Lightweight Unbiased Review-based Recommendation Based on Knowledge Distillation

Guipeng Xv*, Hao Liu*, Meifang Zeng*, Chenyuan Zhang*, Wei Zhou *

School of Informatics
Xiamen University
Xiamen, China
{23020211153907, 23020211153898, 23020211153986, 23020211153988, 23020211153997}@stu.xmu.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 and biased recommendationThe two problems make them inadequate to handle real recommendation scenarios. Though there exist studies working on addressing each issue seperately, none of them consider solving these problems together under a unified framework. LUME is a novel framework that addresses these problems simultaneously. LUME uses multi-teacher ensemble and debiased knowledge distillation to aggregate knowledge from multiple pre-trained RRS, and generate a small, debiased student recommender. Extensive experiments on various real-world benchmarks demonstrate that LUME successfully tackles the two problem and has superior performance than the state-of-the-art knowledge distillation based recommender systems.

Introduction

Recommender Systems (RS) are powering our everyday life. RS help in picking our favorite movies to watch, desirable products to purchase, interested news feed to follow, and even our next friends to connect to. Reviews are valuable auxiliary feedback in RS, as they provide explanations on various aspects of a product and guide users towards purchase. Due to the widespread prevalence of online reviewing sites, review-based RS (RRS) have attracted a great amount of attention (Aggarwal 2016). Though existing RRS provide high-quality and interpretable recommendations (Lyu et al. 2021), they still suffer from several problems.

P1: High Computational Complexity. The state-of-the-art RRS typically adopt deep neural networks with a great number of models from parameters. For example, the number of parameters of RRS models are much larger than the non-review-based matrix factorization. The high computational complexity leads to unaffordable inference time and storage cost, and it is difficult to deploy these cumbersome RRS in practice. Therefore, it is beneficial to develop a lightweight RRS model, while retaining the recommendation accuracy as complex RRS models.

P2: Biased Recommendations. Various types of biases naturally exist in RS (Chen et al. 2019), which profoundly affects the quality of recommendations. For instance, the recently found sentiment bias (Lin et al. 2021) leads to unsatisfied recommendations for certain users. The RRS generate more significant errors on critical users (i.e., users who write fewer positive reviews) than on positive users (i.e., users who post more positive reviews). Another example is the well-known popularity bias (Wei et al. 2021). RRS perform much worse on long-tail items than on popular items. And as time goes by, the bias will accumulate and cause Matthew's phenomenon. Therefore, to deliver fair treatments for the whole user/item universe, debiasing techniques are indispensable to RRS to increase the degraded performance on affected groups of users and items.

In the literature, model compression and debiasing have been studied for RS. However, each of these works addresses one problem only. Lacking consideration of any of the aforementioned problems will result in sub-optimal, ineffective and/or inefficient RRS models that are inadequate to handle real recommendation scenarios.

Recently, knowledge distillation (KD) (Guo et al. 2019) has attracted increasing attention due to its ability to produce a much smaller *student* model while retaining the ability of the original large *teacher* model. Inspired by recent studies on KD, we plan to present a Lightweight Unbiased Multi-teacher Ensemble (abbreviated as LUME) model to make consistent, high-quality and unbiased review-based recommendations.

LUME trains a HeadTeacher model to capture the correlations among teachers (i.e., individual RRS) and transfers the common knowledge shared within multiple teachers to the lightweight student model via a KD process with different losses tackle the problems faced by existing RRS. Finally, the lightweight student model is used for the review-based recommendation task.

The major contributions of this paper are summarized as follows:

- We design a novel framework, LUME, which simultaneously addresses the three problems of high computational complexity, and bias in RRS.

*These authors contributed equally.

- 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.

Related Work

Review-based RS.

The research of Recommender Systems (RS) for alleviating information overload problem has a long history. Review-based RS, utilizing available review texts in RS to provide high-quality recommendations with better interpretations, is an important branch of RS. Traditional review-based RS have utilized latent semantic analysis, LDA and latent factor model, to model reviews and provide better recommendations. Recently, deep neural networks and deep learning based techniques, including CNN, LSTM, Auto-encoder, Transformer and the attention mechanism, have significantly facilitated the development of review-based RS.

Bias in RS.

Although RS have been successfully deployed in many applications, the ubiquitous bias problem in RS is still hard to handle (Chen et al. 2019). In the literature, several biases have been observed in RS:

Selection Bias (van der Maaten and Hinton 2008): the observed ratings in RS are not a representative sample of all ratings.

Conformity Bias (Liu, Cao, and Yu 2016): users in RS rate similarly due to various social factors but doing so does not conform with their own preferences.

Position Bias (Hinton, Vinyals, and Dean 2015): users tend to interact with items in higher position of the recommendation list even if they are not relevant.

Popularity Bias (Wei et al. 2021): RS prefer to recommend popular items more frequently than their original popularity in the data.

Exposure Bias (Liu et al. 2020): users are only exposed to part of the data and unobserved interactions do not always indicate dislike.

Sentiment Bias (Lin et al. 2021): RS models make significantly more accurate recommendations on users/items having more positive feedback than on users/items having more negative feedback.

The first debias category of these works is utilizing causal inference to discover the actual cause of an user-item interaction in RS (Guo et al. 2019). There are other works trying to improve the design of the model architectures so that biases can be mitigated (Chen et al. 2018). Lastly, unlike above works which debias in the training phase, some works (Seo et al. 2017) design unbiased metrics for evaluating RS better in the testing phase.

Knowledge Distillation in RS.

Knowledge distillation (KD) (Guo et al. 2019) learns a small student model from one or multiple large teacher models to reduce model complexity. In RS community, several KD-based methods have been proposed. (Wang et al. 2018) designs the ranking distillation framework that trains a smaller student model to learn to rank items from both the training data and the supervision of a larger teacher model. (Lee, Park, and Lee 2021) improves the ranking distillation by sampling items in the soft target according to their rankings.

Interestingly, KD not only helps reduce the complexity of RS models, but also brings additional benefits to RS, e.g., extracting the core and useful information from the larger teacher model to avoid inaccurate recommendations (Liu et al. 2019).

Proposed Solution

Overview

Fig. 1 provides an overview of LUME that mainly consists of two parts. Given a set of pre-trained RRS models (i.e., teachers), LUME first learns a *HeadTeacher* model to fuse the knowledge from multiple teachers and further improves the quality of common knowledge, which is contained in HeadTeacher, using label blending and teacher selection on the training set, and adaptive model update on the validation set. Then, LUME trains the student model using the guidance from the HeadTeacher via the teacher loss $\mathcal{L}_t$ and the student loss $\mathcal{L}_s$, mitigates biases via a debiasing loss $\lambda_x\mathcal{L}_x$, and strengthens generalization via two generalization losses $\mathcal{L}_y$ and $\mathcal{L}_z$. The overall loss for training student model is defined as:

$$\mathcal{L} = \lambda_t\mathcal{L}_t + \lambda_s\mathcal{L}_s + \lambda_x\mathcal{L}_x + \lambda_y\mathcal{L}_y + \lambda_z\mathcal{L}_z, \qquad (1)$$

where λ_t,λ_s,λ_x,λ_y and λ_z are loss weights.

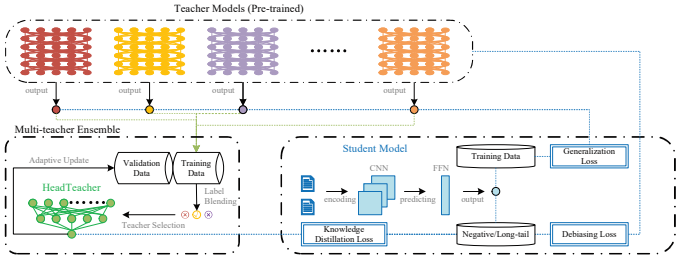

Figure 1: Overview of LUME.

Multi-teacher Ensemble

To reduce the high computational complexity of review-based RS, LUME follows the idea of KD. A vanilla KD method uses the logits of a large deep model as the teacher knowledge to guide the learning of the student model. Different from existing KD-based RS that leverage a single-teacher architecture, LUME uses a multi-teacher architecture. Multi-teacher architecture is more suitable for real review-based recommendation scenarios since the performance of different RRS models is "diverse". The multiple, complex "teacher" models are pre-trained on the same set of training samples, and then LUME transfers the knowledge from multiple teacher models and trains a student model with less parameters. This way, the student model will not be easily misled by a single teacher if the teacher performs poorly in some cases.

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

Now, the question is how to fuse the knowledge from multiple teachers. A natural approach is to use an ensemble model M_{Θ^e} to integrate the predictions of multiple teachers, and make one prediction for a sample:$\hat{X}^e_{u,i}$. Ensemble learning, which combines the insights of multiple machine learning models to facilitate accurate decisions, has been extensively studied. It is questionable whether abnormal predictions from some teachers should be incorporated in training the ensemble model. To overcome the above problem, we propose the *multi-teacher ensemble* to generate a *HeadTeacher*.

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

Then, the HeadTeacher takes the output of each teacher model $\hat{X}^t_{u,i}$, if $l(t,u,i) = 1$, and make a fused prediction, i.e., $\hat{X}^e_{u,i} = M_{\Theta^e}(concate(l(t,u,i)\hat{X}^t_{u,i},\ t \in \mathcal{T}))$where $concate(\cdot)$ is a concatenated vector. The HeadTeacher uses a two-layer feed-forward network (FFN). In the first layer, predictions from individual teachersare aggregated to generate the probability of different rating values. In the second layer, different rating values are aggregated to formed the predicted rating.

$$z_{u,i} = w_1(concate(l(t,u,i)\hat{X}^t_{u,i}, t \in \mathcal{T})) + b_1, \quad (2)$$

$$\hat{X}^e_{u,i} = w_2^T z_{u,i} + b_2, \quad (3)$$

where $z_{u,i} \in \mathbb{R}^{5\times 1}$ indicates the probability distribution of ratings, $w_1, w_2, b_1, b_2 \in \mathbb{R}^{5\times 1}$ are learnable weight vector and bias vector, respectively.

Furthermore, we use a subset of the testing data as a validation set $\mathcal{DV}$ to improve the generalization of LUME, i.e., adaptive model update step. 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 behind it is similar to model-agnostic meta-learning: 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 the model parameters will produce large improvements on the loss function.

Knowledge Distillation

Formally, a student model is denoted as M^s, parameterized with Θ^s, which makes predictions $\hat{X}^s_{u,i} = M^s_{\Theta^s}(P_u, Q_i)$ for each user profile P_u and item profile Q_i.

Given the historical feedback, we construct a user profile 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 $P_u \in R^{N_u * N_w}$, 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 consruct an item profile Q_i by concatenating all reviews written on item i.

The design goal of the student model in LUME is to make it as lightweight as possible. In RRS, the recommendation model usually consists of an encoding module that learns feature representations of textual reviews and a prediction module that generates outputs based on user, item, and review features.

There are a variety of choices in designing a lightweight architecture with an encoding module and a prediction module for the student model. We experimentally find that Convolutional Neural Network (CNN), as an encoding module, generates stable performance, since CNNs capture proximity information in short review texts. To reduce the computational complexity, we use the same CNN module for both user profile and item profile.

The prediction module in the student model is a one-layer FFN that predicts the ratings in one to five stars. The student model in LUME in KD is optimized via a *teacher distillation loss* $\mathcal{L}_t$ and a *student loss* $\mathcal{L}_s$. The teacher distillation loss is used to help the student model to mimic the behavior of the HeadTeacher. Recall that the HeadTeacher contains two layers, where the output of the first layer (i.e., logits $z_{u,i}$ in Eq. 2) carries ensemble knowledge from various individual teacher models, by predicting the probability of one to five rating stars, i.e., $z_{u,i,c} = Pr(X_{u,i} = c)$, $c \in \{1,2,3,4,5\}$. However, the student model outputs numerical rating values instead of discrete rating categories. Thus, the cross-entropy loss used in many KD systems 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 $\mathcal{L}_t$:

$$\mathcal{L}_t = \sum_{u\in U, i\in I, X_{u,i}\neq 0} (\sum_c c \cdot z_{u,i,c} - \hat{X}^s_{u,i})^2, \quad (4)$$

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

The student loss $\mathcal{L}_s$ in LUME is defined between the ground truth rating value $X_{u,i}$ and the output of the student model to encourage the student model to make accurate predictions:

$$\mathcal{L}_s = \sum_{u\in U, i\in I, X_{u,i}\neq 0} (\hat{X}^s_{u,i} - X_{u,i})^2, \quad (5)$$

Debiasing

In the following, we use the sentiment bias (Lin et al. 2021), which exists in most RRS, as the example to illustrate how LUME mitigate biases.

Sentiment bias is defined as the divergence between recommendation performance on positive users/items and negative users/items. The positive and negative users/items are decided based on an unsupervised sentiment analysis tool like TextBlob [1]. which returns a sentiment polarity value for each user/item profile.

We rank users by their sentiment polarity values so that we can extract the top 10% users as positive users $\mathcal{U}^+$, and the bottom 10% users as negative users $\mathcal{U}^-$. Similarly, we have positive items $\mathcal{I}^+$ and negative items $\mathcal{I}^-$. Then, the user sentiment bias and the item sentiment bias for a RS model can be obtained as $BU(RS) = E(RS,\mathcal{U}^-,\mathcal{I}) - E(RS,\mathcal{U}^+,\mathcal{I})$, and $BI(RS) = E(RS,\mathcal{U}, I^- - E(RS,\mathcal{U}, I^+)$, respectively. E is the evaluation metric, e.g., MSE.

Intuitively, to reduce sentiment bias, the student model must be enhanced to provide better predictions on negative users/items. Lin et al. explain that increasing the embedding variance on negative items can effectively mitigate sentiment bias. (Lin et al. 2021) Thus, LUME poses constraints on the embedding vectors. We propose $E_v(t)$ to evaluate teacher model t, based on how much the embedding vectors of negative items spread out in the batch containing samples $\mathcal{S}$:

$$E_v(t) = \sum_{X_{u,i}\in\mathcal{S}\ \&\ i\in\mathcal{I}^-} \|e_i^t - e^t(\bar{\mathcal{S}})\|_2^2, \tag{6}$$

where $\bar{e^t}(\mathcal{S})$ 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 model x (i.e., $\hat{\mathbf{X}}^{\mathcal{X}}_{u,i}$) to guide the student model and reduce sentiment bias on negative items via the following debiasing loss:

$$\mathcal{I}_x = \sum_{u\in\mathcal{U},i\in\mathcal{I}^-,X_{u,i}\neq 0} (\hat{X}^{\mathcal{X}}_{u,i} - \hat{X}^{\mathcal{S}}_{u,i})^2 \tag{7}$$

Generalization

If teacher models do not agree with each other, we increase the uncertainty of student model's output. We first select samples $\mathcal{O}$ in the batch (i.e., $X_{u,i} \in \mathcal{S}$) using the following evaluation function:

$$E_o(u,i) = \sum_{t\in\mathcal{T}} \sum_{X_{u,i}\in\mathcal{S}} (\hat{X}^t_{u,i} - \mathcal{X}_{u,i})^2, \tag{8}$$

where $\mathcal{X}_{u,i}$ is the average output of all teacher models for the sample $X_{u,i}$. If the variance of teacher model outputs (i.e., $E_o(u,i)$) is large, LUME use the entropy-based regularizer $\mathcal{L}_y$ to increase the uncertainty of the final output:

$$\mathcal{L}_y = \sum_{u\in\mathcal{U},i\in\mathcal{I}^-,E_o(u,i)>\phi} \sum_{c=1}^{5} p(u,i,c)\log p(u,i,c), \tag{9}$$

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 model, we can obtain $p(u,i,c) = Pr(X_{u,i} = c)$, 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 to low-value ratings, we present the error function $E_g(t)$, in Eq. 10, to evaluate if a teacher model t provides unbiased predictions on low ratings in a set of ratings $\mathcal{S}$:

$$E_g(\mathcal{S},t) = \sum_{X_{u,i}\in\mathcal{S},X_{u,i}<3} (\hat{X}^t_{u,i} - X_{u,i})^2. \tag{10}$$

When the best teacher model z, in terms of the smallest E_g is selected, we can use the output of z (i.e., $\hat{X}^z$) to strengthen the student model's performance on low-value ratings:

$$\mathcal{L}_z = \sum_{u\in\mathcal{U},i\in\mathcal{I},X_{u,i}\neq 0} (\hat{X}^z_{u,i} - \hat{X}^s_{u,i})^2. \tag{11}$$

Table 1: Statistics of the data.

Dataset	#Users	#Items	#Reviews	Sparsity
Food	14,683	8,715	151,253	99.8818
Kindle	68,225	61,936	982,618	99.9767
Games	826,769	50,212	1,324,753	99.9968
Electronics	192,405	63,003	1,689,188	99.9861
Yelp	1,070,074	36,490	3,766,145	99.9904

Experiments

Experiment Setup

Datasets. We use five public datasets, including four Amazon review dataset (McAuley and Leskovec 2013) and the Yelp dataset [2]. We apply 5-core preprocessing on datasets to make sure each user/item has at least five ratings. We use 8:1:1 training/validation/test split. The statistics of the data are shown in Tab 1.

Teacher Models and Other Competitors. Five state-of-the-art RRS models are used as teacher models and competitors: DeepCoNN (Zheng, Noroozi, and Yu 2017), MPCN (Tang and Wang 2018), NARRE (Chen et al. 2018), DAML (Li et al. 2020) and D_ATTN (Kang et al. 2020). We use public code [3], with the default parameter settings, for the five RRS. Other baselines include simple RRS and state-of-the-art KD-based RS: CNN, CNN+KD, CNN+KDgate (Zhu et al. 2020) and BD+BPR (Kweon, Kang, and Yu 2021). We use the public implementation for BD [4].

Recommendation Performance

We use MSE and NDCG@k to evaluate the performance for rating prediction and top-k recommendation, two prevalent tasks in RS (Aggarwal 2016). By default, we report the result when k = 5.

Tab. 2 shows the performance of different methods. For each method, we also calculate the ratio of its performance with respect to LUME's performance, i.e., "ratio/LUME".

[1] https://textblob.readthedocs.io

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

[3] https://github.com/noveens/reviews4rec

[4] https://github.com/WonbinKweon/BD_WWW2021

Table 2: 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 3: 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	2,518,432
ratio/LUME	1.0000	1.0000	0.4645	1.0000	1.0000

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. In the remaining datasets, LUME generates comparable results, i.e., ratio/LUME approaches 1.0. (2) LUME consistently outperforms other KD-based competitors for both rating prediction and top-k recommendation.

Model Complexity

To validate whether KD is able to reduce the complexity of RRS, we report the number of parameters, which includes word embedding vectors and all internal variables, for different methods in Tab. 3. We can observe that the number of parameters of each KD-based method is an order of magnitude fewer than that of each teacher model. Compared among KD-based methods, we can see BD+BPR has the least parameters because it is not a Review-based RS and its performance is signifcantly worse than LUME's as illustrated in Tab. 2. This observation suggests that LUME has achieved a good balance between model complexity and recommendation quality.

Biases and Generalization

Following (Lin et al. 2021), we adopt the user sentiment bias (BU) and the item sentiment bias (BI) to evaluate the degree of sentiment bias in RRS:

$$BU = \sum_{u \in U^-, i \in I} (\hat{X}_{u,i} - X_{u,i})^2 - \sum_{u \in U^+, i \in I} (\hat{X}_{u,i} - X_{u,i})^2, \tag{12}$$

$$BI = \sum_{u \in U, i \in I^-} (\hat{X}_{u,i} - X_{u,i})^2 - \sum_{u \in U, i \in I^+} (\hat{X}_{u,i} - X_{u,i})^2, \tag{13}$$

Table 4: User sentiment bias and Item sentiment bias of different methods. Bold entries suggest that LUME outperforms the competitor.

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

where the notations have been described in Debiasing subsection.

Lower BU and BI suggest more fair and unbiased recommendations. As shown in Tab. 4, we can see that LUME provides lower BU and BI than teacher models and competitors in most datasets, i.e., ratio/LUME >1.0. The improvement is particularly significant on MPCN. By comparing LUME with CNN+KD which does not use the debiasing 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. 2).

In summary, we can conclude that LUME provides consistent high-quality and unbiased recommendations for different cases including the hard-to-predict low-rating reviews.

Conclusion

Analyzing valuable information contained in online textual reviews can greatly enhance recommendation quality and interpretability. Consequently, review-based recommender systems have received increased research interest. However, review based recommendation systems face two severe challenges: high computational complexity and biased recommendations. This paper presents LUME, a Lightweight Unbiased Multi-teacher Ensemble for review-based recommendation, which address the three challenges simultaneously. The knowledge distillation process of LUME is specially designed to handle biase in review based recommender systems. Extensive experiments on various real-world benchmarks demonstrate that LUME can generate more accurate recommendation performance with much fewer model parameters.

References

Aggarwal, C. C. 2016. *Recommender Systems - The Textbook*. Springer. ISBN 978-3-319-29657-9.

Chen, C.; Zhang, M.; Liu, Y.; and Ma, S. 2018. Neural Attentional Rating Regression with Review-level Explanations. In Champin, P.; Gandon, F.; Lalmas, M.; and Ipeirotis, P. G., eds., *Proceedings of the 2018 World Wide Web Conference on World Wide Web, WWW 2018, Lyon, France, April 23-27, 2018*, 1583–1592. ACM.

Chen, X.; Zhang, Y.; Xu, H.; Qin, Z.; and Zha, H. 2019. Adversarial Distillation for Efficient Recommendation with External Knowledge. *ACM Trans. Inf. Syst.*, 37(1): 12:1–12:28.

Guo, H.; Yu, J.; Liu, Q.; Tang, R.; and Zhang, Y. 2019. PAL: a position-bias aware learning framework for CTR prediction in live recommender systems. In Bogers, T.; Said, A.; Brusilovsky, P.; and Tikk, D., eds., *Proceedings of the 13th ACM Conference on Recommender Systems, RecSys 2019, Copenhagen, Denmark, September 16-20, 2019*, 452–456. ACM.

Hinton, G. E.; Vinyals, O.; and Dean, J. 2015. Distilling the Knowledge in a Neural Network. *CoRR*, abs/1503.02531.

Kang, S.; Hwang, J.; Kweon, W.; and Yu, H. 2020. DE-RRD: A Knowledge Distillation Framework for Recommender System. In d'Aquin, M.; Dietze, S.; Hauff, C.; Curry, E.; and Cudré-Mauroux, P., eds., *CIKM '20: The 29th ACM International Conference on Information and Knowledge Management, Virtual Event, Ireland, October 19-23, 2020*, 605–614. ACM.

Kweon, W.; Kang, S.; and Yu, H. 2021. Bidirectional Distillation for Top-K Recommender System. In Leskovec, J.; Grobelnik, M.; Najork, M.; Tang, J.; and Zia, L., eds., *WWW '21: The Web Conference 2021, Virtual Event / Ljubljana, Slovenia, April 19-23, 2021*, 3861–3871. ACM / IW3C2.

Lee, J.; Park, S.; and Lee, J. 2021. Dual Unbiased Recommender Learning for Implicit Feedback. In *SIGIR*, 1647–1651. ACM.

Li, R.; Wu, X.; Chen, X.; and Wang, W. 2020. Few-Shot Learning for New User Recommendation in Location-based Social Networks. In Huang, Y.; King, I.; Liu, T.; and van Steen, M., eds., *WWW '20: The Web Conference 2020, Taipei, Taiwan, April 20-24, 2020*, 2472–2478. ACM / IW3C2.

Lin, C.; Liu, X.; Xv, G.; and Li, H. 2021. Mitigating Sentiment Bias for Recommender Systems. In Diaz, F.; Shah, C.; Suel, T.; Castells, P.; Jones, R.; and Sakai, T., eds., *SIGIR '21: The 44th International ACM SIGIR Conference on Research and Development in Information Retrieval, Virtual Event, Canada, July 11-15, 2021*, 31–40. ACM.

Liu, D.; Cheng, P.; Dong, Z.; He, X.; Pan, W.; and Ming, Z. 2020. A General Knowledge Distillation Framework for Counterfactual Recommendation via Uniform Data. In Huang, J.; Chang, Y.; Cheng, X.; Kamps, J.; Murdock, V.; Wen, J.; and Liu, Y., eds., *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval, SIGIR 2020, Virtual Event, China, July 25-30, 2020*, 831–840. ACM.

Liu, D.; Li, J.; Du, B.; Chang, J.; and Gao, R. 2019. DAML: Dual Attention Mutual Learning between Ratings and Reviews for Item Recommendation. In Teredesai, A.; Kumar, V.; Li, Y.; Rosales, R.; Terzi, E.; and Karypis, G., eds., *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2019, Anchorage, AK, USA, August 4-8, 2019*, 344–352. ACM.

Liu, Y.; Cao, X.; and Yu, Y. 2016. Are You Influenced by Others When Rating?: Improve Rating Prediction by Conformity Modeling. In Sen, S.; Geyer, W.; Freyne, J.; and Castells, P., eds., *Proceedings of the 10th ACM Conference on Recommender Systems, Boston, MA, USA, September 15-19, 2016*, 269–272. ACM.

Lyu, Y.; Yin, H.; Liu, J.; Liu, M.; Liu, H.; and Deng, S. 2021. Reliable Recommendation with Review-level Explanations. In *37th IEEE International Conference on Data Engineering, ICDE 2021, Chania, Greece, April 19-22, 2021*, 1548–1558. IEEE.

McAuley, J. J.; and Leskovec, J. 2013. Hidden factors and hidden topics: understanding rating dimensions with review text. In Yang, Q.; King, I.; Li, Q.; Pu, P.; and Karypis, G., eds., *Seventh ACM Conference on Recommender Systems, RecSys '13, Hong Kong, China, October 12-16, 2013*, 165–172. ACM.

Seo, S.; Huang, J.; Yang, H.; and Liu, Y. 2017. Interpretable Convolutional Neural Networks with Dual Local and Global Attention for Review Rating Prediction. In Cremonesi, P.; Ricci, F.; Berkovsky, S.; and Tuzhilin, A., eds., *Proceedings of the Eleventh ACM Conference on Recommender Systems, RecSys 2017, Como, Italy, August 27-31, 2017*, 297–305. ACM.

Tang, J.; and Wang, K. 2018. Ranking Distillation: Learning Compact Ranking Models With High Performance for Recommender System. In Guo, Y.; and Farooq, F., eds., *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2018, London, UK, August 19-23, 2018*, 2289–2298. ACM.

van der Maaten, L.; and Hinton, G. 2008. Visualizing Data using t-SNE. *Journal of Machine Learning Research*, 9(86): 2579–2605.

Wang, X.; Zhang, R.; Sun, Y.; and Qi, J. 2018. KDGAN: Knowledge Distillation with Generative Adversarial Networks. In *NeurIPS*, 783–794.

Wei, T.; Feng, F.; Chen, J.; Wu, Z.; Yi, J.; and He, X. 2021. Model-Agnostic Counterfactual Reasoning for Eliminating Popularity Bias in Recommender System. In *KDD*, 1791–1800. ACM.

Zheng, L.; Noroozi, V.; and Yu, P. S. 2017. Joint Deep Modeling of Users and Items Using Reviews for Recommendation. In *WSDM*, 425–434. ACM.

Zhu, J.; Liu, J.; Li, W.; Lai, J.; He, X.; Chen, L.; and Zheng, Z. 2020. Ensembled CTR Prediction via Knowledge Distillation. In *CIKM*, 2941–2958. ACM.