Lightweight Unbiased Multi-teacher Ensemble for Review-based Recommendation

Guipeng Xv* †
School of Informatics, Xiamen University
Xiamen, China
xvguipeng.xgp@alibaba-inc.com

Xinyi Liu*
School of Informatics, Xiamen University
Xiamen, China
xinyiliu@stu.xmu.edu.cn

Chen Lin‡
School of Informatics, Xiamen University
Xiamen, China
chenlin@xmu.edu.cn

Hui Li
School of Informatics, Xiamen University
Xiamen, China
hui@xmu.edu.cn

Chenliang Li
School of Cyber Science and Engineering, Wuhan University
Wuhan, China
cllee@whu.edu.cn

Zhenhua Huang
School of Computer Science, South China Normal University
Guangzhou, China
huangzhenhua@m.scnu.edu.cn

ABSTRACT

Review-based recommender systems (RRS) have received an increasing interest since reviews greatly enhance recommendation quality and interpretability. However, existing RRS suffer from high computational complexity, biased recommendation and poor generalization. The three problems make them inadequate to handle real recommendation scenarios. Previous studies address each issue separately, while none of them consider solving three problems together under a unified framework. This paper presents LUME (a Lightweight Unbiased Multi-teacher Ensemble) for RRS. LUME is a novel framework that addresses the three problems simultaneously. LUME uses multi-teacher ensemble and debiased knowledge distillation to aggregate knowledge from multiple pretrained RRS, and generates a small, unbiased student recommender which generalizes better. Extensive experiments on various real-world benchmarks demonstrate that LUME successfully tackles the three problems and has superior performance than state-of-the-art RRS and knowledge distillation based RS.

CCS CONCEPTS

• **Information systems** → **Recommender systems**.

KEYWORDS

review-based recommender systems, bias in recommender systems, knowledge distillation

*Both authors contributed equally to this research.
†Work done during internship at Alibaba Group.
‡Corresponding author. Supported by the Natural Science Foundation of China (No. 61972328), Alibaba Group through Alibaba Innovative Research program.

CIKM '22, October 17–21, 2022, Atlanta, GA, USA

ACM ISBN 978-1-4503-9236-5/22/10…$15.00
https://doi.org/10.1145/3511808.3557629

ACM Reference Format:
Guipeng Xv, Xinyi Liu, Chen Lin, Hui Li, Chenliang Li, and Zhenhua Huang. 2022. Lightweight Unbiased Multi-teacher Ensemble for Review-based Recommendation. In *Proceedings of the 31st ACM International Conference on Information and Knowledge Management (CIKM '22), October 17–21, 2022, Atlanta, GA, USA.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3511808.3557629

1 INTRODUCTION

Online reviews are valuable feedback in recommender systems, as they provide explanations on various aspects of a product and guide users towards purchase. Due to the prevalence of online reviewing sites, Review-based Recommender Systems (RRS) have attracted a great amount of attention [1, 2, 11, 17, 25, 31, 35].

Although existing RRS provide high-quality and interpretable recommendations, they still suffer from issues such as **high computational complexity, biased recommendations, and poor generalization**. (1) As shown in Fig. 1 (a), since state-of-the-art RRS typically adopt deep neural networks to analyze review contents, they have much more parameters than the non-review-based, shallow recommendation models such as Matrix Factorization (MF) [12]. (2) As shown in Fig. 1(b), RRS are recently found to show sentiment bias [16], i.e., they generate more significant errors on critical users who write fewer positive reviews than on positive users who post more positive reviews. (3) As shown in Fig. 1(c), RRS make inaccurate (i.e., large median MAE) and unreliable (i.e., large variance) predictions on low-rating reviews, which are with insufficient training samples but more valuable than high-rating reviews [22]. These issues severely affect the efficiency in terms of inference time and storage cost, and the quality and fairness of recommendations, when RRS are deployed in practice.

In the literature, model compression [5, 14, 24, 28], debiasing [3, 30] and generalization [15] have been studied for RS. However, existing works address these problems separately. Furthermore, the three issues are correlated, e.g., model complexity and generalization ability, generalization ability to low-rating reviews and bias against critical users. Lacking consideration of any of the above problems will result in sub-optimal, ineffective and/or inefficient RRS that are inadequate to handle real recommendation scenarios.

Inspired by recent advances on Knowledge Distillation (KD) [8], we propose Lightweight Unbiased Multi-teacher Ensemble (LUME) for the review-based recommendation task. LUME first captures high quality, generalizable common knowledge shared within multiple teacher RRS, and then trains a lightweight student model and mitigates the biases via a KD process. Experiments on various real-world RS datasets verify the superiority of LUME to make consistent, high-quality and unbiased review-based recommendations.

In summary, our contributions are three-fold. (1) We design a novel framework, LUME, which simultaneously alleviates high computational complexity, bias and poor generalization of RRS. (2) Unlike most existing KD-based RS that only learn from one teacher model, LUME compresses and accelerates multiple teachers by fusing common knowledge and adapting it to the student model. (3) The KD process of LUME is specially designed to handle biases

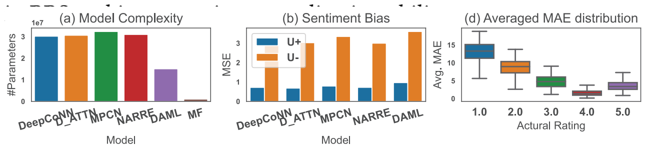

Figure 1: (a) Number of parameters in five state-of-the-art RRS (i.e., DeepCoNN [35], MPCN [25], NARRE [2], DAML [17] and D_ATTN [11]) and conventional MF (with embedding/factor size = 50). (b) RRS produce much higher Mean Square Error (MSE) on critical users $\mathcal{U}^-$ than on positive users $\mathcal{U}^+$. $\mathcal{U}^-, \mathcal{U}^+$ are decided based on the sentiment scores of reviews as in [16]. (c) Averaged MAE (i.e., MAE averaged over different RRS on every sample) on one-star rating samples has a large variance and a large median value.

2 RELATED WORK

Review-based Recommender Systems (RRS). Traditional RRS have utilized latent semantic analysis [34], LDA [27] or latent factor model [20] to model reviews and provide better recommendations. Recently, deep learning based techniques such as CNN [35]MLP [11], LSTM [23, 26], Autoencoder [36], and the attention mechanism [2, 17, 25] have significantly facilitated the development of RRS [31].

Biases in RS. Several biases have been observed in RS [4], including selection bias [19], conformity bias [18], position bias [9], popularity bias [32], and exposure bias [29]. The ubiquitous sentiment bias problem [16] in RRS is hard to handle. To mitigate biases in RS, numerous debiasing methods [4] adopt causal inference methods [29], or regularization methods [3, 16].

Knowledge Distillation (KD) for RS. A number of recent studies have investigated KD [8, 10] in RS, where a small student model learns to rank items [14, 24] from one large teacher models to reduce model complexity. The ranking distillation framework can be enhanced by a three-player game where a discriminator is introduced to learn the true data distribution from the teacher [28]. The student and the teacher can learn from each other simultaneously [13, 33] to enhance the interpretability or the performance of RS. For RRS, an adversarial distillation framework [5] is proposed to make review predictions.

3 OUR METHOD: LUME

As shown in Fig. 2, LUME mainly consists of two parts. Given a set of pre-trained RRS (i.e., teachers), LUME first learns a *HeadTeacher* model to fuse the knowledge from multiple teachers and further

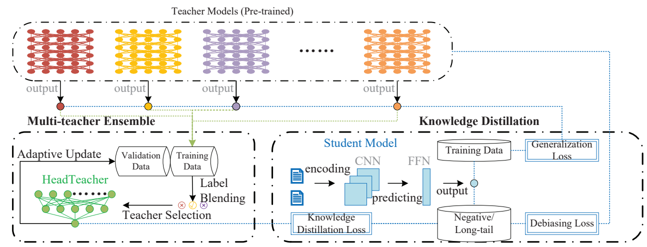

Figure 2: Overview of LUME.

improves the quality of common knowledge, using three steps: label blending, teacher selection, and adaptive model update (Sec. 3.1). Then, LUME trains a student model using the guidance from the HeadTeacher, mitigates biases, and strengthens generalization in a KD process (Sec. 3.2).

3.1 Multi-teacher Ensemble

Different from existing KD-based RS [5, 14, 24, 28] that leverage a single-teacher architecture, LUME uses a multi-teacher architecture, so that the student will not be easily misled by a single teacher if the teacher performs poorly in some cases. A natural approach to fuse multiple teachers is to use an ensemble model M^e, parameterized by Θ^e to integrate the predictions of multiple teachers, i.e., ensemble learning [6]. However, as illustrated in Sec. 1:different RRS produce inconsistent predictions on hard cases, which introduces noise to the KD process. It is questionable whether abnormal predictions from some teachers should be incorporated into the ensemble model. To overcome the above problem, we propose the *multi-teacher ensemble* to generate the HeadTeacher.

Suppose that we have a number of teacher models, where each teacher model $t \in \mathcal{T}$ gives the prediction $\hat{\mathbf{X}}^t_{u,i}$ for the rating $\mathbf{X}_{u,i}$ of a user $u \in \mathcal{U}$ on an item $i \in \mathcal{I}$. The teacher models are independently pre-trained on the training set $\mathcal{DS}$, and they are fixed during the training phase of the later KD process.

We first use a **label blending** step which traverses the training set $\mathcal{DS}$ and removes low-quality teacher predictions to fuse outputs from multiple teachers. A label $l(t,u,i)$ is assigned for each teacher t on every prediction $\hat{\mathbf{X}}^t_{u,i}$ to indicate whether the prediction should be utilized in training the HeadTeacher. If the deviation between the prediction and the actual rating, i.e., $|\hat{\mathbf{X}}^t_{u,i} - \mathbf{X}_{u,i}|$, is larger than a predefined threshold ζ, $\hat{\mathbf{X}}^t_{u,i}$ will be considered as abnormal and it will not benefit the ensemble learning.

Teacher selection. Then, the HeadTeacher takes the output of each teacher model $\hat{\mathbf{X}}^t_{u,i}$, if $l(t,u,i)$ equals 1, and make a fused prediction. The HeadTeacher uses a two-layer feed-forward network (FFN). In the first layer, predictions from teachers are aggregated to generate the probabilities of different rating values:

$$\mathbf{z}_{u,i} = \mathbf{w}_1\Big(concate\big(l(t,u,i)\hat{\mathbf{X}}^t_{u,i}, t \in \mathcal{T}\big)\Big) + \mathbf{b}_1, \tag{1}$$

where $concate(\cdot) \in \mathcal{R}^{|\mathcal{T}|\times 1}$ is a concatenated vector, $\mathbf{z}_{u,i} \in \mathbb{R}^{5\times 1}$ indicates the probability distribution of ratings. $\mathbf{w}_1 \in \mathcal{R}^{5\times|\mathcal{T}|}$ and $\mathbf{b}_1 \in \mathbb{R}^{5\times 1}$ are learnable weight vector and bias vector, respectively. In the second layer, different rating values are aggregated to form the predicted rating:

$$\hat{\mathbf{X}}^e_{u,i} = \mathbf{w}_2^T \mathbf{z}_{u,i} + b_2, \tag{2}$$

where $\mathbf{w}_2 \in \mathbb{R}^{5\times1}$ and b_2 are learnable parameters.

Adaptive model update. Besides, we use a subset of the testing data as a validation set $\mathcal{DV}$ to improve the generalization of LUME. We derive the gradient of the HeadTeacher parameters in the validation set and carry a small number of trials to update the ensemble model.The motivation is similar to model-agnostic meta-learning [7]: Since RRS will be updated using a gradient-based method on new data (including low-rating reviews) that they can not learn well (i.e., poor generalization), LUME is designed to find model parameters that are sensitive to new data so that small changes in model parameters will produce large improvements on the loss function.

3.2 Knowledge Distillation

Formally, the student model is denoted as M^s, parameterized with Θ^s, which makes predictions $\hat{\mathbf{X}}^s_{u,i} = M_{\Theta^s}(\mathbf{P}_u, \mathbf{Q}_i)$ for each user profile $\mathbf{P}_u$ and item profile $\mathbf{Q}_i$. We construct a user profile $\mathbf{P}_u$ by concatenating all reviews written by user u. An embedding vector is used to represent each review token, and thus a user profile is defined as $\mathbf{P}_u \in \mathcal{R}^{N_u \times N_w}$, where N_u is the maximal number of reviews that LUME includes in a user profile, and N_w is the number of the tokens that LUME considers for each review from its beginning. Similarly, we construct an item profile $\mathbf{Q}_i$ by concatenating all reviews written on item i.

The architecture of the student model, as most RRS models, consists of an encoding module that learns feature representations of textual reviews and a prediction module. The goal of the student model in LUME is to make it as lightweight as possible. We experimentally find that Convolutional Neural Network (CNN), as an encoding module, generates stable performance. To reduce the computational complexity, we use the same CNN module for both user profiles and item profiles. The prediction module in the student model is a one-layer FFN that predicts the ratings in one to five stars.

The student optimizes a combined loss that helps the student mimic the behavior of the HeadTeacher via a *teacher distillation loss* $\mathcal{L}_t$, *student loss* $\mathcal{L}_s$, and mitigates biases via a *debiasing loss* $\mathcal{L}_x$, and strengthens generalization via a *generalization losses* $\mathcal{L}_g$. The overall loss for training the student model is defined as:

$$\mathcal{L} = \lambda_t \mathcal{L}_t + \lambda_s \mathcal{L}_s + \lambda_x \mathcal{L}_x + \lambda_g \mathcal{L}_g, \tag{3}$$

where λ_t, λ_s, λ_x, λ_g are loss weights.

Teacher distillation loss. Recall that the HeadTeacher contains two layers, where the output of the first layer (i.e., logits $\mathbf{z}_{u,i}$ in Eq. 1) carries ensemble knowledge from different teachers, by predicting the probabilities of one to five rating stars, i.e., $\mathbf{z}_{u,i,c} = Pr(\mathbf{X}_{u,i} = c), c \in \{1,2,3,4,5\}$. However, the student outputs numerical rating values instead of discrete rating categories. Thus, the cross-entropy loss used in many KD systems [8] is infeasible for RRS.

To transfer the ensemble knowledge in M^e to M^s, LUME uses the logits as supervision signals and optimizes the MSE loss between logits and the student model's output as the teacher loss $\mathcal{L}_t$:

$$\mathcal{L}_t = \sum_{u\in\mathcal{U},\, i\in\mathcal{I},\, \mathbf{X}_{u,i}\neq0} \Big(\sum_c c \cdot \mathbf{z}_{u,i,c} - \hat{\mathbf{X}}^s_{u,i}\Big)^2, \tag{4}$$

where $c = \{1,2,3,4,5\}$ refers to discrete ratings in RS, $\mathbf{z}_{u,i,c}$ is the logit output from the first layer of HeadTeacher on the neuron for c (Eq. 1).

Student loss. $\mathcal{L}_s$ in LUME is defined between the ground truth rating value $\mathbf{X}_{u,i}$ and the output of the student to encourage the student to make accurate predictions:

$$\mathcal{L}_s = \sum_{u\in\mathcal{U},\, i\in\mathcal{I},\, \mathbf{X}_{u,i}\neq0} \big(\hat{\mathbf{X}}^s_{u,i} - \mathbf{X}_{u,i}\big)^2. \tag{5}$$

Debiasing loss. In the following, we use the sentiment bias, which exists in most RRS [16], as the example to illustrate how LUME mitigate biases. The idea can be generalized to other biases (e.g., popularity bias). Intuitively, to reduce sentiment bias, the student model must be enhanced to provide better predictions on negative users/items. We propose $E_v(\mathcal{S}, t)$ to evaluate teacher model t, based on how much the embedding vectors of negative items spread out in the batch containing samples $\mathcal{S}$: $E_v(\mathcal{S}, t) = \sum_{\mathbf{X}_{u,i}\in\mathcal{S}\,\&\, i\in\mathcal{I}^-} \|\mathbf{e}^t_i - \bar{\mathbf{e}^t}(\mathcal{S})\|^2_2$, where $\bar{\mathbf{e}^t}(\mathcal{S})$ is the mean embedding vector in the set $\mathcal{S}$. When the best teacher model x, in terms of the smallest E_v is selected, we can use the output of x (i.e., $\hat{\mathbf{X}}^x_{u,i}$) to guide the student model and reduce sentiment bias on negative items via the following debiasing loss:

$$\mathcal{L}_x = \sum_{u\in\mathcal{U},\, i\in\mathcal{I}^-,\, \mathbf{X}_{u,i}\neq0} \big(\hat{\mathbf{X}}^x_{u,i} - \hat{\mathbf{X}}^s_{u,i}\big)^2. \tag{6}$$

Generalization losses. The Generalization loss $\mathcal{L}_g$ is defined as $\mathcal{L}_g = \lambda_y \mathcal{L}_y + \lambda_z \mathcal{L}_z$. If teacher models do not agree with each other, we increase the uncertainty of student model's output. We first select samples O in the batch (i.e., $\mathbf{X}_{u,i} \in \mathcal{S}$) using the following evaluation function: $E_o(u,i) = \sum_{t\in\mathcal{T}} \sum_{\mathbf{X}_{u,i}\in\mathcal{S}} \big(\hat{\mathbf{X}}^t_{u,i} - \mathcal{X}_{u,i}\big)^2$, where $\mathcal{X}_{u,i}$ is the average output of all teacher models for the sample $\mathbf{X}_{u,i}$. If the variance of teacher model outputs (i.e., $E_o(u,i)$) is large, LUME uses the entropy-based regularizer $\mathcal{L}_y$ to increase the uncertainty of the final output:

$$\mathcal{L}_y = \sum_{u\in\mathcal{U},\, i\in\mathcal{I}^-,\, E_o(u,i)>\phi} \sum_{c=1}^{5} p(u,i,c)\log p(u,i,c), \tag{7}$$

where ϕ denotes a predefined threshold to judge whether teachers agree or not. Simply connecting a FFN layer with softmax to the prediction layer of the student, we can obtain $p(u,i,c) = Pr(\mathbf{X}_{u,i} = c)$, which denotes the probability that user u gives item i a rating of c, where $0 \le p(u,i,c) \le 1$, $\sum_c p(u,i,c) = 1$, and $c \in \{1,2,3,4,5\}$.

To further enhance the generalization on low-value ratings, we present the error function $E_g(\mathcal{S}, t)$, to evaluate whether a teacher model t provides unbiased predictions on low ratings in a set of ratings $\mathcal{S}$: $E_g(\mathcal{S}, t) = \sum_{\mathbf{X}_{u,i}\in\mathcal{S}\,\&\,\mathbf{X}_{u,i}<3} \big(\hat{\mathbf{X}}^t_{u,i} - \mathbf{X}_{u,i}\big)^2$. When the best teacher model z, in terms of the smallest E_g is selected, we can use the output of z (i.e., $\hat{\mathbf{X}}^z$) to strengthen the student model's performance on low-value ratings:

$$\mathcal{L}_z = \sum_{u\in\mathcal{U},\, i\in\mathcal{I},\, \mathbf{X}_{u,i}\neq0} \big(\hat{\mathbf{X}}^z_{u,i} - \hat{\mathbf{X}}^s_{u,i}\big)^2. \tag{8}$$

4 EXPERIMENTS

Experiments are conducted on four Amazon datasets [21] and Yelp dataset. We apply 5-core pre-processing on Yelp to make sure each user/item has at least five ratings. We use 8:1:1 training/validation/test split. Five state-of-the-art RRS models are used as teacher models and competitors: DeepCoNN [35], MPCN [25], NARRE [2], DAML [17], D_ATTN [11]. Other baselines include simple RRS and state-of-the-art KD-based RS: (1) CNN: we train a student network with a CNN encoding module and a FFN prediction layer via the student loss in Eq. 5. This baseline does not

Table 1: MSE and NDCG@5 of different methods. Bold entries suggest that LUME outperforms the competitor.

DataSet	Games		Food		Kindle		Electronics		Yelp	
Model	MSE	NDCG	MSE	NDCG	MSE	NDCG	MSE	NDCG	MSE	NDCG
DeepCoNN	1.1581	0.7857	0.9942	0.8905	0.6962	0.8863	1.2912	0.8836	1.3294	0.8262
ratio/LUME	**1.0247**	**0.9015**	**1.0131**	**0.9965**	**1.0334**	1.0051	**1.0163**	1.0063	**1.0203**	1.0208
D_ATTN	1.1687	0.8534	1.0060	0.8173	0.6960	0.8900	1.2906	0.8867	1.3498	0.8112
ratio/LUME	**1.0341**	**0.9792**	**1.0252**	**0.9146**	**1.0331**	1.0093	**1.0158**	1.0098	**1.0360**	1.0022
MPCN	1.4325	0.7924	1.1966	0.8109	0.9077	0.8531	1.4075	0.8426	1.5145	0.7388
ratio/LUME	**1.2675**	**0.9092**	**1.2194**	**0.9075**	**1.3473**	**0.9675**	**1.1078**	**0.9596**	**1.1624**	**0.9128**
NARRE	1.1189	0.8686	0.9669	0.8921	0.6612	0.8929	1.2588	0.8944	1.2958	0.8272
ratio/LUME	0.9900	**0.9967**	0.9853	**0.9983**	0.9814	1.0126	0.9908	1.0186	0.9946	1.0220
DAML	1.1155	0.8546	0.9672	0.8911	0.7213	0.8872	1.3240	0.8890	1.3840	0.8138
ratio/LUME	0.9870	**0.9806**	0.9856	**0.9972**	**1.0707**	1.0061	**1.0421**	1.0124	**1.0622**	1.0054
CNN	1.1608	0.8625	0.9933	0.8924	0.6910	0.8781	1.2908	0.8721	1.3165	0.8019
ratio/LUME	**1.0271**	**0.9897**	**1.0122**	**0.9987**	**1.0257**	**0.9958**	**1.0160**	**0.9932**	**1.0104**	**0.9907**
CNN+KD	1.1377	0.8731	0.9845	0.8856	0.6751	0.8815	1.2732	0.8757	1.3035	0.8074
ratio/LUME	**1.0066**	1.0018	**1.0033**	**0.9910**	**1.0021**	**0.9997**	**1.0021**	**0.9973**	**1.0005**	**0.9975**
BD+BPR	3.8520	0.7569	3.5768	0.7698	2.8416	0.7730	3.2652	0.7894	3.2651	0.7021
ratio/LUME	**3.4082**	**0.8685**	**3.6450**	**0.8615**	**4.2179**	**0.8766**	**2.5700**	**0.8990**	**2.5060**	**0.8674**
CNN+KDgate	1.1416	0.8569	0.9855	0.8918	0.7009	0.8804	1.2874	0.8760	1.3125	0.7995
ratio/LUME	**1.0101**	**0.9832**	**1.0043**	**0.9980**	**1.0404**	**0.9984**	**1.0133**	**0.9976**	**1.0074**	**0.9878**
LUME	1.1302	0.8715	0.9813	0.8936	0.6737	0.8818	1.2705	0.8781	1.3029	0.8094

Table 2: Statistics of model parameters.

Method	**DeepCoNN**	**D_ATTN**	**MPCN**	**NARRE**	**DAML**
#Parameters	30,187,864	30,626,334	32,365,264	31,596,293	30,998,429
ratio/LUME	11.9868	12.1609	12.8514	12.5460	12.3086

Method	**CNN**	**CNN+KD**	**BD+BPR**	**CNN+KDDgate**	**LUME**
#Parameters	2,518,432	2,518,442	1,169,700	2,518,752	2,518,442
ratio/LUME	1.0000	1.0000	0.4645	1.0000	1.0000

use KD. (2) CNN+KD: we train a student network (the same as the CNN baseline) via the teacher loss in Eq. 4 and the student loss in Eq. 5. (3) CNN+KDgate: we train the student network (the same as the CNN baseline) with multiple teachers via a gating network [37]. (4) BD+BPR: we train a BPR student network using the bidirectional distillation framework [13]. Our source codes and parameter settings are publicly available[1].

4.1 Recommendation Performance

We use MSE and NDCG@5, to evaluate recommendation performance. Tab. 1 shows the results. For each method, we also calculate the ratio of its performance with respect to LUME's performance, i.e., "ratio/LUME". We can observe that: (1) LUME provides superior recommendations in terms of MSE and NDCG@5 than teacher models. In most datasets, the teacher models generate worse recommendations, i.e., higher MSE with ratio/LUME > 1.0 and lower NDCG with ratio/LUME < 1.0. (2) LUME consistently outperforms other KD-based competitors for both rating prediction and top-k recommendation.

4.2 Model Complexity

We can see that the number of parameters of each KD-based method is an order of magnitude fewer than that of each teacher model in Tab. 2. Compared among KD-based methods, BD+BPR has the least parameters. However, its performance is significantly worse than LUME's as illustrated in Tab. 1. This observation shows that LUME has achieved a good balance between model complexity and recommendation quality.

4.3 Bias Mitigation and Generalization

As shown in Tab. 3, we can see that LUME provides lower user sentiment bias (BU [16]) and item sentiment bias (BI [16]) than teacher models and competitors in most datasets, i.e., ratio/LUME > 1.0. By comparing LUME with CNN+KD which does not use the debiasing

Table 3: User sentiment bias and Item sentiment bias of different methods.

DataSet	Games		Food		Kindle		Electronics		Yelp	
Model	BU	BI	BU	BI	BU	BI	BU	BI	BU	BI
DeepCoNN	1.4471	0.8090	1.2958	0.8749	1.0811	0.7665	1.5338	1.2952	1.5279	1.2712
ratio/LUME	**1.0815**	**1.0239**	**1.0249**	**1.0812**	**1.0737**	**1.0879**	**1.0411**	**1.0382**	**1.0565**	**1.1087**
D_ATTN	1.5247	0.7582	1.3244	0.9047	1.0723	0.7633	1.4579	1.2635	1.5448	1.2599
ratio/LUME	**1.1395**	0.9596	**1.0475**	**1.1180**	**1.0650**	**1.0833**	0.9895	**1.0128**	**1.0682**	**1.0988**
MPCN	2.3533	1.7431	1.6758	1.3502	1.7521	1.0606	1.9555	1.5603	2.0117	2.3487
ratio/LUME	**1.7588**	**2.2062**	**1.3255**	**1.6686**	**1.7401**	**1.5053**	**1.3273**	**1.2507**	**1.3910**	**2.0484**
NARRE	1.3435	0.8244	1.2759	0.8067	1.0024	0.7044	1.4132	1.1890	1.4379	1.1538
ratio/LUME	**1.0041**	**1.0434**	**1.0092**	0.9969	0.9955	0.9997	0.9592	0.9531	0.9943	**1.0063**
DAML	1.3454	0.8119	1.2999	0.8279	1.2250	0.8485	1.7306	1.4007	1.7473	1.8656
ratio/LUME	**1.0055**	**1.0276**	**1.0282**	**1.0231**	**1.2166**	**1.2042**	**1.1746**	**1.1228**	**1.2082**	**1.6271**
CNN	1.4757	0.8516	1.2978	0.8648	1.0744	0.7477	1.5561	1.3249	1.4931	1.2093
ratio/LUME	**1.1029**	**1.0778**	**1.0265**	**1.0687**	**1.0670**	**1.0612**	**1.0562**	**1.0620**	**1.0324**	**1.0547**
CNN+KD	1.3854	0.7916	1.2759	0.8311	1.1032	0.7181	1.4976	1.2640	1.4762	1.1936
ratio/LUME	**1.0354**	**1.0019**	**1.0092**	**1.0271**	**1.0956**	**1.0192**	**1.0165**	**1.0132**	**1.0207**	**1.0410**
BD+BPR	0.5308	0.9731	0.1134	0.9188	1.9816	1.2120	2.3084	1.9884	1.7702	0.8750
ratio/LUME	0.3967	**1.2316**	0.0897	**1.1354**	**1.9680**	**1.7201**	**1.5668**	**1.5939**	**1.2240**	0.7631
CNN+KDgate	1.4828	0.8238	1.4152	0.9844	1.0658	0.7523	1.6582	1.3365	1.5485	0.6008
ratio/LUME	**1.1082**	**1.0427**	**1.1194**	**1.2165**	**1.0585**	**1.0677**	**1.1255**	**1.0713**	**1.0707**	0.5240
LUME	1.3380	0.7901	1.2643	0.8092	1.0069	0.7046	1.4733	1.2475	1.4462	1.1466

Table 4: Popularity bias and low value rating bias of different methods.

DataSet	Games		Food		Kindle		Electronics		Yelp	
Model	Pop	Gen	Pop	Gen	Pop	Gen	Pop	Gen	Pop	Gen
DeepCoNN	0.2817	2.6316	0.1230	2.6900	0.0200	2.4923	0.3034	3.8933	0.4369	1.8121
ratio/LUME	**1.1679**	**1.1617**	**1.3774**	**1.0250**	**1.9231**	**1.1350**	**1.0320**	0.9505	**1.0633**	**1.0325**
D_ATTN	0.2620	2.5432	0.0975	2.9195	0.0207	2.3779	0.3055	3.8267	0.4289	1.7960
ratio/LUME	**1.0862**	**1.1227**	**1.0918**	**1.1124**	**1.9904**	**1.0829**	**1.0391**	0.9342	**1.0438**	**1.0234**
MPCN	0.1816	3.5803	0.0974	3.7089	0.0636	3.6926	0.3057	4.7618	0.6882	2.2369
ratio/LUME	0.7529	**1.5805**	**1.0907**	**1.4132**	**6.1154**	**1.6816**	**1.0398**	**1.1625**	**1.6749**	**1.2746**
NARRE	0.2780	2.5140	0.1304	2.7454	0.0375	2.2483	0.3321	3.8746	0.4566	1.6684
ratio/LUME	**1.1526**	**1.1098**	**1.4602**	**1.0461**	**3.6058**	**1.0239**	**1.1296**	0.9459	**1.1112**	0.9507
DAML	0.2478	2.5670	0.1511	2.7885	0.0411	2.4755	0.3531	4.4006	0.6791	1.9426
ratio/LUME	**1.0274**	**1.1332**	**1.6920**	**1.0625**	**3.9519**	**1.1273**	**1.2010**	**1.0743**	**1.6527**	**1.1069**
CNN	0.2419	2.7303	0.1005	2.8199	0.0242	2.4103	0.2905	4.2088	0.4181	1.8658
ratio/LUME	**1.0029**	**1.2053**	**1.1254**	**1.0745**	**2.3269**	**1.0976**	**0.9881**	**1.0275**	**1.0175**	**1.0631**
CNN+KD	0.2664	2.5347	0.1127	2.6312	0.0163	2.2258	0.2949	4.2157	0.4117	1.7661
ratio/LUME	**1.1045**	**1.1189**	**1.2620**	**1.0026**	**1.5673**	**1.0136**	**1.0031**	**1.0292**	**1.0019**	**1.0063**
BD+BPR	0.5124	2.8332	0.2003	2.6767	0.0956	3.1203	0.5424	4.2100	0.9653	3.2654
ratio/LUME	**2.1244**	**1.2507**	**2.2430**	**1.0795**	**9.1923**	**1.4210**	**1.8449**	**1.0278**	**2.3492**	**1.8606**
CNN+KDgate	0.2427	2.2664	0.1128	3.0311	0.0244	2.2956	0.2967	4.1775	0.4362	1.7588
ratio/LUME	**1.0062**	**1.0005**	**1.2632**	**1.1549**	**2.3462**	**1.0454**	**1.0092**	**1.0198**	**1.0616**	**1.0022**
LUME	0.2412	2.2653	0.0893	2.6245	0.0104	2.1959	0.2940	4.0962	0.4109	1.7550

loss, we verify that debiasing loss can effectively reduce the sentiment bias. Note that, by further analyzing the results, we find that although BD+BPR produces lower BU, its results are meaningless because BD+BPR produces "equally" poor rating predictions (i.e., high MSE) for all users (See Tab. 1).

Similarly, we compute the popularity bias *Pop*, which is the recommendation performance divergence between popular items and long-tail items, and low value rating bias *Gen* between high rating values and low rating values. As shown in Tab. 4, we can find that: (1) LUME consistently provides better recommendations on long-tail items and thus, produces smaller popularity bias. Except MPCN on Games dataset, the teacher models and other competitors generate higher popularity bias (i.e., $ratio/LUME > 1.0$) than LUME. (2) LUME also outperforms teacher models and competitors in terms of Gen on most datasets, showing that it has a better generalization ability and handles different cases (including low-rating cases) well.

5 CONCLUSION

The presented LUME framework addresses high computational complexity, biased recommendation and poor generalization simultaneously. LUME fuses knowledge from multiple teachers and derives a HeadTeacher to transfer the common knowledge. It can easily deal with different biases, such as sentiment bias and popularity bias, and have a stronger generalization ability on low-rating reviews.

[1] https://github.com/XMUDM/LUME

REFERENCES

[1] Charu C. Aggarwal. 2016. *Recommender Systems - The Textbook*. Springer.
[2] Chong Chen, Min Zhang, Yiqun Liu, and Shaoping Ma. 2018. Neural Attentional Rating Regression with Review-level Explanations. In *WWW*. 1583–1592.
[3] Jiawei Chen, Hande Dong, Yang Qiu, and Xiangnan He. 2021. AutoDebias: Learning to Debias for Recommendation. In *SIGIR*. 21–30.
[4] Jiawei Chen, Hande Dong, Xiang Wang, Fuli Feng, Meng Wang, and Xiangnan He. 2020. Bias and Debias in Recommender System: A Survey and Future Directions. *arXiv Preprint* (2020). https://arxiv.org/abs/2010.03240
[5] Xu Chen, Yongfeng Zhang, Hongteng Xu, Zheng Qin, and Hongyuan Zha. 2019. Adversarial Distillation for Efficient Recommendation with External Knowledge. *ACM Trans. Inf. Syst.* 37, 1 (2019), 12:1–12:28.
[6] Thomas G. Dietterich. 2000. Ensemble Methods in Machine Learning. In *Multiple Classifier Systems*, Vol. 1857. 1–15.
[7] Chelsea Finn, Pieter Abbeel, and Sergey Levine. 2017. Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks. In *ICML*, Vol. 70. 1126–1135.
[8] Jianping Gou, Baosheng Yu, Stephen J. Maybank, and Dacheng Tao. 2021. Knowledge Distillation: A Survey. *Int. J. Comput. Vis.* 129, 6 (2021), 1789–1819.
[9] Huifeng Guo, Jinkai Yu, Qing Liu, Ruiming Tang, and Yuzhou Zhang. 2019. PAL: a position-bias aware learning framework for CTR prediction in live recommender systems. In *RecSys*. 452–456.
[10] Geoffrey E. Hinton, Oriol Vinyals, and Jeffrey Dean. 2015. Distilling the Knowledge in a Neural Network. *arXiv Preprint* (2015). https://arxiv.org/abs/1503.02531
[11] Dongmin Hyun, Chanyoung Park, Min-Chul Yang, Ilhyeon Song, Jung-Tae Lee, and Hwanjo Yu. 2018. Review Sentiment-Guided Scalable Deep Recommender System. In *SIGIR*. 965–968.
[12] Yehuda Koren, Robert M. Bell, and Chris Volinsky. 2009. Matrix Factorization Techniques for Recommender Systems. *Computer* 42, 8 (2009), 30–37.
[13] Wonbin Kweon, SeongKu Kang, and Hwanjo Yu. 2021. Bidirectional Distillation for Top-K Recommender System. In *WWW*. 3861–3871.
[14] Jae-woong Lee, Minjin Choi, Jongwuk Lee, and Hyunjung Shim. 2019. Collaborative Distillation for Top-N Recommendation. In *ICDM*. 369–378.
[15] Ruirui Li, Xian Wu, Xiusi Chen, and Wei Wang. 2020. Few-Shot Learning for New User Recommendation in Location-based Social Networks. In *WWW*. 2472–2478.
[16] Chen Lin, Xinyi Liu, Guipeng Xv, and Hui Li. 2021. Mitigating Sentiment Bias for Recommender Systems. In *SIGIR*. 31–40.
[17] Donghua Liu, Jing Li, Bo Du, Jun Chang, and Rong Gao. 2019. DAML: Dual Attention Mutual Learning between Ratings and Reviews for Item Recommendation. In *KDD*. 344–352.
[18] Yiming Liu, Xuezhi Cao, and Yong Yu. 2016. Are You Influenced by Others When Rating?: Improve Rating Prediction by Conformity Modeling. In *RecSys*. 269–272.
[19] Benjamin M. Marlin, Richard S. Zemel, Sam T. Roweis, and Malcolm Slaney. 2007. Collaborative Filtering and the Missing at Random Assumption. In *UAI*. 267–275.
[20] Julian J. McAuley and Jure Leskovec. 2013. Hidden factors and hidden topics: understanding rating dimensions with review text. In *RecSys*. 165–172.
[21] Julian J. McAuley, Rahul Pandey, and Jure Leskovec. 2015. Inferring Networks of Substitutable and Complementary Products. In *KDD*. 785–794.
[22] Sangwon Park and Juan L.Nicolau. 2015. Asymmetric effects of online consumer reviews. *Annals of Tourism Research* 50 (2015), 67–83. Issue 2015.
[23] Peijie Sun, Le Wu, Kun Zhang, Yanjie Fu, Richang Hong, and Meng Wang. 2020. Dual Learning for Explainable Recommendation: Towards Unifying User Preference Prediction and Review Generation. In *WWW*. 837–847.
[24] Jiaxi Tang and Ke Wang. 2018. Ranking Distillation: Learning Compact Ranking Models With High Performance for Recommender System. In *KDD*. 2289–2298.
[25] Yi Tay, Anh Tuan Luu, and Siu Cheung Hui. 2018. Multi-Pointer Co-Attention Networks for Recommendation. In *KDD*. 2309–2318.
[26] Quoc-Tuan Truong and Hady W. Lauw. 2019. Multimodal Review Generation for Recommender Systems. In *WWW*. 1864–1874.
[27] Hongning Wang, Yue Lu, and ChengXiang Zhai. 2011. Latent aspect rating analysis without aspect keyword supervision. In *KDD*. 618–626.
[28] Xiaojie Wang, Rui Zhang, Yu Sun, and Jianzhong Qi. 2018. KDGAN: Knowledge Distillation with Generative Adversarial Networks. In *NeurIPS*. 783–794.
[29] Yixin Wang, Dawen Liang, Laurent Charlin, and David M. Blei. 2020. Causal Inference for Recommender Systems. In *RecSys*. 426–431.
[30] Jae woong Lee, Seongmin Park, and Jongwuk Lee. 2021. Dual Unbiased Recommender Learning for Implicit Feedback. In *SIGIR*. 1647–1651.
[31] Le Wu, Xiangnan He, Xiang Wang, Kun Zhang, and Meng Wang. 2021. A Survey on Neural Recommendation: From Collaborative Filtering to Content and Context Enriched Recommendation. *arXiv Preprint* (2021). https://arxiv.org/abs/2104.13030
[32] Yang Zhang, Fuli Feng, Xiangnan He, Tianxin Wei, Chonggang Song, Guohui Ling, and Yongdong Zhang. 2021. Causal Intervention for Leveraging Popularity Bias in Recommendation. In *SIGIR*. 11–20.
[33] Yuan Zhang, Xiaoran Xu, Hanning Zhou, and Yan Zhang. 2020. Distilling Structured Knowledge into Embeddings for Explainable and Accurate Recommendation. In *WSDM*. 735–743.
[34] Yin Zhang, Yueting Zhuang, Jiangqin Wu, and Liang Zhang. 2009. Applying probabilistic latent semantic analysis to multi-criteria recommender system. *AI Commun.* 22, 2 (2009), 97–107.
[35] Lei Zheng, Vahid Noroozi, and Philip S. Yu. 2017. Joint Deep Modeling of Users and Items Using Reviews for Recommendation. In *WSDM*. 425–434.
[36] Jin Peng Zhou, Zhaoyue Cheng, Felipe Pérez, and Maksims Volkovs. 2020. TAFA: Two-headed Attention Fused Autoencoder for Context-Aware Recommendations. In *RecSys*. 338–347.
[37] Jieming Zhu, Jinyang Liu, Weiqi Li, Jincai Lai, Xiuqiang He, Liang Chen, and Zibin Zheng. 2020. *Ensembled CTR Prediction via Knowledge Distillation*. 2941–2958.