Multi-criteria and Review-based Overall Rating Prediction

Edgar Ceh–Varela[0000−0001−6277−2741], Huiping Cao[0000−0002−1350−1846], and Tuan Le[0000−0003−0667−9184]

Department of Computer Science
New Mexico State University
eceh@nmsu.edu, hcao@cs.nmsu.edu, tuanle@nmsu.edu

Abstract. An overall rating cannot reveal the details of user's preferences toward each feature of a product. One widespread practice of e-commerce websites is to provide ratings on predefined aspects of the product and user-generated reviews. Most recent multi-criteria works employ aspect preferences of users or user reviews to understand the opinions and behavior of users. However, these works fail to learn how users correlate these information sources when users express their opinion about an item. In this work, we present **M**ulti-task & **M**ulti-**C**riteria **R**eview-based **R**ating (**MMCRR**), a framework to predict the overall ratings of items by learning how users represent their preferences when using multi-criteria ratings and text reviews. We conduct extensive experiments with three real-life datasets and six baseline models. The results show that *MMCRR* can reduce prediction errors while learning features better from the data.

Keywords: multi-criteria · multi-task · rating prediction.

1 Introduction

Multi-criteria recommender systems (*MCRS*) have been developed to increase the recommender systems (*RS*) performance. Most recent multi-criteria works employ user prefernces [14,20] or reviews [11,2] to understand users'opinions and analyze their behaviors. However, these works fail to consider the analytical tasks users need to follow to express their opinion about an item in different forms and how these tasks are related (i.e., summarizing the textual reviews into criteria, and finally into an overall rating). This process is challenging, as users have different scales for their ratings and for the intensity of some of the opinion words used in their reviews [18]. For example, a user could write the sentence "This is a good hotel" and rate the hotel with five stars, while another user for the same sentence could use four stars.

Considering that these forms for expressing user preferences are related, this study proposes a new **M**ulti-task & **M**ulti-**C**riteria **R**eview-based **R**ating (**MMCRR**) framework to predict the overall rating that a user would give to a new item. *MMCRR* uses a multi-task learning (MTL) approach to learn how

users represent their preferences when using multi-criteria ratings and text reviews and the relationship between these two. Furthermore, it can also predict each criterion rating because of the multi-task process, which can be presented to explain why a given overall rating.

This paper's contributions are as follows: (*i*) we present a model that takes advantage of MTL to use multi-criteria ratings and text reviews to learn the relationship between these types of evaluations. (*ii*) our approach can predict the overall and criterion ratings that a user would give to an item. Moreover, the model does not need the item's multi-criteria or individual review during the prediction phase. (*iii*) we evaluate the model's performance on three real-life datasets against baseline models from the most utilized types of *MCRS*.

The remainder of this paper is organized as follows: Section 2 presents the related literature. Section 3 details the proposed methodology. Section 4 outlines the experimental settings used to test our proposed model. The results showing the validity of our proposed method and its effectiveness are in Section 5. Finally, our conclusions are in Section 6.

2 Related work

2.1 *MCRS* with user preferences

Models in this category try to learn the relationships between the criteria ratings and the overall rating. User preferences are known directly from the user's explicit ratings on the items' features. Different works have extended existing single-criteria Collaborative Filtering (CF) techniques to work with multi-criteria ratings [1]. A method to learn and rank user preferences over each criterion and find each item's dominant criterion is presented in [14]. These ranks are used to predict the overall rating using CF techniques. In [10], multi-criteria ratings are decomposed into k separate single rating problems using matrix factorization techniques (e.g., SVD). An aggregation function is then extracted using different machine learning approaches. These works only exploit the user's preference from multi-criteria ratings, which can fail short on representing the user's whole opinion towards an item.

2.2 Multi-criteria Review-based *RS*

Works from this category consider an implicit relationship between the user's overall rating and her expressed comment. The most widely used method is the extraction of aspects and their polarity scores (i.e., positive or negative sentiments) [11,12]. Aspects are extracted using different Natural Language Processing (NLP) techniques. Similarly, other works [17,2] generate latent feature representations from reviews. These representations are then used as an input for the overall rating prediction. These works only analyze user reviews without considering the explicit multi-criteria ratings.

Moreover, some *MCRS* works [7,5,9] jointly use reviews and ratings for the target item as additional input for the prediction. However, in real-life recommendations, we do not know at recommendation time what the user would write or how the user would rate the criteria. Our proposed solution does not have this problem as it uses historical reviews, and criteria ratings only for training.

2.3 Multi-Task Learning (MTL)

In many situations, we may have a task composed of related sub-tasks, and where each of them shares some features. MTL [3] allows solving all these sub-tasks, and consequently, the overall task, in an end-to-end learning model. In MTL, each task provides regularization to the other tasks, which is especially useful in scenarios where part of the data is not available at test time. Recently, different works [2,19] have started to use MTL in recommendation problems.

In this work, we exploit the multi-task learning paradigm. We consider two related tasks: (*i*) an associated task, used during training, which predicts the overall rating given the multi-criteria ratings, and (*ii*) a main task, composed of two subtasks, to predict the multi-criteria and overall ratings based on historical user and item reviews.

3 Proposed approach

3.1 Problem definitions

Let $U = \{u_1, \cdots, u_M\}$ and $I = \{i_1, \cdots, i_N\}$ be the set of users and set of items that users have rated respectively. We denote H_{UI} the set of reviews written by U about I.

Definition 1 (Reviews document). *A reviews document $H_u \subset H_{UI}$ is the set of reviews written by a user $u \in U$. Similarly, $H_i \subset H_{UI}$ is the set of reviews about an item $i \in I$.*

Each item $i \in I$ is described by a set of aspects (i.e., criteria) $K = \{k_0, k_1, \cdots, k_K\}$. A user $u \in U$ can rate a criterion k with a rating $r_{u,i,k}$, a non-negative real number. We consider the overall rating as one particular aspect, denoted as $r_{u,i,0}$.

Definition 2 (Multi-criteria ratings). *Is the set of ratings following the function $R : U \times I \rightarrow R_0 \times R_1 \times \cdots \times R_k$.*

Where R_0 represents the overall ratings, and R_1 to R_k is the rating values for each criterion. For the rest of the paper, when we mention the multi-criteria ratings, we will not consider R_0.

Definition 3 (Problem statement). *Having the set of reviews H_{UI}, the set of users U, the set of items I, and the set of multi-criteria ratings from R, the **problem** is to find a function $r_{u,i,0} = f(U, I)$, where $r_{u,i,0}$ is the overall rating that a user $u \in U$ would give to an item $i \in I$ which he/she has not interacted with before.*

3.2 MMCRR model

Figure 1 shows the general architecture of the *MMCRR* model. It consists of two parts, one for each task of our MTL approach. During training, *MMCRR* uses the multi-criteria representation from the associated task to adjust the feature representation and to improve the accuracy in the main task. Moreover, user and item embeddings are shared between the two tasks allowing them to create better representations of users and items. For clarity, we use bold lowercase for vectors and bold uppercase for matrices.

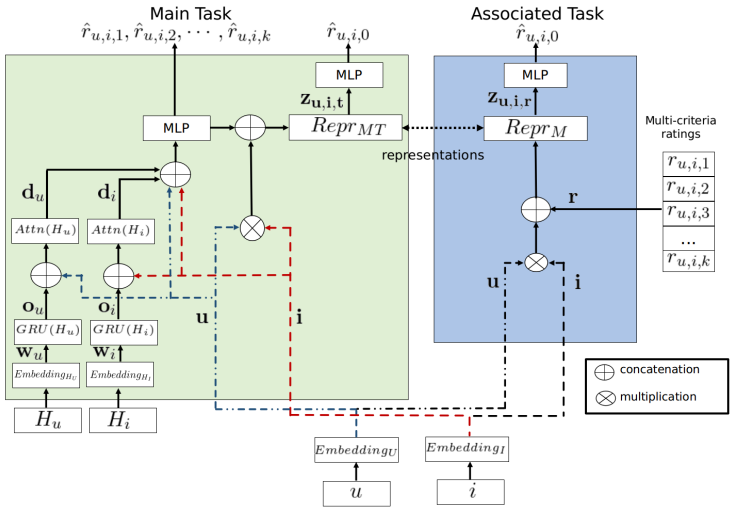

Fig. 1: MMCRR model components

3.3 Associated task

This task is only used during the model training, given that the criteria ratings are not available for the final overall rating predictions. It aims to learn a representation of the user's preferences for each item's criterion. Figure 1 shows in blue the architecture of this task. The inputs for this task are the user u and item i IDs, and the ratings given by u to each of i's criterion. Each user u and item i is represented with an embedding vector. These vectors represent the intrinsic properties of u and i learned from the data. We use two embedding layers $Embedding_U \in R^{U \times D_U}$ and $Embedding_I \in R^{I \times D_I}$, where D_U and D_I are the vectors' dimensionalities. Then, $\mathbf{u} \in R^{D_U}$ and $\mathbf{i} \in R^{D_I}$ are the embedding vectors for each user and item, respectively.

Similarly, let $r_{u,i,k}$ be the rating of user u to item i in criterion k. We represent the criteria ratings given by a user u to an item i as $\mathbf{r}$. The interaction of

embedding vectors $\mathbf{u}$ and $\mathbf{i}$ is obtained using an element-wise product, which has been demonstrated to be highly effective [6]. Finally, this interaction and $\mathbf{r}$, are passed to a neural network ($Repr_M$) to obtain the features representation as

$$\mathbf{z_{u,i,r}} = \sigma(\mathbf{W_r}\,[\mathbf{u} \times \mathbf{i}, \mathbf{r}] + \mathbf{b_r}) \tag{1}$$

where $\mathbf{W_r}$ and $\mathbf{b_r}$ denote the weight matrix and bias vector, respectively. σ is the activation function and $\mathbf{z_{u,i,r}}$ is the features representation. The representation is used to predict the overall rating as:

$$\hat{r}_{u,i,0} = \mathbf{w_r}^{\top}(\sigma(\mathbf{W_o z_{u,i,r}} + \mathbf{b_o})) \tag{2}$$

where $\mathbf{w_r}$ denotes the weights of the prediction layer, $\mathbf{W_o}$ and $\mathbf{b_o}$ are the parameters, σ is the activation function, and $\hat{r}_{u,i,0}$ is the overall rating prediction for a user u and item i.

3.4 Main task

This task aims to predict the overall rating and, in an auxiliary way, the multi-criteria ratings that a user u would give to an item i. This task consists of several steps: (*i*) word sequence encoding, (*ii*) word-level attention, (*iii*) multi-criteria rating prediction, and (*iv*) overall rating prediction. Figure 1 presents in green the architecture of this task. We detail this task in the following sections.

Word sequence encoding. We rely on the GRU [4] gating mechanism. Assume that a document (i.e., H_u or H_i) contains T words. The words in this document are represented as w_t with $t \in [1, T]$. The main task transforms the raw document into a vector representation, on which we build a multi-criteria rating predictor. Without loss of generality, we present how we build the document level vector for H_u, the steps for H_i are similar. Each word is embedded as a vector $\mathbf{w} \in R^{W_U}$, using an embedding layer $Embedding_{H_U} \in R^{W_U \times D_W}$, where W_U is the size of the vocabulary for the users' documents, and D_W is the dimensionality of the embedding vector. The sequence of words embeddings $\mathbf{w}_t$ for each document is the GRU input to get the contextual information of each word. This step is represented as $\mathbf{o_u} = GRU\,(\mathbf{w}_t)\,, t \in [1, T]$, where $\mathbf{o_u}$ contains the output features h_t for each t.

Word-level attention. The same words may have a different intention for different users. Hence, we introduce an attention mechanism to extract those words that are important to the document's meaning, considering the user who wrote it. Then, we aggregate those informative words to form a document vector. Concretely,

$$e_t = \tanh\left(\mathbf{W}_w[\mathbf{o_u}, \mathbf{u}] + \mathbf{b}_w\right) \tag{3} \qquad \alpha_t = \frac{\exp\left(e_t\right)}{\sum_t \exp\left(e_t\right)} \tag{4} \qquad \mathbf{d_u} = \sum_t \alpha_t \mathbf{o_u} \tag{5}$$

where e_t is a hidden representation of $\mathbf{o_u}$ and $\mathbf{u}$, α_t is the normalized importance weight. Finally, we compute a user document vector $\mathbf{d_u}$ as a weighted sum of the weights α_t and $\mathbf{o_u}$.

Multi-criteria rating prediction. After obtaining the document representations $\mathbf{d_u}$ and $\mathbf{d_i}$ ($\mathbf{d_i}$ is the item's document vector), we proceed to predict each criterion's rating. The embedding vectors $\mathbf{u}$ and $\mathbf{i}$, along with $\mathbf{d_u}$ and $\mathbf{d_i}$, are passed to a two-layer neural network to obtain the features representation.

$$\mathbf{z_{u,i,k}} = \sigma(\mathbf{W_{k1}}(\sigma(\mathbf{W_{k0}}\,[\mathbf{u}, \mathbf{i}, \mathbf{d_u}, \mathbf{d_i}] + \mathbf{b_{k0}})) + \mathbf{b_{k1}}) \tag{6}$$

where $\mathbf{W_{k0}}$, $\mathbf{W_{k1}}$, $\mathbf{b_{k0}}$, and $\mathbf{b_{k1}}$ denote the weight matrices and bias vectors, respectively. σ is the activation function, and $\mathbf{z_{u,i,k}}$ is the feature representation. This representation is used to predict a vector with the multi-criteria ratings as follows:

$$\hat{\mathbf{r}}_{u,i,k} = \mathbf{w_k}^\top \mathbf{z_{u,i,k}} \tag{7}$$

where $\mathbf{w_k}$ denotes the weights of the prediction layer, and $\hat{\mathbf{r}}_{u,i,k}$ is a vector with the multi-criteria rating predictions for a user u and an item i.

Overall rating prediction. To predict the overall rating, the interaction of $\mathbf{u}$ and $\mathbf{i}$, along with $\hat{\mathbf{r}}_{u,i,k}$, are passed to a simple neural network to get the features representation ($Repr_{MT}$).

$$\mathbf{z_{u,i,t}} = \sigma(\mathbf{W_t}\,[\mathbf{u} \times \mathbf{i}, \hat{\mathbf{r}}_{u,i,k}] + \mathbf{b_t}) \tag{8}$$

where $\mathbf{W_t}$ and $\mathbf{b_t}$ denote the weight matrix and bias vector, respectively. σ is the activation function, and $\mathbf{z_{u,i,t}}$ is the features representation. Then, we get the overall rating as:

$$\hat{r}_{u,i,0} = \mathbf{w_t}^\top(\sigma(\mathbf{W_t z_{u,i,t}} + \mathbf{b_t})) \tag{9}$$

where $\mathbf{w_t}$ denotes the weights of the prediction layer, $\mathbf{W_t}$ and $\mathbf{b_t}$ are the parameters, σ is the activation function, and $\hat{r}_{u,i,0}$ is the overall rating prediction for a user u and item i.

3.5 Model optimization

First, for the associated task, given the ground-truth overall rating of user u on item i, and the predicted overall rating calculated using Equation 2, the loss of this rating prediction is defined as follows:

$$Loss_{AT} = (r_{u,i,0} - \hat{r}_{u,i,0})^2 \tag{10}$$

Second, the main task performs two additional tasks: predicting the aspect ratings and predicting the overall rating. First, given the ground-truth aspect ratings of user u on item i and the aspect rating predictions using Equation 7, the loss of rating predictions is defined as:

$$Loss_A = \sum_{j=1}^{k} (r_{u,i,j} - \hat{r}_{u,i,j})^2 \tag{11}$$

Then, for the ground-truth overall rating and the predicted rating from Equation 9, we have:

$$Loss_{MT} = (r_{u,i,0} - \hat{r}_{u,i,0})^2 \tag{12}$$

The associated and main tasks are closely related; therefore, their representations need to be close to each other. Hence, we add a representation loss as:

$$Loss_R = \|\mathbf{z_{u,i,r}} - \mathbf{z_{u,i,t}}\|_2 \tag{13}$$

Finally, to optimize all model parameters, we try to minimize the following loss function:

$$Loss = \frac{1}{|N|} \sum_{r_{ui} \in N} (\lambda(Loss_{MT} + Loss_A + Loss_R) + (1 - \lambda)Loss_{AT}) + \alpha_\theta |\theta|_F^2 \tag{14}$$

where N is the training set and λ determines the relative importance of the tasks. This optimization function considers $Loss_R$ when both tasks are trained (i.e., $0 < \lambda < 1$). We use the Frobenius norm regularization term $|\theta|_F^2 = \sum_i \theta_i^2$, where θ stands for the parameters to optimize, and α_θ is the penalty term.

4 Experimental settings

4.1 Datasets

We use three real-life datasets to test our model. These datasets have rating values and written reviews for the items. The first two correspond to non-alcoholic (**NALC**) and regular beer (**BEER**) reviews collected from the *BeerAdvocate* website by [11]. They have ratings on four beer aspects (i.e., feel, look, smell, and taste) and an overall rating. The third dataset has hotel reviews from TripAdvisor (**TRIP**) collected by [16]. This dataset includes ratings on seven aspects in each review (i.e., value, room, location, cleanliness, check-in/front desk, service, business service), and an overall rating. For the three datasets, the rating range is from 1 to 5 stars. Table 1 shows the description of these datasets. We remove

Table 1: Dataset descriptions

	NALC	BEER	TRIP
Instances	1,201	1,585,887	67,155
Users	582	33,372	60,107
Items	162	66,051	1,850
Avg. words per review	110.24	123.84	177.70
Aspects	4	4	7
Avg. overall rating	2.58	3.82	4.07
std. overall rating	1.01	0.72	1.16
Aut. Read. Index	8.06	9.09	11.73
Sparsity	~ 0.91	~ 0.99	~ 0.97

instances considered outliers. For example, an instance having all its aspects with ratings of 5 stars, but an overall rating of 1. Further, we remove users with

less than five reviews. We transform each review to lowercase and remove all numbers and special characters. For BEER and TRIP, we randomly get a sample of 50,000 instances. We randomly split each dataset into 80% for training, 10% for evaluation, and 10% for testing. For each user, we concatenate the text from her reviews. We do the same for the reviews about each item. We specify a maximum text size of 10,000 words. We use "<UNK>" to replace those words with a frequency below 10.

4.2 Baselines

We choose baselines that only use user and item IDs as input for the prediction phase, given that additional input information (i.e., multi-criteria ratings and user's review) is not present in such a phase. We compare our proposed model with the following baselines:

1. Multi-criteria rating-based models:

MultiDim [1]: This method extends the standard CF approach. It calculates similarity using multi-dimensional distance metrics to reflect multi-criteria information.

PrefLearn [14]: This algorithm learns and ranks the user's preferences over different criteria. Similarly, it finds and ranks the dominant criterion of each item. Then, it uses these rankings with modified versions of *UBCF* and *IBCF* algorithms. The results of these algorithms are unified to obtain the final overall rating.

AggFunc [10]: This model assumes that the overall rating and the multi-criteria ratings have a relationship. It decomposes the multi-criteria rating space into k single-rating recommendation problems. It uses *SVD* to predict each criterion's missing ratings and a two-layer Neural Network to learn an aggregation function for predicting the overall rating based on the known multi-criteria ratings.

2. Multi-criteria review-based models:

AspectBased [12]: This model extracts relevant aspects and sentiment scores from the user's reviews. The sentiment scores associated with each extracted aspect are used as ratings. Then it uses a *Multi-criteria User-to-User* algorithm to predict the overall rating.

GRURec: We implemented this baseline based on [2,8]; this model uses two bidirectional GRU RNN to get latent preference factors for H_u and H_i. Both vectors are then concatenated with the user and item embeddings. The resulting vector is then used as input for an MLP neural network to predict the overall rating.

3. Multi-criteria rating- & review-based models:

AggFunc+GRURec: We use an ensemble method to combine Multi-criteria rating-based models and Multi-criteria review-based models. This model calculates the overall rating using the average of the ratings predicted by **AggFunc** and **GRURec**.

4.3 Hyperparameters

We use Pytorch on a Tesla P100 GPU with 16 GB of RAM. We use a hidden layer size of 128, an embedding dimension of 256 for users and items, a learning rate of 0.001 with a learning scheduler. The word embedding dimensions are set to 100 and initialize with pre-trained GloVe embeddings. The activation function is LeakyReLU. The model is trained with a batch size of 256 for NALC and 128 for BEER and TRIP. We use Adam as the optimizer. On the embedding layers, we use a dropout percentage of 0.2. For the loss, we initially set $\lambda = 0.5$, for the regularization, α_θ is set to 0.01, and for epochs, we use an early stopping strategy based on the validation loss.

5 Experimental Results

5.1 Model performance

We use the standard metric, *Root Mean Square Error (RMSE)*. A smaller RMSE value indicates better performance. Table 2 shows the average performance of the six baselines and *MMCRR* after running the tests three times. From these results, we observe the following:

(1) For methods using criteria ratings, the *AggFunc* method has the best results for all datasets, indicating that using the preferences for each criterion is useful. (2) For methods using reviews, the *AspectBased* method results show that using only the sentiment of latent aspects extracted from the reviews as the multi-criteria ratings do not provide good results. These results happen because different user's words can have different sentiment strengths, which implies a different rating scale [15]. The results of *GRURec* are better, showing that it can find more useful representations of users and items based on the reviews. (3) Combining models for criteria ratings and reviews improves those models' results just using text.

We can see that *MMCRR* outperforms all baselines, which indicates that it can learn the analytical tasks that a user makes when expressing preferences about an item.

5.2 Effect of parameter λ

Figure 2 shows the results of varying the parameter λ from Equation 14. When λ increases, the contribution from the associated task decreases. For the NALC and TRIP datasets, RMSE error is lower around the middle, meaning that there is a contribution from both tasks. RMSE starts with high values in the BEER dataset, and as λ increases, these values decrease and stabilize. The results obtained by these metrics are coherent with our definition of *MMCRR*.

Method	NALC	BEER	TRIP
MultiDim	1.0251	0.6059	1.171
PrefLearn	0.932	0.6045	1.1235
AggFunc	0.9314	0.5939	1.073
AspectBased	1.2958	0.8856	1.4238
GRURec	0.9726	0.6473	1.164
AggFunc+GRURec	0.9349	0.6974	1.081
MMCRR	**0.873**	**0.579**	**0.9751**

Table 2: Performance results for all datasets ($\lambda = 0.5$)

Method	NALC	BEER	TRIP
MMCRR_NA	1.0346	0.6374	1.2314
MMCRR_NR	0.9003	**0.5778**	0.9881
MMCRR_NRA	1.0130	0.6398	1.2869
MMCRR	**0.873**	0.579	**0.9751**

Table 3: Comparing different losses

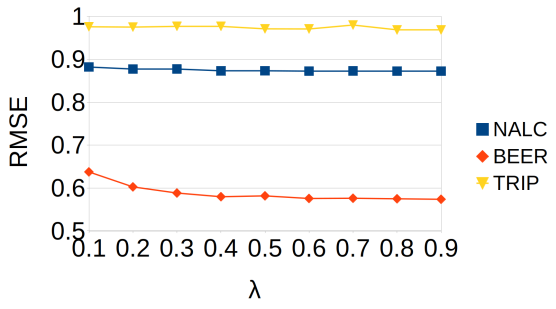

Fig. 2: MMCRR varying λ for each dataset (considering $Loss_R$)

5.3 Effect of loss functions

We create three different variations of Equation 14. First, *MMCRR_NA*, which does not consider the loss from Equation 11. Second, *MMCRR_NR*, which does not consider the loss from Equation 13. Finally, *MMCRR_NRA* does not consider both losses. Table 3 shows the results. *MMCRR_NRA* has higher error values than *MMCRR*, showing that these losses allow the model to minimize the prediction errors and fit the data better. Only for BEER, *MMCRR_NR* has a lower value; we attribute it to the dataset sparsity (i.e., $\sim 99\%$).

5.4 Performance of the Main task

We test the performance of using only the *Main* task (Section 3.4). Table 4 shows the results. The λ values are the ones where we get better results for *MMCRR*. We can see that *MMCRR* has better performance overall, as it uses the *Associated* task to learn the multi-criteria preference representations for the users. These representations, along with the shared user and item representations, help the *Main* task to improve its learning.

Method	NALC $\lambda = 0.4$	BEER $\lambda = 0.9$	TRIP $\lambda = 0.5$
MMCRR_Main	0.8829	0.5717	0.9907
MMCRR	**0.8731**	**0.5702**	**0.977**

Table 4: Comparing performance of the *Main* task

Method	NALC	BEER	TRIP
AggFunc	0.7950	0.5382	**1.0015**
MMCRR	**0.7377**	**0.4027**	1.4372

Table 5: RMSE median for the aspect predictions

5.5 Metrics for aspect ratings

Although not the primary goal of *MMCRR*, we analyze how it predicts the aspect ratings. Only the *AggFunc* baseline predicts individual aspect ratings. Table 5 shows the RMSE medians for all item aspects. Only for TRIP does not show a better performance. Recall that these ratings are predicted based on item reviews and the user and item representations. Therefore, given that the TRIP dataset has seven aspects to predict and it has more complex text (i.e., Automatic Reading Index [13] is higher, see Table 1) than the other datasets, the model's job is more challenging.

6 Conclusions

We proposed *MMCRR*, a framework leveraging multi-task learning to address the problem of predicting at the same time the ratings for each item's criterion and its overall rating. We use three real-life datasets with six baseline models. Our results show that *MMCRR* makes predictions with a lower prediction error than these baselines. Moreover, our model can learn the users' rating profiles and predict ratings considering how users write their reviews. Similarly, we show how our approach also reduces the prediction error for the criteria ratings.

Acknowledgements. This work has been supported by the National Council of Science and Technology of Mexico (CONACYT) #602434/440684, and National Science Foundation of USA (NSF) #1633330, #1757207, and #1914635.

References

1. Adomavicius, G., Kwon, Y.: New recommendation techniques for multicriteria rating systems. IEEE Intelligent Systems **22**(3), 48–55 (2007)
2. Bansal, T., Belanger, D., McCallum, A.: Ask the gru: Multi-task learning for deep text recommendations. In: Proceedings of the 10th ACM Conference on Recommender Systems. pp. 107–114 (2016)
3. Caruana, R.: Multitask learning. Machine learning **28**(1), 41–75 (1997)
4. Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., Bengio, Y.: Learning phrase representations using rnn encoder-decoder for statistical machine translation. arXiv preprint arXiv:1406.1078 (2014)

5. Ding, Y., Li, S., Yu, W., Wang, J., Liu, M.: A unified neural model for review-based rating prediction by leveraging multi-criteria ratings and review text. Cluster Computing **22**(4), 9177–9185 (2019)
6. He, X., Chua, T.S.: Neural factorization machines for sparse predictive analytics. In: Proceedings of the 40th International ACM SIGIR conference on Research and Development in Information Retrieval. pp. 355–364 (2017)
7. Jin, Z., Li, Q., Zeng, D.D., Zhan, Y., Liu, R., Wang, L., Ma, H.: Jointly modeling review content and aspect ratings for review rating prediction. In: Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval. pp. 893–896 (2016)
8. Li, P., Tuzhilin, A.: Latent multi-criteria ratings for recommendations. In: Proceedings of the 13th ACM Conference on Recommender Systems. pp. 428–431 (2019)
9. Li, P., Tuzhilin, A.: Learning latent multi-criteria ratings from user reviews for recommendations. IEEE Transactions on Knowledge and Data Engineering (2020)
10. Majumder, G.S., Dwivedi, P., Kant, V.: Matrix factorization and regression-based approach for multi-criteria recommender system. In: International Conference on Information and Communication Technology for Intelligent Systems. pp. 103–110. Springer (2017)
11. McAuley, J., Leskovec, J., Jurafsky, D.: Learning attitudes and attributes from multi-aspect reviews. In: 2012 IEEE 12th International Conference on Data Mining. pp. 1020–1025. IEEE (2012)
12. Musto, C., de Gemmis, M., Semeraro, G., Lops, P.: A multi-criteria recommender system exploiting aspect-based sentiment analysis of users' reviews. In: Proceedings of the eleventh ACM conference on recommender systems. pp. 321–325 (2017)
13. Senter, R., Smith, E.A.: Automated readability index. Tech. rep., CINCINNATI UNIV OH (1967)
14. Sreepada, R.S., Patra, B.K., Hernando, A.: Multi-criteria recommendations through preference learning. In: Proceedings of the Fourth ACM IKDD Conferences on Data Sciences. pp. 1–11 (2017)
15. Tang, D., Qin, B., Liu, T., Yang, Y.: User modeling with neural network for review rating prediction. In: Twenty-Fourth International Joint Conference on Artificial Intelligence (2015)
16. Wang, H., Lu, Y., Zhai, C.: Latent aspect rating analysis on review text data: a rating regression approach. In: Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining. pp. 783–792 (2010)
17. Wang, H., Lu, Y., Zhai, C.: Latent aspect rating analysis without aspect keyword supervision. In: Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining. pp. 618–626 (2011)
18. Wang, J., De Vries, A.P., Reinders, M.J.: Unifying user-based and item-based collaborative filtering approaches by similarity fusion. In: Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval. pp. 501–508 (2006)
19. Wang, N., Wang, H., Jia, Y., Yin, Y.: Explainable recommendation via multi-task learning in opinionated text data. In: The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval. pp. 165–174 (2018)
20. Zheng, Y.: Utility-based multi-criteria recommender systems. In: Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing. pp. 2529–2531 (2019)