applied sciences

MDPI

Article

An Adversarial Deep Hybrid Model for Text-Aware Recommendation with Convolutional Neural Networks

Xiaolin Zheng and Disheng Dong *

School of Computer Science, Zhejiang University, Hangzhou 310007, China; xlzheng@zju.edu.cn
* Correspondence: eson@zju.edu.cn; Tel.: +86-177-0651-3759

Received: 17 November 2019; Accepted: 18 December 2019; Published: 24 December 2019

Abstract: The standard matrix factorization methods for recommender systems suffer from data sparsity and cold-start problems. Thus, in real-world scenarios where items are commonly associated with textual data such as reviews, it becomes necessary to build a hybrid recommendation model that can fully utilize the text features. However, existing methods in this area either cannot extract good features from the texts due to their order–insensitive document modeling approaches or fail to learn the hybrid model in an effective way due to their complexity of inferring the latent vectors. To this end, we propose a deep hybrid recommendation model which seamlessly integrates matrix factorization with a Convolutional Neural Network (CNN), a powerful text feature extraction tool with the capability of detecting the information of word orders. Unlike previous works which use content features as prior knowledge to regularize the latent vectors, we combine CNN into MF in an additive manner to allow training CNN with direct learning signals. Furthermore, we propose an adversarial training framework to learn the hybrid recommendation model, where a generator model is built to learn the distribution over the pairwise ranking pairs while training a discriminator to distinguish generated (fake) and real item pairs. We conduct extensive experiments on three real-world datasets to demonstrate the effectiveness of our proposed model against state-of-the-art methods in various recommendation settings.

Keywords: hybrid recommendation model; matrix factorization; generative adversarial learning; convolution neural network

1. Introduction

In recent years, many commercial websites have employed recommender systems to provide better services for their customers. Recommender systems aim to match items to users through exploring the hidden connections between users and items. One of the most successful techniques for recommender systems is collaborative filtering based upon Matrix Factorization (MF) [1–7]. A standard MF-based approach aims to discover latent similarities among users and items from user–item interaction data, such as clicks and ratings, by learning user and item vectors whose dot products predict the users' preference scores over items.

Although these MF-based methods have been shown to achieve promising recommendation results, they typically suffer from data sparsity and cold-start problems. In real-world scenarios, a user commonly interacts with only a tiny number of items in the system and an item is also consumed by only a minority of users. As a result, standard MF-based methods may fail to capture effective collaborative information based on the extremely sparse user–item interactions In addition, it is practically infeasible for matrix factorization to obtain meaningful latent factors of new users or items which are associated with no interaction data (a.k.a., cold-start problem). To address these issues, plenty

of work have attempted to build hybrid recommendation models by exploiting auxiliary content data information to aid MF-based collaborative filtering [8–16]. A common approach to building a hybrid model is to bridge together the content feature modeling and matrix factorization. These methods range from non-DL (Deep Learning) to DL-based models, with their major difference lying in their ways of learning auxiliary features from the content data. Deep hybrid models [9,10,13–16], by using DL techniques to extract deep content features, have recently shown superior performance than traditional non-DL based models, which adopts shallow representation learning techniques, such as LDA [8,11,17].

In principle, these hybrid models can be built from a probabilistic perspective, where content features are incorporated as priors into MF to regularize the latent user and item vectors [8,9,16,18]. However, this probabilistic treatment involves manually tuning additional regularization parameters and limits the use of other non-MF based collaborative filtering methods. Besides, since the exact inference for latent variables is usually intractable, the estimation of model parameters has to rely on approximation methods, such as maximum a posteriori inference (i.e., maximum likelihood estimation with regularization terms). Moreover, most existing deep hybrid models detect the content features through auto-encoders [9,13,15]. However, auto-encoders assume the bag-of-words model by ignoring contextual text information such as surrounding words and word orders. Thus, they are unable to fully capture document features in text-aware recommendation scenarios where items are commonly associated with text information such as reviews .

In order to address the aforementioned problems, this paper *first* develops an adversarial training framework to learn a deep hybrid recommendation model. Specifically, we specify the recommender model as a generator to generate item ranking pairs while introducing an adversarial discriminator that evolves to discriminate between the real data and the generated data. Compared with maximum likelihood estimation, this indirect approach to learning the probability distribution offers more flexibility and has shown promising results in learning latent variable models [19–22]. Maximum likelihood estimation is theoretically equivalent to maximizing the Kullback–Leibler divergence between the data distribution and modeled distribution, whereas the adversarial paradigm corresponds to maximizing the Jensen–Shannon divergence, which may thus boost the capability of learning the distributional features over the collaborative patterns [19,21]. *Second*, in our training framework, the specification of generator is a flexible design choice and we apply a Convolutional Neural Network (CNN) [23–26] to extract deep features in the document data. CNN has proven to be a powerful technique to capture local features of images or documents through modeling components such as local receptive fields, shared weights and subsampling [27–30]. We seamlessly integrate the CNN content feature extraction module into our hybrid model to enhance both cold start and wart start recommendation.

To summarize, our main contributions are:

- We propose an adversarial training paradigm to learn a hybrid recommendation model, which, compared to maximum a posteriori estimation, enjoys more powerful capability of the learning the distribution over user behavior data. Specifically, we train a pair-wise ranking model (generator) to capture the distribution over positive-negative item pairs, while in the mean time learning a discriminator to guide the training of the generator. The parameters of the two component models are learned by letting them play a mini-max game.
- The generator is specified as a deep hybrid model, where a Convolutional Neural network (CNN) is seamlessly incorporated into matrix factorization to extract high-level features from the content data. Unlike previous methods, which use content features to regularize latent vectors, we combine CNN into MF in an additive manner, allowing CNN to be trained via direct learning signals without tuning additional regularization parameters.
- We conduct extensive experiments on three real-world datasets and show that our model outperforms the state-of-the-art methods in both cold-start and warm-start recommendation

settings. Also, we demonstrate the effectiveness of the proposed adversarial learning paradigm and CNN in document feature extraction in our proposed hybrid model.

2. Related Work

2.1. Hybrid Recommendation Model

To deal with data sparsity and cold-start problems, hybrid recommendation models resort to exploiting auxiliary content data (a.k.a. side information) descriptive of users or items, such as demographics of a user and document descriptions of a movie, to aid collaborative filtering [8,12,17,27,28,31,32]. Typical methods include Collective Matrix Factorization (CMF) [12] which simultaneously factorizes several matrices to inject the content information into latent user and item vectors, and Collaborative Topic Regression (CTR) [8,17] which combines matrix factorization with the topic models that discover latent topics of the content data. However, these models use shallow feature learning methods and are, thus, unable to extract deep knowledge in the content data.

As a powerful feature extraction technique, Deep Learning (DL) has recently gained wide popularity for building hybrid recommendation models [9,10,14,23–26,30,33]. The key idea of these deep hybrid models is to combine MF based collaborative filtering methods with deep learning techniques, where deep content features are incorporated as prior knowledge into matrix factorization. The superiority of these models over previous non-DL methods [8,12,17,31] mainly lies in the introduction of DL for extracting high-level features from the content data.

From a probabilistic perspective, both DL and non-DL hypbrid recommendation models essentially share the same goal with the standard plain MF-based methods [2,8,9,34,35] and focus on approximating the real distribution over user–item interactions. However, the complex structure of these hybrid (especially DL-based) models makes it hard to exactly infer the latent user and item vectors. Therefore, they typically resort to Maximum A Posteriori estimation (MAP) to learn model parameters and infer the latent variables.

2.2. Generative Adversarial Network (GAN)

Generative Adversarial Network (GAN) has recently proven to be effective at learning data distributions in various applications such as image analysis and natural language processing [19,22,36,37]. Instead of optimizing the likelihood of data (i.e., minimizing the Kullback-Leibler divergence between the data distribution and model distribution), GAN aims to find a data distribution that is capable of generating samples which are indistinguishable from real data samples by a discriminator. The superiority of such likelihood-free learning mechanism is that it does not require inference of latent variables and thus can be more applicable in learning latent variable models. Another advantage of GAN training is that its objective is to minimize the Jensen-Shannon divergence rather than Kullback-Leibler divergence between the data distribution and model distribution [20–22,38].

Despite the tremendous success of GAN in learning distributions with latent variables, little attention has been paid to the area of recommender systems. Recently, IRGAN [39] builds a recommender model from an information retrieval perspective and brings the idea of GAN into item recommendation, where a conditional softmax generator is used to generate relevant items for a given user (query), and a discriminator is used to distinguish the generated samples from the positive observations [39]. This implementation is reasonable in IR where the retrieval module has been given a candidate set of items. However, it requires ranking the whole set of items and, as a result, cannot scale well to a large number of items in item recommendation. The scalability problem becomes even more serious when modeling content data associated with items and thus the effectiveness of learning a hybrid model in an adversarial framework remains unclear in the literature.

3. Adversarial Deep Collaborative Filtering

3.1. General Framework

We deal with binary implicit feedback such as clicks and purchases, where only positive interactions are observed. In particular, we aim to approximate the underlying distribution over user behavioral data by a parameterized model p_θ. Typically, the data distribution can be expressed in three forms: point-wise [2,3], pair-wise [40–42] and list-wise [43,44]. While the point-wise distribution is easy to implement, it ignores the ranking nature of recommendation. On the other hand, the list-wise approaches fail to model inter-list loss and are inefficient on large scale datasets. We therefore focus on learning the pair-wise distribution, which has previously shown to be effective in both non-DL and DL based collaborative filtering models [40,45,46].

Following [40,41], we construct a pair-wise training set as follows:

$$S = \{\mathbf{s} = (u, j, j') | u \in U \wedge j \in I_u \wedge j' \in I / I_u\}, \tag{1}$$

where U and I represent the whole user and item set, respectively, and I_u represents the set of items that user u has interacted with. A sample $\mathbf{s} = (u, j, j')$ is used to indicate that user u prefers item j over item j' [40,41]. Accordingly, we use $p_\theta(\mathbf{s} = (u, j, j'))$ to model the probability that user u likes item j better than item j'. Note that p_θ is not a single distribution but a series of correlated distributions over the training data S, subject to the totality order constraint that $p_\theta(\mathbf{s} = (u, j, j')) = 1 - p_\theta(\mathbf{s} = (u, j', j))$ when $j \neq j'$ [40].

Given the training set S, the learning task is to estimate the parameters of the generator p_θ. Instead of maximizing the likelihood of observing S, we follow the idea of generative adversarial network (GAN) [19] to learn the parameters θ by simultaneously training two models: a generative model p_θ that generates data samples, and a discriminative model D_w parameterized by w that estimates the probability that a sample comes from the training data rather than the generator. The training procedure is to let the two components play a mini-max game: p_θ varies its parameters to generate data that maximize the probability of D_w making a mistake, whereas D_w is trained to draw a clear distinction between the ground-truth samples and the generated ones. When the two components reach a balance, we obtain the desired model p_θ that is able to produce samples indistinguishable from real data by D_w.

Formally, we perform the following min-max optimization over the generator parameters θ and discriminator parameters w:

$$\min_\theta \max_w \{\mathrm{E}_{\mathbf{s}\sim p_r}[\log D_w(\mathbf{s})] + \mathrm{E}_{\mathbf{s}\sim p_\theta}[\log(1 - D_w(\mathbf{s}))]\}, \tag{2}$$

where p_r denotes underlying real data distribution. We discuss the specific parameterization of p_θ and D_w in the following subsections. We dub the proposed method as Adversarial Deep Hybrid Recommendation (ADHR) model and illustrate its learning framework in Figure 1.

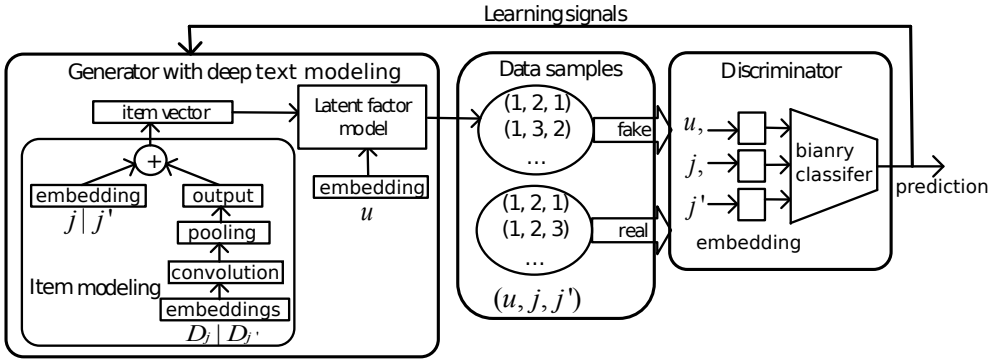


Figure 1. Illustration of Adversarial Deep Hybrid Recommendation (ADHR) model. ADHR consists of two components: a generator and a discriminator. The generator is a deep hybrid model to capture the pairwise ranking distribution where a convolutional neural network is used to extract high-level item content features. The discriminator is a binary classifier based upon latent factor model, which takes the user and item embeddings in a sample pair as input and predicts the probability of the sample being real. While the generator provides fake samples for training the discriminator, the discriminator provides learning signals that drive the optimization of the generator.

3.2. Generator with Deep Text Modeling

The generator hinges on predicting the user–item score $\hat{r}_{uj}$ that indicates the preference of a user u towards an item j. We adopt a commonly used latent factor model in this paper [1,3,4], leaving more complex models based on neural networks [47,48] to future work. Specifically, we define $\hat{r}_{uj}$ as:

$$\hat{r}_{uj} = p_u \cdot q_j + b_j, \tag{3}$$

where p_u and q_j represent the latent vector for user u and item j, respectively. Compared with plain MF, a bias term is included in Equation (3) and b_j is the bias term for item j. Here, p_u is a user ID embedding and the modeling of q_j will be discussed in the following subsection. By letting q_j to be an item ID embedding, Equation (3) reduces to the ordinary latent factor model.

Integrating CNN into MF

We consider text-aware recommendation where item j is a document associated with a sequence of word tokens denoted by D_j. To incorporate document features, we model the item vector q_j with the following additive approach:

$$q_j = g(D_j) + v_j, \tag{4}$$

where $g(\cdot)$ is a function that takes a raw document as input and is used to establish a high-level feature representation of the document. An item embedding vector v_j is included as an offset term and corresponds to the original item vector of the plain MF. By plugging Equation (4) into Equation (3), the CNN-based document modeling is integrated into the latent item vectors of MF and this additive manner allows the training signals to directly flow into the CNN.

This paper builds a model for $g(\cdot)$ based on Convolutional Neural Network (CNN), which is one of the most popular deep learning tools to extract high-level features from text data [29]. Unlike previous order–insensitive models [8,9], CNN has the benefit of exploiting the additional information inherent in word orders [29,30], which has recently been demonstrated to be critical for performing text-aware hybrid recommendation [10,14]. It is worth pointing out the difference between our model and ConvMF [10], which considers text information as the prior knowledge to regularize item latent vectors and thus requires to tune additional trade-off hyper-parameters for training. By contrast, our CNN is directly incorporated into the representations of items and thus enjoys more explicit flow of the learning signals than the regularization approach.

We design the CNN module $g(\cdot)$ as follows. First, we transform an item j's document D_j into the concatenation of a sequence of k-dimensional embeddings:

$$\mathbf{x} = e_1 \oplus e_2 \oplus \cdots \oplus e_n, \tag{5}$$

where n is the total number words in the text, $\oplus$ is the concatenation operator, and $e_i \in R^k$ represents the embedding vector of the i-th word in the document. Let $x_{i:i+h-1}$ refer to the concatenated embedding representation of a window of h words:

$$x_{i:i+h-1} = e_i \oplus e_{i+1} \oplus \cdots \oplus e_{i+h-1}. \tag{6}$$

We apply a convolutional filter $w_c^t \in R^{hk}$ to $x_{i:i+h-1}$ to produce a new feature c_i^t for $t = 1, \cdots, n_c$:

$$c_i^t = relu(w_c^t \cdot x_{i:i+h-1}) + b^t, \tag{7}$$

where $b^t \in R$ is a bias term for filter t, $relu$ is the rectified linear unit function, and n_c represents the total number of filters used. After applying to each possible window of words in the document, the filter w_c^t produces a feature vector $\mathbf{c}^j \in R^{n-h+1}$:

$$\mathbf{c}^t = [c_1^t, c_2^t, \cdots, c_{n-h+1}^t]. \tag{8}$$

Here, as one filtering weight captures only one type of contextual features, we use multiple filtering weights to detect multiple types of contextual features. To deal with the varied length of documents, we apply a max-pooling operation over all the feature vectors:

$$\tilde{\mathbf{c}} = [max\{\mathbf{c}^1\}, max\{\mathbf{c}^2\}, \cdots, max\{\mathbf{c}^{n_c}\}]. \tag{9}$$

By doing so, we transform documents of varied length into fixed-length vectors. Finally, we apply a two-layer nonlinear neural network on top of $\tilde{\mathbf{c}}$ to obtain:

$$g(D_j) = tanh(W_2\{tanh(W_1\tilde{\mathbf{c}} + b_1)\} + b_2), \tag{10}$$

where $tanh$ is the hyperbolic tangent function, W_1, W_2 are projection weight matrices, and b_1, b_2 represent bias vectors.

Based on the preference score and CNN document feature modeling, the generator generates a sample $\mathbf{s} = (u, j, j')$ with probability:

$$p_\theta(\mathbf{s}) = \delta(\hat{r}_{uj} - \hat{r}_{uj'}), \tag{11}$$

where $\delta(x) = \frac{1}{1+e^{-x}}$ is the sigmoid function. It is easy to verify that $p_\theta(\mathbf{s} = (u, j, j')) = 1 - p_\theta(\mathbf{s} = (u, j', j))$. Note that in order to generate samples from p_θ, we have to first select a personalized item pair as the input to the generator.

3.3. Discriminator Based on Latent Factor Model

The goal of the discriminator is to classify a given sample as real or fake. Theoretically, any binary classifier will fit the task. In our experiments, we apply the similar latent factor model as in generator with user and item embeddings as the latent vectors, which is similar to Equation (3) with no document modeling. Also, the output of the discriminator is binary and we thus build the discriminator on top of the user–item score by a similar logistic classifier given by Equation (11). Note that these embeddings are different parameters from the ones used in the generator. Besides, the output of the discriminator model, by contrast, indicates the probability of a sample being real. We

also experimented with standard Multi-Layer Perceptron (MLP) as the discriminator, where we used the concatenation of embeddings as its input and found that the latent factor model performed better.

Latent factor model, as discussed previously, satisfies the totality order constraint [40] which brings an additional optimization advantage in our learning setting. That is, minimizing the probability of a generated (fake) sample (u, j, j') being real is equivalent to maximizing the probability of (u, j', j) being real. This behavior is desirable if (u, j', j) happens to be a real sample, as one of the training goals for D_w is to maximize the probability of the truthful data. If the real sample is (u, j, j'), the objective in Equation (2) would force D_w, regardless of its specification as latent factor model or MLP, to output a probability of 0.5 for both (u, j, j') and (u, j', j), thus reaching a balance between p_r and p_θ.

3.4. Training

3.4.1. Optimizing Discriminator

The objective for the discriminator is to maximize the log-likelihood of correctly distinguishing between the true and generated data samples. With p_θ fixed, we update D_w by maximizing the objective in Equation (2). Equivalently, we minimize the following binary cross-entropy loss:

$$w^* = \arg\min_w -\{\mathrm{E}_{\mathbf{s}\sim p_r}[\log D_w(\mathbf{s})] + \mathrm{E}_{\mathbf{s}\sim p_\theta}[\log(1 - D_w(\mathbf{s}))]\}. \tag{12}$$

More specifically, we first draw samples from the two distributions p_θ and p_r, and then train a binary classifier D_w based on the two sets of samples. We perform a gradient descent algorithm to learn the model parameters w and incorporate L_2 regularization term into the objective loss, with λ as the trade-off parameter.

As the real data distribution p_r is usually unknown to us, sampling from p_r is considered equivalent to randomly selecting samples from the training data [19,39,49]. The samples of p_r provide triple indexes as input to generator p_θ, which is able to generate a sample in every Bernoulli trial due to the totality constraint of the pairwise ordering scheme. Specifically, suppose a real data sample $\mathbf{s} = (u, j, j')$ is selected from p_r, then accordingly, we run a Bernoulli trial with probability given by $p_\theta(\mathbf{s} = (u, j, j'))$. If the result is 1, a sample $\mathbf{s}' = (u, j, j')$, which is the same as given input $\mathbf{s}$, is regarded to be "generated" by the current p_θ; otherwise, $\mathbf{s}' = (u, j', j)$ by exchanging the ranking positions between item j' and j in the $\mathbf{s}$.

However, the formulation of the training set S shown by Equation (1) introduces noisy samples, since not all observed items are preferred over unobserved items. To address this problem, we first select samples from S, and then remove those for which the value of p_θ is less than ϵ. The motivation is to remove the noisy samples which are considered to be impossible observations by a good (to some extent) model. We set ϵ to zero at the start, so that we can learn a relative good model. Then, as training progresses, we gradually increase its value until it reaches an upper bound, which was set to 0.3 in our experiments. Note that the idea of resorting to the model at training time is not completely new and has been similarly applied in [50].

3.4.2. Optimizing Generator

On the other hand, the generative model p_θ intends to minimize the objective in Equation (2). It fits the underlying true distribution by randomly generating samples to fool the discriminator. Formally, keeping the discriminator D_w fixed, we update the parameters θ by maximizing the probability that samples from p_θ are classified as real by the discriminator:

$$\begin{aligned} \theta^* &= \arg\min_\theta \mathrm{E}_{\mathbf{s}\sim p_\theta}[\log(1 - D_w(\mathbf{s}))]. \\ &= \arg\max_\theta \mathrm{E}_{\mathbf{s}\sim p_\theta}[D_w(\mathbf{s})]. \end{aligned} \tag{13}$$

From the perspective of reinforcement learning [51–53], p_θ acts as the policy to select actions (samples), in which the reward for an action $\mathbf{s}$ is given by the term $\log D_w(\mathbf{s})$. The goal of Equation (13) is to shift the policy through its parameters to increase the reward of generated samples, as judged by $\log D_w(\cdot)$. Following [39,51–53], we adopt the policy gradient method and calculate the gradient with respect to the generator parameters as follows:

$$\begin{aligned} \bigtriangledown \mathrm{E}_{\mathbf{s}\sim p_\theta}[\log D_w(\mathbf{s})] &= \sum_{\mathbf{s}} \log D_w(\mathbf{s}) \bigtriangledown p_\theta(\mathbf{s}) \\ &= \sum_{\mathbf{s}} \log D_w(\mathbf{s}) p_\theta(\mathbf{s}) \bigtriangledown \log p_\theta(\mathbf{s}) \\ &= \mathrm{E}_{\mathbf{s}\sim p_\theta}[\log D_w(\mathbf{s}) \bigtriangledown \log p_\theta(\mathbf{s})]. \end{aligned} \tag{14}$$

Here, $\log D_w(\cdot)$ based on the discriminator acts as learning weights and assigns relatively large weights (rewards) to those samples that are incorrectly modeled by the current p_θ.

One typical solution for the last step is to perform Monte Carlo approximation and compute the average of multiple samples. However, this would distort the function of $\log D_w(\cdot)$, since the weights of different samples would counter-weigh each other in the training process. This is a direct result of the generation process of the generator. Therefore, we perform stochastic gradient decent and update the parameters using one sample at a time (shown in Algorithm 1). We note that the update rule has a very similar form to LambdaRank [54,55] and the difference is that they use current item ranking list to compute the learning weights while we exploit signals from the discriminator. It is noteworthy that theoretically we can sample the unobserved item ranking pairs that are not in the training set S, thus improving the model's generalization ability.

Overall, the adversarial training is an alternating optimization with respect to the generator parameters θ and the discriminator parameters w, i.e., alternating between Equations (12) and (13). Following [39], we pre-train the generative model to ensure a stable model training process in experiments. The convergence of the proposed model ADHR is monitored through early stopping on a validation dataset. The overall optimization procedure is illustrated in Algorithm 1.

Algorithm 1: Optimization Algorithm for Our Proposed Model ADHR

Input: generator p_θ; discriminator D_w; and training set S.
Initialize p_θ with a pre-trained model;
repeat
  for *train* D_w **do**
    Draw a sample $\mathbf{s} = (u, j, j')$ uniformly from the pairwise training data S;
    Run a Bernoulli trial with $p_\theta(\mathbf{s})$ to generate a fake sample $\mathbf{s}'$;
    Update $w \leftarrow w + \eta(\bigtriangledown \log D_w(\mathbf{s}) + \bigtriangledown \log(1 - D_w(\mathbf{s}')) - \lambda w)$.
  end
  for *train* p_θ **do**
    Select a triple (u, j, j'), where i and j, j' are randomly drawn from U and I, respectively;
    Run a Bernoulli trial with $p_\theta((u, j, j'))$ to generate a training sample $\mathbf{s}$;
    Update $\theta \leftarrow \theta - \eta'(\log D_w(\mathbf{s}) \bigtriangledown \log p_\theta(\mathbf{s}) + \lambda' \theta)$.
  end
until *Convergence*;

3.4.3. Complexity

Just like other GAN training [19,21,39], the complexity of the algorithm highly relies on the number of iterations, each of which is of linear complexity with respect to the number of training samples. As we perform stochastic gradient descent based on bootstrap sampling of training triples, only a fraction of a full cycle of the data is sufficient for convergence. On the other hand, compared

with previous deep hybrid models [9,10], the extra cost of our model mainly concentrates on training discriminator, which can be ignored compared with the cost of training the deep generator model.

4. Experiment

In this section, we conduct extensive experiments to show that: (1) our proposed model ADHR outperforms the state-of-the-art methods in both warm-start and cold-start recommendation settings; and (2) ADHR is effective in learning the distribution over user behavioral data and capturing the contextual features in the document. We will first present the experimental settings, followed by analyzing the above research topics one by one.

4.1. Experimental Setting

4.1.1. Evaluation Scheme

4.1.2. Datasets

We perform experiments on three real-world datasets obtained from MovieLens (http://grouplens.org/datasets/movielens/) and Amazon (http://jmcauley.ucsd.edu/data/amazon/). These datasets consist of users' 5-star ratings on items. For Amazon, the dataset contains users' purchases of instant video and we consider the reviews as the description documents associated with items. For MovieLens, we use plot summary of corresponding items from IMDB (http://www.imdb.com/). To adapt to implicit feedback recommendation, we follow a widely adopted approach [10,39,40] to preprocess all datasets and transform ratings into binaries to indicate whether users have rated items. Similar to [8,10,16], the documents are preprocessed with the following procedures: (1) remove stop words; (2) set the maximum document length to 300; (3) select top 8000 distinct words as vocabulary; (4) remove non-vocabulary words from document. The statistics of the final datasets are given in Table 1, where we use Avg. to indicate the average number of words per document.

Table 1. Statistics of the evaluation datasets.

Dataset	#Records	#Users	#Items	Sparsity	Avg.
MovieLen-1m	993,482	6040	3544	95.36%	97.09
MovieLen-10m	9,945,875	69,878	10,073	98.59%	92.05
Amazon	135,188	29,757	15,149	99.97%	73.03

4.1.3. Evaluation Methodology

We test the models on held-out datasets for both warm start and cold start recommendation, with the evaluation settings described as follows.

Warm Start. We adopt 5-fold cross-validation. For each user, we do a 5-fold split of their interactions. We always assign items with less than 5 interactions to training set. This ensures that there are no cold start items in the test set. The removed interactions simulate the unobserved data (zero entries in the matrix) in the real scenario. After learning, we predict interactions across all unobserved items for each user.

Cold Start. We again adopt 5-fold cross-validation. We consider the cold item recommendation setting and the situation is similar to recommending for cold users. We evenly split the set of all items into 5 folds. Similarly, we always assign items with less than 5 interactions to training set. We fit the models on the training set items and form predictive per user ranking of items in the test set.

Metrics. After training, a top-N ranked list of unobserved items can be returned for each user, which are then compared to the test data for performance evaluation. We adopt two widely-used performance metrics: Recall@N and truncated Normalized Discounted Cumulative Gain (NDCG@N). Formally, let T represent the binary user–item interaction matrix of the test data and define π_u as the

column indexes of items in the recommended ranking list for user u. $\pi_u(k)$ returns the item column index in the k-th ranking position for user u.

Recall@N for user u is defined as:

$$\text{Recall@}N(u, \pi_u) = \frac{1}{min(N, \sum_{j=1}^{n} T(u,j))} \sum_{k=1}^{N} T(u, \pi_u(k)),$$

where the denominator calculates the minimum between the truncation number N and the number of items that user u has interacted with. We use Recall@N to denote the average of Recall@$N(u, \pi_u)$ over all users. Recall@N considers whether the recommended items are in the user list.

DCG@N for user u is computed by:

$$\text{DGG@}N(u, \pi_u) = \sum_{k=1}^{N} \frac{2^{T(u,\pi_u(k))} - 1}{log(k+1)}.$$

NDCG@$N(u, \pi_u)$ is the normalized DCG@$N(u, \pi_u)$ with respect to all possible rankings. We use NDCG@N to denote the average of NDCG@$N(u, \pi_u)$ over all users. NDCG takes into account the ranking positions of the recommended items and similarly, we report NDCG@N as the average measure over all users.

4.1.4. Baselines

We compare our proposed model with the following baselines, ranging from classical MF-based method to state-of-the-art hybrid recommendation models.

- **BPR** (http://www.mymedialite.net): BPR (Bayesian Personalized Ranking) [40] is a classical MF-based recommendation model which focuses on optimizing a pair-wise rank-award objective loss.
- **IRGAN** (https://github.com/geek-ai/irgan): IRGAN [39] is a state-of-the-art non-hybrid recommendation method and learns from user–item interactions only with no content feature modeling. It applies list-wise learning with softmax based generator in an adversarial training framework.
- **CTR** (https://github.com/blei-lab/ctr): Collaborative Topic Regression [8] is a classical non-DL based hybrid model, which combines topic modeling with matrix factorization.
- **CDL** (https://github.com/js05212/CDL): Collaborative Deep Learning [9] is a state-of-the-art DL-based hybrid model, which couples denoising auto-encoders with matrix factorization.
- **CVAE** (http://eelxpeng.github.io/research/): Collaborative Variational Autoencoder [16] is another state-of-the-art DL-based hybrid model, which learns deep features from content data in an unsupervised manner and also captures relationships between items and users from both content and implicit feedback.

4.1.5. Implementation Details

For all the baselines, we use the publicly accessible code and try our best to tune their hyper-parameters on our datasets. We implement our proposed model ADHR using Pytorch (https://pytorch.org) with Nvidia GeForce GTX 1080 GPU. The model parameters are updated based on batch gradient with Adam optimizer [56]. As for the CNN architecture, the word embeddings were randomly initialized with dimension of 200. We used window sizes (h) of $3, 4, 5$ with 100 (n_c) feature filters each. In experiments, we perform grid search to find the best performing hyper-parameters for all comparion methods based on validation datasets. We also used dropout to prevent CNN from over-fitting [57], with the dropout rate set as 0.1. In order to balance between the unseen and observed samples in training generator, we updates the parameters of generator 10 times more often than updating the discriminator. The batch sizes for training the generator and discriminator are 512 and 256, respectively. We perform grid search to select the learning rate from $\{0.001, 0.005, 0.01, 0.05, 0.1\}$

and the regularization parameter from $\{0.0001, 0.001, 0.01, 0.1\}$. The besting performing values of the L_2 regularization parameters and the learning rates for different datasets are reported in Table 2.

Table 2. Best performing hyper-parameters on the three datasets.

Dataset	Learning Rate η	Learning Rate η'	Regularization λ	Regularization λ'
MovieLens1m	0.005	0.01	0.001	0.001
MovieLens10m	0.05	0.1	0.001	0.01
Amazon	0.01	0.01	0.005	0.001

4.2. Qualitative Performance Comparison Results

We report the comparison performance results of all methods in various settings. The experiments show that our proposed method ADHR is able to achieve superior performance against state-of-the-art methods in both cold-start and warm-start recommendation settings and the performance improvement are consistent across the metrics and the three datasets.

4.2.1. General Comparison Results and Analysis

Tables 3 and 4 show the performance results of all comparison methods in warm-start and cold-start recommendation settings, respectively. As BPR and IRGAN are unable to perform cold-start recommendation, we omit them in Table 3 and compare the hybrid models only. Similar to [9,16], we fix the dimension of latent user and item vectors as 50 for all models and truncate recommended list at 100 for both metrics. The following observations can be noted from the results.

Table 3. Warm-start top-20 recommendation performance of all comparison methods.

	MovieLens1m		MovieLens10m		Amazon	
Model	**Recall**	**NDCG**	**Recall**	**NDCG**	**Recall**	**NDCG**
BPR	0.259	0.273	0.182	0.191	0.043	0.051
IRGAN	0.327	0.355	0.224	0.232	0.067	0.071
CTR	0.343	0.365	0.292	0.301	0.081	0.086
CDL	0.401	0.420	0.311	0.316	0.088	0.090
CVAE	0.421	0.432	0.317	0.325	0.093	0.094
ADHR	**0.451**	**0.453**	**0.333**	**0.339**	**0.103**	**0.104**
Improve (abs)	0.030	0.021	0.025	0.014	0.010	0.010
Improve (%)	7.1%	4.9%	5.0%	4.3%	10.7%	10.6%

Table 4. Cold-start top-20 recommendation performance of all comparison methods.

	MovieLens1m		MovieLens10m		Amazon	
Model	**Recall**	**NDCG**	**Recall**	**NDCG**	**Recall**	**NDCG**
CTR	0.301	0.325	0.232	0.241	0.062	0.066
CDL	0.322	0.346	0.261	0.276	0.071	0.082
CVAE	0.351	0.372	0.277	0.285	0.081	0.086
ADHR	**0.373**	**0.388**	**0.289**	**0.298**	**0.090**	**0.094**
Improve (abs)	0.022	0.016	0.012	0.013	0.009	0.008
Improve (%)	6.3%	4.3%	4.3%	4.6%	11.1%	9.3%

First, among the baselines, we see from Table 3 that IRGAN shows a significant performance improvement over BPR in the warm-start recommendation settings. The results are consistent with [39], indicating the powerful learning ability of adversarial training paradigm. We observe that there is a large performance gap between IRGAN and the hybrid recommendation methods. Specifically, IRGAN shows worse performance results than CTR, which only applies shallow feature learning

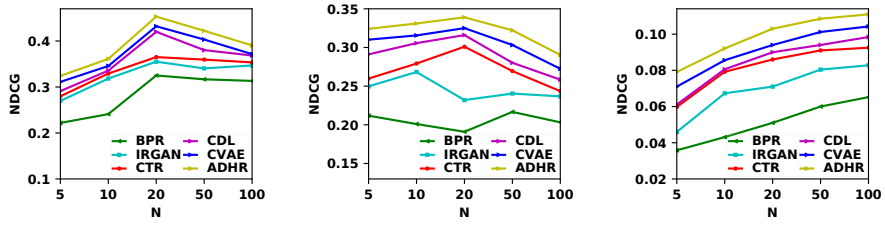
method LDA for content modeling. These findings indicate the very necessity of incorporating content features for text-aware recommendation.

Second, among the hybrid baseline models, DL-based methods CDL and CVAE outperform the non-DL model CTR in both warm-start and cold-start recommendation settings. The results are consistent with the findings in [9,16] and the performance improvements are attributed to the fact that deep learning methods can extract more effective features from the text data to aid collaborative filtering than that of the shallow feature learning method like LDA used in CTR.

Lastly, we can see that our proposed model ADHR delivers consistent performance improvements over all baseline across the three datasests on both metrics. ADHR delivers better performance than CDL and CVAE despite the fact that both belong to DL-based hybrid models. ADHR outperforms the best performing baseline CVAE across all three datasets by a margin of 4.3%∼10.8% in the warm-start recommendation setting and by a margin of 4.3%∼11.1% in the cold-start recommendation setting. The results demonstrate that CNN is better at capturing textual information than auto-encoders and our proposed adversarial framework is effective at learning the hybrid recommendation model.

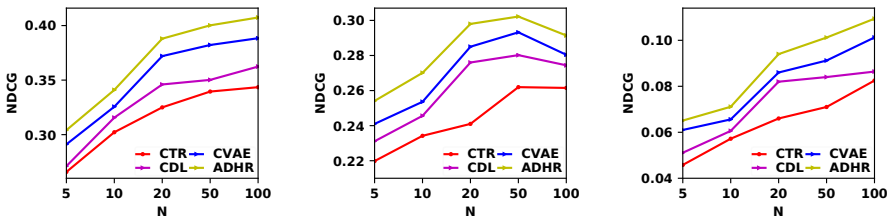
4.2.2. Performance Comparison w.r.t. Truncated Value *N*

To further compare against the baselines, we illustrate the NDCG@*N* of all comparison methods w.r.t. the truncated value *N* in the warm-start and cold-start recommendation settings in Figures 2 and 3, respectively. We can see that our proposed model ADHR demonstrates consistent improvements over other methods across all positions on the two datasets. In the warm-start recommendation setting, ADHR outperforms the best performing baseline CVAE by a margin of 4.2%∼5.2%, 4.3%∼6.6% and 7.2%∼11.3% on MovieLens1m, MovieLens10m and Amazon, respectively. In the cold-start recommendation setting, ADHR outperforms CVAE by a margin of 3.6%∼5.4%, 4.1%∼6.3% and 9.2%∼12.1% on MovieLens1m, MovieLens10m and Amazon, respectively. For Amazon, since the dataset is even sparser, the improvement is more significant than on the other two datasets. The results further validate the effectiveness of our proposed approach in utilizing content data to assist collaborative filtering.



(a) MovieLens1m (b) MovieLens10m (c) Amazon

Figure 2. Warm-start recommendation performance w.r.t. different values of N.

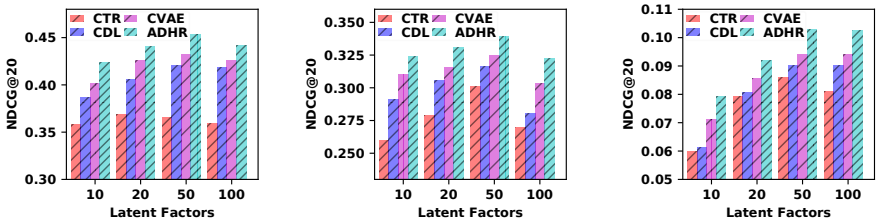


(a) MovieLens1m (b) MovieLens10m (c) Amazon

Figure 3. Cold-start recommendation performance w.r.t. different values of N.

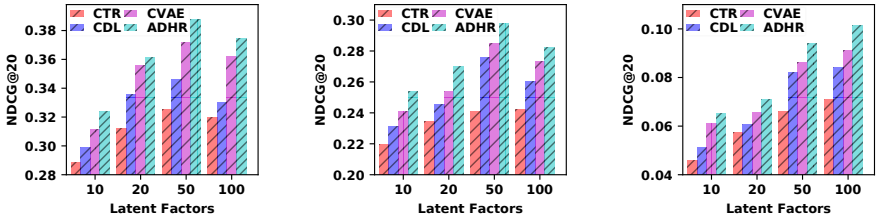
4.2.3. Performance Comparison w.r.t. Latent Factors

Moreover, to further compare the hybrid recommendation models, we study their NDCG@20 performance with respect to the number of latent factors on the three datasets. We report the results in the cold-start and warm-start recommendation settings in Figures 4 and 5, respectively. Again, we observe that our proposed model ADHR delivers consistently better performance than the best performing baseline CVAE across different latent factors. In the warm-start recommendation setting, ADHR outperforms CVAE by a margin of 4.3%∼5.5%, 3.1%∼5.2% and 7.1%∼11.5% on MovieLens1m, MovieLens10m and Amazon, respectively. In the cold-start recommendation setting, ADHR outperforms CVAE by a margin of 4.3%∼5.5%, 4.1%∼6.3% and 6.1%∼12.3% on MovieLens1m, MovieLens10m and Amazon, respectively.



(a) MovieLens1m (b) MovieLens10m (c) Amazon

Figure 4. Warm-start recommendation performance w.r.t. latent dimensionality.



(a) MovieLens1m (b) MovieLens10m (c) Amazon

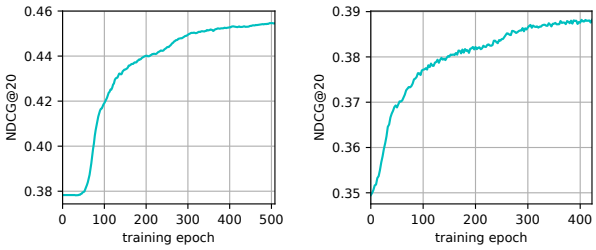
Figure 5. Cold-start recommendation performance w.r.t. latent dimensionality.

4.3. Effectiveness of Adversarial Learning and CNN

We analyze the converge of our proposed model ADHR in both warm-start and cold-start recommendation settings and demonstrate the effectiveness of our proposed adversarial training paradigm in learning the data distribution. Moreover, we show the effectiveness of CNN in capturing the contextual features in document modeling in the proposed model ADHR.

4.3.1. Convergence

To show the effectiveness of adversarial learning, we study the convergence of our proposed model ADHR with respect to training epochs on MovieLens1m in Figure 6. The starting points in the figure represent the performance results of a pre-trained model, which is of the same form as the generator model of ADHR, but is trained using typical stochastic gradient decent as adopted by BPR [40–42]. We used the best pre-trained model though tuning the hyper-parameters for MAP. From the results, we observe that our proposed model ADHR can deliver consistent improvement over the pre-trained model from the beginning of adversarial training. We also tried with training ADHR with some non-best pre-trained models and found ADHR was consistently able to converge to the best performance. These results demonstrate that our proposed pairwise adversarial training framework is effective at capturing the collaborative ranking distributions over the user–item interaction data.



(a) Warm-start recommendation (b) Cold-start recommendation

Figure 6. Convergence of our proposed model ADHR on MovieLens1m.

4.3.2. CNN in Document Modeling

The performance of our proposed hybrid recommendation model is affected by the effectiveness of CNN in capturing the contextual features of the item documents. As the convolution filtering weights play the key role of detecting text features in CNN, we study whether the contextual meaning of phrases are captured by the convolution matrices.

To investigate the CNN's capability in our proposed hybrid model ADHR, we select the convolution filtering weight W_c^{11} and W_c^{86} from the model trained on MovieLens10m dataset. The convolution weights capture the contextual meaning of the phrases shown in Table 5. Although the term "trust" seem similar to each other in the two phrases, it has subtlety different contextual meanings captured by the convolution weights. Specifically, the "trust" in the first phase is a verb while in the second one it is used as noun. To very this distinction, we change the term "trust" in the phrases and study the change of weight values. As shown in Table 5, when replacing "trust" with "believe" and "faith" in the phase captured by W_c^{11}, the weight value for "believe" is higher than that for "faith". By contrast, in the case of W_c^{86}, the weight value for "believe" is lower than that for "faith", which matches our expectation that the "trust" word shows different meanings in the two phases. The results thus demonstrate that the CNN module in our model is able to distinguish the different contextual meaning of words in documents via the convolution filtering weights.

Table 5. Case study of Convolution Filtering Weights of ADHR.

Phrase captured by W_c^{11}	$\max(c^{11})$	Phrase captured by W_c^{86}	$\max(c^{86})$
people **trust** the man	0.0712	betray his **trust** finally	0.1004
Test phrases for W_c^{11}	$\max(c_{test}^{11})$	Test phrases for W_c^{86}	$\max(c_{test}^{86})$
people **believe** the man	**0.0398**	betray his **believe** finally	0.0632
people **faith** the man	0.0365	betray his **faith** finally	**0.698**

Similar to [58–62], we apply a simple tool to further analyze the predictions of our proposed hybrid model ADHR. Specifically, we produce a heat-map where we associate each input word with its impact on the output prediction in Figure 7. Suppose item j is recommended to user u. That is, we assume $\hat{r}_{uj}$ is large. Then let $[e_1, e_2, \cdots, e_n]$ be the sequence of word embeddings associated with item j. We generate the heat-map's value for word t by converting the gradient with respect to the word into a scalar. Since the word is discrete, we compute $||\frac{d\hat{r}_{uj}}{de_t}||$ instead and obtain the gradient by back-propagating through the CNN module $g(\cdot)$.

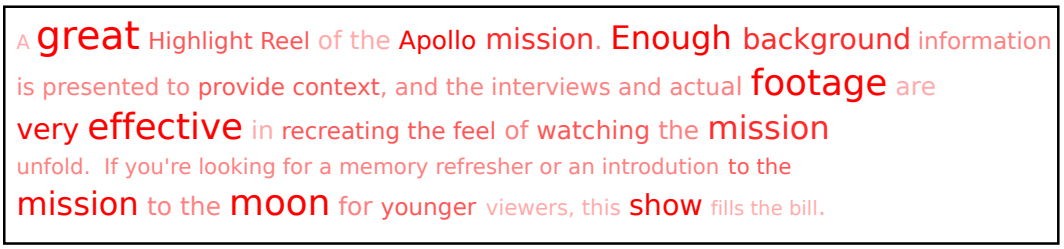


Figure 7. Saliency of each word in a review regarding an instant video in the Amazon dataset. Size and color of the words indicate the influence on final predicted score. The important chunks of words such as "great" and "footage" are learned by the model.

5. Conclusions

In this paper, we propose an adversarial training framework to learn a deep hybrid recommendation model, which seamlessly integrates a Convolutional Neural Network (CNN) into latent factor model to exploit deep content features as well as the collaborative patterns from user–item interactions. Specifically, our main work can be concluded as follows.

First, we propose a hybrid recommendation model that incorporates a Convolutional Neural Network (CNN) to learn the deep text features. As CNN is an order-sensitive textual feature extractor, it offers the benefit of exploiting the additional information in word orders.

Second, with the aim of reflecting the ranking nature of top-N recommendation, we propose an adversarial learning framework to capture the pair-wise ranking distribution over user–item interactions in an effective way.

Finally, we have conducted extensive experiments on three real-world datasets to demonstrate that our proposed model is able to deliver the state-of-the-art performance across both warm-start and cold-start recommendation settings.

Author Contributions: Conceptualization, X.Z., D.D.; Methodology, X.Z. and D.D.; Software, X.Z. and D.D.; Validation; X.Z., D.D.; Formal analysis, Z.Z. and D.D.; Investigation, X.Z.; Resources, X.Z.; Data Curation, X.Z.; Writing–original draft preparation, X.Z. and D.D.; Writing–Review& Editing, Z.Z. and D.D.; Visualization, D.D.; Supervision, X.Z.; Project Administration, X.Z.; Funding Acquisition, X.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported in part by the National Key R&D Program of China (No. 2018YFB1403001), the National Natural Science Foundation of China (No. U1509221), and the Zhejiang Provincial Key R&D Program, China (No. 2017C03044).

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript (in alphabetical order):

ADHR	Adversarial Deep Hybrid Recommendation
BPR	Bayesian Personalized Ranking
CDL	Collaborative Deep Learning
CNN	Convolutional Neural Network
CTR	Collaborative Topic Regression
CVAE	Collaborative Variational Auto-Encoder
DL	Deep Learning
GAN	Generative Adversarial Net
LDA	Latent Dirichlet Allocation
MF	Matrix Factorization

References

1. Koren, Y.; Bell, R. Advances in Collaborative Filtering. In *Recommender Systems Handbook*; Springer: Berlin/Heidelberg, Germany, 2011; pp. 145–186.
2. Salakhutdinov, R.; Mnih, A. Probabilistic Matrix Factorization. In Proceedings of the 20th International Conference on Neural Information Processing Systems, Vancouver, BC, Canada, 3–6 December 2007; Volume 8, pp. 1257–1264.
3. Hu, Y.; Koren, Y.; Volinsky, C. Collaborative Filtering for Implicit Feedback Datasets. In Proceedings of the 2008 Eighth IEEE International Conference on Data Mining, Pisa, Italy, 15–19 December 2008; Volume 10, pp. 263–272.
4. Koren, Y.; Bell, R.; Volinsky, C. Matrix Factorization Techniques for Recommender Systems. *Computer* **2009**, *8*, 30–37. [CrossRef]
5. Bobadilla, J.; Ortega, F.; Hernando, A.; Gutiérrez, A. Recommender Systems Survey. *Knowl.-Based Syst.* **2013**, *24*, 109–132. [CrossRef]
6. Isinkaye, F.O.; Folajimi, Y.O.; Ojokoh, B.A. Recommendation Systems: Principles, Methods and Evaluation. *Egypt. Inform. J.* **2015**, *13*, 261–273. [CrossRef]
7. Kumar, P.; Thakur, R.S. Recommendation System Techniques and Related Issues: A Survey. *Int. J. Inf. Technol.* **2018**, *6*, 495–501. [CrossRef]
8. Wang, C.; Blei, D.M. Collaborative Topic Modeling for Recommending Scientific Articles. In Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Diego, CA, USA, 21–24 August 2011; Volume 9, pp. 448–456.
9. Wang, H.; Wang, N.; Yeung, D.-Y. Collaborative Deep Learning for Recommender Systems. In Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Sydney, Australia, 10–13 August 2015; Volume 10, pp. 1235–1244.
10. Kim, D.; Park, C.; Oh, J.; Lee, S.; Yu, H. Convolutional Matrix Factorization for Document Context-Aware Recommendation. In Proceedings of the 10th ACM Conference on Recommender Systems, Boston, MA, USA, 15–19 September 2016; Volume 8, pp. 233–240.
11. Blei, D.M.; Ng, A.Y.; Jordan, M.I. Latent Dirichlet Allocation. *J. Mach. Learn. Res.* **2003**, *30*, 993–1022.
12. Hu, L.; Cao, J.; Xu, G.; Cao, L.; Gu, Z.; Zhu, C. Personalized Recommendation via Cross-domain Triadic Factorization. In Proceedings of the 22nd International Conference on World Wide Web, Rio de Janeiro, Brazil, 13–17 May 2013; Volume 12, pp. 595–606.
13. Li, S.; Kawale, J.; Fu, Y. Deep Collaborative Filtering via Marginalized Denoising Auto-encoder. In Proceedings of the 24th ACM International on Conference on Information and Knowledge Management, Melbourne, Australia, 19–23 October 2015; Volume 10, pp. 811–820.
14. Bansal, T.; Belanger, D.; McCallum, A. Ask the GRU: Multi-task Learning for Deep Text Recommendations. In Proceedings of the 10th ACM Conference on Recommender Systems, Boston, MA, USA, 15–19 September 2016; Volume 8, pp. 107–114.
15. Zheng, L.; Noroozi, V.; Yu, P.S. Joint Deep Modeling of Users and Items Using Reviews for Recommendation. In Proceedings of the Tenth ACM International Conference on Web Search and Data Mining, Cambridge, UK, 6–10 February 2017; Volume 10, pp. 425–434.
16. Li, X.; She, J. Collaborative Variational Autoencoder for Recommender Systems. In Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Halifax, NS, Canada, 13–17 August 2017; Volume 10, pp. 305–314.
17. McAuley, J.; Leskovec, J. Hidden Factors and Hidden Topics: Understanding Rating Dimensions with Review Text. In Proceedings of the 7th ACM Conference on Recommender Systems, Hong Kong, China, 12–16 October 2013; Volume 8, pp. 165–172.
18. van den Oord, A.; Dieleman, S.; Schrauwen, B. Deep Content-based Music Recommendation. In Proceedings of the 26th International Conference on Neural Information Processing Systems, Lake Tahoe, Nevada, 5–10 December 2013; Volume 9, pp. 2643–2651.
19. Goodfellow, I.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; Bengio, Y. Generative Adversarial Nets. In *Advances in Neural Information Processing Systems 27*; Neural Information Processing Systems Foundation: Montreal, QC, Canada, 2014; Volume 10, pp. 2672–2680.

20. Huszár, F. How (not) to Train Your Generative Model: Scheduled Sampling, Likelihood, Adversary? *arXiv* **2015**, arXiv:1511.05101.
21. Martin, A.; Soumith, C.; Léon, B. Wasserstein Generative Adversarial Networks. In Proceedings of the 34th International Conference on Machine Learning, Sydney, Australia, 6–11 August 2017; Volume 10, pp. 214–223.
22. Salimans, T.; Goodfellow, I.; Zaremba, W.; Cheung, V.; Radford, A.; Chen, X. Improved Techniques for Training GANs. In Proceedings of the 30th International Conference on Neural Information Processing Systems, Barcelona, Spain, 5–10 December 2016; Volume 9, pp. 2234–2242.
23. Alex, K.; Sutskever, I.; Hinton, G. ImageNet Classification with Deep Convolutional Neural Networks. In *Advances in Neural Information Processing Systems 25*; Curran Associates, Inc.: New York, NY, USA, 2012; Volume 9, pp. 1097–1105.
24. Tandon, N.; Varde, A.; de Melo, G. Commonsense Knowledge in Machine Intelligence. *ACM SIGMOD Rec. J.* **2017**, *12*, 49–52. [CrossRef]
25. Abidha, P.; Manish, P.; Aparna, V. Object Detection with Neural Models, Deep Learning and Common Sense to Aid Smart Mobility. In Proceedings of the IEEE 30th International Conference on Tools with Artificial Intelligence (ICTAI), Volos, Greece, 7 November 2018; Volume 4, pp. 859–863.
26. Ketaki, G.; Aparna, S.V.; Xu, D. Sentiment Analysis of Twitter Data with Hybrid Learning for Recommender Applications. In Proceedings of the 2018 9th IEEE Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON), Columbia, NY, USA, 8–10 November 2018.
27. Liu, H.; Singh, P. ConceptNet—A Practical Commonsense Reasoning ToolKit. *BT Technol. J.* **2004**, *22*, 211–226. [CrossRef]
28. Whitelaw, C.; Garg, N.; Argamon, S. Using Appraisal Groups for Sentiment Analysis. In Proceedings of the 14th ACM International Conference on Information and Knowledge Management, Bremen, Germany, 31 October–5 November 2005; Volume 7, pp. 625–631.
29. Kalchbrenner, N.; Grefenstette, E.; Blunsom, P. A Convolutional Neural Network for Modelling Sentences. In Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, Baltimore, MD, USA, 22–27 June 2014; Volume 6, pp. 1746–1751.
30. Qian, Y.; Dong, J.; Wang, W.; Tan, T. Learning and Transferring Representations for Image Steganalysis Using Convolutional Neural Network. In Proceedings of the 2016 IEEE International Conference on Image Processing (ICIP), Phoenix, AK, USA, 25–28 September 2016; Volume 5, pp. 2752–2756.
31. Zhou, K.; Yang, S.-H.; Zha, H. Functional Matrix Factorizations for Cold-start Recommendation. In Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval, Beijing, China, 24–28 July 2011; Volume 10, pp. 315–324.
32. Maciej, K. Metadata embeddings for user and item cold-start recommendations. In Proceedings of the 2nd Workshop on New Trends on Content-Based Recommender Systems Co-Located with 9th ACM Conference on Recommender Systems, Vienna, Austria, 16–20 September 2015; Volume 8, pp. 14–21.
33. Bengio, Y.; Courville, A.; Vincent, P. Representation Learning: A Review and New Perspectives. *IEEE Trans. Pattern Anal. Mach. Intell.* **2013**, *35*, 1798–1828. [CrossRef] [PubMed]
34. Salakhutdinov, R.; Mnih, A. Bayesian Probabilistic Matrix Factorization Using Markov Chain Monte Carlo. In Proceedings of the 25th International Conference on Machine Learning, Helsinki, Finland, 5–9 July 2008; Volume 8, pp. 880–887.
35. Charlin, L.; Ranganath, R.; McInerney, J.; Blei, D.M. Dynamic Poisson Factorization. In Proceedings of the 9th ACM Conference on Recommender Systems, Vienna, Austria, 16–20 September 2015; Volume 8, pp. 155–162.
36. Yu, L.; Zhang, W.; Wang, J.; Yu, Y. SeqGAN: Sequence Generative Adversarial Nets with Policy Gradient. In Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, San Francisco, CA, USA, 4–9 February 2017; Volume 7, pp. 2852–2858.
37. Xu, T.; Zhang, P.; Huang, Q.; Zhang, H.; Gan, Z.; Huang, X.; He, X. Attngan: Fine-grained Text to Image Generation with Attentional Generative Adversarial Networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, CA, USA, 4–9 June 2018; Volume 9, pp. 1316–1324.
38. Gulrajani, I.; Ahmed, F.; Arjovsky, M.; Dumoulin, V.; Courville, A. Improved Training of Wasserstein GANs. In Proceedings of the 31st International Conference on Neural Information Processing Systems, Long Beach, CA, USA, 3–9 December 2017; Volume 11, pp. 5769–5779.

39. Wang, J.; Yu, L.; Zhang, W.; Gong, Y.; Xu, Y.; Wang, B.; Zhang, P.; Zhang, D. IRGAN: A Minimax Game for Unifying Generative and Discriminative Information Retrieval Models. In Proceedings of the 40th International ACM SIGIR conference on Research and Development in Information Retrieval, Tokyo, Japan, 7–11 August 2017; Volume 10, pp. 515–524.
40. Rendle, S.; Freudenthaler, C.; Gantner, Z.; Schmidt-Thieme, L. BPR: Bayesian Personalized Ranking from Implicit Feedback. In Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence, Montreal, QC, Canada, 18–21 June 2009; Volume 10, pp. 452–461.
41. Rendle, S.; Freudenthaler, C. Improving Pairwise Learning for Item Recommendation from Implicit Feedback. In Proceedings of the 7th ACM International Conference on Web Search and Data Mining, New York, NY, USA, 24–28 February 2014; Volume 10, pp. 273–282.
42. He, X.; He, Z.; Du, X.; Chua, T.-S. Adversarial Personalized Ranking for Recommendation. In Proceedings of the 41st International ACM SIGIR Conference on Research, Development in Information Retrieval, Ann Arbor, MI, USA, 8–12 July 2018; Volume 10, pp. 355–364.
43. Shi, Y.; Larson, M.; Hanjalic, A. List-wise Learning to Rank with Matrix Factorization for Collaborative Filtering. In Proceedings of the Fourth ACM Conference on Recommender Systems, Barcelona, Spain, 26–30 September 2010; Volume 4, pp. 269–272.
44. Xu, T.; Zhang, P.; Huang, Q.; Zhang, H.; Gan, Z.; Huang, X.; He, X. CLiMF: Learning to Maximize Reciprocal Rank with Collaborative Less-is-More Filtering. In Proceedings of the Sixth ACM Conference on Recommender Systems, Dublin, Ireland, 2–9 April 2012; Volume 8, pp. 139–146.
45. Wu, Y.; DuBois, C.; Zheng, A.X.; Ester, M. Collaborative Denoising Auto-encoders for Top-n Recommender Systems. In Proceedings of the Ninth ACM International Conference on Web Search and Data Mining, San Francisco, CA, USA, 22–25 February 2016; Volume 10, pp. 153–162.
46. Chen, T.; Sun, Y.; Shi, Y.; Hong, L. On Sampling Strategies for Neural Network-based Collaborative Filtering. In Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Halifax, NS, Canada, 13–17 August 2017; Volume 10, pp. 767–776.
47. He, X.; Liao, L.; Zhang, H.; Nie, L.; Hu, X.; Chua, T.-S. Neural Collaborative Filtering. In Proceedings of the 26th International Conference on World Wide Web, Perth, Australia, 3–7 April 2017; Volume 10, pp. 173–182.
48. Zhang, S.; Yao, L.; Sun, A.; Tay, Y. Deep Learning Based Recommender System: A Survey and New Perspectives. *ACM Comput. Surv.* **2019**, *38*, 5. [CrossRef]
49. Isola, P.; Zhu, J.Y.; Zhou, T.; Efros, A.A. Image-to-Image Translation with Conditional Adversarial Networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; Volume 10, pp. 1125–1134.
50. Zhang, W.; Chen, T.; Wang, J.; Yu, Y. Optimizing Top-n Collaborative Filtering via Dynamic Negative Item Sampling. In Proceedings of the 36th International ACM SIGIR Conference on Research and Development in Information Retrieval, Dublin, Ireland, 28 July–1 August 2013; Volume 4, pp. 785–788.
51. Lagoudakis, M.G.; Parr, R. Least-Squares Policy Iteration. *J. Mach. Learn. Res.* **2003**, *43*, 1107–1149.
52. David, S.; Guy, L.; Nicolas, H.; Thomas, D.; Daan, W.; Martin, R. Deterministic Policy Gradient Algorithms. *Proc. Mach. Learn. Res.* **2014**, *32*, 387–395.
53. Sutton, R.S.; McAllester, D.; Singh, S.; Mansour, Y. Policy Gradient Methods for Reinforcement Learning with Function Approximation. In Proceedings of the 12th International Conference on Neural Information Processing Systems, Denver, CO, USA, 29 November–4 December 1999; Volume 7, pp. 1057–1063.
54. Donmez, P.; Svore, K.M.; Burges, C.J.C. On the Local Optimality of LambdaRank. In Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval, Boston, Massachusetts, 19–23 July 2009; Volume 8, pp. 460–467.
55. Burges, C.J.C. *On the Local Optimality of LambdaRank*; Technical Report; Microsoft Research: Redmond, WA, USA, 2010.
56. Kingma, D.P.; Ba, J. Adam: A Method for Stochastic Optimization. *arXiv* **2014**, arXiv:1412.6980.
57. Srivastava, N.; Hinton, G.; Krizhevsky, A.; Sutskever, I.; Salakhutdinov, R. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *J. Mach. Learn. Res.* **2014**, *30*, 1929–1958.
58. Jozefowicz, R.; Zaremba, W.; Sutskever, I. An Empirical Exploration of Recurrent Network Architectures. In Proceedings of the 32nd International Conference on International Conference on Machine Learning, Lille, France, 6–11 July 2015; Volume 9, pp. 2342–2350.

59. Mahendran, A.; Vedaldi, A. Visualizing deep convolutional neural networks using natural pre-images. *Int. J. Comput. Vis.* **2016**, *13*, 233–255. [CrossRef]
60. Collobert, R.; Weston, J.; Bottou, L.; Karlen, M.; Kavukcuoglu, K.; Kuksa, P. Natural Language Processing (Almost) from Scratch. *J. Mach. Learn. Res.* **2011**, *45*, 2493–2537.
61. Pennington, J.; Socher, R.; Manning, C. Glove: Global Vectors for Word Representation. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, Doha, Qatar, 25–29 October 2014; Volume 12, pp. 1532–1543.
62. Shen, Y.; He, X.; Gao, J.; Deng, L.; Mesnil, G. A Latent Semantic Model with Convolutional-Pooling Structure for Information Retrieval. In Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management, Shanghai, China, 3–7 November 2014; Volume 10, pp. 101–110.

© 2019 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (http://creativecommons.org/licenses/by/4.0/).