Adversarial Distillation for Efficient Recommendation with External Knowledge

XU CHEN, School of Software, Tsinghua University
YONGFENG ZHANG, Department of Computer Science, Rutgers University
HONGTENG XU, Department of ECE, Duke University
ZHENG QIN, School of Software, Tsinghua University
HONGYUAN ZHA, College of Computing, Georgia Institute of Technology

Integrating external knowledge into the recommendation system has attracted increasing attention in both industry and academic communities. Recent methods mostly take the power of neural network for effective knowledge representation to improve the recommendation performance. However, the heavy deep architectures in existing models are usually incorporated in an embedded manner, which may greatly increase the model complexity and lower the runtime efficiency.

To simultaneously take the power of deep learning for external knowledge modeling as well as maintaining the model efficiency at test time, we reformulate the problem of recommendation with external knowledge into a *generalized distillation framework*. The general idea is to free the complex deep architecture into a separate model, which is only used in the training phrase, while abandoned at test time. In particular, in the training phrase, the external knowledge is processed by a comprehensive teacher model to produce valuable information to teach a simple and efficient student model. Once the framework is learned, the teacher model is abandoned, and only the succinct yet enhanced student model is used to make fast predictions at test time. In this article, we specify the external knowledge as user review, and to leverage it in an effective manner, we further extend the traditional *generalized distillation framework* by designing a Selective Distillation Network (SDNet) with *adversarial adaption* and *orthogonality constraint* strategies to make it more robust to noise information.

Extensive experiments verify that our model can not only improve the performance of rating prediction, but also can significantly reduce time consumption when making predictions as compared with several state-of-the-art methods.

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

Additional Key Words and Phrases: Recommendation system, personalization, distillation network, external knowledge, adversarial training

Authors' addresses: X. Chen, School of Software, Tsinghua University, Beijing, China, 100084; email: xu-ch14@mails.tsinghua.edu.cn; Y. Zhang, Department of Computer Science, Rutgers University, NJ 08854-8014, USA; email: yongfeng.zhang@rutgers.edu; H. Xu, Department of ECE, Duke University, NC 27708, USA; email: hongteng.xu@duke.edu; Z. Qin, School of Software, Tsinghua University, Beijing, China, 100084; email: qinzh@mails.tsinghua.edu.cn; H. Zha, College of Computing, Georgia Institute of Technology, GA 30332, USA; email: zha@cc.gatech.edu.

1046-8188/2018/12-ART12 $15.00
https://doi.org/10.1145/3281659

ACM Reference format:
Xu Chen, Yongfeng Zhang, Hongteng Xu, Zheng Qin, and Hongyuan Zha. 2018. Adversarial Distillation for Efficient Recommendation with External Knowledge. *ACM Trans. Inf. Syst.* 37, 1, Article 12 (December 2018), 28 pages.
https://doi.org/10.1145/3281659

1 INTRODUCTION

External knowledge, such as user review or product image, has been widely incorporated into modern recommender systems, because it can not only reveal more comprehensive user/item properties to enhance the recommendation performance, but also provides opportunities to make the recommendations more explainable [6, 7, 49, 50, 52].

Among the different types of external knowledge, user review is a most popular one due to its easy accessibility and high effectiveness [5]. To incorporate textual information into a recommender system, early methods, such as HFT [31], RBLT [41], and CTR [43], mainly focused on how to effectively integrate topic model (e.g., LDA [1]) with matrix factorization (MF). However, as the bag-of-words (BOG) assumption in topic models is limited in discovering semantic information, these models can hardly extract sufficient valuable knowledge from user reviews to enhance the recommendation performance [51].

Very recently, with the growth of deep learning technology, several models, such as DeepCoNN [51] and TransNet [3], were designed to leverage convolutional neural network (CNN [21]) for a more comprehensive review on semantic mining, which were built upon a similar framework as shown in Figure 1(a). For a user-item pair (i, j), suppose the user has m reviews $\boldsymbol{W}^i = \{\boldsymbol{W}^i_1, \boldsymbol{W}^i_2, \ldots \boldsymbol{W}^i_m\}$, and the item has n reviews $\boldsymbol{W}^j = \{\boldsymbol{W}^j_1, \boldsymbol{W}^j_2, \ldots \boldsymbol{W}^j_n\}$ in the training set. To predict the rating from i to j, all the reviews in $\boldsymbol{W}^i$ and $\boldsymbol{W}^j$ are respectively concatenated into two documents, which are then fed into CNN review processors to extract useful features. The final rating is derived by interacting the output from different review processors. Despite effectiveness, we note that the deep architecture (i.e., CNN) is embedded in the framework, which greatly increases the model complexity, and will lower the runtime efficiency that is important for a practical recommender system. So in this article, we would like to ask: *"Can we not only take the power of deep learning for external knowledge modeling, but also keep a runtime-efficient model?"*

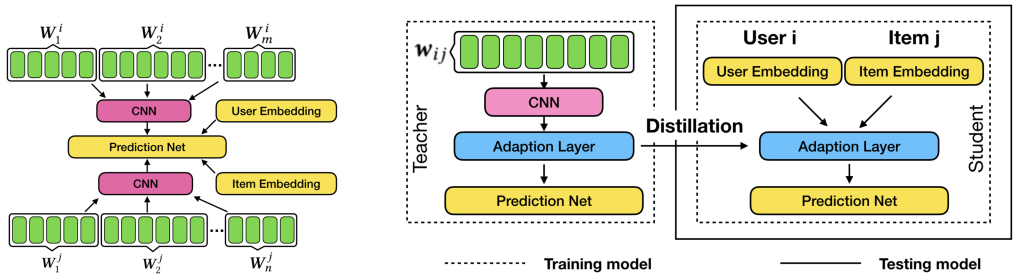
To answer this question, we reformulate the problem of recommendation with external knowledge into a *generalized distillation framework* (GDF), where the complex architecture (e.g., CNN) is moved into a separate component, which is only used in the training phrase, while abandoned at test time. In general, there are two components in our framework: one is a comprehensive teacher model (e.g., CNN), which is effective in extracting valuable information from the external knowledge. The other is a simple student model, which is efficient when making predictions. We would like to take both of their advantages to build an effective as well as runtime-efficient model. Specifically, in the training phrase, where we usually have more time budget, both the teacher and student models are utilized for parameter learning. By connecting the top layers between the teacher and student models, the valuable information learned by the teacher model is transfered (distilled) into the student model to help enhance its prediction ability. Once the framework is learned, the parameters in the student model will be biased to encode valuable information from the external knowledge, and we only use the efficient student model to make fast inference at test time. A high-level comparison between our model and several previous methods is presented in Table 1.

As an implementation of this framework, we select user review as the external knowledge. The overall principle can be seen in Figure 1(b). The student model is specified as a user-item prediction network, which takes a pair of user-item IDs as input and output a rating, while the teacher

Table 1. A High-Level Comparison Between Our Model with Previous Methods

Properties	HFT	RBLT	CTR	DeepCoNN	TransNet	SDNet
Reference	[31]	[41]	[43]	[51]	[3]	-
Model Depth	shallow	shallow	shallow	deep	deep	deep
Effectiveness	↓	↓	↓	↑	↑	↑
Runtime Efficiency	↑	↑	↑	↓	↓	↑

↑ means relative high and ↓ means relative low.



(a) Existing framework (b) Our knowledge distillation framework

Fig. 1. (a) The core architecture of many existing review-based recommendation models. $\{\boldsymbol{W}^i_1, \boldsymbol{W}^i_2, \ldots \boldsymbol{W}^i_m\}$ and $\{\boldsymbol{W}^j_1, \boldsymbol{W}^j_2, \ldots \boldsymbol{W}^j_n\}$ are the reviews of user i and item j in the training set, which are respectively concatenated before inputting into the CNN networks to make the final rating prediction. (b) Our idea of decomposing user review modeling into a teacher model, which only exists in the training phase, and will not be leveraged when making predictions. $\boldsymbol{w}_{ij}$ is the review from user i to item j.

model is implemented by a review prediction network, which maps a piece of user review into a rating. To enhance the student model by the powerful teacher machine, we can straightforwardly follow previous methods [15, 28] to fully connect their top layers for knowledge transfer. However, in the task of review-based recommendation, such methods can be less effective because the review information processed by the teacher model and the user/item properties encoded in the student model do not always agree with each other (see Figure 2): on one hand, many words in user reviews are not directly related to user preferences (or item features). In Figure 3, for example, the words *I ordered it for my birthday*, *I gave it away to my niece*, and *I had bought this for my husband* do not reflect user/item properties, and should not be considered when learning the user/item representations. On the other hand, many user preferences (or item features) cannot be discovered from review information. For example, a user may like the style of some clothes by looking at the product image, however, such direct visual preferences can hardly be expressed in her textual reviews. As a result, not all the user/item representations should be leveraged to receive the transferred review information.

To transfer just the right amount of valuable knowledge from the review information into the user/item representations, in this article, we carefully design a novel Selective Distillation Network (SDNet for short) tailored for the task of review-based recommendation. In particular, the top layer in each of the teacher and student models is designed to contain two types of features: one is the shared features, which are used for transferring valuable review information. The other is the private features, which are reserved to capture the mismatching knowledge (see Figures 2 and 4). Ideally, the shared features should only transfer valuable knowledge, and at the same time, all the valuable knowledge should be included in the shared features. To achieve

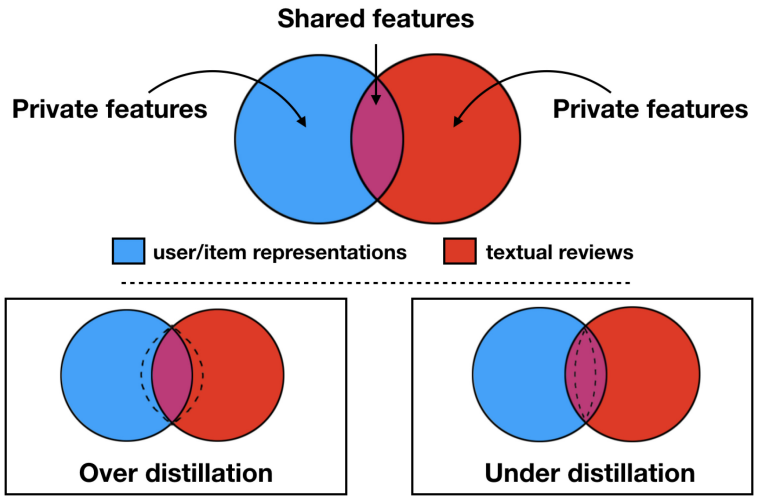


Fig. 2. The top sub-figure illustrates the relationship between the review information and the user/item representations. They not only share some common knowledge, but also enjoy their own private information. The private features in the review information means the user-item irrelevant knowledge, while the private features in the user/item representations indicate user/item properties that cannot be reflected in the review information. The bottom sub-figure shows two unfavored cases. In the first case, the linking strategy between the two components is less effective, and some user/item irrelevant information exists in the shared features, which will introduce noise when conducting distillation. In the second case, on the contrary, only a part of useful information has been leveraged for distillation, which will fail to fully take advantage of review information for effective user/item representation learning.

this goal, on one hand, we design the *adversarial adaption* strategy to connect the teacher and student models for effective knowledge transfer. On the other hand, we adopt the *orthogonality constraint* to forcefully have the shared and private features in each component encode different information, so as to push the valuable knowledge leaked in the private features into the shared features following the supervision signal.

Contributions. The main contributions of this article can be summarized as follows:

- We propose to reformulate recommendation with external knowledge into a generalized distillation framework, based on which we can not only take advantage of deep architectures for external knowledge modeling, but also can keep the proposed model computationally efficient at test time.
- We design a novel neural network, SDNet, for solving the mismatching problem between the teacher and student model in the generalized distillation framework. And we apply this method to the task of review-based recommendation for performance evaluation.
- We conduct extensive experiments to verify that, compared with the state-of-the-art methods, our models are not only able to improve the recommendation performance, but also can greatly reduce the computational time when making predictions.

In the following part of this article, we first describe the preliminaries in Section 2, and then illustrate our models in Section 3. In Section 4, we verify the effectiveness and efficiency of our methods with experimental results, and the related work and conclusions of this work are presented in Sections 5 and 6.

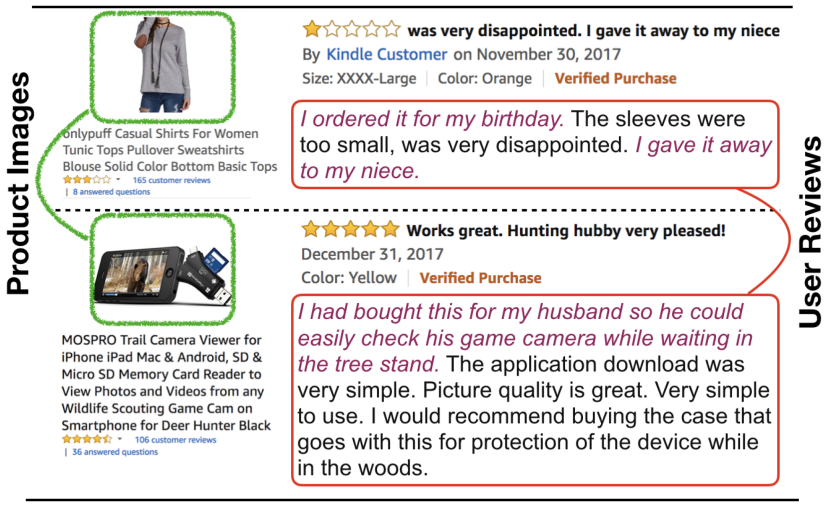


Fig. 3. An example of external knowledge on Amazon. In the review information, the words irrelevant with user preferences or item properties are highlighted in italic fonts.

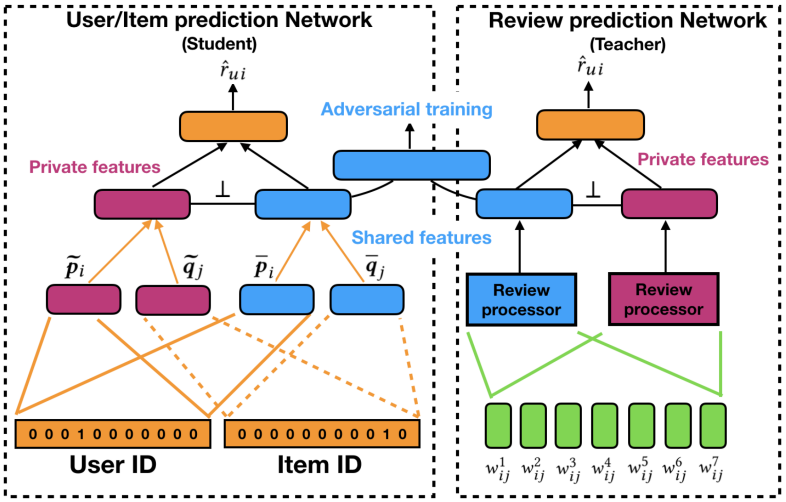


Fig. 4. Our overall framework. The left user-item prediction network and the right review prediction network are connected in an adversarial manner, and the shared and private features in the top layer of each network are encouraged to be orthogonal (denoted as $\perp$).

2 PRELIMINARIES

We first formulate the task of review-based recommendation, and shortly review several existing methods. Then, we briefly introduce the generalized distillation framework, highlighting its potential advantages for building an effective as well as efficient review-based recommender model.

2.1 Review-Based Recommendation

Problem Formulation. Suppose there are N users $\mathcal{U} = \{u_1, u_2, \ldots, u_N\}$ and M items $\mathcal{V} = \{v_1, v_2, \ldots, v_M\}$ in our system. Let r_{ij} represent a real-value explicit rating that user i scores on item j (e.g., an integer between 1 and 5 on Amazon). The set of all observed ratings is defined as

Table 2. Notations and Descriptions

Notations	Descriptions
$\mathcal{U}$	The set of N users $\{u_1, u_2, \ldots, u_N\}$.
$\mathcal{V}$	The set of M items $\{v_1, v_2, \ldots, v_M\}$.
D	The embedding dimension.
$\widetilde{\boldsymbol{p}}_i$, $\overline{\boldsymbol{p}}_i$	The private and shared embedding of user i.
$\widetilde{\boldsymbol{q}}_j$, $\overline{\boldsymbol{q}}_j$	The private and shared embedding of item j.
r_{ij}, $\mathcal{R}$	The rating scored by user i on item j, and the set of all ratings.
l_{ij}, $\boldsymbol{w}_{ij}$, $\mathcal{W}$	The length and word list of user i's review on item j: $\{w_{ij}^1, w_{ij}^2, \ldots, w_{ij}^{l_{ij}}\}$, and the set of all reviews $\{\boldsymbol{w}_{ij} \vert i \in \mathcal{U}, j \in \mathcal{V}\}$.
$\boldsymbol{W}^i$, $\boldsymbol{W}^j$	The set of reviews related to user i and item j.
λ	The regularization coefficient in the user-item prediction network.
b_0, b_i, $\boldsymbol{a_i}$	The parameters in the FM layer of user-item prediction network.
$\boldsymbol{t}_{ij}$, $\boldsymbol{s}_{ij}$	The complete private and shared embedding.
$\boldsymbol{t}_{w_{ij}}$, $\boldsymbol{s}_{w_{ij}}$	The outputs from shared and private review processors.
$\boldsymbol{V}$, g	The weighting and bias parameters in the review prediction network.
$\boldsymbol{V}^i$, $\boldsymbol{V}^f$, $\boldsymbol{V}^o$, $\boldsymbol{V}^c$	The weighting parameters in LSTM.
$\boldsymbol{g}^i$, $\boldsymbol{g}^f$, $\boldsymbol{g}^o$, $\boldsymbol{g}^c$	The bias parameters in LSTM.
t, f_j	The window size, and the output from the j-th filter in CNN.
λ_{1-4}	The weighting parameters that balance different objective functions.
σ, tanh, h	The sigmoid, hyperbolic tangent, and ReLU activation functions.

$\mathcal{R} = \{r_{ij}\}$, which is usually very sparse in real recommender systems. In addition, we also have the review information which reflects more comprehensive user preferences and item properties. Let $\boldsymbol{w}_{ij} = \{w_{ij}^1, w_{ij}^2, \ldots, w_{ij}^{l_{ij}}\}$ be a review posted by user i for item j, where w_{ij}^t ($t \in \{1, 2, \ldots, l_{ij}\}$) is the t-th word in the review, and l_{ij} is the review length. The set of all reviews is defined as $\mathcal{W} = \{\boldsymbol{w}_{ij}\}$, and we denote $\boldsymbol{W}^i$ and $\boldsymbol{W}^j$ as the set of reviews related to user i and item j, respectively. Given the training set of $\{\mathcal{U}, \mathcal{V}, \mathcal{R}, \mathcal{W}\}$, our task is to learn a predictive function f, which maps each user-item pair in the testing set into a real number.

It is noted that, when learning f, we can use all the information including $\{\mathcal{U}, \mathcal{V}, \mathcal{R}, \mathcal{W}\}$ in the training set. Whereas, when testing, for a user-item pair (i, j), we cannot use its review information $\boldsymbol{w}_{ij}$, because in real scenarios we have to predict the rating from a user to her non-interacted items, which means these items still have not received reviews from the user. For easy reference, we list the notations used throughout the paper in Table 2.

Typical Models. Using review information to enhance the recommendation performance is getting increasing attention in both research and industry communities [5]. As well known, MF is an effective recommender model, and TM are good at textual information modeling. So early researchers mainly focused on how to design effective integration manners between MF and TM with different assumptions for solving the problem of review-based recommendation [31, 41, 43]. A major problem of these methods is that the BoG assumption in the topic models usually discover collaborative information based on the lexical similarity. However, in real scenarios, user reviews can be very diverse and flexible; sentences with low percentage of token overlapping may still exhibit high semantic similarity. Failing to capture semantic information may degrade the accuracy of user/item profiling, which may lower the recommendation performance.

To solve this problem, recent methods [3, 51] took the power of deep learning in semantic mining to tame the unstructured review information, which obtained significant improvement against the

above methods. The key idea of these models is as follows: for a user-item pair in the testing set, all their review information in the training set is respectively concatenated into two different documents, which are then fed into CNN to extract useful features. The final rating is computed by interacting the output from different CNNs through a higher-level prediction network (see Figure 1(a)). Formally, for a user-item pair (i, j), the rating is predicted by

$$r_{ij} = g(\text{CNN}(\text{concat}(\boldsymbol{W}^i)), \text{CNN}(\text{concat}(\boldsymbol{W}^j))), \tag{1}$$

where $\text{CNN}(\cdot)$ is a convolutional neural network [21], and different reviews in $\boldsymbol{W}^i$ and $\boldsymbol{W}^j$ are cancatenated before inputting into CNN. g is a prediction network. Although this architecture has shown promising results for the recommendation task of rating prediction [3, 51], we note that the deep model (i.e., CNN) is integrated in an embedded manner, which may greatly degrade the runtime efficiency that is important for a real recommender system.

2.2 Generalized Distillation Framework

Recently, GDFas a novel learning paradigm is becoming more and more popular. It improves the traditional (student) model learning by taking advantage of a more powerful (teacher) machine [28]. In particular, the teacher model will leverage its power to guide the student model in the training phrase. While at runtime, the teacher model is abandoned, and the enhanced student model will be used to make predictions. A possible analogy to this learning paradigm is human learning with a teacher: when a student learns a course in school—for example, calculus—the teacher can provide additional insights, which cannot be easily discovered by herself, to help her understand some complex concepts. Hopefully this will make the student learn better and think deeper. However, when later in life the student faces a calculus problem, she will not be able to rely on the teacher's expertise anymore.

Formally, for a training dataset $\{(x_t, x_t^*, y_t)\}_{t=1}^n$ with two types of input x_t and x_t^*, GDF first learns the teacher model h_T based on $\{(x_t^*, y_t)\}_{t=1}^n$ by optimizing the following objective function:

$$h_T = \arg\min_{h \in \mathcal{H}_T} \frac{1}{n} \sum_{t=1}^{n} \ell(y_t, h(x_t^*)) + \Omega(\|h\|), \tag{2}$$

where $\mathcal{H}_T$ is the class of teacher function, $\Omega : \mathbb{R} \mapsto \mathbb{R}$ is an increasing function that regularizes the learning process. $\ell(\cdot)$ is a function that measures the distance between y_t and $h(x_t^*)$.

Then the student model h_S is learned from $\{(x_t, y_t)\}_{t=1}^n$ by minimizing the distance between $h_S(x_t)$ with the true label y_t as well as the output from the teacher $h_T(x_t^*)$.

$$h_S = \arg\min_{h \in \mathcal{H}_S} \frac{1}{n} \sum_{t=1}^{n} [\lambda \ell(y_t, h(x_t)) + (1 - \lambda)\ell(h_T(x_t^*), h(x_t))], \tag{3}$$

where $\lambda \in [0, 1]$ balances the importance between fitting the ground truth and the output from the teacher model. The learning of teacher and student models can also be combined into a joint end-to-end optimization problem [28].

In this framework, when $x_t^* = x_t$, the teacher model is usually more complex than the student model, and the power of h_T comes from its expressiveness, which is usually stronger than h_S. When $x_t^* \neq x_t$, x_t^* can be seen as the privileged information [42], which is usually more informative than x_t, and the power of h_T comes from its learned additional knowledge from x_t^*, which can guide the student to find better representations. It is noted that the privileged information is only accessible in the training phrase, but not available at test time [42]. For example, the clinic report can be seen as its radio-graph's privileged information. As a doctor can only write the report after seeing the radio-graph, the clinic report can only be used in the training phrase, but is unavailable at test time.

2.3 Review-Based Recommendation as Generalized Distillation Framework

In this section, we bridge the above two research areas, highlighting the potential advantages. To begin with, we re-organize the user behavioral data in the above settings as a set of quadruples $O = \{(i, j, \boldsymbol{w}_{ij}, r_{ij})_t\}_{t=1}^{n}$, where the element $(i, j, \boldsymbol{w}_{ij}, r_{ij})$ means user i interacts with item j by reviewing $\boldsymbol{w}_{ij}$ and rating r_{ij}. The rationalities of formulating review-based recommendation as a generalized distillation framework comes from two aspects:

- —As mentioned above, the crux of existing review-based models' low runtime efficiency lies in the embedded deep architecture. Under the generalized distillation framework, the complex model component can be separated into the teacher model, which is only used in the training phrase, and does not influence the runtime efficiency.
- —In a review-based recommender system, the ID information (e.g., i, j) is mainly used to distinguish different users (or items), and its rating prediction ability comes from the collaborative filtering assumption. Whereas, the review information (e.g., $\boldsymbol{w}_{ij}$) can explicitly reflect more comprehensive user/item properties and reveal the specific reasons for the predicted results (e.g., r_{ij}). See Figure 3 for an example; the upper review manifests that the reasons behind the low rating (i.e., "1" start) is that the user does not like small sleeves. Similarly, the bottom review "…The application download was very simple…" tells why the user gives the item a "5" star. The review information can usually provide additional valuable knowledge to understand the predicted ratings, which, obviously, is valuable privileged information to help enhance the rating prediction performance.

Based on these analyses, we regard user review as privileged information [42], and revisited the task of review-based recommendation by a generalized distillation framework.

In particular, we decompose the original architecture in Figure 1(a) into two parts (see Figure 1(b)): one is a teacher model based on $\{(\boldsymbol{w}_{ij}, r_{ij})_t\}_{t=1}^{n}$, and the other is a student model with $\{(i, j, r_{ij})_t\}_{t=1}^{n}$. The teacher model is a complex deep architecture for effective review modeling, while the student model is a simple collaborative filtering method for high runtime efficiency. In the training phrase, the teacher model is connected with the student model on the top layers to transfer valuable review information for learning optimal user/item representations (according to Equations (2) and (3)). Whereas, at runtime, only the efficient student model is leveraged for making fast predictions. By this learning paradigm, we can simultaneously take advantage of the teacher model's effectiveness as well as the student model's runtime efficiency.

Although this idea seems promising, user reviews are usually very noisy in real settings, and directly transferring (distilling) all the textual information into the student model can be less effective (as mentioned in Section 1). To solve this problem, we propose to partially connect the outputs between the teacher and student model, based on which we design *adversarial adaption* and *orthogonality constraint* strategies to guarantee the effectiveness of the knowledge distillation process.

3 SDNET: SELECTIVE DISTILLATION NETWORK

In this section, we first describe our model structure in depth, then we analyze the relationship of our proposed model with several other conventional models to highlight the importance of our model for personalized recommendation.

3.1 Model Structure

Essentially, we reformulate review-based recommendation into a generalized distillation framework, and the overall structure can be seen in Figure 4. It contains two components: one is a

succinct user-item prediction network, which inputs a user-item pair ID and outputs a rating. The other is a comprehensive review prediction network, which maps a user review into a rating. In the training phrase, by connecting these two models on their top layers, the strong review prediction network will transfer its learned effective review knowledge into the user-item prediction network to help enhance its prediction ability. To guarantee the effectiveness of the transferring process, two strategies—*adversarial adaption* and *orthogonality constraint*—are carefully designed. Once the framework is learned, only the efficient yet enhanced user-item prediction network is utilized to make fast predictions at test time. In the following, we describe the architectures as well as the design rationalities of different components more in depth.

3.1.1 User-Item Prediction Network (Student Model). Predicting the rating for a user-item pair (also called rating prediction, for short) is a classical recommendation problem. Our user-item prediction network is designed to solve this task, which takes user and item IDs as input, and outputs a rating. To learn a rating prediction model, traditional methods usually represent a user (or an item) by a unified low rank embedding. However, in our model, we encode user preferences (or item properties) into different embeddings according to whether they can be reflected in the review information. Formally, we map each user i into a shared embedding $\overline{\boldsymbol{p}}_i \in \mathbb{R}^D$ as well as a private embedding $\widetilde{\boldsymbol{p}}_i \in \mathbb{R}^D$, and similarly, each item j is projected into $\overline{\boldsymbol{q}}_j \in \mathbb{R}^D$ and $\widetilde{\boldsymbol{q}}_j \in \mathbb{R}^D$, respectively. The shared embeddings are leveraged to receive knowledge from the textual reviews, while the private ones are reserved for profiling other features.

To predict the final ratings, we first concatenate the embeddings of the shared and private features as $\boldsymbol{s}_{ij} = [\overline{\boldsymbol{p}}_i, \overline{\boldsymbol{q}}_j]$ and $\boldsymbol{t}_{ij} = [\widetilde{\boldsymbol{p}}_i, \widetilde{\boldsymbol{q}}_j]$, respectively. Then the result is computed by

$$\hat{r}_{ij} = PREDICT(\boldsymbol{s}_{ij}, \boldsymbol{t}_{ij}), \tag{4}$$

where $PREDICT(\cdot)$ is a predictive function, which is specified as a factorization machine (FM) layer upon the concatenate operation; that is,

$$\hat{r}_{ij} = FM([\boldsymbol{s}_{ij}, \boldsymbol{t}_{ij}]), \tag{5}$$

$$FM(\{z_i\}_{i=1}^n) = b_0 + \sum_{i=1}^{n} b_i z_i + \sum_{i=1}^{n} \sum_{j=(i+1)}^{n} (\boldsymbol{a_i}^T \boldsymbol{a_j}) z_i z_j, \tag{6}$$

where $[\cdot, \cdot]$ is the concatenate operation and $b_0, b_i \in \mathbb{R}$ and $\boldsymbol{a_i} \in \mathbb{R}^K$ are the weighting parameters.

The rationalities behind this design are as follows: (1) FM is effective in capturing nested variable interactions, which is important in our task for the heterogeneous information fusion between the knowledge learned from user reviews and the review-independent information encoded in the private embeddings. (2) We have explored to add some non-linear layers between the concatenate and FM layers; however, it did not lead to improved performance, and lowered the runtime efficiency at test time.

Finally, the objective function to be minimized in a user-item prediction network is

$$L_{UI} = \frac{1}{|\mathcal{R}|} \sum_{(i,j)\in\mathcal{R}} \left((r_{ij} - \hat{r}_{ij})^2 + \lambda(||\boldsymbol{s}_{ij}||_2^2 + ||\boldsymbol{t}_{ij}||_2^2) \right), \tag{7}$$

where the last two terms are regularizers to prevent over fitting, and λ is the corresponding regularization coefficient.

3.1.2 Review Prediction Network (Teacher Model). The review prediction network maps a piece of user review into a real-value rating, which is similar to the task of sentiment analysis (SA) [37]. As mentioned in Section 1, we hope to transfer higher-quality review information into the student

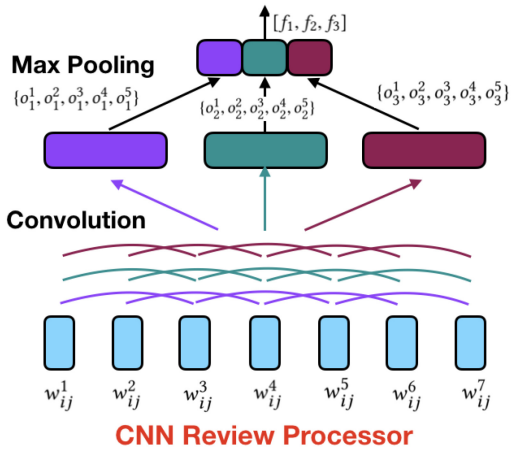


Fig. 5. Implementation of the CNN review processor.

model, while the user/item irrelevant knowledge should be automatically filtered out. To achieve this goal, we design two types of review processors (see Figure 4): one is used for connecting with the student model for distilling user/item related knowledge (we call it shared review processor), while the other is reserved for capturing the remaining information (we call it private review processor). In our model, the shared and private review processors are based on the same architecture with different parameters. The implementation details, such as CNN or LSTM review processors, will be described in the following. For review $\boldsymbol{w}_{ij}$, suppose the shared and private embeddings from the corresponding review processors are $\boldsymbol{s}_{\boldsymbol{w}_{ij}} \in \mathbb{R}^{2D}$ and $\boldsymbol{t}_{\boldsymbol{w}_{ij}} \in \mathbb{R}^{2D}$, respectively. We predict the rating by a linear affine function as

$$\hat{r}_{\boldsymbol{w}_{ij}} = \boldsymbol{V} \cdot [\boldsymbol{s}_{\boldsymbol{w}_{ij}}, \boldsymbol{t}_{\boldsymbol{w}_{ij}}] + g, \tag{8}$$

where $[\cdot, \cdot]$ is the concatenate operation, $\boldsymbol{V}$ is the weighting parameter, and g is the bias. Similar to the user-item prediction network, we want to minimize the root mean square error (RMSE) between the predicted ratings and the real ones. As a result, the loss function of review prediction network is

$$L_{Review} = \frac{1}{|\mathcal{R}|} \sum_{(i,j) \in \mathcal{R}} (r_{ij} - \hat{r}_{\boldsymbol{w}_{ij}})^2. \tag{9}$$

With the development of deep learning technology, textual information can be modeled by many promising methods, among which CNN [21] and long short-term memory (LSTM) [16] are two widely used ones. In the following, we describe how to implement our review processors based on CNN and LSTM, respectively (see Figures 5 and 6).

CNN Review Processor. Let $\boldsymbol{W}_{1:k} \in \mathbb{R}^{d \times k}$ be the embedded matrix of word list $\boldsymbol{w}_{ij}$, where each column of $\boldsymbol{W}_{1:k}$ corresponds to a d-dimensional word embedding. Suppose there are s filters in our CNN architecture, each of which is associated with a parameter $\boldsymbol{K}_j \in \mathbb{R}^{d \times t}$, where t is the window size. Then, the output f_j from the j-th filter is computed by

$$o_j^l = ReLU(\boldsymbol{W}_{l:(l+t-1)} * \boldsymbol{K}_j + b_j), \tag{10}$$

$$f_j = \max\{o_j^1, o_j^2, \ldots, o_j^{k-t+1}\}, \tag{11}$$

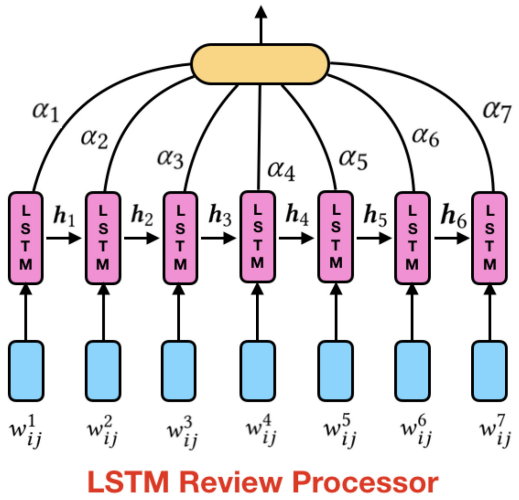


Fig. 6. Implementation of the LSTM review processor.

where b_j is the bias, $ReLU(x) = \max\{0, x\}$ is the active function [36], and $*$ is the Frobenius inner product operation. At last, the final output of the review processor is

$$output = h(\boldsymbol{H} \cdot [f_1, f_2, \ldots f_s]), \tag{12}$$

where $h(\cdot)$ is the Rectified Linear Unit (ReLU) [30] active function with $\boldsymbol{H}$ as its parameter. It should be noted that the shared and private CNN review processors are implemented using the same architecture; however, as their parameters are separated, they can learn different knowledge, and play different roles in the model learning process.

LSTM Review Processor. In the LSTM framework, suppose $\boldsymbol{W}_t$ is the embedding of word w_{ij}^t. At each step, the hidden state $\boldsymbol{h}_t$ as well as cell state $\boldsymbol{c}_t$ are computed by

$$\boldsymbol{i}_t = \sigma(\boldsymbol{V}^i \cdot [\boldsymbol{h}_{t-1}, \boldsymbol{W}_t] + \boldsymbol{g}^i), \tag{13}$$
$$\boldsymbol{f}_t = \sigma(\boldsymbol{V}^f \cdot [\boldsymbol{h}_{t-1}, \boldsymbol{W}_t] + \boldsymbol{g}^f), \tag{14}$$
$$\boldsymbol{o}_t = \sigma(\boldsymbol{V}^o \cdot [\boldsymbol{h}_{t-1}, \boldsymbol{W}_t] + \boldsymbol{g}^o), \tag{15}$$
$$\hat{\boldsymbol{c}}_t = tanh(\boldsymbol{V}^c \cdot [\boldsymbol{h}_{t-1}, \boldsymbol{W}_t] + \boldsymbol{g}^c), \tag{16}$$
$$\boldsymbol{c}_t = \boldsymbol{f}_t \odot \boldsymbol{c}_{t-1} + \boldsymbol{i}_t \odot \hat{\boldsymbol{c}}_t, \tag{17}$$
$$\boldsymbol{h}_t = \boldsymbol{o}_t \odot tanh(\boldsymbol{c}_t), \tag{18}$$

where $\{\boldsymbol{V}^i, \boldsymbol{V}^f, \boldsymbol{V}^o, \boldsymbol{V}^c, \boldsymbol{g}^i, \boldsymbol{g}^f, \boldsymbol{g}^o, \boldsymbol{g}^c\}$ are the weighting parameters to be learned, $\odot$ is the element-wise multiplication, and σ is sigmoid function.

Different words may play different roles in a sentence embedding. So instead of leveraging Vanilla LSTM, we introduce an attention mechanism, which has been verified to be effective in many areas [4, 13, 46]. In particular, we first compute the attention weight for each word w_{ij}^t by

$$\alpha_t = \frac{\exp(\boldsymbol{e}^T \boldsymbol{h}_t)}{\sum_{j=1}^{k} \exp(\boldsymbol{e}^T \boldsymbol{h}_j)}, \tag{19}$$

where $\boldsymbol{e}$ is the parameter that maps $\boldsymbol{h}_t$ into a scaler. Then, the output of the review processor is the weighted average over the hidden states, which is

$$output = \sum_{t=1}^{k} \alpha_t \boldsymbol{h}_t. \tag{20}$$

Similarly, shared and private LSTM review processors are implemented using the same architecture with different parameters.

3.1.3 Knowledge Distillation. We connect the top layers between the user-item prediction and review prediction networks. We hope the rating-aware representation of a user review, which is more informative in profiling users/items, can bias the user/item representations toward better representations. Ideally, review prediction network should distill the exact amount of valuable information into the user-item representations to help them fit the ground truth, as shown in the top subfigure in Figure 2. However, two unfavorable cases may arise if knowledge is not properly distilled from reviews to representations. One is over-distillation (in Figure 2), where some user/item irrelevant information is distilled through the shared embeddings, which will introduce noise to the learning of representations. Another is under-distillation, where the shared embeddings did not distill sufficient user/item relevant knowledge.

Inspired by these intuitions, we propose two strategies to control the knowledge distillation procedure, which are *adversarial adaption* between shared embeddings, and *orthogonality constraints* between shared and private embeddings, respectively.

Adversarial Adaption. Adversarial mechanism as a powerful training technology has been successfully applied in many fields (e.g., picture style transfer [23], super-resolution [22], and image-to-image translation [18]) for feature adaption because of its high performance. The basic idea of adversarial training is first proposed in generative adversarial net (GAN) [9]. It aims to learn a generative distribution P_z to match the true data distribution P_{data}, where z is a random noise vector. Specifically, GAN contains a generator G and a discriminator D, where G learns to generate samples that are similar to the ones from P_{data}, while D learns to discriminate whether such samples come from G or P_{data}. The final network is optimized by a min-max objective function as follows:

$$\min_G \max_D \Big(E_{x\sim P_{data}}[\log D(x)] + E_{z\sim P_z}[\log(1 - D(G(z)))]\Big). \tag{21}$$

In our model, the user-item prediction network (generator G) tries to generate $\boldsymbol{s}_{ij}$ that cannot be distinguished from $\boldsymbol{s}_w$ in the review prediction network by a discriminator D, which is trained to discover which network the shared embedding comes from. Formally, our model gives the following min-max objective function:

$$L_{GAN} = \min_{\Phi} \max_D \Big(E_{\boldsymbol{w}\sim \boldsymbol{W}}[\log D(\boldsymbol{s}_{\boldsymbol{w}})] + E_{(i,j)\sim S}[\log(1 - D(\boldsymbol{s}_{ij}))]\Big), \tag{22}$$

where W is the set of all reviews, Φ is the parameter set in user-item prediction network, and D is specialized as the sigmoid function.

As mentioned before, the adversarial learning paradigm has been verified to be very effective for feature adaption. Basically, in our model, the discriminator strengthens itself by focusing on more difficult $\boldsymbol{s}_{ij}$ provided by the generator, which—in turn—will set a higher request for the generator to produce better $\boldsymbol{s}_{ij}$ in the next training epoch. By mutually enhancing each other, the generator can effectively produce $\boldsymbol{s}_{ij}$ that cannot be distinguished from $\boldsymbol{s}_w$ even by a strong discriminator.

Previous studies [18, 38] also found that integrating GAN loss with a traditional loss, such as L_2, can lead to improved performance. This has also been verified in our experiments. In particular,

we tested with both L_1 and L_2 losses, and finally adopt L_2 loss $L_{Tra} = E_{(i,j)\sim\mathcal{R}}(||\boldsymbol{s}_{\boldsymbol{w}_{ij}} - \boldsymbol{s}_{ij}||_2)$ in our model due to its better performance.

With the above process, we are actually trying to solve the problem of over-distillation in Figure 2. On one hand, to fit the ground truth, the irrelevant information in the shared embedding will be penalized by the backpropagated supervision signal, which forces the shared embedding to only encode user/item relevant information. On the other hand, under the adversarial learning paradigm, the relevant information of the shared embedding is encouraged to be effectively distilled, such that the latent factors can learn more useful knowledge to predict user ratings.

Orthogonality Constraint. The above strategy tries to enforce that the shared embedding only encodes relevant information, however, such information may also exist in the private embeddings (corresponding to the under-distillation case in Figure 2). To push the user-item related knowledge from the private embeddings into the shared ones, we introduce orthogonality constraint in our model.

Essentially, we hope the overlap between the shared and private embeddings to be as small as possible. For easy implementation, we include the following orthogonal constraint as part of the loss, which has also been demonstrated to be effective by many previous studies [2, 27]:

$$L_{Orth} = \frac{1}{|\mathcal{R}|} \sum_{(i,j)\in\mathcal{R}} \left(\boldsymbol{s}_{ij}^T \boldsymbol{t}_{ij} + \boldsymbol{s}_{\boldsymbol{w}_{ij}}^T \boldsymbol{t}_{\boldsymbol{w}_{ij}}\right), \tag{23}$$

where the shared and private embeddings in both networks are encouraged to be orthogonal.

3.1.4 Overall Objective Function. By putting all the above together, the final objective function to be optimized is

$$L = L_{UI} + \lambda_1 L_{Review} + \lambda_2 L_{GAN} + \lambda_3 L_{Tra} + \lambda_4 L_{Orth}, \tag{24}$$

where $\lambda_{1\sim4}$ are the hyper-parameters. Once the model has been learned, the review information will be distilled into the user-item latent factors, and we only leverage *user-item prediction network* to compute the final ratings, where the runtime computational complexity is only determined by the FM operation, that is, $O(4D * K)$.

3.2 Further Discussion

For better understandings of our model, in this section, we provide some further analysis.

3.2.1 Comparison between LSTM and CNN Review Processor. Generally speaking, both LSTM and CNN review processors can be utilized in our final framework to model textual features, and by the following experiments, we found that they can achieve similar performance on most datasets. The difference is that the CNN review processor usually enables us to have higher training efficiency, while in the LSTM review processor, we can identify which words are more important according to their corresponding attention weights, which makes our model more transparent and explainable. This is an *efficiency-explainability* tradeoff, which inspires us to select different review processors according to the specific real-world applications.

3.2.2 Comparison between SDNet and TransNet. TransNet is a recently proposed algorithm [3], which obtained the state-of-the-art performance in the field of review-based recommendation. In general, both TransNet and SDNet include two components, and the source and target network in TransNet can be largely corresponded to the user-item prediction and review prediction network in SDNet, respectively. Instead of inputting massive user reviews into the source network as in TransNet, SDNet processes all the textual information in the review prediction network (similar to target network in TransNet), which is designed based on the following motivations:

(i) The strategy of merging all the review information as in TransNet can be less effective in terms of user profiling for the following reasons:
   — A user's reviews for different items may be quite diverse in terms of topic and sentiment. Indiscriminately leveraging all the other products' reviews to predict the current item's rating may be less effective. For example, when a user purchases a desktop, her previous preference on mobile phones may be less useful or even misleading.
   — If we take the dynamics of user preferences into consideration, the merged reviews may be ambiguous in terms of user profiling, that is, a user may dislike science-fiction movies in early years, but she may become fond of them later; as a result, there may exist very conflicting attitudes towards science-fiction movies in her merged reviews.

(ii) As we only use the compact and mathematically simple user-item prediction network at test time, the runtime efficiency will be significantly improved for real-world recommender systems. At the same time, by using the carefully designed adversarial-orthogonal knowledge distillation strategy, the review prediction network in our model can effectively filter the noise textual information, and distill the useful user/item-related information into the target network to help predict the ground truth. According to our experiments, this design not only enhances the test efficiency, but also achieves better performance in terms of rating prediction.

3.2.3 Comparison between SDNet and Generalized Distillation Framework. As mentioned in Section 2.2, SDNet is a special generalized distillation framework tailored for review-based recommendation. The power of the teacher model (i.e., review prediction network) comes from the review information, which is usually very noisy in real-world scenarios. Different from previous generalized distillation network, the teacher in our framework is designed to have the ability to discriminatively select useful knowledge to teach the student to fit the ground truth (as described in Section 3.1.3). Although SDNet is designed for review-based recommendation, it can also be easily adapted to other scenarios, where the teacher model has to process noise information sources.

4 EXPERIMENTS

In this section, we evaluate our proposed model, SDNet, based on real-world datasets. We begin by introducing the experimental setup, then we conduct extensive experiments to demonstrate the superiority of our proposed methods. In a summary, we study the following research questions:

— **RQ 1**: What is the performance of SDNet as compared with baseline models?
— **RQ 2**: How do the hyper-parameters influence the performance of SDNet?
— **RQ 3**: Is SDNet efficient when making predictions when comparing with baseline models?
— **RQ 4**: How do different components in SDNet contribute to the task of rating prediction?
— **RQ 5**: What is the effect of model pre-training for SDNet?
— **RQ 6**: What is the distilled knowledge learned by SDNet?

4.1 Experimental Setup

Datasets. We conduct our experiments based on the Amazon dataset[1] [10, 32], which is sampled from the well-known site www.amazon.com. It contains a large number of user rating and review behaviors, and for comprehensive evaluation, we conduct our experiments on the datasets with different sparsities:

[1] http://jmcauley.ucsd.edu/data/amazon/.

Table 3. Basic Statistics of Our Datasets

Datasets	#Users	#Items	#Words	#Total(R)	#Test(R)	#WPR	Density
Amazon Instant Video	5,130	1,685	85,480	37,126	3,729	93.55	0.433%
Automotive	2,928	1,835	46,400	20,473	1,828	86.96	0.381%
Baby	19,445	7,050	149,783	160,792	15,918	100.93	0.122%
Beauty	2,2363	12,101	175,974	198,502	22,569	90.58	0.071%
Cell Phones and Accessories	27,879	10,429	211,194	194,439	18,225	93.46	0.067%
Clothing Shoes and Jewelry	39,387	23,033	163,696	278,677	26,071	61.17	0.031%
Digital Music	5,541	3,568	143,566	64,706	6,109	204.73	0.327%
Grocery and Gourmet Food	1,4681	8,713	178,860	151,254	18,471	95.19	0.118%
Health and Personal Care	38,609	18,534	343,501	346,355	39,113	96.56	0.048%
Home and Kitchen	66,519	28,237	451,879	551,682	59,675	99.71	0.029%
Musical Instruments	1,429	900	30,988	10,261	918	92.28	0.798%
Office Products	4,905	2,420	126,230	53,258	6,575	148.46	0.449%
Patio Lawn and Garden	1,686	962	51,835	13,272	1,353	160.44	0.818%
Pet Supplies	19,856	8,510	153,947	157,836	16,294	89.95	0.093%
Sports and Outdoors	35,598	18,357	303,145	296,337	31,339	88.86	0.045%
Tools and Home Improvement	16,638	10,217	210,193	134,476	13,900	111.66	0.079%
Toys and Games	19,412	11,924	189,720	167,597	1,5481	101.90	0.072%
Video Games	24,303	10,672	556,049	231,780	29,585	210.55	0.089%
Raw Amazon Instant Video	426,910	23,962	334,179	583,914	7,173	54.34	0.0057%
Raw Digital Music	478,201	266,393	718,686	835,953	23,212	77.64	0.0007%
Raw Musical Instruments	338,967	83,025	539,873	499,730	8,504	89.88	0.0018%

Total(R) means the total number of records, Test(R) means the number of records used for testing, and WPR is the average number of words per review.

— **5-core Datasets:** These datasets are relatively dense, where the users (or items) with less than five reviews are filtered out. We select 18 categories for our model evaluation. The statistics of these datasets are summarized in the second block of Table 3, from which we can see they cover different data characters; for example, the *Patio Lawn and Garden* and *Musical Instruments* datasets are smaller with higher densities, while the *Home and Kitchen* and *Health and Personal Care* datasets are much larger and sparser. In general, the total number of user-item interactions in these datasets ranges from 10,261 to 551,682, while the densities of them fall between 0.029% and 0.818%.
— **Raw Datasets:** These are sparser datasets without filtering. Three categories including *Musical Instruments, Amazon Instant Video*, and *Digital Music* are utilized in our experiments, and their statistics can be seen in the third block of Table 3.

Evaluation Protocols. In our experiments, following [3, 51], the performance is evaluated by the mean square error (MSE) as follows:

$$MSE = \frac{1}{|S|} \sum_{(i,j) \in S} (r_{ij} - \hat{r}_{ij})^2, \tag{25}$$

where S is the set of all user-item interaction pairs in testing set and r_{ij} and $\hat{r}_{ij}$ are the real and predicted ratings. A lower MSE means a better performance.

Baselines. In our experiments, we use the following representative and state-of-the-art baselines:

— **MF:** This is a basic matrix factorization method [20]. The user-item rating matrix is estimated by the multiplication of two low-rank matrices.
— **PMF:** This method [34] generalizes MF into a probability form, where the user/item latent factors are assumed to follow Gaussian distribution. We learn the model parameters by stochastic gradient decent (SGD).
— **CTR:** This is a well-known probabilistic recommender model [43] leveraging textual information.
— **DeepCoNN:** This is the first deep model designed for review-based recommendation [51], where user reviews are modeled by the CNN.
— **TransNet, TransNet-Ext:** They are the state-of-the-art methods [3] for review-based recommendation. The difference between them is that TransNet only inputs all the reviews that belong to the current user/item, while TransNet-Ext further includes user/item latent factors in the source network. We implement them using the code provided by the authors.[2]

Implementation Details. For each user, we randomly split her interactions into the training (70%), validation (10%), and testing (20%) sets. The items which only exist in the validation or testing set but not in the training set are removedt. Following [3], the reviews in the validation and testing sets are masked as they are not available in real scenarios. A model's final result is reported as the performance on the testing set when MSE on the validation set reaches the lowest point. In our model, grid search technology is utilized to determine different parameters. In particular, the learning rate is determined in the range of $\{0.2, 0.1, 0.02, 0.01, 0.002, 0.001\}$, and the batch size is tuned in $\{10, 20, 50, 100, 150, 200\}$. The regularizer coefficient λ and hyper-parameters λ_{1-4} are selected from the range of $\{10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 1\}$ and $\{0.01, 0.1, 1, 10\}$, respectively. The dimension of the user/item representation is tuned in $\{10, 20, 30, \ldots, 90, 100\}$. When preparing the user reviews, we pre-process them leveraging the Stanford Core NLP tool.[3] The word embeddings are pre-trained based on the Skip-gram model,[4] and the embedding size is set as 64.

4.2 RQ 1: Performance Comparison

The overall performance is shown in Table 4, from which we have the following observations.

— By leveraging review information, CTR, DeepCoNN, TransNet, and TransNet-Ext performed better than MF and PMF, which highlights the importance of textual features for rating prediction in the field of recommender systems. Furthermore, TransNet or TransNet-Ext performed better than DeepCoNN on most datasets, which was consistent with the results in [3]. On considering that the main difference between TransNet (TransNet-Ext) and DeepCoNN is whether or not to separate the review and rating modeling by different networks, this observation actually verifies the effectiveness of the teacher-student learning paradigm.
— Encouragingly, our model (the better version between CNN and LSTM review processors) can consistently obtain the best performance on both dense and sparse datasets, and on average, it can improve the performance by about 3.68% as compared with the best baselines. This is within expectation because DeepCoNN, TransNet, and TransNet-Ext indiscriminately utilize all the review information in the modeling process, which may introduce too much noise as described in Section 3.2.2. In our model, however, all the user reviews have to be firstly processed through the review prediction network, and according to our

[2] https://github.com/rosecatherinek/TransNets.
[3] https://stanfordnlp.github.io/CoreNLP/.
[4] http://mccormickml.com/2016/04/19/word2vec-tutorial-the-skip-gram-model/.

Table 4. Comparison of MSE Between the Baselines and Our Models

Dataset	MF	PMF	CTR	DCN	Tran	TranE	SDNet		imp
							CNN	LSTM	
Amazon Instant Video	1.401	1.268	1.231	1.195	1.189	1.033*	**0.988**	0.994	4.3%
Automotive	1.303	1.026	1.020	0.986	0.977	0.960*	**0.929**	0.937	3.2%
Baby	1.371	1.231	1.229	1.220	1.203*	1.211	1.136	**1.132**	5.9%
Beauty	1.542	1.312	1.261	1.259	1.253	1.245*	**1.214**	1.219	2.4%
Cell Phones and Accessories	1.751	1.421	1.342	1.332	1.256*	1.283	**1.226**	1.230	2.4%
Clothing Shoes and Jewelry	1.821	1.312	1.291	1.250*	1.255	1.297	**1.192**	1.199	4.6%
Digital Music	1.511	1.290	1.211	1.170	1.113	1.036*	0.981	**0.974**	6.0%
Grocery and Gourmet Food	1.682	1.473	1.246	1.276	1.118	1.072*	**1.037**	1.044	3.2%
Health and Personal Care	1.571	1.259	1.244	1.207*	1.209	1.243	**1.180**	1.187	2.2%
Home and Kitchen	1.862	1.321	1.280	1.215	1.201	1.187*	**1.141**	1.144	3.9%
Musical Instruments	1.323	1.205	0.715	0.709	0.711	0.707*	0.681	**0.678**	4.1%
Office Products	0.991	0.901	0.891	0.887	0.876	0.785*	**0.756**	0.762	3.7%
Patio Lawn and Garden	1.491	1.256	1.183	1.147	1.161	1.004*	0.966	**0.964**	4.0%
Pet Supplies	1.422	1.305	1.311	1.347	1.304	1.271*	1.237	**1.224**	3.7%
Sports and Outdoors	0.996	0.971	0.961	0.961	0.943*	0.952	**0.919**	0.926	2.5%
Tools and Home Improvement	1.431	1.223	1.112	1.102	1.056	0.988*	0.960	**0.952**	3.6%
Toys and Games	1.256	1.149	0.981	0.910	1.024	0.901*	0.878	**0.876**	2.7%
Video Games	1.543	1.512	1.358	1.268*	1.286	1.273	1.199	**1.190**	6.2%
Raw Amazon Instant Video	1.421	1.341	1.201	1.162	1.143	1.139*	**1.091**	1.110	4.2%
Raw Digital Music	1.001	0.931	0.881	0.821	0.811	0.801*	**0.785**	0.792	2.0%
Raw Musical Instruments	1.321	1.223	1.104	1.079	1.074*	1.083	**1.046**	1.053	2.6%

The starred values are the best baseline results, and the bolded numbers indicate the best performance of each dataset. Improvements (labeled by "Imp") of SDNet (the better version between CNN and LSTM review processors) from the best baseline on all the datasets are significant at $p = 0.01$ with paired t-test. DCN, Tran, and TranE are short for DeepCoNN, TransNet, and TransNet-Ext, respectively.

designed selective knowledge distillation mechanism, it can produce more useful and clean information to help the user/item latent factors to fit the ground-truth, which finally leads to better performance.

—Another observation was that the improvement margin between our model and the best baseline was even larger on the datasets such as Digital Music and Video Games, where the user reviews are relatively longer. The reason can be that long reviews may contain more useless or even noisy text, and with the help of the selective distillation mechanism, our model can filter this noisy information more effectively, which leads to better performance.

—Interestingly, we find that the difference between the CNN and LSTM review processors was not obvious on most datasets. The reason may be that although LSTM is good at capturing word sequential information—which is important to many NLP tasks such as language generation [24] and machine translation [29]—our review prediction network is more likely to focus on the sentiment and opinion-rich part of the text, and the result does not heavily rely on the specific position of the sentiment words.

4.3 RQ 2: Parameter Analysis

In this section, we analyze the influence of two important parameters, that is, the user/item complete latent factor dimension $2D$ and the regularizer coefficient λ. We first fix λ as 1, and

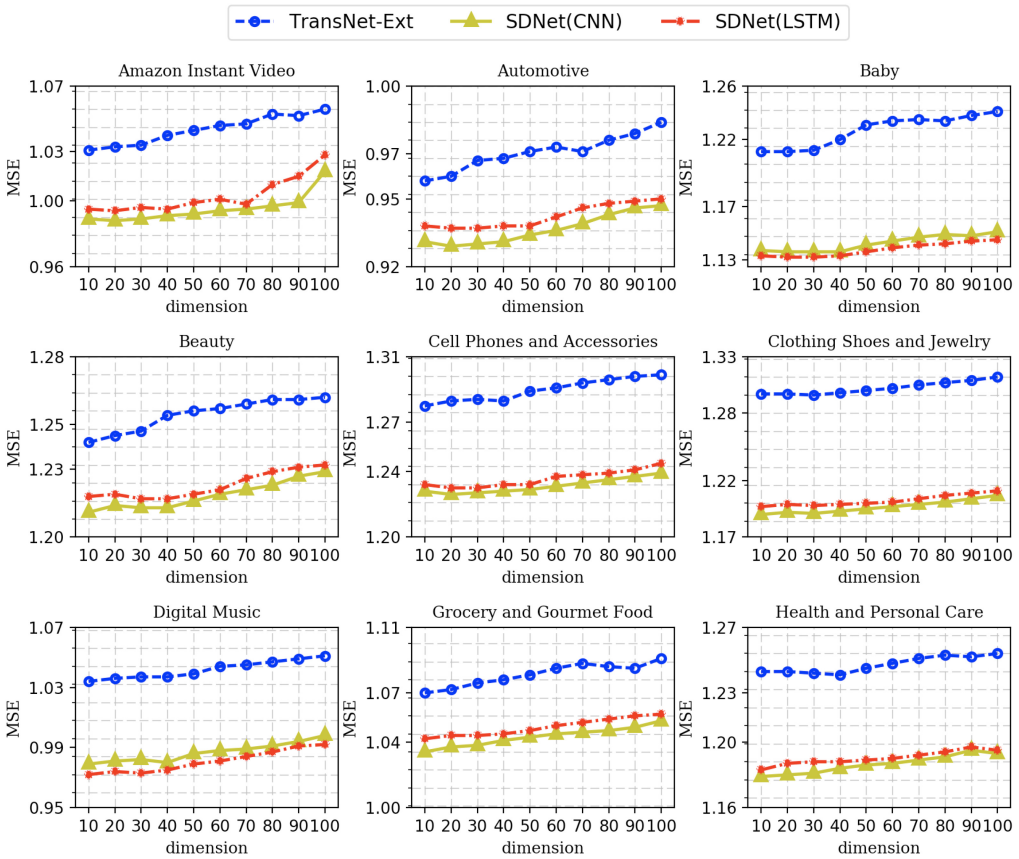


Fig. 7. Influence of the latent factor dimension.

observe the model performance by tuning $2D$ in the range of $\{10, 20, 30, \ldots, 90, 100\}$. Then, to explore the best performance, we set $2D$ as its optimal value, and evaluate our models when λ changes in $\{10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 1\}$. In this experiment, the other parameters follow the settings in Section 4.1, and we present the results on nine datasets including *Amazon Instant Video, Automotive, Baby, Beauty, Cell Phones and Accessories, Clothing Shoes and Jewelry, Digital Music, Grocery and Gourmet Food*, and *Health and Personal Care*.

4.3.1 Influence of the Latent Factor Dimension. In this experiment, we present the results of TransNet-Ext for reference, because it can usually achieve the best performance in the baselines. From the results shown in Figure 7, we can see that

- —our models can consistently outperform TransNet-Ext on all the datasets when selecting different latent factor dimensions. This further verifies the effectiveness and rationality of our idea to make use of review information in a selective manner;
- —for all thesenine datasets, SDNet was relatively robust to different dimensions, and a small number of latent factors can usually lead to good performance. We also notice that introducing more latent factors did not significantly improve (sometimes even lower) the performance. This implies that although increasing the parameter complexity improves

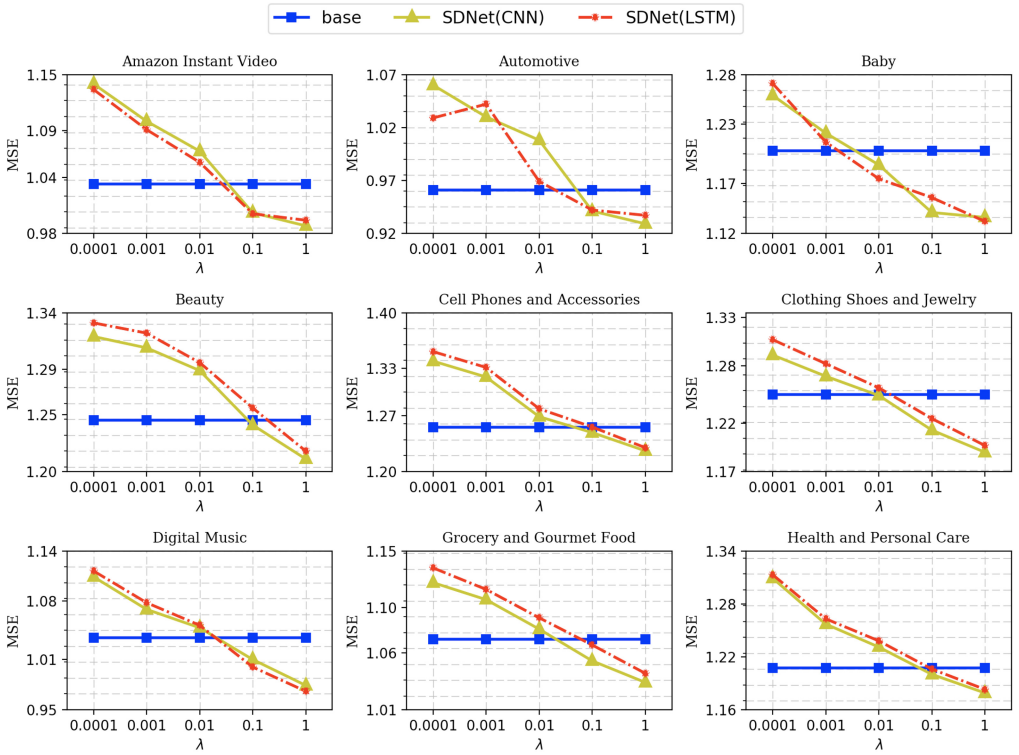


Fig. 8. Influence of the regularizer coefficient.

the expressive power of the models, it may also lead to an over-fitting problem which decreases the final performance.

4.3.2 Influence of the Regularizer Coefficient. Regularization is an effective method to improve the model generalization capability. In this section, we study the influence of the regularizer coefficient λ on the model performance. We plot the best baseline for reference in the final results. From Figure 8, we can see that the performance varies a lot with different λ's. In particular, when λ is small, our models did not perform very well, and even failed to overcome the baseline. However, with the increase of λ, the performance continued to rise, and finally outperformed the baseline significantly. This observation suggests that λ is very important to our model, and in practice a larger value may be a favorable choice.

4.4 RQ 3: Efficiency Comparison

In real-world scenarios, the runtime efficiency of a recommender system is very important. In this section, we evaluate the time consumption of different models for predicting the ratings in the test set. We conduct this experiment on a sever with 1 TITAN X GPU, 256G memory and 40 cores, and the model parameters follow the settings in Section 4.1. We present the results on eight datasets, and results on the other datasets are similar. From the results shown in Table 5, we can see that

—the runtime efficiency of SDNet was comparable with the simple model PMF. Considering SDNet can leverage complex teacher models for effective knowledge extraction to enhance

Table 5. Efficiency Comparison

Dataset	PMF	DeepCoNN	TransNet	TransNet-Ext	SDNet
Amazon Instant Video	1.825s	17.904s	18.480s	17.762s	1.939s
Automotive	0.911s	3.609s	3.671s	3.521s	0.953s
Cell Phones and Accessories	3.004s	78.539s	79.237s	84.530s	3.126s
Digital Music	5.212s	89.712s	89.506s	91.119s	5.694s
Grocery and Gourmet Food	8.945s	93.462s	94.450s	93.218s	9.167s
Musical Instruments	0.483s	1.935s	1.943s	1.955s	0.544s
Office Products	1.452s	30.176s	35.229s	34.975s	1.521s
Patio Lawn and Garden	0.302s	5.427s	5.671s	5.891s	0.360s

We do not distinguish CNN and LSTM versions of SDNet because they share the same model at test time.

the model performance, this result manifests that our model can simultaneously enjoy the advantages of a complex model's effectiveness and a simple model's runtime efficiency;

— the difference among DeepCoNN, TransNet, and TranNet-Ext was not obvious, while SDNet was significantly faster than the best of them on all the datasets. Specifically, SDNet obtained on average 12.8 times of speed up when comparing with the best of DeepCoNN, TransNet, and TranNet-Ext. More encouragingly, on the relative large datasets: *Cell Phones and Accessories*, *Digital Music*, *Grocery and Gourmet Food*, and *Office Products*, SANet was even on average 17.7 faster than the best of DeepCoNN, TransNet, and TranNet-Ext.

This observation verifies the efficiency of our designed distillation structure, and is actually as expected, because our student model leveraged in the testing phase does not need text processing components, which is quite succinct and efficient as compared with the baseline models. It is quite remarkable that SDNet can save more time as the number of test records increases, which verifies its great advantage in a real-world recommender system, where there are usually an extremely large number of users and items.

4.5 RQ 4: Model Ablation: Effect of Different Components in SDNet

For better understanding our framework, in this section, we study the contributions of its different components. In particular, we combine different parts in Equation (24) to form six variations of our model as follows:

— $SDNet(-L_{Review})$: In this situation, we only use L_{UI} as the objective function, and our method reduces to a traditional collaborative filtering model, where review information is not included in the modeling process.

— $SDNet(-L_2)$: In this version, we use $L_{UI} + L_{Review} + L_{GAN} + L_{Orth}$ as the objective function, and our user-item prediction and review prediction network are connected solely by a L_{GAN} loss.

— $SDNet(-L_{GAN})$: This method optimizes $L_{UI} + L_{Review} + L_2 + L_{Orth}$. It not only leverages L_2 loss to connect different networks, but also adds orthogonality constraint to enhance the distillation effectiveness.

— $SDNet(-L_{Orth})$: This model optimizes $L_{UI} + L_{Review} + L_2 + L_{GAN}$, and we combine L_2 and L_{GAN} loss together for knowledge distillation.

— $SDNet(-L_{GAN}, L_{Orth})$: This model only optimizes $L_{UI} + L_{Review} + L_2$, which does not use adversarial and orthogonality constraint strategies.

— $SDNet$: This is the overall model, which optimizes $L_{UI} + L_{Review} + L_2 + L_{GAN} + L_{Orth}$ together.

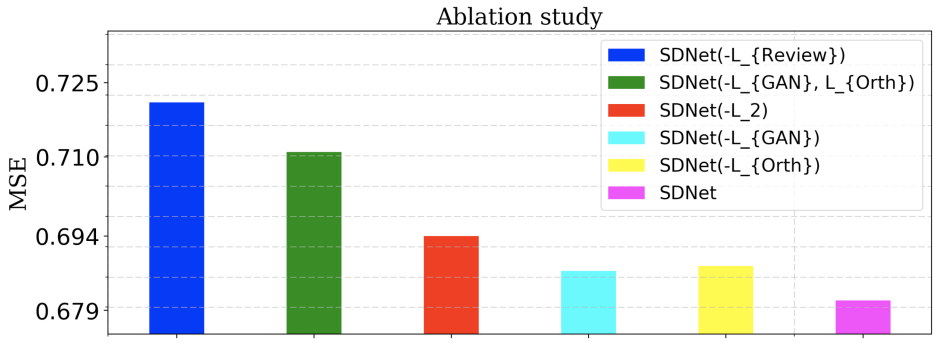


Fig. 9. The effect of different components in our model.

We conduct this experiment based on the dataset of *Musical Instruments*, and CNN [21] is utilized to implement our review processors due to its higher training efficiency. From the results shown in Figure 9, we can see the following:

- —As expected, only leveraging user-item prediction network (i.e., L_{UI}) performed worst among all the methods, which demonstrated that review information can indeed provide us with valuable signals to enhance the performance of rating prediction.
- —$SDNet(-L_{GAN}, L_{Orth})$ performed worse than both $SDNet(-L_{GAN})$ and $SDNet(-L_{Orth})$, while the difference between $SDNet(-L_{GAN})$ and $SDNet(-L_{Orth})$ is little. These observations verify the effectiveness of our two designed strategies. adversarial adaption and orthogonality constraint, and also manifest that L_{GAN} and L_{Orth} contributed almost the same to the final performance.
- —$SDNet(-L_{GAN})$ performed slightly better than $SDNet(-L_2)$, and by combining L_{GAN} and L_2 together, $SDNet$ obtained better performance than both of them. These observations manifest that in our problem, L_2 is a little more effective than L_{GAN}, but they may play different positive roles, and their combination can take both of their strengths to further promote the final performance. This is actually consistent with many previous studies [18, 38] in the field of computer vision, which verifies that the strategy of combing a regular (e.g., L_2) with an adversarial loss can be a generally effective technique for different feature adaption.
- —$SDNet$ was better than $SDNet(-L_{Orth})$. This manifests that by pushing user-item related information from the private embeddings into the shared ones, L_{Orth} played a positive role for enhancing the final result, which verifies the usefulness of our orthogonality constraint strategy.

4.6 RQ 5: Model Pre-Training

Previous studies [11, 14] have manifested that initializing the model parameters by some pre-trained values can positively affect the performance as well as the convergence rate. In this section, we evaluate the usage of pre-training in our model. Specifically, to study the effects of different components, we do not pre-train our model as a whole; instead, we evaluate the effect of pre-training the user-item prediction network (called ui pre-training, for short) and the review prediction network (called review pre-training, for short), separately. The parameters follow the settings in Section 4.1, and we report the performance of our model (in terms of MSE) with different training epochs. On considering the training efficiency, we leverage CNN to implement the

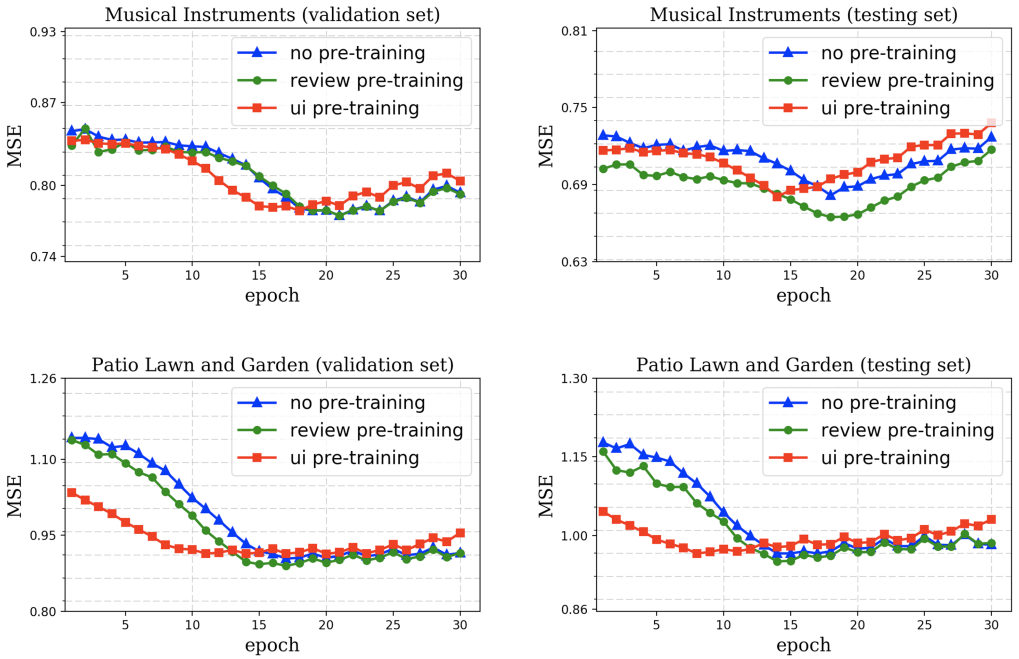


Fig. 10. The effects of different model pre-training strategies. We report the MSE value (vertical axis) as the number of training epochs (horizontal axis) changes from 1 to 30 (more training epochs lead to worse performance). The squared and circled lines represent results of applying pre-training on review prediction network (named as review pre-training) and user-item prediction network (named as ui pre-training), respectively, and result of training from scratch (named as no pre-training) is denoted by the triangled line.

review processors. For clarity, we only present the results on the datasets of *Musical Instruments* and *Patio Lawn and Garden* (see Figure 10); we can see

—Pre-training the user-item prediction network can provide us with a faster convergence speed, but it did not obviously enhance the model performance. In particular, on the dataset of *Musical Instruments*, our model with ui pre-training converged after about 15 epochs on both validation and testing sets. However, without pre-training, we have to spend about 18 epochs until the performances tend to be stable (or worse). In addition, the best performance of our model with ui pre-training on the testing set ($MSE = 0.680$) did not improve much as compared with the result from totally training from scratch ($MSE = 0.681$). Similar results can also be found on the *Patio Lawn and Garden* dataset.
These observations imply that our model is relatively robust to the parameters in user-item prediction network, and different initializations do not influence the final results very much.
—Pre-training review prediction network consistently led to better performances on both datasets. Specifically, the lowest MSE on the testing set of *Musical Instruments* and *Patio Lawn and Garden* can achieve about 0.664 and 0.951, which improve the results of no pre-training strategy by about 2.5% and 1.6%, respectively.
These findings indicate that the parameters in the review prediction network are of key importance to the final performance, and properly pre-training them can help to find better local optima.

Table 6. Visualization of the Distilled Knowledge

Rating	Review
5.0	I normally get elixir strings but these sounded too good to *pass* up. They are indeed some high tech *strings* with *features* including *break resistance* and improved tuning stability. They feel and sound good so grab a pack and get jamming you'll definitely want to check these out.
5.0	I have a pretty crappy ear so i rely on it to *tune* correctly responds fast display is incredible. I wish my car dash looked like this thing fits easily on the headstock of the guitar in front or behind tunes by *sensing* the *vibration* of the instrument not *using* a *microphone*.
2.0	To star after *returning* it back to brookmays. The seller I am really disappointed by the seller brookmays' ignorant *responses* to my return of the defective guitar pickup. After two weeks of the returned item was delivered i didn't *hear anything* from the seller *regarding* the return.
4.0	This amp cable is perfect. It fits my need for practicing. It's just the right length for sitting on the *couch strumming daily*. The cable also *looks* nice and the fender look I *would* highly recommend this and give it a no-brainer 5 star rating. I have nothing negative to say about this.
3.0	Looking for options around the old dropped pick syndrome. I was skeptical as the pick at first glance seems slicker than *most*, however, once *between thumb* and *index finger* these hang on.

We present five pieces of user review in the musical instruments dataset. The words most related to the shared and private embeddings are labeled by underlined and italic fonts, respectively.

4.7 RQ 6: Visualization of the Distilled Knowledge

For a more thorough understanding of our proposed model, we would like to know what information has been distilled from the review prediction network to the user-item prediction network. With this purpose, in this section, we present some examples from the dataset of *Musical Instruments* to illustrate the distilled information learned by our model. In particular, we use the attentional LSTM described in Section 3.1.2 to implement our review processor, and according to the attention weight α, we highlight top-5 most related words in the shared ($\boldsymbol{s}_{\boldsymbol{w}_{ij}}$) and private ($\boldsymbol{t}_{\boldsymbol{w}_{ij}}$) embeddings. Again, the model parameters follow the settings in Section 4.1. The results are presented in Table 6, from which we can see the following:

—To fit the ground-truth in the user-item prediction network, the shared embeddings can successfully capture opinionated/sentimental signals that are more informative in terms of user/item profiling, and the highlighted textual information can effectively reflect the sentiment of the users. For example, in the first line, the words *good, high* learned by the shared embeddings are consistent with the user's high rating to the item. Whereas, in the third line, when the user's rating is low, the distilled knowledge is also negative, such as the highlighted words *disappointed, defective*, and *ignorant*.

—With the help of orthogonality constraint, the words with highest attention weights in different review processors were very different. Take the fourth line, for example; the shared embedding focused more on words such as *recommend, highly, nice*, and so forth, which directly reflect the user preference. On the contrary, the texts highlighted by the private embedding, such as *would, looks, daily*, and so forth, tend to be some general-purpose words, which are irrelevant with the user preference or item properties, and similar patterns can be found in the other reviews.

5 RELATED WORK

In this section, we review the research areas related to our work.

5.1 Recommendation with Textual Information

In recent years, textual information has been widely used in recommender systems to enhance performance. In early stages, the designed models mainly focused on the combination of Latent Dirichlet Allocation (LDA) [1] and traditional MF. Specifically, Collaborative Topic modeling for Recommendation (CTR) [43] designed a probabilistic graphical model to link the topic distribution with the item latent factors. The Hidden Factor and Topics (HFT) model [31] introduced a soft-max function to adapt review topics with user (or item) latent factors. By jointly maximizing the likelihood of textual reviews as well as minimizing the MSE between the predicted and actual ratings, it can benefit from both review and rating information. Ratings meet Reviews (RMR) [26] leveraged mixture Gaussian distribution to model the rating information, which obtained enhanced performance especially for the cold-start settings. The Rating-Boosted Latent Topics model (RBLT) [41] introduced user sentiment orientations for the features discussed in reviews, and proposed to learn item recommendability as well as user preference distributions in a shared topic space for more accurate modeling.

Overall, these methods made use of collaborative information discovered from textual features solely based on the lexical similarity. However, in real-world applications, sentences with low percentage of token overlapping may still exhibit high semantic similarity, and taking the semantic of words into consideration is important for review-based recommendation.

Recently, many deep architectures have been proposed to infuse textual information into recommender systems. Specifically, the Collaborative Deep Learning (CDL) model [44] integrated the stacked denoising autoencoder (SDAE) with MF into a unified probabilistic graphical framework, and used the learned representations from SDAE to enhance the item latent factors. However, when extracting textual features, the word contexts were ignored in CDL. To capture such information, Convolutional Matrix Factorization (ConvMF) [19] infused CNN into MF, and achieved improved performance. These studies mainly focused on mining textual information to enhance the item representations, but failed to consider user preferences reflected in the text corpus.

Deep Cooperative Neural Networks (DeepCoNN) [51] proposed to jointly model user behaviors and item properties from the review text based on coupled CNNs, which obtained significant improvement against CTR, HFT, and CDL in terms of rating prediction. However, as mentioned in [3], the predicted ratings in DeepCoNN exhibit high correlation with their corresponding reviews; although we can assume such reviews to be available at training time, they are not available at test time. To solve this problem, Transformational Neural Networks (TransNet) [3] extended DeepCoNN by introducing an additional layer to approximate the review of a target user-item pair, which obtained improved performance on the Amazon dataset.

Compared with the above models, our method is different in two aspects: on one hand, we reformulate review-based recommendation into a generalized distillation framework, where we drop the review input component as in DeepCoNN and TransNet, which greatly increases the runtime efficiency. On the other hand, we link textual information with user-item latent factors in a selective manner for avoiding noise in the user reviews to improve the final performance.

5.2 Generalized Distillation Framework

Generalized distillation framework is a recently proposed learning paradigm, which unifies the techniques of knowledge distillation and privileged information. In this section, we review these research areas, highlighting their differences from our model.

5.2.1 Knowledge Distillation. Deep neural network has been demonstrated to be effective in many domains, such as computer vision [21], neural language processing [40], and information retrieval [14]. However, as there are usually numerous parameters in the deep and wide architectures, the low runtime efficiency and high memory demand have became two major obstacles for such effective models to be practical in real settings. Knowledge distillation framework is a method to tackle this problem. In particular, Hinton et al. [15] proposed to train deep neural networks by a *teacher-student* framework. The knowledge learned by the comprehensive teacher model is infused into the compact student model. To this end, the student is trained to predict the output of the teacher, as well as the true classification labels. Hu et al. [17] proposed to distill (structural) information from logical rules into the weights in neural networks to enhance the performance of sentiment analysis and named entity recognition. Mou et al. [35] focused on distilling word embeddings for NLP tasks. They designed an encoding method to transfer task-specific knowledge from a set of high-dimensional embeddings. Their method can not only reduce model complexity by a large margin, but also retain high accuracy, showing a good compromise between efficiency and performance.

5.2.2 Learning Using Privileged Information. Learning using privileged information (LUPI) [42] as a new learning paradigm is becoming more and more popular. It aims to solve the learning problem in the scenario where some important (privileged) information is only available in the training phase, while not accessible at test time. Recently, many researchers adapt this idea into different application areas, For example, Li el at. [25] leveraged textual features as privileged information to enhance the learning of image classification. Yan et al. [47] applied LUPI to the active learning task for solving the computer vision problem. Fouad et al. [8] utilized the "hints" in privileged information to help learning global metric in the original data space.

Different from previous studies, we adapt the basic idea of generalized distillation framework to personalized recommendation, with textual reviews as the privileged information for teacher modeling, which, to the best of our knowledge, is the first time in the recommendation community. Considering the specific nature of the problem, we selectively distill the teacher's knowledge extracted from user reviews into the student models to decrease potential noise. In addition, we explore to link the teacher and student models in an adversarial manner for more effective knowledge distillation.

5.3 GANs

GAN [9] is a popular learning paradigm. It consists of two parts, the generator (G) and the discriminator (D), where G is trained to produce high-quality fake samples, while D tries to detect whether an instance is generated from G or the real dataset. This idea has been widely applied to many research areas such as computer vision, neural language processing, and information retrieval. In the domain of computer vision, Reed et al. [39] leveraged GAN to transfer visual concepts from characters to pixels. For more effective visual communication, Zhu et al. [53] proposed to learn the natural image manifold directly from data using a GAN network. In the field of neural language processing, Yu et al. [48] used policy gradient reinforcement learning to generate sequences by backpropagating the error from the discriminator. Miyato et al. [33] extend adversarial training to the text domain by applying perturbations to the word embeddings in a recurrent neural network rather than to the original input itself. Very recently, Wang et al. [45] applied the idea of GAN into the field of information retrieval (IR), and demonstrate its effectiveness in three specific tasks including document retrieval, recommendation system, and question answering. He et al. [12] designed adversarial personalized ranking (APR) to enhance the robustness of a recommender model for improving its generalization performance.

Different from these studies, we apply the technique of adversarial training into the field of review-based recommendation, where we regard user-item interactions and review information as two different modalities, and adversarially connect to encourage information transfer.

6 CONCLUSION

In this article, we proposed to formulate the problem of recommendation with external knowledge into a generalized distillation framework. As an implementation of our idea, we take user reviews as the external knowledge, and further developed a SDNet to transfer informative review signals from the teacher model into the student model for effective user/item representation learning. We designed two key strategies for knowledge distillation, adversarial adaption and orthogonality constraint, to guarantee the quality of the knowledge distilled between different networks. Extensive experiments verified that our model can significantly outperform many state-of-the-art methods in terms of both effectiveness and efficiency perspectives.

There are many possible directions to extend this work. We can design more advanced review processors to further improve the performance. Although our focus in this article is user reviews, we can also apply our idea to other external knowledge such as images or even videos. In addition, our selective distillation network (SDNet) is a general framework, which can be applied to other research tasks beyond personalized recommendation, such as computer vision and natural language processing tasks.

REFERENCES

[1] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent Dirichlet allocation. *Journal of Machine Learning Research* 3 (2003), 993–1022.
[2] Konstantinos Bousmalis, George Trigeorgis, Nathan Silberman, Dilip Krishnan, and Dumitru Erhan. 2016. Domain separation networks. In *NIPS*.
[3] Rose Catherine and William Cohen. 2017. TransNets: Learning to transform for recommendation. In *Recsys*.
[4] Jingyuan Chen, Hanwang Zhang, Xiangnan He, Liqiang Nie, Wei Liu, and Tat-Seng Chua. 2017. Attentive collaborative filtering: Multimedia recommendation with item-and component-level attention. In *SIGIR*.
[5] Li Chen, Guanliang Chen, and Feng Wang. 2015. Recommender systems based on user reviews: The state of the art. *User Modeling and User-Adapted Interaction* 25, 2 (2015), 99–154.
[6] Xu Chen, Zheng Qin, Yongfeng Zhang, and Tao Xu. 2016. Learning to rank features for recommendation over multiple categories. In *SIGIR*.
[7] Xu Chen, Yongfeng Zhang, Qingyao Ai, Hongteng Xu, Junchi Yan, and Zheng Qin. 2017. Personalized key frame recommendation. In *SIGIR*.
[8] Shereen Fouad, Peter Tino, Somak Raychaudhury, and Petra Schneider. 2013. Incorporating privileged information through metric learning. *IEEE Transactions on Neural Networks and Learning Systems* 24, 7 (2013), 1086–1098.
[9] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *NIPS*.
[10] Ruining He and Julian McAuley. 2016. Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In *WWW*.
[11] Xiangnan He and Tat-Seng Chua. 2017. Neural factorization machines for sparse predictive analytics. In *SIGIR*.
[12] Xiangnan He, Zhankui He, Xiaoyu Du, and Tat-Seng Chua. 2018. Adversarial personalized ranking for recommendation. In *SIGIR*.
[13] Xiangnan He, Zhenkui He, Jingkuan Song, Zhenguang Liu, Yu-Gang Jiang, and Tat-Seng Chua. 2018. NAIS: Neural attentive item similarity model for recommendation. *IEEE Transactions on Knowledge and Data Engineering* 1, 1–1.
[14] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural collaborative filtering. In *WWW*.
[15] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network. *stat* (2015).
[16] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation* 9, 8 (1997), 1735–1780.
[17] Zhiting Hu, Xuezhe Ma, Zhengzhong Liu, Eduard Hovy, and Eric Xing. 2016. Harnessing deep neural networks with logic rules. In *ACL*.

[18] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A. Efros. 2017. Image-to-image translation with conditional adversarial networks. In *CVPR*.
[19] Donghyun Kim, Chanyoung Park, Jinoh Oh, Sungyoung Lee, and Hwanjo Yu. 2016. Convolutional matrix factorization for document context-aware recommendation. In *Recsys*.
[20] Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix factorization techniques for recommender systems. *Computer* 8 (2009), 30–37.
[21] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. 2012. ImageNet classification with deep convolutional neural networks. In *NIPS*.
[22] Christian Ledig, Lucas Theis, Ferenc Huszár, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, et al. 2017. Photo-realistic single image super-resolution using a generative adversarial network. In *CVPR*.
[23] Chuan Li and Michael Wand. 2016. Precomputed real-time texture synthesis with Markovian generative adversarial networks. In *ECCV*.
[24] Piji Li, Zihao Wang, Zhaochun Ren, Lidong Bing, and Wai Lam. 2017. Neural rating regression with abstractive tips generation for recommendation. In *SIGIR*.
[25] Wen Li, Li Niu, and Dong Xu. 2014. Exploiting privileged information from web data for image categorization. In *ECCV*.
[26] Guang Ling, Michael R. Lyu, and Irwin King. 2014. Ratings meet reviews, a combined approach to recommend. In *Recsys*.
[27] Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. 2017. Adversarial multi-task learning for text classification. In *ACL*.
[28] David Lopez-Paz, Léon Bottou, Bernhard Schölkopf, and Vladimir Vapnik. 2015. Unifying distillation and privileged information. *stat* (2015).
[29] Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *EMNLP*.
[30] Andrew L. Maas, Awni Y. Hannun, and Andrew Y. Ng. 2013. Rectifier nonlinearities improve neural network acoustic models. In *ICML*.
[31] Julian McAuley and Jure Leskovec. 2013. Hidden factors and hidden topics: Understanding rating dimensions with review text. In *Recsys*.
[32] Julian McAuley, Christopher Targett, Qinfeng Shi, and Anton van den Hengel. 2015. Image-based recommendations on styles and substitutes. In *SIGIR*.
[33] Takeru Miyato, Andrew M. Dai, and Ian Goodfellow. 2016. Virtual adversarial training for semi-supervised text classification. *stat* (2016).
[34] Andriy Mnih and Ruslan R. Salakhutdinov. 2008. Probabilistic matrix factorization. In *NIPS*.
[35] Lili Mou, Ran Jia, Yan Xu, Ge Li, Lu Zhang, and Zhi Jin. 2016. Distilling word embeddings: An encoding approach. In *CIKM*.
[36] Vinod Nair and Geoffrey E. Hinton. 2010. Rectified linear units improve restricted Boltzmann machines. In *ICML*.
[37] Bo Pang, Lillian Lee, et al. 2008. Opinion mining and sentiment analysis. *Foundations and Trends® in Information Retrieval* 2, 1–2 (2008), 1–135.
[38] Deepak Pathak, Philipp Krahenbuhl, Jeff Donahue, Trevor Darrell, and Alexei A. Efros. 2016. Context encoders: Feature learning by inpainting. In *CVPR*.
[39] Scott Reed, Zeynep Akata, Xinchen Yan, Lajanugen Logeswaran, Bernt Schiele, and Honglak Lee. 2016. Generative adversarial text to image synthesis. In *ICML*.
[40] Richard Socher, Yoshua Bengio, and Christopher D. Manning. 2012. Deep learning for NLP (without magic). In *Tutorial Abstracts of ACL 2012*.
[41] Yunzhi Tan, Min Zhang, Yiqun Liu, and Shaoping Ma. 2016. Rating-boosted latent topics: Understanding users and items with ratings and reviews. In *IJCAI*.
[42] Vladimir Vapnik and Akshay Vashist. 2009. A new learning paradigm: Learning using privileged information. *Neural Networks* 22, 5–6 (2009), 544–557.
[43] Chong Wang and David M. Blei. 2011. Collaborative topic modeling for recommending scientific articles. In *KDD*.
[44] Hao Wang, Naiyan Wang, and Dit-Yan Yeung. 2015. Collaborative deep learning for recommender systems. In *SIGKDD*.
[45] Jun Wang, Lantao Yu, Weinan Zhang, Yu Gong, Yinghui Xu, Benyou Wang, Peng Zhang, and Dell Zhang. 2017. IRGAN: A minimax game for unifying generative and discriminative information retrieval models. In *SIGIR*.
[46] Jun Xiao, Hao Ye, Xiangnan He, Hanwang Zhang, Fei Wu, and Tat-Seng Chua. 2017. Attentional factorization machines: Learning the weight of feature interactions via attention networks. In *IJCAI*.
[47] Yan Yan, Feiping Nie, Wen Li, Chenqiang Gao, Yi Yang, and Dong Xu. 2016. Image classification by cross-media active learning with privileged information. *IEEE Transactions on Multimedia* 18, 12 (2016), 2494–2502.

[48] Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. 2017. SeqGAN: Sequence generative adversarial nets with policy gradient. In *AAAI*.
[49] Y. Zhang, Q. Ai, X. Chen, and W. Croft. 2017. Joint representation learning for top-N recommendation with heterogeneous information sources. *CIKM* (2017).
[50] Yongfeng Zhang, Guokun Lai, Min Zhang, Yi Zhang, Yiqun Liu, and Shaoping Ma. 2014. Explicit factor models for explainable recommendation based on phrase-level sentiment analysis. In *SIGIR*.
[51] Lei Zheng, Vahid Noroozi, and Philip S. Yu. 2017. Joint deep modeling of users and items using reviews for recommendation. In *WSDM*.
[52] Feida Zhu, Yongfeng Zhang, Neil Yorke-Smith, Guibing Guo, and Xu Chen. 2018. IFUP: Workshop on multi-dimensional information fusion for user modeling and personalization. In *WSDM*.
[53] Jun-Yan Zhu, Philipp Krähenbühl, Eli Shechtman, and Alexei A. Efros. 2016. Generative visual manipulation on the natural image manifold. In *ECCV*.

Received April 2018; revised August 2018; accepted September 2018