COLLABORATIVE DEEP EMBEDDING VIA DUAL NETWORKS

Yilei Xiong & Dahua Lin
Department of Information Engineering
The Chinese University of Hong Kong
{xy014,dhlin}@ie.cuhk.edu.hk

Haoying Niu, Jiefeng Cheng & Zhenguo Li
Noah's Ark Lab, Huawei Technologies Co. Ltd.
{niu.haoying,cheng.jiefeng,li.zhenguo}@huawei.com

ABSTRACT

Despite the long history of research on recommender systems, current approaches still face a number of challenges in practice, *e.g.* the difficulties in handling new items, the high diversity of user interests, and the noisiness and sparsity of observations. Many of such difficulties stem from the lack of expressive power to capture the complex relations between items and users. This paper presents a new method to tackle this problem, called *Collaborative Deep Embedding*. In this method, a pair of *dual networks*, one for encoding items and the other for users, are jointly trained in a collaborative fashion. Particularly, both networks produce embeddings at multiple aligned levels, which, when combined together, can accurately predict the matching between items and users. Compared to existing methods, the proposed one not only provides greater expressive power to capture complex matching relations, but also generalizes better to unseen items or users. On multiple real-world datasets, this method outperforms the state of the art.

1 INTRODUCTION

What do consumers really want? – this is a question to which everyone wishes to have an answer. Over the past decade, the unprecedented growth of web services and online commercial platforms such as *Amazon*, *Netflix*, and *Spotify*, gives rise to a vast amount of business data, which contain valuable information about the customers. However, *"data don't speak for themselves"*. To accurately predict what the customers want, one needs not only the data, but also an effective means to extract useful messages therefrom.

There has been extensive study on recommender systems. Existing methods roughly fall into two categories, namely content-based filtering (Pazzani & Billsus, 2007) and collaborative filtering (Mnih & Salakhutdinov, 2008; Hu et al., 2008; Yu et al., 2009). The former focuses on extracting relevant features from the content, while the latter attempts to exploit the common interest among groups of users. In recent efforts, hybrid methods (Agarwal & Chen, 2009; Van den Oord et al., 2013) that combine both aspects have also been developed.

Whereas remarkable progress has been made on this topic, the state of the art remains far from satisfactory. The key challenges lie in several aspects. First, there is a large *semantic gap* between the true cause of a matching and what we observe from the data. For example, what usually attracts a book consumer is the implied emotion that one has to feel between the lines instead of the occurrences of certain words. It is difficult for classical techniques to extract such deep meanings from the observations. Second, the *cold-start* issue, namely making predictions for unseen items or users, has not been well addressed. Many collaborative filtering methods rely on the factorization of the matching matrix. Such methods implicitly assume that all the users and items are known in advance, and thus are difficult to be applied in real-world applications, especially online services.

The success of deep learning brings new inspiration to this task. In a number of areas, including image classification (Krizhevsky et al., 2012), speech recognition (Hinton et al., 2012), and natural language understanding (Socher et al., 2011), deep learning techniques have substantially pushed forward the state of the art. The power of deep networks in capturing complex variations and bridging semantic gaps has been repeatedly shown in previous study. However, deep models were primarily used for classification or regression, *e.g.* translating images to sentences. How deep networks can be used to model cross-domain relations remains an open question.

In this work, we aim to explore deep neural networks for learning the *matching relations* across two domains, with our focus placed on the matching between items and users. Specifically, we propose a new framework called *Collaborative Deep Embedding*, which comprises a pair of *dual networks*, one for encoding items and the other for users. Each network contains multiple embedding layers that are aligned with their dual counterparts of the other network. Predictions can then be made by coupling these embeddings. Note that unlike a conventional network, the dual networks are trained on two streams of data. In this paper, we devise an algorithm that can jointly train both networks using *dual mini-batches*. Compared to previous methods, this method not only narrows the semantic gap through a deep modeling architecture, but also provides a natural way to generalize – new items and new users can be encoded by the trained networks, just like those present in the training stage.

On a number of real world tasks, the proposed method yields significant improvement over the current state-of-the-art. It is worth stressing that whereas our focus is on the matching between items and users, *Collaborative Deep Embedding* is a generic methodology, which can be readily extended to model other kinds of cross-domain relations.

2 Related Work

Existing methods for recommendation roughly fall into two categories: *content-based methods* (Pazzani & Billsus, 2007) and *collaborative filtering (CF)* (Mnih & Salakhutdinov, 2008; Hu et al., 2008; Yu et al., 2009). Specifically, *content-based methods* rely primarily on feature representation of the content, in which recommendations are often made based on feature similarity (Slaney et al., 2008). Following this, there are also attempts to incorporate additional information, such as meta-data of users, to further improve the performance (McFee et al., 2012). Instead, *collaborative filtering* exploits the interaction between users and items. A common approach to CF is to derive latent factors of both users and items through matrix factorization, and measure the degree of matching by their inner products. Previous study (Ricci et al., 2011) showed that CF methods tend to have higher recommendation accuracy than content-based methods, as they directly target the recommendation task. However, practical use of CF is often limited by the *cold start problem*. It is difficult to recommend items without a sufficient amount of use history. Issues like this motivated *hybrid methods* (Agarwal & Chen, 2009; Van den Oord et al., 2013) that combine both aspects of information, which have showed encouraging improvement. Our exploration is also along this line.

Despite the progress on both family of methods, the practical performance of state-of-the-art still leaves a lot to be desired. This, to a large extent, is due to the lack of capability of capturing complex variations in interaction patterns. Recently, deep learning (Bengio, 2009) emerges as an important technique in machine learning. In a number of successful stories (Krizhevsky et al., 2012; Hinton et al., 2012; Socher et al., 2011), deep models have demonstrated remarkable representation power in capturing complex patterns. This power has been exploited by some recent work for recommendation. Van den Oord et al. (2013) applies deep learning for music recommendation. It uses the latent item vector learned by CF as ground truth to train a deep network for extracting content features, obtaining considerable performance gain. However the latent vectors for known users and items are not improved. Wang & Wang (2014) proposed an extension to this method, which concatenates both the CF features and the deep features, resulting in slight improvement.

Wang & Blei (2011) showed that CF and topic modeling, when combined, can benefit each other. Inspired by this, Wang et al. (2015) proposed *Collaborative Deep Learning (CDL)*, which incorporates CF and deep feature learning with a combined objective function. This work represents the latest advances in recommendation methods. Yet, its performance is still limited by several issues, *e.g.* the difficulties in balancing diversified objectives and the lack of effective methods for user encoding. An important aspect that distinguishes our work from CDL and other previous methods is that it encodes *both* items and users through a pair of deep networks that are jointly trained, which substantially

enhance the representation power on both sides. Moreover, the objective function of our learning framework directly targets the recommendation accuracy, which also leads to better performance.

3 COLLABORATIVE DEEP EMBEDDING

At the heart of a recommender system is *matching model*, namely, a model that can predict whether a given *item* matches the interest of a given *user*. Generally, this can be formalized as below. Suppose there are m users and n items, respectively indexed by i and j. Items are usually associated with *inherent* features, *e.g.* the descriptions or contents. Here, we use $\mathbf{x}_j$ to denote the observed features of the j-th item. However, inherent information for users is generally very limited and often irrelevant. Hence, in most cases, users are primarily characterized by their *history*, *i.e.* the items they have purchased or rated. Specifically, the user history can be *partly* captured by a *matching matrix* $\mathbf{R} \in \{0,1\}^{m \times n}$, where $\mathbf{R}(i,j) = 1$ indicates that the i-th user purchased the j-th item and gave a positive rating. Note that $\mathbf{R}$ is often an *incomplete* reflection of the user interest – it is not uncommon that a user does not purchase or rate an item that he/she likes.

3.1 DUAL EMBEDDING

To motivate our approach, we begin with a brief revisit of collaborative filtering (CF), which is widely adopted in practical recommender systems. The basic idea of CF is to derive vector representations for both users and items by factorizing the matching matrix $\mathbf{R}$. A representative formulation in this family is the *Weighted Matrix Factorization (WMF)* (Hu et al., 2008), which adopts an objective function as below:

$$\sum_i \sum_j c_{ij} (\mathbf{R}_{ij} - \mathbf{u}_i^T \mathbf{v}_j)^2 + \lambda_u \sum_i \|\mathbf{u}_i\|_2^2 + \lambda_v \sum_j \|\mathbf{v}_j\|_2^2. \tag{1}$$

Here, $\mathbf{u}_i$ and $\mathbf{v}_j$ denote the vector representations of the i-th user and the j-th item, c_{ij} the confidence coefficient of an observed entry, and λ_u, λ_v the regularization coefficients. Underlying such methods lies a common assumption, namely, all users and items must be known *a priori*. As a result, they will face fundamental difficulties when handling new items and new users.

Encoding Networks. In this work, we aim to move beyond this limitation by exploring an alternative approach. Instead of pursuing the embeddings of a given set of items and users, our approach jointly learns a pair of *encoding networks*, respectively for items and users. Compared to CF, the key advantage of this approach is that it is *generalizable by nature*. When new items or new users come, their vector embeddings can be readily derived using the learned encoders.

Generally, the items can be encoded based on their own inherent features, using, for example, an auto-encoder. The key question here, however, is *how to encode users*, which, as mentioned, have no inherent features. Again, we revisit conventional CF methods such as WMF and find that in these methods, the user representations can be expressed as:

$$\mathbf{u}_i = \operatorname*{argmin}_{\mathbf{u}} \sum_j c_{ij} \|\mathbf{R}_{ij} - \mathbf{u}_i^T \mathbf{v}_j\|^2 + \lambda_u \sum_i \|\mathbf{u}_i\|^2 = \left(\mathbf{V}\mathbf{C}_i\mathbf{V}^T + \lambda_u \mathbf{I}\right)^{-1} \mathbf{V}\mathbf{r}_i. \tag{2}$$

Here, $\mathbf{V} = [\mathbf{v}_1, \ldots, \mathbf{v}_n]$ is a matrix comprised of all item embeddings, each column for one; $\mathbf{r}_i$ is the i-th row of $\mathbf{R}$ treated as a column vector, which represents the history of the i-th user; and $\mathbf{C}_i = \mathrm{diag}(c_{i1}, \ldots, c_{in})$ captures the confidence weights.

The analysis above reveals that $\mathbf{u}_i$ is a linear transform of $\mathbf{r}_i$ as $\mathbf{u}_i = \mathbf{W}_u \mathbf{r}_i$, where the transform matrix $\mathbf{W}_u$ depends on the item embeddings $\mathbf{V}$. This motivates our idea of user encoding, that is, to use a *deep neural network* instead the linear transform above, as

$$\mathbf{u}_i = g(\mathbf{r}_i; \mathbf{W}_u), \tag{3}$$

where g denotes a nonlinear transform based on a deep network with parameters $\mathbf{W}_u$. As we will show in our experiments, by drawing on the expressive power of deep neural networks, the proposed way of user encoding can substantially improve the prediction accuracy.

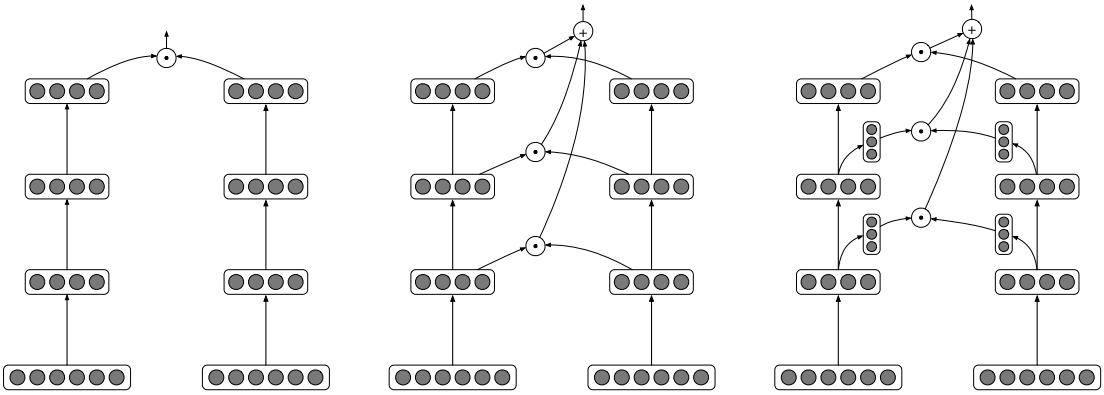

Figure 1: This figure shows three different designs of the dual networks. Here, $\odot$ indicates dot product and $\oplus$ indicates summation. (a) The basic design adopts the MLP structure for each network. (b) The multi-level design integrates the dot products of embeddings at different levels to produce the prediction. (c) In the branching design, the embeddings (except those of the top level) used in the dot products are produced by transform branches. In this way, the main abstraction paths won't be directly twisted.

Overall Formulation. By coupling an *item-network* denoted by $f(\mathbf{x}_j; \mathbf{W}_v)$ and a *user-network* g as introduced above, we can predict the *matching* of any given pair of user and item based on the inner product of their embeddings, as $\langle f(\mathbf{x}; \mathbf{W}_v), g(\mathbf{r}; \mathbf{W}_u)\rangle$. The inputs to these networks include $\mathbf{x}$, the inherent feature of the given item, and $\mathbf{r}$, the history of the given user on a set of *reference items*. With both encoding networks, we formulate the learning objective as follows:

$$\min_{\mathbf{W}_u, \mathbf{W}_v} \sum_i \sum_j c_{ij} \|\mathbf{R}_{ij} - \langle f(\mathbf{x}_j; \mathbf{W}_v), g(\mathbf{r}_i; \mathbf{W}_u)\rangle\|^2. \tag{4}$$

Here, $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_n]$ denotes the input features of all reference items. This formulation differs from previous ones in two key aspects: (1) Both users and items are encoded using deep neural networks. The learning objective above encourages the cooperation of both networks such that the coupling of both sides yield the highest accuracy. Hence, the user-network parameters $\mathbf{W}_u$ depends on the item embeddings $\mathbf{V}$, and likewise for the item-network. (2) The learning task is to estimate the parameters of the encoding networks. Once the encoding networks are learned, they encode users and items in a uniform way, no matter whether they are seen during training. In other words, new users and new items are no longer *second-class citizens* – they are encoded in exactly the same way as those in the training set.

Comparison with CDL. The *Collaborative Deep Learning (CDL)* recently proposed by Wang et al. (2015) was another attempt to tackle the *cold-start* issue. This method leverages the item features by aligning the item encoder with the embeddings resulted from matrix factorization. In particular, the objective function is given as follows:

$$\sum_{ij} c_{ij}(\mathbf{R}_{ij} - \mathbf{u}_i^T \mathbf{v}_j)^2 + \lambda_v \sum_j \|\mathbf{v}_j - f_e(\tilde{\mathbf{x}}_j, \boldsymbol{\theta})\|^2 + \lambda_n \sum_j \|\tilde{\mathbf{x}}_j - f_r(\tilde{\mathbf{x}}_j, \boldsymbol{\theta})\|^2 + \lambda_u \sum_i \|\mathbf{u}_i\|^2 + r(\boldsymbol{\theta}). \tag{5}$$

Here, a *Stacked Denoising Autoencoder (SDAE)* (Vincent et al., 2010) with parameter $\boldsymbol{\theta}$ is used to encode the items, based on $\{\tilde{\mathbf{x}}_j\}$, noisy versions of their features. Compared to our formulation, CDL has several limitations: (1) The objective is to balance the SDAE reconstruction error and the matching accuracy, which does not necessarily lead to improved recommendation. Tuning this balance also turns out to be tricky. (2) *Only items are encoded*, while the representations of the users are still obtained by matrix factorization. As a result, its expressive power in capturing user interest remains limited. (3) There are inconsistencies between known items and new ones – the embedding of known items is resulted from a tradeoff between the matching accuracy and the fidelity to SDAE features, while the embedding of new items are purely based on SDAE encoding.

3.2 NETWORK ARCHITECTURE DESIGNS

Our model consists of two networks, namely the *item-network* f and the *user-network* g. We went through a progressive procedure in designing their architectures, obtaining three different designs, from *basic design*, *multi-level design*, to *multi-level branching design*. Each new design was motivated by the observation of certain limitations in the previous version.

The basic design, as shown in Figure 1 (a) adopts the multilayer perceptron as the basic architecture, using $\tanh$ as the nonlinear activation function between layers[1]. The top layer of the item-network produces a vector $f(\mathbf{x}_j; \mathbf{W}_v)$ for each item; while that of the user-network produces a dual vector $g(\mathbf{r}_i; \mathbf{W}_u)$ for each user. During training, the loss layer takes their inner products and compares them with the ground-truth $\mathbf{R}(i, j)$.

Each layer in these networks generates a vector representation. We observe that representations from different layers are *complementary*. Representations from lower layers tend to be closer to the inputs and preserve more information; while those from higher layers focus on deeper semantics. The representations from these levels have their respective values, as different users tend to focus on different aspects of an item. Following this intuition, we reach a multi-level design, as shown in Figure 1 (b). In this design, dot products between dual embeddings at corresponding levels are aggregated to produce the final prediction.

There is an issue of the multi-level design – the output of each intermediate layer actually plays two roles. On one hand, it is the input to the next layer for further abstraction; on the other hand, it also serves as a facet to be matched with the other side. These two roles require different properties of the representations. Particularly, for the former role, the representation needs to preserve more information for higher-level abstraction; while for the latter, those parts related to the current level of matching need to be emphasized. To address this issue, we design a *multi-level branching architecture*, as shown in Figure 1 (c). In this design, a *matching branch* is introduced to transform the representation at each level to a form that is more suitable for matching. This can also be considered as learning an alternative metric to measure the *matchness* between the embeddings. As we will show in our experiments, this design can considerably improve the prediction accuracy.

4 TRAINING WITH DUAL MINI-BATCHES

A distinctive aspect of our training algorithm is the use of *dual mini-batches*. Specifically, in each iteration, B_v items and B_u users are selected. In addition to the item features and user histories, the corresponding part of the matching matrix $\mathbf{R}$ will also be loaded and fed to the network. Here, the two batch sizes B_v and B_u can be different, and they should be chosen according to the sparsity of the matching matrix $\mathbf{R}$, such that each *dual mini-batch* can cover both positive and zero ratings.

During the backward pass, the loss layer that compares the predictions with the ground-truth matchings will produce two sets of gradients, respectively for items and users. These gradients are then back-propagated along respective networks. Note that when the multi-level designs (both with and without branching) are used, each intermediate layer will receive gradients from two sources – those from the upper layers and those from the dual network (via the dot-product layer). Hence, the training of one network would impact that of the other.

The entire training procedure consists of two stages: *pre-training* and *optimization*. In the pre-training stage, we initialize the item-network with unsupervised training (Vincent et al., 2010) and the user-network randomly. The unsupervised training of the item-network allows it to capture the feature statistics. Then both networks will be jointly refined in a layer-by-layer fashion. Particularly, we first tune the one-level networks, taking the dot products of their outputs as the predictions. Subsequently, we stack the second layers on top and refine them in a similar way. Empirically, we found that this layer-wise refinement scheme provides better initialization. In the optimization stage, we adopt the SGD algorithm with momentum and use the dual mini-batch scheme presented above. In this stage, the training is conducted in *epochs*. Each epoch, through multiple iterations, traverses the whole matching matrix $\mathbf{R}$ without repetition. The order of choosing mini-batches is arbitrary and will be shuffled at the beginning of each epoch. Additional tricks such as dropout and batch normalization are employed to further improve the performance.

[1]The choice of $\tanh$ as the activation function is based on empirical comparison.

5 EXPERIMENTS

We tested our method on three real-world datasets with different kinds of items and matching relations:

1. **CiteULike**, constructed by Wang & Blei (2011), provides a list of researchers and the papers that they interested. Each paper comes with a text document that comprises both the title and the abstract. In total, it contains $5,551$ researchers (as users) and $16,980$ papers (as items) with 0.22% density. The task is to predict the papers that a researcher would like.
2. **MovieLens+Posters** is constructed based on the *MovieLens 20M Dataset* (Harper & Konstan, 2016), which provides about $20M$ user ratings on movies. For each movie, we collect a movie poster from TMDb and extract a visual feature therefrom using a convolutional neural network (Szegedy et al., 2016) as the item feature. Removing all those movies without posters and the users with fewer than 10 ratings, we obtain a dataset that contains $76,531$ users and $14,101$ items with 0.24% density. In this dataset, all 5 ratings are considered as positive matchings.
3. **Ciao** is organized by Tang et al. (2012) from a product review site, where each product comes with a series of reviews. The reviews for each product are concatenated to serve as the item content. We removed those items with less than 5 rated users and the users with less than 10 ratings. This results in a dataset with $4,663$ users and $12,083$ items with 0.25% density. All ratings with 40 or above (the rating ranges from 0 to 50) are regarded as positive matchings.

5.1 EVALUATION

The performance of a recommender system can be assessed from different perspective. In this paper, we follow Wang & Blei (2011) and perform the evaluation from the retrieval perspective. Specifically, a fraction of rating entries are omitted in the training phase, and the algorithms being tested will be used to predict those entries. As pointed out by Wang & Blei (2011), as the ratings are implicit feedback (Hu et al., 2008) – some positive matchings are not reflected in the ratings, *recall* is more suitable than *precision* in measuring the performance. In particular, we use *Recall@*M averaged over all users as the performance metric. Here, for a certain user, *Recall@*M is defined as follows:

$$\text{recall@}M = \frac{\text{the number of items a user likes in top } M \text{ recommendations}}{\text{the total number of items the user likes}}.$$

In our experiments, the value of M varies from 50 to 200.

Following Wang & Blei (2011), we consider two tasks, *in-matrix prediction* and *out-matrix prediction*. Specifically, we divide all users into two disjoint parts, *known* and *unknown*, by the ratio of 9 to 1. The *in-matrix prediction* task only considers known items. For this task, all rating entries are split into three disjoint sets: *training*, *validation* and *testing*, by the ratio $3:1:1$. It is ensured that all items in the validation and testing sets have appeared in the training stage (just that part of their ratings were omitted). The *out-matrix prediction* task is to make predictions for the items that are completely unseen in the training phase. This task is to test the performance of generalization and the capability of handling the *cold-start* issue.

5.2 COMPARISON WITH OTHER METHODS

We compared our method, which we refer to as *DualNet* with two representative methods in previous work: (1) *Weighted Matrix Factorization (WMF)* (Hu et al., 2008), a representative method for for collaborative filtering (CF), and (2) *Collaborative deep learning (CDL)* (Wang et al., 2015), a hybrid method that combines deep encoding of the items and CF, which represents the latest advances in recommendation techniques.

On each dataset, we chose the design parameters for each method via grid search. The parameter combinations that attain best performance on the validation set are used. For our *DualNet* method, we adopt a three-level branching configuration, where the embedding dimensions of each network, from bottom to top, are set to $200, 200, 50$. For *WMF*, the latent dimension is set to 300 on CDL and 450 on other datasets. For *CDL*, the best performance is attained when the structure of SDAE is configured to be $(2000, 1000, 300)$, with drop out ratio 0.1. Other design parameters of CDL are set as $a = 1.0, b = 0.01, lu = 1, lv = 10, ln = 1000, lw = 0.0005$.

Table 1: Comparison of performance on three datasets. The performances are measure with the metric *Recall@*M. We report the results where M are set to 50, 100, and 200.

	CiteULike1			*CiteULike2*		
	50	100	200	50	100	200
WMF	22.14%	32.58%	43.65%	40.45%	50.28%	59.95%
CDL	25.02%	36.57%	48.32%	39.49%	52.02%	64.41%
DualNet	**30.41%**	**41.71%**	**52.24%**	**41.26%**	**53.80%**	**65.21%**
	MovieLens			*Ciao*		
	50	100	200	50	100	200
WMF	37.14%	48.81%	60.25%	14.46%	19.66%	26.22%
CDL	38.11%	49.73%	61.00%	17.90%	24.55%	**32.53%**
DualNet	**44.95%**	**59.15%**	**72.56%**	**17.94%**	**24.58%**	32.52%

Table 2: Comparison for out-matrix predictions on CiteULike

	Recall@50	Recall@100	Recall@200
CDL	32.18%	43.90%	56.36%
DualNet	**47.51%**	**56.59%**	**66.36%**

Note that on *CiteULike*, there are two ways to split the data. One is the scheme in (Wang et al., 2015), and the other is the scheme in (Wang & Blei, 2011), which is the one presented in the previous section. Note that in the former scheme, a fixed number of ratings from each user are selected for training. This may result in some testing items being missed in the training set. To provide a complete comparison with prior work, we use both schemes in our experiments, which are respectively denoted as *CiteULike1* and *CiteULike2*.

Table 1 compares the performance of *WML*, *CDL*, and *DualNet* on all three datasets (four data splitting settings). From the results, we observed: (1) Our proposed *DualNet* method outperforms both *WML* and *CDL* on all datasets. On certain data sets, the performance gains are substantial. For example, on *MovieLens*, we obtained average recalls at 44.95%, 59.15%, and 72.56% respectively when $M = 50, 100, 200$. Comparing what CDL achieves (38.11%, $49,73\%$, and 61.00%), the relative gains are around 18%. On other data sets, the gains are also considerable. (2) The performance gains vary significantly across different datasets, as they are closely related to the *relevance* of the item features. Particularly, when the item features are pertinent to the user interest, we may see remarkable improvement when those features are incorporated; otherwise, the performance gains would be relatively smaller.

5.3 Detailed Study

We conducted additional experiments on *CiteULike* to further study the proposed algorithm. In this study, we investigate the performance of out-matrix prediction, the impact of various modeling choices, *e.g.* multi-level branching, as well as the influence of training tactics.

Out-matrix prediction. As mentioned, the out-matrix prediction task is to examine an algorithm's capability of handling new items, *i.e.* those unseen in the training stage. For this task, we compared *CDL* and *DualNet* on the *CiteULike* dataset. *WML* is not included here as it is not able to handle new items. Table 2 shows the results. It can be clearly seen that *DualNet* outperforms *CDL* by a notable margin. For example, *Recall@50* increases from 32.18% to 47.51% – the relative gain is 47.6%, a very remarkable improvement. The strong generalization performance as demonstrated here is, to a large extent, ascribed to our basic formulation, where the encoding networks uniformly encode both known and new items.

Multi-level branching. We compared three different designs presented in Section 3: *basic design*, *multi-level design*, and *multi-level branching design*. From the results shown in Table 3, we can observe limited improvement of the multi-level design over the basic one. More significant performance

Table 3: Comparison of different network architecture designs on CiteULike

	Recall@10	Recall@50	Recall@100
basic	15.86%	38.86%	51.03%
multi-level	16.89%	39.92%	51.26%
multi-level branching	17.43%	40.31%	51.78%

gains are observed when the *branching design* is introduced. This shows that the *branches* contribute a lot to the overall performance.

Noise injection. Sometimes we noticed overfitting during training *i.e.* the validation performance gets worse while the training loss is decreasing. To tackle this issue, we inject noises to the inputs, *i.e.* setting a fraction of input entries to zeros. Generally, we observed that noise injection has little effect for *Recall@*M on in-matrix predictions when $M < 30$. However, it can considerably increase the recall for large M value or *out-matrix predictions*. Particularly, on *CiteULike*, it increases in-matrix *Recall@300* from 67.3% to 71.2%, and out-matrix *Recall@50* from 38.6% to 47.5%.

Unsuccessful Tactics. Finally, we show some tactics that we have tried and found to be not working. (1) Replacing the weighted Euclidean loss with *logistic loss* would lead to substantial degradation of the performance (sometimes by up to 20%). Also, when using logistic loss, we observed severe overfitting. Rendle et al. (2009) proposed Bayesian Personalized Recommendation (BPR) which directly targets on ranking. We tested this on CiteULike with parameters tuned to obtain the optimal performance. Our experimental results showed that its performance is similar to WMF. Particularly, the Recall@50, 100, 200 for BPR are respectively 39.11%, 49.16%, 59.96%, while those for WMF are 40.45%, 50.25%, 59.95%.

(2) Motivated by the observation that positive ratings are sparse, we tried a scheme that ignores a fraction of dual mini-batches that correspond to all zero ratings, with an aim to speed up the training. Whereas this can reduces the time needed to run an epoch, it takes significantly more epochs to reach the same level of performance. As a result, the overall runtime is even longer.

6 Conclusions and Future Work

This paper presented a new method for predicting the interactions between users and items, called *Collaborative Deep Embedding*. This method uses *dual networks* to encode users and items respectively. The user-network and item-network are trained jointly, in a collaborative manner, based on two streams of data. We obtained considerable performance gains over the state-of-the-art consistently on three large datasets. The proposed method also demonstrated superior generalization performance (on out-matrix predictions). This improvement, from our perspective, is ascribed to three important reasons: (1) the expressive power of deep models for capturing the rich variations in user interests, (2) the collaborative training process that encourages closely coupled embeddings, and (3) an objective function that directly targets the prediction accuracy.

We consider this work as a significant step that brings the power of deep models to relational modeling. However, the space of deep relational modeling remains wide open – lots of questions remain yet to be answered. In future, we plan to investigate more sophisticated network architectures, and extend the proposed methodology to applications that involve more than two domains.

References

Deepak Agarwal and Bee-Chung Chen. Regression-based latent factor models. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 19–28. ACM, 2009.

Yoshua Bengio. Learning deep architectures for ai. *Foundations and trends® in Machine Learning*, 2(1):1–127, 2009.

F Maxwell Harper and Joseph A Konstan. The movielens datasets: History and context. *ACM Transactions on Interactive Intelligent Systems (TiiS)*, 5(4):19, 2016.

Geoffrey Hinton, Li Deng, Dong Yu, George E Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N Sainath, et al. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *Signal Processing Magazine, IEEE*, 29(6):82–97, 2012.

Yifan Hu, Yehuda Koren, and Chris Volinsky. Collaborative filtering for implicit feedback datasets. In *Data Mining, 2008. ICDM'08. Eighth IEEE International Conference on*, pp. 263–272. Ieee, 2008.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pp. 1097–1105, 2012.

Brian McFee, Luke Barrington, and Gert Lanckriet. Learning content similarity for music recommendation. *Audio, Speech, and Language Processing, IEEE Transactions on*, 20(8):2207–2218, 2012.

Andriy Mnih and Ruslan R Salakhutdinov. Probabilistic matrix factorization. In J. C. Platt, D. Koller, Y. Singer, and S. T. Roweis (eds.), *Advances in Neural Information Processing Systems 20*, pp. 1257–1264. Curran Associates, Inc., 2008. URL `http://papers.nips.cc/paper/3208-probabilistic-matrix-factorization.pdf`.

Michael J Pazzani and Daniel Billsus. Content-based recommendation systems. In *The adaptive web*, pp. 325–341. Springer, 2007.

Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. Bpr: Bayesian personalized ranking from implicit feedback. In *Proceedings of the twenty-fifth conference on uncertainty in artificial intelligence*, pp. 452–461. AUAI Press, 2009.

Francesco Ricci, Lior Rokach, and Bracha Shapira. *Introduction to recommender systems handbook*. Springer, 2011.

Malcolm Slaney, Kilian Weinberger, and William White. Learning a metric for music similarity. In *International Symposium on Music Information Retrieval (ISMIR)*, 2008.

Richard Socher, Cliff C Lin, Chris Manning, and Andrew Y Ng. Parsing natural scenes and natural language with recursive neural networks. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pp. 129–136, 2011.

Christian Szegedy, Sergey Ioffe, and Vincent Vanhoucke. Inception-v4, inception-resnet and the impact of residual connections on learning. *arXiv preprint arXiv:1602.07261*, 2016.

J. Tang, H. Gao, and H. Liu. mTrust: Discerning multi-faceted trust in a connected world. In *Proceedings of the fifth ACM international conference on Web search and data mining*, pp. 93–102. ACM, 2012.

Aaron Van den Oord, Sander Dieleman, and Benjamin Schrauwen. Deep content-based music recommendation. In *Advances in Neural Information Processing Systems*, pp. 2643–2651, 2013.

Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, and Pierre-Antoine Manzagol. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *The Journal of Machine Learning Research*, 11:3371–3408, 2010.

Chong Wang and David M Blei. Collaborative topic modeling for recommending scientific articles. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 448–456. ACM, 2011.

Hao Wang, Naiyan Wang, and Dit-Yan Yeung. Collaborative deep learning for recommender systems. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1235–1244. ACM, 2015.

Xinxi Wang and Ye Wang. Improving content-based and hybrid music recommendation using deep learning. In *Proceedings of the ACM International Conference on Multimedia*, pp. 627–636. ACM, 2014.

Kai Yu, John Lafferty, Shenghuo Zhu, and Yihong Gong. Large-scale collaborative prediction using a nonparametric random effects model. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pp. 1185–1192. ACM, 2009.