252**	**20.749**	**1.957**	**4.915**	**4.976**	**11.568**	**5.314**	**13.648**
Global Part	3.035	9.965	4.860	15.462	1.084	2.770	3.492	8.250	4.199	11.050
Aspect Part	2.401	8.237	5.200	16.700	1.677	4.395	3.006	7.568	3.781	11.246
Aspect vs. Global	−20.890	−17.341	6.996	8.007	54.705	58.664	−13.918	−8.267	−9.955	1.774

We follow the short form convention adopted in Table 4 to name the datasets. The best performance of each measure on each dataset is highlighted in bold. The last block shows the percentage of improvements (or decrements for negative values) achieved by Aspect Part compared with Global Part (Aspect vs. Global).

Table 11. The Corresponding Precision and Recall Results of AARM and Its Variants on Five Datasets for RQ5

Measures(%)	Movies		CDs		Clothings		Cell Phones		Beauty	
	Recall	Precision	Recall	Precision	Recall	Precision	Recall	Precision	Recall	Precision
AARM	**7.140**	**1.834**	**9.965**	**2.716**	**3.292**	**0.511**	**8.014**	**1.259**	**7.947**	**1.818**
Global Part	4.485	1.206	6.760	2.057	1.802	0.295	5.645	0.895	6.171	1.507
Aspect Part	3.512	0.936	7.686	2.020	2.925	0.451	5.187	0.794	6.036	1.249
Aspect vs. Global	−21.695	−22.388	13.698	−1.799	62.320	52.881	−8.113	−11.285	−2.188	−17.120

them. The capability of enabling similar words shifted similarly makes the "pre-training + trainable linear transformation" strategy more suitable for our task.

Comparing the performance of Random+Tune with Pre-train+Tune in Tables 8 and 9, we can find that Pre-train+Tune outperforms Random+Tune in larger datasets like "Movies and TV" and "CDs and Vinyl" (refer to Table 1), while Random+Tune performs better in smaller datasets like "Clothing, Shoes and Jewelry," "Cell Phones and Accessories," and "Beauty" (refer to Table 1). This may be caused by the fact that when the training data is not sufficient, the Pre-train+Tune strategy may not be able to transform the pre-trained embeddings for the new task and thus lose the original similarity between words [18]. The Random+Tune strategy which assigns a much smaller random initial value to the embedding matrix is easier to be optimized for the new task in an end-to-end style.

4.9 Model Ablation: Effect of Global Module and Aspect Module (RQ5)

In this section, we examine the roles of the Global Interactions part and Aspect Interactions part in the results of AARM. As shown in Figure 1, given the user and product as input, the two parts of AARM worked separately. Then the outputs of these two parts are merged and input into the output layer to estimate the score. To verify the effect of the Aspect Interactions part, we remove the Global Interactions part from AARM, and directly input the result of the Aspect Interactions part into the output layer. This variant of AARM is referred to as "Aspect Part" in Tables 10 and 11. Similarly, another variant of AARM which is referred to as "Global Part" in Tables 10 and 11 is constructed by removing the Aspect Interactions part from AARM to verify the effect of the Global Interactions Part.

From Tables 10 and 11, we can find that AARM significantly outperforms Aspect Part and Global Part. This result indicates that our combination strategy based on concatenation is valid. And the Global Interactions part, which is designed to capture the user preferences that have not been mentioned in review texts, is an effective complement to the Aspect Interactions part.

Table 12. The Distributions of the Number of Shared Aspects Between a User and a Product on the Five Datasets

Datasets	0	1	2	3	4	5	>5
Cell Phones and Accessories	26.34%	28.95%	19.73%	11.31%	6.09%	3.26%	4.33%
Beauty	35.31%	29.85%	16.37%	8.21%	4.22%	2.29%	3.76%
Clothing, Shoes, and Jewelry	12.09%	24.90%	25.50%	17.92%	10.07%	5.01%	4.50%
Movies and TV	30.98%	27.26%	15.54%	8.73%	5.21%	3.30%	8.98%
CDs and Vinyl	3.06%	10.71%	13.91%	13.36%	11.39%	9.29%	38.27%

From left to right, the columns present the ratios of different user-product pairs which have specific numbers of shared aspects. Specifically, the last column represents the ratio of user-product pairs which have more than five shared aspects.

As compared with Global Part, Aspect Part performs better in two datasets while it falls behind in the other three datasets. Because the Aspect Part connects users and products via the interactions between their aspects, its performance may be influenced by the number of interactions between related aspects. To verify this viewpoint, we traverse all the users and products in a dataset to construct all the possible user-product pairs, and then count the number of shared aspects of each user-product pair. A shared aspect of a user-product pair is an aspect which has been mentioned in both the user and the product's reviews. The distributions of the number of shared aspects of each user-product pair on the five datasets are shown in Table 12.

From Tables 10, 11, and 12, we can find that Aspect Part usually performs better on datasets which have more shared aspects between each user-product pair in general. For example, Aspect Part substantially outperforms the Global Part in the "CDs and Vinyl" and "Clothing, Shoes, and Jewelry" datasets which have the smallest ratios of 0 shared aspects (see the second column in the table). And for datasets "Movies and TV," "Cell Phones and Accessories," and "Beauty" where more than 20% user-product pairs do not have any shared aspects, Global Part outperforms Aspect Part.

4.10 Case Study of Attention Layers

The user-level and aspect-level attention modules are important parts of AARM. The user-level attention module (refer to Equation (7)) is employed to capture a user's varied preferences on aspects. And the aspect-level attention module (refer to Equation (4)) is designed to enhance the interactions between meaningful aspect pairs, like the interactions between the same or similar aspects, and reduce the influence of the interactions between the two irrelevant aspects. To illustrate the roles of these two attention modules in AARM, we randomly selected some examples for qualitative analysis.

In Table 13, we show the user-level attention values of a user "A1P9UMP1XSE6MI" in the "Cellphones and Accessories" dataset when examining different products. The first column is the ids of four products in the dataset and their aspect sets. Each product has 15 aspects, which is the 75% quantile of the sizes of all product aspect sets in the dataset. The rest of the columns show the aspects of the user (the second row from top to bottom) and the attention values that are assigned to these aspects when facing the aforementioned four products. From each product's aspect set, we can find that product "B00EOE6FUW" is a "usb charger," "B005HS5MKS" is a "bluetooth earpiece," and "B002VPE1NO" and "B00E8GYIRI" are the "shell case" of cellphones. The shared aspects of each user-product pair and corresponding attention values are highlighted in red.

As shown in Table 13, when examining a product, the user-level attention module can find the aspects which are related to the product and assign higher attention values to them. First, all the shared aspects (highlighted in red) of each user-product pair are assigned much higher attention values. Second, the user-level attention module can assign higher values to aspects that

Table 13. A Case Study of the User-level Attention Module

	Aspects of User A1P9UMP1XSE6MI								
Products and Their Aspects	sound quality	shell case	grommets	quality	impact protection	usb cords	bluetooth earpiece	usb plug	grab
B00EOE6FUW: usb, usb cable, charging device, colors, cable, usb charger, car trip, usb cords, usb end, nokia lumia, usb chargers, car chargers, wiggle, ultra, usb plug	0.0013	0.0008	0.0058	0.0014	0.0003	**0.4406**	0.0034	**0.5389**	0.0075
B005HS5MKS: peeve, sound quality, sizes, bluetooth earpiece, downside, quality, protection, looks	0.4161	0.0103	0.1416	0.1392	0.0126	0.0371	0.1780	0.0174	0.0477
B002VPE1NO: metallic, shell case, shell, looks, grip, finish, impact protection, protection, iphone cases, grommets, rubber strips, plastic, case w, armor, air case	0.0109	**0.4785**	**0.1309**	0.0084	**0.1464**	0.0199	0.0197	0.0102	0.1751
B00E8GYIRI: impact protection, protection, shell, packing snapon cases, plastic, plastic case, case, scuff, bulk, matte phone protection, polycarbonate, iphone cases, shell case	0.0077	**0.6295**	0.0248	0.0042	**0.1929**	0.0160	0.0144	0.0121	0.0984

The first column (from the left) shows ids and aspect sets of four products from the "Cell Phones and Accessories" dataset. The rest of the columns show the aspects of the user (the second row from top to bottom) and the attention values assigned to these aspects when facing the aforementioned four products. In each row, the aspects mentioned in both the user and product's reviews and their corresponding attention values are highlighted in red.

are related to the product but have not been mentioned in the product's reviews. For example, when examining the shell cases "B002VPE1NO" and "B00E8GYIRI," "grab" is assigned a higher weight, although it is not in the product's aspect set. This is because there are some related aspects of "grab" in the two products' aspect sets which are captured by our attention module (refer to Figure 4).

The examples in Table 13 indicate why AARM can outperform A_Static and No-UserAtt (refer to Tables 6 and 7). The user's aspect set consists of three unrelated kinds of aspects: (1) "sound quality," "quality," and "bluetooth earpiece"; (2) "usb cords" and "usb plug"; (3) "shell case," "grommets," "impact protection," and "grab." In this case, No-UserAtt would assign the same weights to aspect 'bluetooth earpiece" and "shell case" when purchasing a bluetooth earpiece. And A_Static would assign same weights to aspect "sound quality" no matter what kinds of products the user is purchasing. By identifying different aspects' different roles when purchasing different products, AARM achieved better performance.

Next, we present how the aspect-level attention module finds the meaningful interactions (i.e., interactions between the shared aspects, synonymous aspects, and similar aspects) from all the aspect interactions between a user and a product. In Figure 4, we show the aspect-level attention values of the interactions between aforementioned user "A1P9UMP1XSE6MI" and product "B002VPE1NO." In the heat map, the columns refer to the product's aspects while the rows refer to the user's aspects. The color of each grid cell represents the attention value assigned to the corresponding interaction. The darker the color in a grid cell, the higher the attention value.

First, we can see that interactions between the shared aspects like "grommets," "impact protection," and "shell case" are captured and assigned higher attention values. Second, the interactions between synonymous aspects are assigned higher weights as compared with unrelated ones. For example, ("shell case," "shell") is assigned the second highest attention value in the interactions

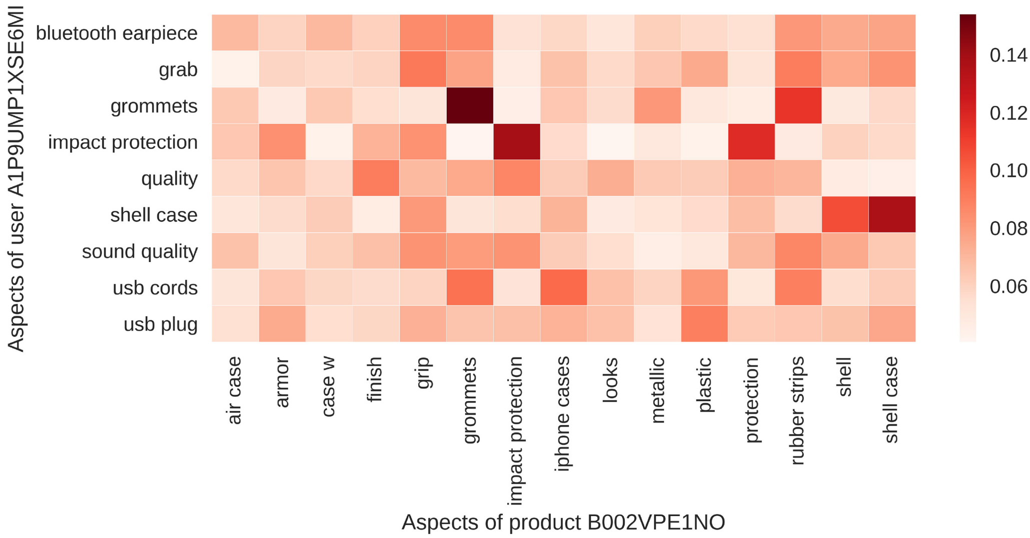

Fig. 4. Heat map of aspect-level attention. The columns refer to aspects of the product while the rows refer to aspects of the user. Darker color in the grid cell means that higher attention value is assigned to the interaction between corresponding aspects by the aspect-level attention module.

between "shell case" and the product's aspects. Third, some interactions between similar aspects are captured. For example, in the interactions with "impact protection," the product's aspects "protection," "armor," and "grip" are assigned high attention values. Finally, for the user's aspects that are unrelated to the product (e.g., "usb plug"), their attention value distributions are more uniform compared to the shared and similar aspects. By assigning higher attention values to meaningful aspect interactions, AARM can alleviate the impact of noisy interactions and overcome the aspect sparsity problem.

5 CONCLUSION AND FUTURE WORK

In this article, we presented an AARM, which carefully captures the interactions between aspects extracted from reviews for recommendation. AARM first calculates the interactions between aspect embeddings to estimate how a product fits a user's requirements on each aspect, and then estimates the user's overall satisfaction on the product by synthesizing the product's performances on each aspect. To deal with the problem that the number of shared aspects between a user and a product is often limited, AARM takes the interactions between different aspects into consideration. With a well-designed aspect-level attention module, not only the shared aspects but also other related aspect pairs can be selected and assigned higher attention values. In addition, we hold the assumption that a user's interests toward aspects are varied when examining different products. To achieve the goal, an attention module which simultaneously considers user and product information is designed in AARM. In the experiments on five real-world datasets, AARM outperforms the state-of-the-art methods on the top-N recommendation task. In particular, compared with multimodal (textual reviews, product images, and numerical ratings) methods JRL and eJRL, AARM can still achieve better results in all datasets. To demonstrate the effectiveness of each component in AARM, a lot of quantitative experiments and qualitative case studies are conducted.

In the future, we would like to extend our work in the following three ways: (1) Applying our method to capture the similarity relation between two different aspects to other recommendation scenes. By using the pre-trained aspect embedding, the aspect embedding transformation module, and the aspect interaction layer, AARM can mimic the cosine similarity and capture the semantics and syntax similarities between two aspects. This strategy can also be used in other recommendation scenes (e.g., recommendation with tags or item metadata) to capture the relation between different elements (like tags or item categories). (2) Extracting aspects with neural network and combining it with AARM. In particular, we would like to jointly train the aspect extraction module and the recommendation module in an end-to-end style. Ideally, the end-to-end training could reduce noisy aspects and mine more domain-specific aspects. (3) Integrating aspect-level sentiment information in AARM. Aspect-level sentiment information is useful to identify a user's likes and dislikes about product features. But existing methods usually use external tools for aspect-level sentiment analysis, which relies on the accuracy of these tools and is usually not able to deal with new reviews. We will study how to extract the sentiment information and integrate it into AARM with end-to-end learning.

REFERENCES

[1] Konstantin Bauman, Bing Liu, and Alexander Tuzhilin. 2017. Aspect based recommendations: Recommending items with the most valuable aspects based on user reviews. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 717–725.

[2] Da Cao, Xiangnan He, Lianhai Miao, Yahui An, Chao Yang, and Richang Hong. 2018. Attentive group recommendation. In *Proceedings of the 41st International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, New York, 645–654.

[3] Da Cao, Xiangnan He, Liqiang Nie, Xiaochi Wei, Xia Hu, Shunxiang Wu, and Tat-Seng Chua. 2017. Cross-platform app recommendation by jointly modeling ratings and texts. *ACM Transactions on Information Systems* 35, 4 (2017), Article 37, 37:1–37:27 pages.

[4] Rose Catherine and William Cohen. 2017. Transnets: Learning to transform for recommendation. In *Proceedings of the 11th ACM Conference on Recommender Systems*. ACM, 288–296.

[5] Chong Chen, Min Zhang, Yiqun Liu, and Shaoping Ma. 2018. Neural attentional rating regression with review-level explanations. In *Proceedings of the 2018 World Wide Web Conference*. International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, Switzerland, 1583–1592.

[6] Chong Chen, Min Zhang, Yiqun Liu, and Shaoping Ma. 2019. Social attentional memory network: Modeling aspect- and friend-level differences in recommendation. In *Proceedings of the 11th ACM International Conference on Web Search and Data Mining*.

[7] Jingyuan Chen, Hanwang Zhang, Xiangnan He, Liqiang Nie, Wei Liu, and Tat-Seng Chua. 2017. 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, 335–344.

[8] Li Chen, Guanliang Chen, and Feng Wang. 2015. Recommender systems based on user reviews: The state of the art. *User Modeling and User-Adapted Interaction* 25, 2 (2015), 99–154.

[9] Xu Chen, Zheng Qin, Yongfeng Zhang, and Tao Xu. 2016. Learning to rank features for recommendation over multiple categories. In *Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 305–314.

[10] Zhiyong Cheng, Xiaojun Chang, Lei Zhu, Rose C. Kanjirathinkal, and Mohan Kankanhalli. 2019. MMALFM: Explainable recommendation by leveraging reviews and images. *ACM Transactions on Information Systems (TOIS)* 37, 2 (2019), 16.

[11] Zhiyong Cheng, Ying Ding, Xiangnan He, Lei Zhu, Xuemeng Song, and Mohan Kankanhalli. 2018. A^3NCF: An adaptive aspect attention model for rating prediction. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*. International Joint Conferences on Artificial Intelligence Organization, 3748–3754.

[12] Zhiyong Cheng, Ying Ding, Lei Zhu, and Mohan Kankanhalli. 2018. Aspect-aware latent factor model: Rating prediction with ratings and reviews. In *Proceedings of the 2018 World Wide Web Conference*. International World Wide Web Conferences Steering Committee, 639–648.

[13] Jin Yao Chin, Kaiqi Zhao, Shafiq Joty, and Gao Cong. 2018. ANR: Aspect-based neural recommender. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*. ACM, 147–156.

[14] Paolo Cremonesi, Yehuda Koren, and Roberto Turrin. 2010. Performance of recommender algorithms on top-N recommendation tasks. In *Proceedings of the 4th ACM Conference on Recommender Systems*. ACM, 39–46.

[15] Ruihai Dong and Barry Smyth. 2017. User-based opinion-based recommendation. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*. AAAI Press, 4821–4825.

[16] Travis Ebesu, Bin Shen, and Yi Fang. 2018. Collaborative memory network for recommendation systems. In *Proceedings of the 41st International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'18)*. ACM, New York, 515–524.

[17] Gayatree Ganu, Yogesh Kakodkar, and AméLie Marian. 2013. Improving the quality of predictions using textual information in online user reviews. *Information Systems* 38, 1 (2013), 1–15.

[18] Yoav Goldberg. 2017. Neural network methods for natural language processing. *Synthesis Lectures on Human Language Technologies* 10, 1 (2017), 1–309.

[19] Yangyang Guo, Zhiyong Cheng, Liqiang Nie, Xin-Shun Xu, and Mohan Kankanhalli. 2018. Multi-modal preference modeling for product search. In *Proceedings of the 2018 ACM on Multimedia Conference*. ACM.

[20] Ruining He and Julian McAuley. 2016. Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In *Proceedings of the 25th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 507–517.

[21] Xiangnan He, Tao Chen, Min-Yen Kan, and Xiao Chen. 2015. TriRank: Review-aware explainable recommendation by modeling aspects. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*. ACM, 1661–1670.

[22] Xiangnan He and Tat-Seng Chua. 2017. Neural factorization machines for sparse predictive analytics. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 355–364.

[23] Xiangnan He, Zhenkui He, Jingkuan Song, Zhenguang Liu, Yu-Gang Jiang, and Tat-Seng Chua. 2018. NAIS: Neural attentive item similarity model for recommendation. *IEEE Transactions on Knowledge and Data Engineering* 30, 12 (2018), 2354–2366.

[24] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural collaborative filtering. In *Proceedings of the 26th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 173–182.

[25] Richang Hong, Lei Li, Junjie Cai, Dapeng Tao, Meng Wang, and Qi Tian. 2017. Coherent semantic-visual indexing for large-scale image retrieval in the cloud. *IEEE Transactions on Image Processing* 26, 9 (2017), 4128–4138.

[26] Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv:1412.6980*.

[27] Y. Koren, R. Bell, and C. Volinsky. 2009. Matrix factorization techniques for recommender systems. *Computer* 42, 8 (2009), 30–37.

[28] Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *Proceedings of the International Conference on Machine Learning*. 1188–1196.

[29] Tie-Yan Liu. 2009. Learning to rank for information retrieval. *Foundations and Trends in Information Retrieval* 3, 3 (March 2009), 225–331.

[30] Xin Luo, Ye Wu, and Xin-Shun Xu. 2018. Scalable supervised discrete hashing for large-scale search. In *Proceedings of the 2018 World Wide Web Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 1603–1612.

[31] Julian McAuley and Jure Leskovec. 2013. Hidden factors and hidden topics: Understanding rating dimensions with review text. In *Proceedings of the 7th ACM Conference on Recommender Systems*. ACM, 165–172.

[32] Xuying Meng, Suhang Wang, Huan Liu, and Yujun Zhang. 2018. Exploiting emotion on reviews for recommender systems. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

[33] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv:1301.3781*.

[34] Weike Pan, Qiang Yang, Wanling Cai, Yaofeng Chen, Qing Zhang, Xiaogang Peng, and Zhong Ming. 2019. Transfer to rank for heterogeneous one-class collaborative filtering. *ACM Transactions on Information Systems* 37, 1 (Jan. 2019), Article 10, 20 pages.

[35] Štefan Pero and Tomáš Horváth. 2013. Opinion-driven matrix factorization for rating prediction. In *International Conference on User Modeling, Adaptation, and Personalization*. Springer, 1–13.

[36] Steffen Rendle. 2010. Factorization machines. In *Proceedings of the 2010 IEEE International Conference on Data Mining*. IEEE Computer Society, 995–1000.

[37] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian personalized ranking from implicit feedback. In *Proceedings of the 25th Conference on Uncertainty in Artificial Intelligence*. AUAI Press, 452–461.

[38] Gerard Salton, Anita Wong, and Chung-Shu Yang. 1975. A vector space model for automatic indexing. *Communications of the ACM* 18, 11 (1975), 613–620.

[39] Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research* 15, 1 (2014), 1929–1958.

[40] Jiwei Tan, Xiaojun Wan, Hui Liu, and Jianguo Xiao. 2018. QuoteRec: Toward quote recommendation for writing. *ACM Transactions on Information Systems* 36, 3 (2018), 34:1–34:36.
[41] Yunzhi Tan, Min Zhang, Yiqun Liu, and Shaoping Ma. 2016. Rating-boosted latent topics: Understanding users and items with ratings and reviews. In *Proceedings of the 25th International Joint Conference on Artificial Intelligence*. AAAI Press, 2640–2646.
[42] Shuaiqiang Wang, Shanshan Huang, Tie-Yan Liu, Jun Ma, Zhumin Chen, and Jari Veijalainen. 2016. Ranking-oriented collaborative filtering: A listwise approach. *ACM Transactions on Information Systems* 35, 2 (2016), 10:1–10:28.
[43] Xiang Wang, Xiangnan He, Fuli Feng, Liqiang Nie, and Tat-Seng Chua. 2018. TEM: Tree-enhanced embedding model for explainable recommendation. In *Proceedings of the 2018 World Wide Web Conference (WWW'18)*. International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, Switzerland, 1543–1552.
[44] Jun Xiao, Hao Ye, Xiangnan He, Hanwang Zhang, Fei Wu, and Tat-Seng Chua. 2017. 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, 3119–3125.
[45] Yinqing Xu, Wai Lam, and Tianyi Lin. 2014. Collaborative filtering incorporating review text and co-clusters of hidden user communities and item groups. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*. ACM, 251–260.
[46] Longqi Yang, Cheng-Kang Hsieh, Hongjian Yang, John P. Pollak, Nicola Dell, Serge Belongie, Curtis Cole, and Deborah Estrin. 2017. Yum-Me: A personalized nutrient-based meal recommender system. *ACM Transactions on Information Systems* 36, 1 (2017), 7:1–7:31.
[47] Wei Zhang and Jianyong Wang. 2016. Integrating topic and latent factors for scalable personalized review-based rating prediction. *IEEE Transactions on Knowledge and Data Engineering* 28, 11 (2016), 3013–3027.
[48] Yongfeng Zhang, Qingyao Ai, Xu Chen, and W. Bruce Croft. 2017. 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, 1449–1458.
[49] Yongfeng Zhang, Guokun Lai, Min Zhang, Yi Zhang, Yiqun Liu, and Shaoping Ma. 2014. Explicit factor models for explainable recommendation based on phrase-level sentiment analysis. In *Proceedings of the 37th International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 83–92.
[50] Yongfeng Zhang, Haochen Zhang, Min Zhang, Yiqun Liu, and Shaoping Ma. 2014. Do users rate or review?: Boost phrase-level sentiment labeling with review-level sentiment classification. In *Proceedings of the 37th International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 1027–1030.
[51] Lei Zheng, Vahid Noroozi, and Philip S. Yu. 2017. Joint deep modeling of users and items using reviews for recommendation. In *Proceedings of the 10th ACM International Conference on Web Search and Data Mining*. ACM, 425–434.

Received September 2018; revised December 2018; accepted January 2019