

1 Attentive Aspect Modeling for Review-aware 2 Recommendation 3

4 XINYU GUAN, Xi'an Jiaotong University, P. R. China
5

6 ZHIYONG CHENG, National University of Singapore, Singapore
7

8 XIANGNAN HE, National University of Singapore, Singapore
9

10 YONGFENG ZHANG, Rutgers University, USA
11

ZHIBO ZHU, Xi'an Jiaotong University, P. R. China
12

QINKE PENG, Xi'an Jiaotong University, P. R. China
13

TAT-SENG CHUA, National University of Singapore, Singapore
14

15 In recent years, many studies extract aspects from user reviews and integrate them with ratings for improving
16 the recommendation performance. The common aspects mentioned in a user's reviews and a product's reviews
17 indicate indirect connections between the user and product. However, these aspect-based methods suffer
18 from two problems. First, the common aspects are usually very sparse, which is caused by the sparsity of
19 user-product interactions and the diversity of individual users' vocabularies. Second, a user's interests on
20 aspects could be different with respect to different products, which are usually assumed to be static in existing
21 methods. In this paper, we propose an Attentive Aspect-based Recommendation Model (AARM) to tackle these
22 challenges. For the first problem, to enrich the aspect connections between user and product, besides common
23 aspects, AARM also models the interactions between synonymous and similar aspects. For the second problem,
24 a neural attention network which simultaneously considers user, product and aspect information is constructed
to capture a user's attention towards 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 top-N recommendation task.
25

26 CCS Concepts: • Information systems → Collaborative filtering; Recommender systems;
27

28 Additional Key Words and Phrases: top-N recommendation, neural network, reviews, aspects.
29

30 ACM Reference Format: 31

32 Xinyu Guan, Zhiyong Cheng, Xiangnan He, Yongfeng Zhang, Zhibo Zhu, Qinke Peng, and Tat-Seng Chua.
33 2018. Attentive Aspect Modeling for Review-aware Recommendation. 1, 1 (September 2018), 26 pages.
34 <https://doi.org/10.1145/nnnnnnnn.nnnnnnnn>
35

36 Authors' addresses: Xinyu Guan, Xi'an Jiaotong University, Systems Engineering Institute, Xi'an, 710049, P. R. China,
37 xinyu_guan@foxmail.com; Zhiyong Cheng, National University of Singapore, School of Computing, Singapore, 117417,
38 Singapore, jason.zy.cheng@gmail.com; Xiangnan He, National University of Singapore, School of Computing, Singapore,
39 117417, Singapore, xiangnanhe@gmail.com; Yongfeng Zhang, Rutgers University, Department of Computer Science, New
40 Brunswick, 08901, USA, zhangyf07@gmail.com; Zhibo Zhu, Xi'an Jiaotong University, Systems Engineering Institute, Xi'an,
41 710049, P. R. China, zdh96.zzb@stu.xjtu.edu.cn; Qinke Peng, Xi'an Jiaotong University, Systems Engineering Institute, Xi'an,
42 710049, P. R. China, qkpeng@mail.xjtu.edu.cn; Tat-Seng Chua, National University of Singapore, School of Computing,
43 Singapore, 117417, Singapore, dcscts@nus.edu.sg.

44 Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee
45 provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the
full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored.
Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires
prior specific permission and/or a fee. Request permissions from permissions@acm.org.
46

47 © 2018 Copyright held by the owner/author(s). Publication rights licensed to ACM.
48 XXXX-XXXX/2018/9-ART \$15.00
49 <https://doi.org/10.1145/nnnnnnnn.nnnnnnnn>

50 1 INTRODUCTION

51 Recommender systems help users find their potentially interested products from an enormous list of
 52 products. Matrix Factorization (MF) methods [22] are widely adopted in recommendation systems
 53 because of its accuracy and scalability. MF methods usually rely on the explicit (e.g., user ratings)
 54 or implicit (e.g., click behaviors) interactions between users and products for recommendation.
 55 However, a rating or binary interaction can only reflect the user's overall attitude towards a product,
 56 which does not include information about the underlying reasons for the user behavior. As a result,
 57 it is difficult for MF methods to model user's fine-grained preferences on specific product features
 58 and provide explanation to recommendations.

59 To tackle these limitations, researches have attempted to utilize reviews to alleviate the data
 60 sparsity problem and provide more explainable recommendations [3, 6, 9, 17]. As accompanying
 61 information of ratings, the textual review expresses user's opinions on different product features,
 62 and thus contains more fine-grained information about the user preference. Different strategies
 63 have been applied to incorporate reviews into MF models, including sentiment analysis [26],
 64 representation learning [4, 37], and topic models [24, 32]. Although these methods have achieved
 65 some progress, the generated vector representations of users and products are still latent and thus
 66 cannot explicitly model user's preference on specific product features, which could impede their
 67 performance.

68 Another direction is to leverage the aspects mentioned in user reviews for recommendation.
 69 In this paper, **aspect** is defined as the words or phrases used by users in their product reviews
 70 to describe product features. For example, "battery life" and "battery duration" are two different
 71 aspects while they refer to the same product feature. There are already some methods which detect
 72 aspects in user reviews and leverage them to model user's fine-grained preferences to specific
 73 product features [12, 38]. For example, EFM [38] conducted aspect-level sentiment analysis to
 74 extract user's preference and product's quality on specific product feature, then incorporated the
 75 results into an MF framework to provide more accurate recommendation. SULM [1] and LRPPM
 76 [7] went beyond EFM [38] by using more effective methods to identify the impact of each aspect
 77 on the overall rating. However, these methods rely highly on the accuracy of external sentiment
 78 analysis tools.

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

87 Motivated by the above concerns, in this paper, we propose an Attentive Aspect-based Rec-
 88 commendation Model (AARM), which can effectively tackle the above two problems. For the first
 89 problem of *aspect sparsity*, AARM models the interactions between synonymous and similar as-
 90 pects to alleviate it, where *synonymous aspects* are the ones referring to the same product feature
 91 (e.g., "battery life" and "battery duration"); and *similar aspects* are those of different features that
 92 are closely related (e.g., "battery life" and "battery capacity"). Intuitively, a user's attention to an
 93 unmentioned aspect can be inferred through its similar aspects. For instance, a user who cares
 94 about "battery capacity" of cellphones may also care about its "charging speed", although "charging
 95 speed" has never been mentioned in this user's reviews. In our model, an aspect extracted from
 96 reviews is first represented as an embedding vector. Then a user u 's satisfaction about v according

99 to aspect a is estimated by calculating the interactions between a and all the aspects mentioned in
 100 v 's reviews. And an attention module is designed to pick up the interactions between meaningful
 101 aspect pairs. In this way, we achieve the goal of capturing the interactions between synonymous
 102 and similar aspects.

103 For the second problem of *identifying user's varied interests on aspects*, AARM introduces another
 104 attention module which takes user, product and aspect information into consideration. In this way,
 105 user's varied interests on aspects can be captured by the product-dependent user attention. Instead
 106 of rating prediction, we target the top-N recommendation task with a pair-wise learning-to-rank
 107 method, which is the most practically used recommendation scenario in real-world systems [11, 33].
 108 To this end, our model estimates a user u 's satisfaction towards an product v by (1) estimating v 's
 109 performances on u 's concerned aspects; and (2) identifying the impacts of these aspects on the
 110 overall satisfaction.

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

- 116 • We propose a novel recommendation method to model the interactions between both the
 117 same and the different aspects, which helps to alleviate the aspect sparsity problem in reviews.
 118 To the best of our knowledge, this is the first attempt to model the interactions between
 119 different aspects to model user preferences in recommendation.
- 120 • We design an attention mechanism in AARM to capture user's varied attentions on different
 121 aspects towards various products, which has been demonstrated to be very effective on
 122 improving the recommendation accuracy.
- 123 • We conduct extensive experiments on real-world datasets to demonstrate the effectiveness of
 124 our model. Experimental results show that our method can achieve superior performance by
 125 a large margin.

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

132 2 RELATED WORK

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

139 2.1 Review-level Methods

140 Review-level methods treat the review as a single piece of information and incorporate it with
 141 ratings. The opinion-driven matrix factorization [26] calculates the overall opinion of a review by
 142 summing up the orientations of opinion words in the text, and then combines it with numerical
 143 ratings for rating prediction. Some methods concatenate all the reviews belonging to a user (or item)
 144 as a user (or item) document, and then employ deep learning methods to learn the continuous vector
 145

representation for the user (or item) [4, 15, 37, 40]. For example, Transnets [4] and DeepCoNN [40] process the user and item documents with convolutional neural network to generate the vector representation for users and items. JRL [37] adopts the PV-DBOW model [23], which is an unsupervised methods 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 recommendation.

2.2 Topic-level Methods

Topic-level methods build probabilistic graphical model to extract topics from reviews. HFT [24] associates topic vectors from reviews with latent factors from ratings to improve rating prediction accuracy. Subsequently, some studies employed more complicated topic models and combined them with matrix factorization model for rating prediction. For example, CMR [35] is a probabilistic graphical model which simultaneously associates the review text and the hidden user communities and item group relationship with numerical ratings. RBLT [32] 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 the vanilla matrix factorization model for rating prediction. 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 user's preference towards 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 paper falls into this category. Ganu et al. [13] manually defined six aspects and four sentiments for restaurant reviews and used a regression-based method for rating prediction. Zhang et al. [38] employed an unsupervised tool for aspect extraction and aspect-level sentiment analysis. Aspect and sentiment outputs from this step were integrated with matrix factorization methods for rating prediction. Chen et al. [7] proposed a tensor-matrix factorization 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. More recently, Cheng et al. [10] proposed a combination of topic-level and aspect-level method, called aspect-aware topic model (ATM). It models user's preferences and product's qualities on aspects and estimate the importance of each aspect for the user. The aspect importance is then integrated into an aspect-aware matrix factorization model for rating prediction task.

197 Specifically, Chen et al. [7] and Cheng et al. [9] also focused on users' varied interests towards
 198 aspects, while [7] paid more attention to a user's interest variation over different categories and [9]
 199 focused on rating prediction task. He et al. [17] did not conduct sentiment analysis but adopted
 200 the aspect frequency information in reviews to construct the user-item-aspect tripartite graph for
 201 recommendation. The improved performance in [17] from baselines verified that the aspect mention
 202 signals in reviews could have already been able to reflect user's interests on aspects. Similarly,
 203 in AARM we do not conduct sentiment analysis on reviews explicitly, which helps to simplify
 204 the model design and implementation. Moreover, AARM considers both the interactions between
 205 different aspects and the user's varied preference towards aspects, which are neglected by previous
 206 studies.

2.4 Attention Mechanism

209 In recent years, many deep learning-based recommendation methods have been proposed and
 210 achieved good performance in many tasks [8, 18, 20, 31, 36]. The attention mechanism which can
 211 assign adaptive weights for a set of features has also been employed in recommendation models
 212 [2, 5, 19]. For example, ACF [5], which focuses on multimedia recommendation, uses a component-
 213 level attention module to find informative components for multimedia items (images/videos), and
 214 a item-level attention module to select representative items to represent users' preferences.
 215 AFM [34], which is an extension of FM machine [18, 27], uses an attention neural network to
 216 discriminate the importance of different feature interactions. Compared with these methods, we
 217 specially design two attention modules for the fine-grained modeling of product features extracted
 218 from user reviews. The user-level attention module in AARM is built to find out the user's most
 219 concerned product features for a candidate product, while the aspect-level attention module is
 220 constructed to select informative aspect interactions.

3 ATTENTIVE ASPECT-BASED RECOMMENDATION MODEL

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

3.1 Preliminaries

227 Given a user set $U = \{u_1, u_2, \dots, u_{|U|}\}$ and a product set $V = \{v_1, v_2, \dots, v_{|V|}\}$, AARM estimates a
 228 satisfaction score \hat{y}_{uv} for an user u towards a product v . The candidate products are then ranked
 229 in a descending order of \hat{y} and the top N products are recommended to u . In our method, aspects
 230 extracted from user reviews are used as the explicit features of users and products. We define
 231 $A = \{a_1, a_2, \dots, a_{|A|}\}$ as the aspect set of the dataset. The aspects that have been mentioned in the
 232 reviews of user u is represented as A_u , which is a subset of A . Similarly, product v 's aspects that
 233 have been mentioned in v 's reviews are represented as A_v . Product v 's rating given by user u is
 234 denoted as $r_{uv} \in R$, where R is the collection of ratings.

235 The structure of AARM is shown in Figure 1. In the input layer, users and products are represented
 236 as binarized sparse vectors using the one-hot encoding method. Above the input layer, the Aspect
 237 Interactions part is used to model the interactions between the aspects from user u 's aspect set A_u
 238 and the aspects from the product v 's aspect set A_v . Because a user's review for a product may not
 239 cover all the factors which can influence the user's satisfaction towards the product, the aspects
 240 extracted from review text may not be able to fully explain the rating. Hence the Global Interactions
 241 part is stacked above the input layer to model the implicit factors which influence user's decision
 242

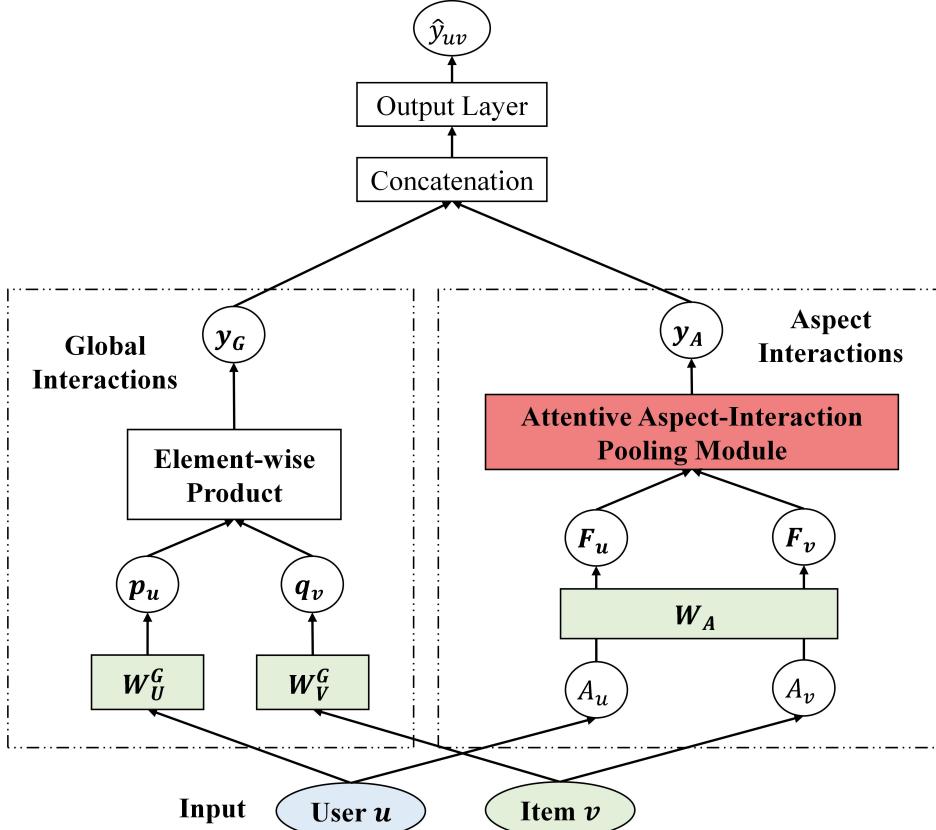


Fig. 1. Attentive Aspect-based Recommendation Model

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 represents 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 i th aspect in A_u is projected to the i th 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 y_A to represent the preference information in user reviews.

3.2.1 Aspect Extraction. Because the main contribution of this paper focuses on how to leverage aspects for personalized recommendation, we refer to external tools for aspect extractions. In this

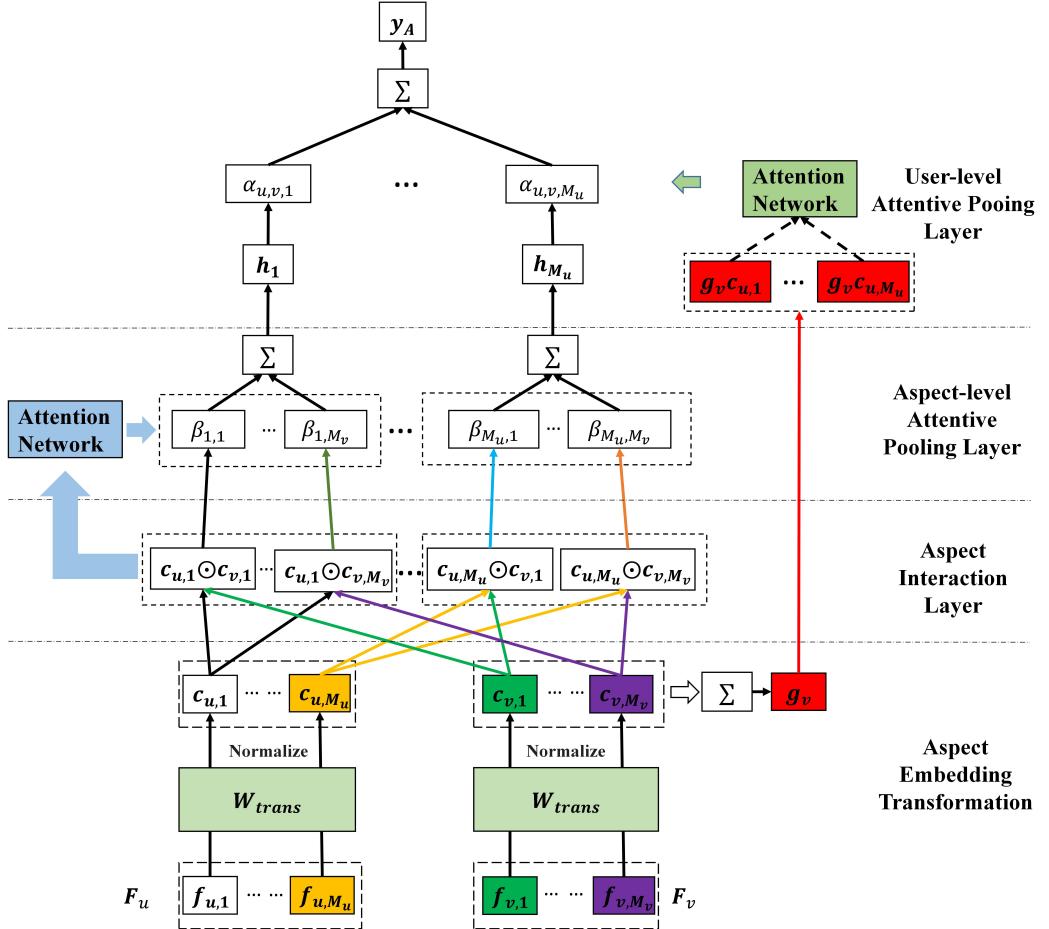


Fig. 2. The Attentive Aspect-Interaction Pooling Module.

paper, we use the Sentires¹, which has been successfully used in [38, 39] 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 paper. 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 set into the same length M_u and pad all the product aspect set into the same length M_v . Taking user aspect set as example, we define a meaningless aspect $<PAD>$ and add it to the end of user aspect sets whose lengths are less than the predefined size M_u . For A_u whose length is larger than M_u , we calculate the *TF-IDF* score [29] 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

¹<http://yongfeng.me/software/>

344 defined as:

$$345 \quad tfidf_u(a) = \frac{tf_u(a)}{\sum_{i \in A_u} tf_u(i)} \cdot \ln \frac{|U|}{df(a) + 1} \quad (1)$$

346 where $tf_u(a)$ is the frequency of a 's occurrence in u 's reviews, $|U|$ is the number of users, and
 347 $df(a)$ is the number of users who mentioned a . All the product aspect sets are padded into the
 350 same length M_v in a similar way.

351 3.2.2 *Attentive Aspect-Interaction Pooling Module*. As shown in Figure 2, given \mathbf{F}_u and \mathbf{F}_v as
 352 input, there are four parts in this module: *aspect embedding transformation*, *aspect interaction layer*,
 353 *aspect-level attentive pooling layer*, and *user-level attentive pooling layer*. The final output of this
 354 module is the vector $\mathbf{y}_A(u, v)$ which represents the overall satisfaction of a user u towards a product
 355 v estimated with review text. In this module, we hold the assumption that u 's overall satisfaction for
 356 v is based on v 's performances on u 's concerned aspects (i.e. aspects from A_u). This module works
 357 as follows. First, for each aspect $a \in A_u$, the aspect interaction layer and aspect-level attentive
 358 pooling layer are employed to estimate v 's performance on a , where the performance is represented
 359 as vector $\mathbf{h}_a(u, v)$. Then the user-level attentive pooling layer is used to estimate u 's preference
 360 towards v by integrating $\mathbf{h}_a(u, v)$ for all the aspect $a \in A_u$ and represent the preference as a vector
 361 $\mathbf{y}_A(u, v)$. Finally, $\mathbf{y}_A(u, v)$ will be combined with the result of Global Interaction part and further
 362 input into the output layer to estimate the user u 's satisfaction score towards the product v .

363 **Aspect Embedding Transformation.** To model the interactions between synonymous and
 364 related aspects, we expect the vector representation of aspect to encode the similarity relation
 365 between aspects. In this paper, the Word2vec model [25], which is able to encode many linguistic
 366 regularities and patterns, is used to pre-train aspect embeddings with the review texts in each
 367 dataset. Instead of directly tuning aspect embeddings with the recommendation task, a trainable
 368 matrix $\mathbf{W}_{trans} \in \mathbb{R}^{d_a \times d_a}$ is defined to customize the pre-trained aspect embedding \mathbf{f} to make it
 369 oriented towards our recommendation task. Then these customized embeddings are normalized as:
 370

$$371 \quad \mathbf{c} = \frac{\mathbf{W}_{trans}\mathbf{f}}{\|\mathbf{W}_{trans}\mathbf{f}\|}. \quad (2)$$

373 Here $\|\mathbf{x}\|$ is the Euclidean norm of \mathbf{x} . In this paper, aspect interaction between two aspects is defined
 374 as the element-wise product of their embedding vectors. By normalizing the aspect embeddings
 375 with their corresponding Euclidean norms, the calculation of interaction between two aspects
 376 is similar to calculating their cosine similarity. As illustrated in [25], if two words have higher
 377 semantics and syntax similarities, their embeddings generated by Word2vec would have larger
 378 cosine similarity. In this way, the results of aspect interactions are associated with the semantics and
 379 syntax relations between aspects, which helps in identifying the synonymous and related aspects.
 380 Alternatively, we can also directly tune the aspect embedding matrix \mathbf{W}_A during the training of
 381 AARM for top-N recommendation. We will compare the performances of these different settings in
 382 the experiment section.

383 **Aspect Interaction Layer.** This layer maps the vector representations of aspects in A_u and
 384 A_v to a set of d_a -dimensional interacted vectors. The aspect interaction between aspect $i \in A_u$
 385 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
 386 output of the aspect interaction layer can be represented as a set of vectors:

$$387 \quad f_{AI}(u, v) = \{\mathbf{c}_i \odot \mathbf{c}_j(x_i x_j)\}_{i \in A_u, j \in A_v}. \quad (3)$$

388 Here $x_i \in \{0, 1\}$ is the masking indicator, where $x_i = 0$ if i is the meaningless aspect $< PAD >$
 389 (defined for padding). To implement the masking operation in AARM, we define an aspect masking
 390 vector $\mathbf{W}_{mask} \in \mathbb{R}^{d_a \times |A|}$, where the column of aspect $< PAD >$ is a zero vector, and the columns of
 392

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 $< PAD >$ is transformed into a zero vector. In this way, we make sure that the interactions between aspect $< PAD >$ 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 [25]. Inspired by this, for aspect pair i and j , the input of attention layer is defined as the element-wise product of their normalized embedding vector \mathbf{c}_i and \mathbf{c}_j . And the aspect-level attention layer is defined as:

$$\hat{\beta}_{i,j} = \mathbf{l}_{att_1}^T (\mathbf{c}_i \odot \mathbf{c}_j) (x_i x_j),$$

$$\beta_{i,j} = \frac{\exp(\hat{\beta}_{i,j})}{\sum_{x \in A_u} \sum_{y \in A_v} \exp(\hat{\beta}_{xy})}. \quad (4)$$

Here $\mathbf{l}_{att_1} \in \mathbb{R}^{d_a}$, 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:

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

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 estimation of the user's overall satisfactory 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 towards a 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 towards 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 an aspect a with respect to a product v is measured by the similarities between a and the aspects that has 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:

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

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:

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

Here $\mathbf{l}_{att_2} \in \mathbb{R}^{d_a}$, 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 .

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 towards product v :

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

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:

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

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:

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

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 towards product v .

3.5 Learning

In this paper, we binarize the ratings scores and train AARM with a pair-wise learning-to-rank method. This makes AARM suitable for recommendation with implicit feedbacks. Given a user u , a triple (u, v^+, v^-) is constructed for pair-wise 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 higher

satisfaction score to the positive pair (u, v^+) than the negative pair (u, v^-) . Hence, the Bayesian Personalized Ranking (BPR) optimization criterion is employed as the objective function of AARM:

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

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 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 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:

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

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 [30] to reduce overfitting. Dropout can prevent complex co-adaptations on training data by randomly dropping some units during training [30]. 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 implementation² 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.

²<https://radimrehurek.com/gensim/>

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	2865	14.72	32.24
CDs and Vinyl	4033	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

4.1 Datasets

We use the "5-core" subsets from the publicly accessible "Amazon product dataset"³ [16] for experiments. Here the "5-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 pair-wise learning-to-rank framework, ratings are binarized to construct positive user-product pairs. We adopt five different product categories from the "Amazon product dataset", i.e., '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 training set, while the rest of 30% records are put into test set. Particularly, we use the exact same splits and evaluation measures as the experimental settings in [37]⁴. This is to guarantee that all the methods are evaluated on exactly the same settings for fair comparisons.

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

³<http://jmcauley.ucsd.edu/data/amazon/>

⁴We would like to thank the authors for sharing us with the datasets and specific splits.

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 musics, 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

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 paper, 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 [37], 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 has been purchased by the user:

$$\text{Recall} = \frac{n_{tp}}{n_{gt}}, \quad (13)$$

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 which has been purchased by the user in the top-N recommendation list:

$$\text{Precision} = \frac{n_{tp}}{N}. \quad (14)$$

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):

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

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:

$$\text{NDCG} = \frac{\text{DCG}}{\text{IDCG}}. \quad (16)$$

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.

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

$$640 \quad 641 \quad HT = \frac{n_{hit}}{|U|}. \quad 642 \quad (17)$$

643 4.4 Baselines and Parameter Settings

644 We compare our method AARM with the following baselines.

645 **BPR-MF** [28]. The matrix factorization (MF) based on Bayesian Personalized Ranking (BPR),
 646 which combines MF-model with a pair-wise learning to rank loss function, is a solid baseline for
 647 top-N recommendation. Only user-product interaction data is used in this method.

648 **BPR-HFT** [24]. The Hidden Factor and Topics (HFT) model associates topics extracted from
 649 reviews with latent factors learned from numerical ratings. It is one of the state-of-the-art review-
 650 based recommendation methods. The original HFT model is a rating prediction method. BPR-HFT
 651 [37] modifies HFT by adding a Bayesian Personalized Ranking loss on top of HFT to generate the
 652 top-N recommendation.

653 **DeepCoNN** [40]. The Deep Cooperative Neural Network is one of the state-of-the-art deep learning-
 654 ing methods for recommendation which utilizes reviews to build user and product representations.
 655 It uses the review-based user and product representations for rating prediction.

656 **JRL** [37]. The Joint Representation Learning model is a state-of-the-art method which integrates
 657 different information sources with deep learning methods for top-N recommendation. Textual
 658 reviews, product images and numerical ratings are jointly used in JRL.

659 **JRL-Review** [37]. JRL-Review is a single-view version of JRL which incorporates textual reviews
 660 for top-N recommendation. JRL-Review employs PV-DBOW model [23] to learn the vector repre-
 661 sentations of users and products from their corresponding reviews. It is one of the state-of-the-art
 662 review-based recommendation methods.

663 **eJRL** [37]. eJRL is another variant of JRL which jointly utilizes textual reviews, product images
 664 and numerical ratings for recommendation. The difference between them is that eJRL prevents
 665 information propagation among different information sources.

666 The hyper-parameters of baselines are tuned on training set with five-fold cross-validation. In
 667 particular, the dimension of latent factors (or embeddings) for BPR-MF, BPR-HFT and DeepCoNN
 668 is 100. For BPR-HFT, the number of topics is 10. For JRL, JRL-Review and eJRL, the embedding size
 669 is set as 300.

670 **Parameter Settings.** We implemented our methods with Tensorflow⁵. When padding user
 671 aspect set to the same size, the maximum size M_u was defined as the 75% quantile of the sizes
 672 of all user aspect sets. Similarly, the maximum size M_v of product aspect set was defined as the
 673 75% quantile of the sizes of all product aspect sets. For embedding layers, we set the dimension
 674 d_g of user and product embeddings in the global interactions module to 128; set the dimension
 675 d_a of aspect embeddings to 128. AARM was optimized with mini-batch Adam [21] because Adam
 676 uses adaptive learning rates for parameters with different update frequencies and converges faster
 677 than vanilla stochastic gradient descent. We tested the learning rate of [0.001, 0.003, 0.01]. For the
 678 coefficient of L^2 regularization, [0.0, 0.0001, 0.01, 0.1] was tested. To prevent overfitting, in dropout
 679 layers, the dropout rate was set to 0.5.

680 The model was trained for a maximum of 300 epochs with early stopping. To build the validation
 681 set, 1000 users are randomly selected from the users in the training set. For each user, one of his
 682 purchased products is randomly drawn from training set as the ground truth product in validation
 683 set. And when evaluating the model on the validation set, for each user, all the products which are

684
 685 ⁵<https://www.tensorflow.org/>

Table 4. The NDCG and hit ratio (HT) results of baselines and the proposed method for RQ1. 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 3 rows.

Measures(%)	Movies		CDs		Clothings		Cell Phones		Beauty	
	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
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
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

Table 5. The corresponding recall and precision results of baselines and the proposed method.

Measures(%)	Movies		CDs		Clothings		Cell Phones		Beauty	
	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
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
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

not paired with the user in training set are added to the candidate set. Then to build 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 epoch, 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.

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 shallow methods (BPR-MF and BPR-HFT), review-based deep learning methods (DeepCoNN and JRL-Review), multi-modal deep learning 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 [37], we directly reproduce their results of shallow methods, review-based deep learning methods and multi-modal deep learning methods for fair comparisons. From Tables 4 and 5, we can see that:

(1) All methods that incorporate reviews for recommendation outperform BPR-MF which only using user-product interaction data. This shows that reviews is an important information source to boost recommendation performance. For methods which merge review information with user-product interaction information, deep learning-based methods such as DeepCoNN and JRL-Review achieve better results than that of the shallow method BPR-HFT. This might be attributed to the better representational capacity of neural models.

(2) Our proposed method AARM outperforms all the shallow methods and review-based deep learning methods on all the datasets in terms of different metrics. Compared to these baselines, AARM make 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 towards aspects. In the following sections, we further analyze how the specific designs of AARM boost its recommendation performance.

(3) 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 leveraged 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 variant, which is termed as "A_Inter" in Figure 3, under the same experimental settings.

As a variant of AARM, the differences between AARM and A_Inter are in Aspect Interactions part. Given a user u and a product v , A_Inter only considers the interactions between shared aspects of u and v , i.e., $a \in A_u \cap A_v$. 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 $a \in A_u$, the Equations (3), (4) and (5) of AARM are replaced with the following equation:

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

Here $x_i \in \{0, 1\}$ is an indicator, where $x_i = 0$ if i is the meaningless aspect $< PAD >$ defined for padding. As A_Inter only considers interactions between the same aspects, no aspect-level attention module is used here. We verify the effect of interactions between different aspects by comparing the performance of A-Inter and AARM on top-N recommendation task. The results of the two methods on the five experimental datasets are shown in Figure 3. All the experimental settings are kept the same to ensure the reliability of results. As shown in Figure 3, AARM substantially outperforms A_Inter 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

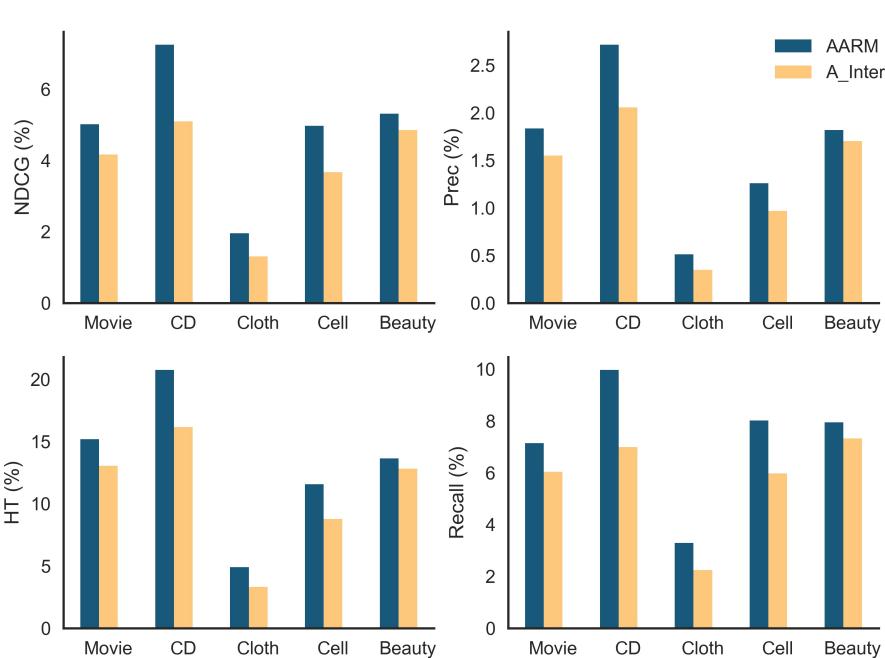


Fig. 3. **Performance of AARM and its variant A_Inter on five datasets for RQ2.** Due to limitation of space, we present dataset ‘Movies and TV’ as ‘Movie’, ‘CDs and Vinyl’ as ‘CD’, ‘Clothing, Shoes and Jewelry’ as ‘Cloth’, ‘Cell Phones and Accessories’ as ‘Cell’ for short.

precision. As the only difference between A-Inter and AARM is by replacing Equations (3), (4) and (5) with Equation (18), the results demonstrate the importance of modeling the interactions between different aspects and the effectiveness of our specially designed aspect-level attentive pooling layer. We will further perform 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 user’s interests towards aspects are varied among different products. And an user-level attentive pooling layer (Equation (6), (7) and (8)), which simultaneously considers user, product and aspect information, is designed to capture user’s different biases towards 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 top-N recommendation task under the same settings.

The differences between AARM, A_Static and No-UserAtt are in the design of user-level attention module. A_Static also assumes that user’s interests towards different aspects are different. But different from AARM, A_Static assumes that a user’s interests towards 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 towards its

Table 6. The NDCG and hit ratio (HT) results of AARM and its variants on five datasets for RQ3. 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).

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

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								
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

aspects, different from AARM, the input of the aspect $a_i \in A_u$ is designed as:

$$\begin{aligned} \mathbf{x}_{u,i} &= \mathbf{g}_u \odot \mathbf{c}_i, \\ \mathbf{g}_u &= \sum_{j \in A_u} \mathbf{c}_j. \end{aligned} \quad (19)$$

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:

$$\begin{aligned} \hat{\alpha}_{u,i} &= \mathbf{l}_{att_2}^T \mathbf{x}_{u,i}, \\ \alpha_{u,i} &= \frac{\exp(\hat{\alpha}_{u,i})}{\sum_{j \in A_u} \exp(\hat{\alpha}_{u,j})}. \end{aligned} \quad (20)$$

Here $\mathbf{l}_{att_2} \in \mathbb{R}^{d_a}$, and $\alpha_{u,i}$ represents the importance of aspect $a_i \in A_u$ with respect to the user u . From Equations (19) and (20), 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 :

$$\mathbf{y}_A(u, v) = \sum_{j \in A_u} \mathbf{h}_j. \quad (21)$$

As shown in Tables 6 and 7, AARM outperforms A_static and No-UserAtt on all the datasets and on all the measures. Remind that the only differences between AARM and A_static are the different

Table 8. The NDCG and hit ratio (HT) results of AARM and its variants on five datasets for RQ4. 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 Pretrain+Tune (Random vs. Pretrain).

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
Pretrain+Tune	4.764	14.320	6.744	19.905	0.802	2.046	4.210	10.191	4.658	12.266
Random vs. Pretrain	-3.296	-2.311	-0.519	-2.321	85.411	77.713	3.420	1.227	2.920	5.756

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								
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
Pretrain+Tune	6.744	1.719	9.270	2.616	1.346	0.216	7.012	1.110	6.969	1.647
Random vs. Pretrain	-3.692	-3.025	-3.376	-7.187	83.952	76.852	2.125	2.252	4.577	3.582

assumptions about user attentions on aspects towards 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 attentions 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 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 [25] 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 [14].

There are also other two 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

Table 10. The NDCG and hit ratio (HT) results of AARM and its variants on five datasets for RQ5. 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).

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

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								
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

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 is presented in Tables 8 and 9 in the row of “Pretrain+Tune”.

As shown in Tables 8 and 9, AARM with the “pretraining + trainable linear transformation” strategy outperforms Random+Tune and Pretrain+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 them. The capability of enabling similar words shifted similarly makes the “pretraining + trainable linear transformation” strategy more suitable for our task.

Comparing the performance of Random+Tune with Pretrain+Tune in Tables 8 and 9, we can find that Pretrain+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 Pretrain+Tune strategy may not be able to transform the pre-trained embeddings for the new task and thus lose the original similarity between words[14]. Random+Tune strategy which assigns a much smaller random initial values to 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 Aspect Interactions part into

981
982
983
984
**Table 12. The distributions of the number of shared aspects between a user and a product on the
five datasets. From left to right, the columns present the ratios of different user-product pairs
which have specific numbers of shared aspects. Specially, the last column represents the ratio of
user-product pairs which have more than five shared aspects.**

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%

992
993
994 the output layer. This variant of AARM is referred as “Aspect Part” in Tables 10 and 11. Similarly,
995 another variant of AARM which is referred as “Global Part” in Tables 10 and 11 is constructed by
996 removing the Aspect Interactions part from AARM to verify the effect of Global Interactions Part.
997

998 From Tables 10 and 11, we can find that AARM significantly outperforms Aspect Part and Global
999 Part. This result indicates that our combination strategy based on concatenation is valid. And the
1000 Global Interactions part, which is designed to capture the user preferences that have not been
1001 mentioned in review texts, is an effective complement to the Aspect Interactions part.

1001 As compared with Global Part, Aspect Part performs better in two datasets while falls behind
1002 in the other three datasets. Because Aspect Part connects users and products via the interactions
1003 between their aspects, its performance may be influenced by the number of interactions between
1004 related aspects. To verify this viewpoint, We traverse all the users and products in a dataset to
1005 construct all the possible user-product pairs, and then count the number of shared aspects of each
1006 user-product pair. A shared aspect of a user-product pair is a aspect which has been mentioned in
1007 both the user and the product’s reviews. The distributions of the number of shared aspects of each
1008 user-product pair on the five datasets are shown in Table 12.

1009 From Tables 10, 11 and 12, we can find that Aspect Part usually performs better on datasets
1010 which have more shared aspects between each user-product pair in general. For example, Aspect
1011 Part substantially outperforms Global Part in “CDs and Vinyl” and “Clothing, Shoes and Jewelry”
1012 datasets which have the smallest ratios of 0 shared aspects (see the 2nd column in the table). And
1013 for datasets “Movies and TV”, “Cell Phones and Accessories” and “Beauty” where more than 20%
1014 user-product pairs do not have any shared aspects, Global Part outperforms Aspect Part.

1015 4.10 Case Study of Attention Layers

1016 The user-level and aspect-level attention modules are important parts of AARM. The user-level
1017 attention module (refer to Equation 7) is employed to capture user’s varied preferences on aspects.
1018 And the aspect-level attention module (refer to Equation 4) is designed to enhance the interactions
1019 between meaningful aspect pairs, like the interactions between the same or similar aspects, and
1020 reduce the influence of the interactions between the two irrelevant aspects. To illustrate the roles
1021 of these two attention modules in AARM, we randomly selected some examples for qualitative
1022 analysis.

1023 In Table 13, we show the user-level attention values of a user ‘A1P9UMP1XSE6MI’ in “Cell
1024 Phones and Accessories” dataset when examining different products. The first column is the ids
1025 of four products in the dataset and their aspect sets. Each product has 15 aspects which is the
1026 75% quantile of the sizes of all product aspect sets in the dataset. The rest of columns show the
1027 aspects of the user (the second row from top to bottom) and the attention values that assigned to
1028

1030
1031 Table 13. A case study of the user-level attention module. The first column (from the left) shows
1032 ids and aspect sets of four products from the “Cell Phones and Accessories” dataset. The rest of
1033 columns show the aspects of the user (the second row from top to bottom) and the attention values
1034 assigned to these aspects when facing aforementioned four products. In each row, the aspects
1035 mentioned in both the user and product’s reviews and their corresponding attention values are
1036 highlighted in red.
1037

Products and Their Aspects	Aspects of User A1P9UMP1XSE6MI								
	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

1048
1049 these aspects when facing aforementioned four products. From each product’s aspect set, we can
1050 find that product ‘B00EOE6FUW’ is a ‘usb charger’, ‘B005HS5MKS’ is a ‘bluetooth earpiece’, and
1051 ‘B002VPE1NO’ and ‘B00E8GYIRI’ are the ‘shell case’ of cell phones. The shared aspects of each
1052 user-product pair and corresponding attention values are highlighted in red.

1053 As shown in Table 13, when examining a product, the user-level attention module can find the
1054 aspects which are related to the product and assign higher attention values to them. First, all the
1055 shared aspects (highlighted in red) of each user-product pair are assigned much higher attention
1056 values. Second, the user-level attention module can assign higher values to aspects that are related
1057 to the product but have not been mentioned in the product’s reviews. For example, when examining
1058 the shell cases ‘B002VPE1NO’ and ‘B00E8GYIRI’, ‘grab’ is assigned higher weight although it is not
1059 in the product’s aspect set. This is because that there are some related aspects of ‘grab’ in the two
1060 products’ aspect sets which are captured by our attention module (refer to Figure 4).

1061 The examples in Table 13 indicate why AARM can outperform A_Static and No-UserAtt (refer to
1062 Tables 6 and 7). The user’s aspect set consists of three unrelated kinds of aspects: 1) ‘sound quality’,
1063 ‘quality’ and ‘bluetooth earpiece’; 2) ‘usb cords’ and ‘usb plug’; 3) ‘shell case’, ‘grommets’, ‘impact
1064 protection’ and ‘grab’. In this case, No-UserAtt would assign same weights to aspect ‘bluetooth
1065 earpiece’ and ‘shell case’ when purchasing a bluetooth earpiece. And A_Static would assign same
1066 weights to aspect ‘sound quality’ no matter what kinds of products the user is purchasing. By
1067 identifying different aspects’ different roles when purchasing different products, AARM achieved
1068 better performance.

1069 Next we present how the aspect-level attention module finds the meaningful interactions (i.e.,
1070 interactions between the shared aspects, synonymous aspects and similar aspects) from all the aspect
1071 interactions between a user and a product. In Figure 4, we show the aspect-level attention values of
1072 the interactions between aforementioned user ‘A1P9UMP1XSE6MI’ and product ‘B002VPE1NO’. In
1073 the heat map, the columns refer to the product’s aspects while the rows refer to the user’s aspects.
1074 The color of each grid cell represents the attention value assigned to the corresponding interaction.
1075 The darker of the color in a grid cell, the higher of the attention value.

1076 First, we can see that interactions between the shared aspects like ‘grommets’, ‘impact protection’
1077 and ‘shell case’ are captured and assigned higher attention values. Second, the interactions between
1078

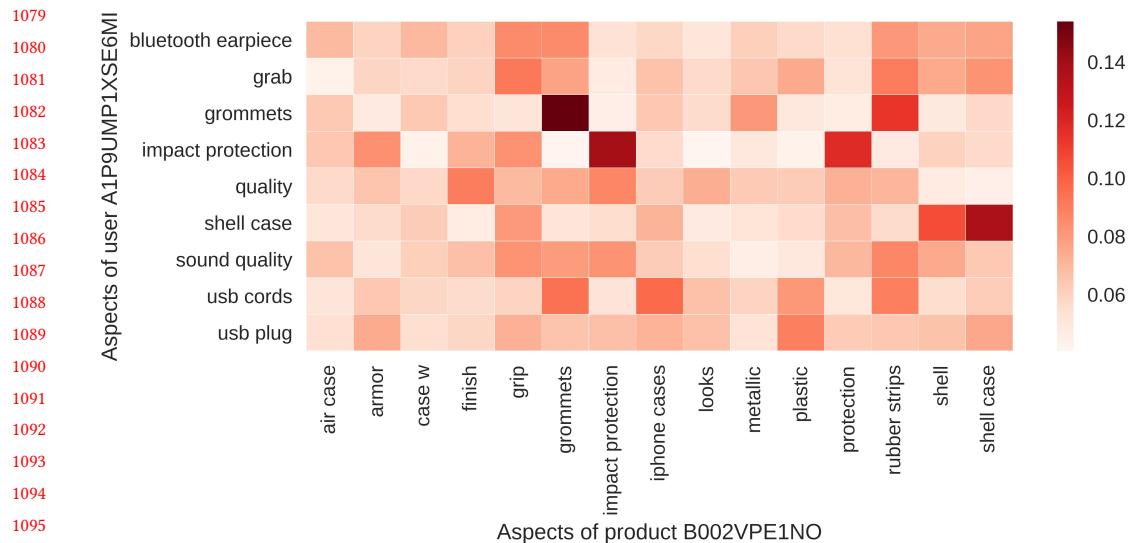


Fig. 4. Heat map of aspect-level attention. The columns refer to aspects of the product while the rows refer to aspects of the user. Darker color in the grid cell means that higher attention value is assigned to the interaction between corresponding aspects by the aspect-level attention module.

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 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 paper, we presented an the Attentive Aspect-based Recommendation Model (AARM) to incorporate 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 satisfactory 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 towards 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 multi-modal (textual reviews, product images and numerical ratings) methods JRL and eJRL, AARM can still achieves better results in all datasets. To demonstrates the effectiveness

1128 of each component in AARM, a lot of quantitative experiments and qualitative case studies are
 1129 conducted.

1130 In the future, we would like to extend our work in the following two ways: (1) Extracting aspects
 1131 with neural network and combining it with AARM. In particular, we would like to jointly train
 1132 the aspect extraction module and the recommendation module in an end-to-end style. Ideally,
 1133 the end-to-end training could reduce noisy aspects and mine more domain-specific aspects. (2)
 1134 Integrating aspect-level sentiment information in AARM. Aspect-level sentiment information is
 1135 useful to identify user's likes and dislikes about product features. But existing methods usually
 1136 use external tools for aspect-level sentiment analysis, which relies on the accuracy of these tools
 1137 and is usually not able to deal with new reviews. We will study how to extract these sentiment
 1138 information and integrate them into AARM with end-to-end learning.

1139 REFERENCES

- 1141 [1] Konstantin Bauman, Bing Liu, and Alexander Tuzhilin. 2017. Aspect based recommendations: Recommending items
 1142 with the most valuable aspects based on user reviews. In *Proceedings of the 23rd ACM SIGKDD International Conference
 1143 on Knowledge Discovery and Data Mining*. ACM, 717–725.
- 1144 [2] Da Cao, Xiangnan He, Lianhai Miao, Yahui An, Chao Yang, and Richang Hong. 2018. Attentive group recommendation.
 1145 In *The 41st International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, New York,
 NY, USA, 645–654.
- 1146 [3] Da Cao, Xiangnan He, Liqiang Nie, Xiaochi Wei, Xia Hu, Shunxiang Wu, and Tat-Seng Chua. 2017. Cross-platform
 1147 app recommendation by jointly modeling ratings and texts. *ACM Transactions on Information Systems* 35, 4, Article 37
 (2017), 37:1–37:27 pages.
- 1148 [4] Rose Catherine and William Cohen. 2017. Transnets: Learning to transform for recommendation. In *Proceedings of the
 1149 Eleventh ACM Conference on Recommender Systems*. ACM, 288–296.
- 1150 [5] Jingyuan Chen, Hanwang Zhang, Xiangnan He, Liqiang Nie, Wei Liu, and Tat-Seng Chua. 2017. Attentive collaborative
 1151 filtering: Multimedia recommendation with item-and component-level attention. In *Proceedings of the 40th International
 1152 ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 335–344.
- 1153 [6] Li Chen, Guanliang Chen, and Feng Wang. 2015. Recommender systems based on user reviews: the state of the art.
 1154 *User Modeling and User-Adapted Interaction* 25, 2 (2015), 99–154.
- 1155 [7] Xu Chen, Zheng Qin, Yongfeng Zhang, and Tao Xu. 2016. Learning to rank features for recommendation over multiple
 1156 categories. In *Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information
 1157 Retrieval*. ACM, 305–314.
- 1158 [8] Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg
 1159 Corrado, Wei Chai, Mustafa Ispir, Rohan Anil, Zakaria Haque, Lichan Hong, Vihan Jain, Xiaobing Liu, and Hemal
 Shah. 2016. Wide & deep learning for recommender systems. In *Proceedings of the 1st Workshop on Deep Learning for
 1160 Recommender Systems*. ACM, 7–10.
- 1161 [9] Zhiyong Cheng, Ying Ding, Xiangnan He, Lei Zhu, Xuemeng Song, and Mohan Kankanhalli. 2018. A^3 NCF: An adaptive
 1162 aspect attention model for rating prediction. In *Proceedings of the Twenty-Seventh International Joint Conference on
 1163 Artificial Intelligence*. International Joint Conferences on Artificial Intelligence Organization, 3748–3754.
- 1164 [10] 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.
- 1165 [11] Paolo Cremonesi, Yehuda Koren, and Roberto Turrin. 2010. Performance of recommender algorithms on top-n
 1166 recommendation tasks. In *Proceedings of the fourth ACM conference on Recommender systems*. ACM, 39–46.
- 1167 [12] Ruihai Dong and Barry Smyth. 2017. User-based opinion-based recommendation. In *Proceedings of the 26th International
 1168 Joint Conference on Artificial Intelligence*. AAAI Press, 4821–4825.
- 1169 [13] Gayatree Ganu, Yogesh Kakodkar, and Amélie Marian. 2013. Improving the quality of predictions using textual
 1170 information in online user reviews. *Information Systems* 38, 1 (2013), 1–15.
- 1171 [14] Yoav Goldberg. 2017. Neural network methods for natural language processing. *Synthesis Lectures on Human Language
 Technologies* 10, 1 (2017), 1–309.
- 1172 [15] 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.
- 1173 [16] Ruining He and Julian McAuley. 2016. Ups and downs: Modeling the visual evolution of fashion trends with one-class
 1174 collaborative filtering. In *proceedings of the 25th international conference on world wide web*. International World Wide
 1175 Web Conferences Steering Committee, 507–517.

1176

- 1177 [17] Xiangnan He, Tao Chen, Min-Yen Kan, and Xiao Chen. 2015. TriRank: Review-aware explainable recommendation by
1178 modeling aspects. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*.
1179 ACM, 1661–1670.
- 1180 [18] 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.
- 1181 [19] Xiangnan He, Zhenkui He, Jingkuan Song, Zhenguang Liu, Yu-Gang Jiang, and Tat-Seng Chua. 2018. NAIS: Neural
1182 attentive item similarity model for recommendation. *IEEE Transactions on Knowledge and Data Engineering* (2018).
- 1183 [20] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural collaborative filtering.
1184 In *Proceedings of the 26th International Conference on World Wide Web*. International World Wide Web Conferences
Steering Committee, 173–182.
- 1185 [21] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*
1186 (2014).
- 1187 [22] Y. Koren, R. Bell, and C. Volinsky. 2009. Matrix factorization techniques for recommender systems. *Computer* 42, 8
1188 (2009), 30–37.
- 1189 [23] Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *International Conference
on Machine Learning*, 1188–1196.
- 1190 [24] Julian McAuley and Jure Leskovec. 2013. Hidden factors and hidden topics: understanding rating dimensions with
1191 review text. In *Proceedings of the 7th ACM conference on Recommender systems*. ACM, 165–172.
- 1192 [25] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector
1193 space. *arXiv preprint arXiv:1301.3781* (2013).
- 1194 [26] 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.
- 1195 [27] Steffen Rendle. 2010. Factorization machines. In *Proceedings of the 2010 IEEE International Conference on Data Mining*.
1196 IEEE Computer Society, 995–1000.
- 1197 [28] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: bayesian personalized
1198 ranking from implicit feedback. In *Proceedings of the twenty-fifth conference on uncertainty in artificial intelligence*.
AUAI Press, 452–461.
- 1199 [29] Gerard Salton, Anita Wong, and Chung-Shu Yang. 1975. A vector space model for automatic indexing. *Commun. ACM*
1200 18, 11 (1975), 613–620.
- 1201 [30] Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout:
1202 a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research* 15, 1 (2014),
1203 1929–1958.
- 1204 [31] 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.
- 1205 [32] Yunzhi Tan, Min Zhang, Yiqun Liu, and Shaoping Ma. 2016. Rating-boosted latent topics: Understanding users and
1206 items with ratings and reviews. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*.
AAAI Press, 2640–2646.
- 1207 [33] Shuaiqiang Wang, Shanshan Huang, Tie-Yan Liu, Jun Ma, Zhumin Chen, and Jari Veijalainen. 2016. Ranking-oriented
1208 collaborative filtering: A listwise approach. *ACM Transactions on Information Systems* 35, 2 (2016), 10:1–10:28.
- 1209 [34] Jun Xiao, Hao Ye, Xiangnan He, Hanwang Zhang, Fei Wu, and Tat-Seng Chua. 2017. Attentional factorization
1210 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.
- 1211 [35] Yingqing Xu, Wai Lam, and Tianyi Lin. 2014. Collaborative filtering incorporating review text and co-clusters of
1212 hidden user communities and item groups. In *Proceedings of the 23rd ACM International Conference on Conference on
Information and Knowledge Management*. ACM, 251–260.
- 1213 [36] Longqi Yang, Cheng-Kang Hsieh, Hongjian Yang, John P. Pollak, Nicola Dell, Serge Belongie, Curtis Cole, and Deborah
1214 Estrin. 2017. Yum-Me: A personalized nutrient-based meal recommender system. *ACM Transactions on Information
Systems* 36, 1 (2017), 7:1–7:31.
- 1215 [37] Yongfeng Zhang, Qingyao Ai, Xu Chen, and W Bruce Croft. 2017. Joint representation learning for top-n recom-
1216 mendation with heterogeneous information sources. In *Proceedings of the 2017 ACM on Conference on Information and
Knowledge Management*. ACM, 1449–1458.
- 1217 [38] Yongfeng Zhang, Guokun Lai, Min Zhang, Yi Zhang, Yiqun Liu, and Shaoping Ma. 2014. Explicit factor models for
1218 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.
- 1219 [39] Yongfeng Zhang, Haochen Zhang, Min Zhang, Yiqun Liu, and Shaoping Ma. 2014. Do users rate or review?: Boost
1220 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.
- 1221
- 1222
- 1223
- 1224
- 1225

- 1226 [40] Lei Zheng, Vahid Noroozi, and Philip S. Yu. 2017. Joint deep modeling of users and items using reviews for rec-
1227 ommendation. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*. ACM,
1228 425–434.

1229

1230

1231

1232

1233

1234

1235

1236

1237

1238

1239

1240

1241

1242

1243

1244

1245

1246

1247

1248

1249

1250

1251

1252

1253

1254

1255

1256

1257

1258

1259

1260

1261

1262

1263

1264

1265

1266

1267

1268

1269

1270

1271

1272

1273

1274