Preliminary Investigation of Alleviating User Cold-Start Problem in E-commerce with Deep Cross-Domain Recommender System

Hanxin Wang
Osaka University
Suita, Osaka, Japan
wang.hanxin@ist.osaka-u.ac.jp

Daichi Amagata
Osaka University
Suita, Osaka, Japan
amagata.daichi@ist.osaka-u.ac.jp

Takuya Maekawa
Osaka University
Suita, Osaka, Japan
maekawa@ist.osaka-u.ac.jp

Takahiro Hara
Osaka University
Suita, Osaka, Japan
hara@ist.osaka-u.ac.jp

Hao Niu
KDDI Research, Inc
Chiyoda-ku, Tokyo, Japan
ha-niu@kddi-research.jp

Kei Yonekawa
KDDI Research, Inc
Chiyoda-ku, Tokyo, Japan
ke-yonekawa@kddi-research.jp

Mori Kurokawa
KDDI Research, Inc
Chiyoda-ku, Tokyo, Japan
mo-kurokawa@kddi-research.jp

ABSTRACT

Many current applications use recommender systems to predict user preferences, aiming at improving user experience and increasing the amount of sales and the usage time that users spent on the application. However, it is not an easy task to recommend items to new users accurately because of the user cold-start problem, which means that recommendation performance will degrade on users with little interaction, particularly for latent users who have never used the service before. In this work, we combine an online shopping domain with information from an ad platform, and then apply deep learning to build a cross-domain recommender system based on shared users of these two domains, to alleviate the user cold-start problem. Experimental results show the effectiveness of our deep cross-domain recommender system on handling user cold-start problem. By our framework, it is possible to recommend products to users of other domain through ad distribution in a more accurate level, and to increase sales amount of online shopping.

CCS CONCEPTS

• **Information systems** → **Collaborative filtering**; **Recommender systems**; *Electronic commerce*; • **Computing methodologies** → **Neural networks**; *Learning from implicit feedback*;

KEYWORDS

Recommender Systems, Cross-domain Recommendation, Deep Learning, Implicit Feedback, E-commerce

WWW '19 Companion, May 13–17, 2019, San Francisco, CA, USA

ACM ISBN 978-1-4503-6675-5/19/05.
https://doi.org/10.1145/3308560.3316596

ACM Reference Format:
Hanxin Wang, Daichi Amagata, Takuya Maekawa, Takahiro Hara, Hao Niu, Kei Yonekawa, and Mori Kurokawa. 2019. Preliminary Investigation of Alleviating User Cold-Start Problem in E-commerce with Deep Cross-Domain Recommender System. In *Companion Proceedings of the 2019 World Wide Web Conference (WWW '19 Companion), May 13–17, 2019, San Francisco, CA, USA.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3308560.3316596

1 INTRODUCTION

In the era of information explosion, it is hard for users of online shopping services to explore all the available products in a limited amount of time. Thus, users prefer to get suggestions of products they might like to buy directly from online vendors. Because of this, recommender systems have become an important part of the e-commerce field, and have also been widely adopted by many other online services, such as online news [15], video sharing websites [1] and music streaming platforms [23].

The key component of the personalized recommender systems, which aims to model users' preferences for items based on their past interactions (such as purchase history), is collaborative filtering (CF) [19]. The most popular CF technique is matrix factorization (MF) [11, 14], which projects users and items to a shared latent space by factorizing a user-item interaction matrix. Given the immense success of deep learning on computer vision and natural language processing, some recent works have also employed deep neural networks (DNNs) for collaborative filtering (e.g. neural collaborative filtering [4]), or used neural networks to model auxiliary information (e.g., Word2vec [13]). With the powerful ability to learn a high-order nonlinear function, deep neural networks are suitable for learning complex relationship between users and items. However, both traditional MF methods and deep-learning-based methods suffer from the user cold-start problem, making it difficult to perform accurate recommendations when a new user comes, in particular when recommending items to users who have never used the service before.

It is important to note that individual users are increasingly accessing a variety of different online services. For example, a user may buy products on an online shopping site and also reads news on other websites. An effective solution for the user cold-start problem is to transfer the knowledge from relevant domains and build a cross-domain recommender system. Since online shopping sites are empirically combined with ad platforms to promote the products, it is not difficult to collect a large amount of browsing histories in the ad domain of users who have bought products on an online shopping site. In this paper, we will use the term *bridge users* to refer to users who have both the purchase history in an online shopping domain and the browsing history in an ad domain.

Motivated by the above observation, we propose a framework to build the deep cross-domain recommender system by introducing side information in other domain. Specifically, we use a method to represent users by their browsing histories on an ad platform, and learn the user-item relationship in an online shopping service, based on the bridge users' purchase histories, to improve the recommendation performance for new users of the online shopping service, who have browsing histories on the ad platform. Through ad distribution, we can even accurately recommend products on online shopping sites to latent users who have never used the service before. We apply different kinds of CF models and perform extensive experiments on a real-world dataset, to show the effectiveness of integrating deep learning and side information from other domain for alleviating the user cold-start problem.

The rest of the paper is organized as follows. We briefly review related work in Section 2. Then we describe the basic information of the real-world dataset we used and our proposed framework in Section 3. The experimental setup and results are presented in Section 4. We finally conclude the paper and suggest some future work in Section 5.

2 RELATED WORK

There has been extensive study on applying deep learning to recommender systems. In this section, we review a set of representative studies related to our work.

The pioneer study on recommender systems, which used neural networks, proposed a two-layer Restricted Boltzmann Machine (RBM) to model users' explicit ratings on items [17]. Because of the strong performance of neural networks, deep learning has become a popular choice for building collaborative recommender systems [20, 21]. The user-based AutoRec [20] aims at learning a complex hidden structure to reconstruct a user's rating, given his or her historical ratings of other items. In order to avoid learning an identity function that always returns the same value as its argument, which performs very poorly on unseen data by general autoencoders, denoising autoencoders (DAEs) [21] have been used to learn from corrupted data, which involves artificially adding some noise to original input data, in order to train a more powerful model that can generalize to unseen data. Although these methods have shown the effectiveness of neural networks for CF, they focused on explicit ratings only, which means that they could fail to learn users' preferences based on positive-only implicit data, such as purchase history and movie viewing activity, because of the lack of negative feedback.

With respect to implicit feedback, there are also some recent works that applied deep learning to improve recommendation performance [4, 26]. [26] proposed a collaborative denoising autoencoder (CDAE) for CF with implicit feedback, which adds an additional user node to the input of the model compared with DAEs. [4] presented a state-of-the-art model called neural matrix factorization (NeuMF), which combines a generalized matrix factorization (GMF) model with a multi-layer feed-forward neural network, to model user-item interactions. However, CF is generally unable to handle new users and new items, which is known as the cold-start problem.

Solutions to the cold-start problem in recommender systems have mainly focused on new items (items which have no interactions with any users). This kind of works have primarily used DNNs to model auxiliary information, such as acoustic features of musics [23], images of items [28], and textual information of items [24]. In contrast to most traditional recommender systems, which are built within a single domain, several recent works have focused on combining data from multiple domains, which is called the cross-domain recommender systems. The approach is to utilize information of a shared set of users between different domains. For example, [6] proposed a method which improves the performance of apps recommendation by transferring knowledge from relevant news reading domain. However, this approach only improves the recommendation performance on shared users, and it is unable to deal with new users. In order to alleviate the user cold-start problem, [25] introduced user-user connections in social networking services (SNSs) to learn the embeddings of new users. We see that it is impractical to reconstruct the social networks for all the bridge users in online shopping site.

Our work combines the advantages of utilizing deep learning as CF technique and applying DNNs to extract users' and items' features. We use information from an ad platform domain, to build a cross-domain recommender system which alleviates the user cold-start problem in a real-world e-commerce site. More specifically, our framework performs more accurate recommendation for new users without previous interactions with the service.

3 FRAMEWORK

In this section, we first briefly describe our overall framework. We represent users by their browsing histories in an ad platform, with no user information in the online shopping domain, and represent items by their titles, subtitles, and descriptions through Word2vec. Then we train NeuMF based on the bridge users of these two domains, and do recommendation for users who have no interaction with the online shopping domain but with sufficient browsing records in the ad platform through ads distribution. Figure 1 illustrates the methodological procedure.

Next, we provide some basic information about the real-world dataset we used, and describe existing solutions for handling implicit feedback. Then, we outline the method to represent users and items in a latent low-dimensional space. Finally, we detail the NeuMF model that we use, and how we modify the model to fit our problem.

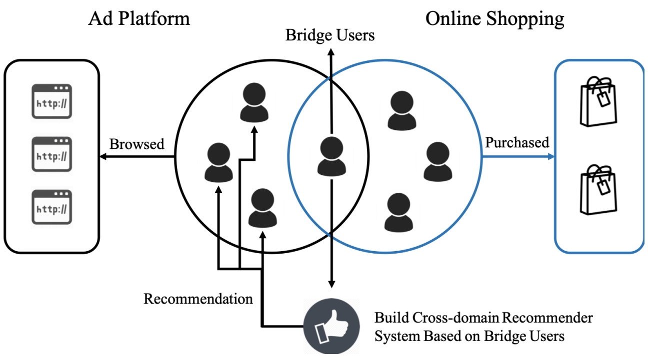

Figure 1: Methodological procedure of our framework

3.1 Learning from implicit feedback

We use a real-world online shopping dataset, an ad platform dataset, and a shared users dataset between these two domains.[1] The online shopping dataset contains users' purchase records from May 11th, 2017 to September 30th, 2017, and product information such as title, textual description, category, and price. In the ad platform dataset, there is a large amount of user browsing history over the same period as that of the online shopping dataset, and when a user accesses a web page where the ads are distributed, an access record will be stored in Ad platform. The shared users dataset is an ID mapping table of bridge users between the online shopping domain and the ad platform domain.

We build the cross-domain recommender system based on users' purchase records, which involves a kind of implicit feedback that can be defined as

$$y_{ui} = \begin{cases} 1, & \text{if user } u \text{ purchased item } i; \\ 0, & \text{otherwise.} \end{cases} \tag{1}$$

It is important to note that a value of 1 for y_{ui} does not mean that user u actually likes item i, but only indicates that there has been an interaction between u and i, or u has interest in i. Similarly, $y_{ui} = 0$ does not necessarily mean that user u does not want to purchase this item, perhaps the user simply did not find the item before, which can be viewed as missing data. Given this fact, implicit feedback contains noisy signals about users' preferences, and naturally lacks negative feedback.

In addressing the problem of recommendation with implicit feedback, several previous works formulated the problem as predicting the scores of unobserved y_{ui}, which are used for ranking the items [5, 7, 16, 26]. Thus, the problem can be seen as learning $\hat{y_{ui}} = f(u, i|\theta)$ by optimizing an objective function, where θ denotes the parameters of the predictive model and f denotes an interaction function that represents the relationship between users and items. For the objective function, we follow [5, 7] to use point-wise loss, which aims at minimizing the point-wise loss between $\hat{y_{ui}}$ and y_{ui}, and sample negative instances from unobserved entries to deal with the absence of negative feedback [4, 5].

[1]Due to the policy of the data providers, we can not reveal the data sources.

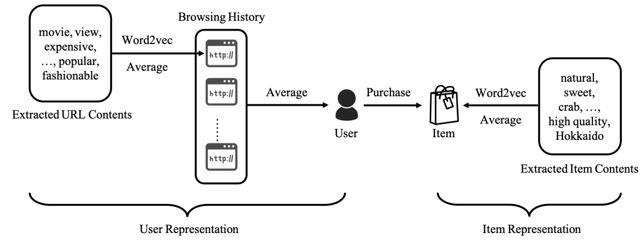

Figure 2: Process for user and item representation

3.2 Representation of Users and Items

Because our target is to recommend products in an online shopping domain to new users or people who have never used the service before, but with an abundant browsing history on an ad platform, we represent users only by corresponding browsing records in the ad platform but not information such as their profile in the online shopping domain, in order to generalize our recommender system to users in other domains. The process for creating representations of users and items is shown in Figure 2.

As different websites have different structures, and there are so many websites available, it is unrealistic to run crawlers for each website to collect contents, so we only extracted representative contents (title, keywords, and description) from web pages in the browsing records. In order to save time and reduce computational cost in the collection phase, we collect all the contents at the URL domain name level.[2] After obtaining a set of textual contents for each browsing record, we use Mecab[3] to do the Japanese text segmentation, and extract only nouns, verbs, and adjectives, filtering out meaningless words like stop words. Then we use a pre-trained Word2vec model to convert each word into a corresponding vector, and represent each browsing record as the average of these vectors. For each user, we similarly average the vectors of his/her browsing records as his/her representation.

We adopt a similar process for item representation, convert textual features such as title, subtitle, and description of each item into corresponding vectors using Word2vec.

3.3 Generalization of NeuMF

The neural matrix factorization model (NeuMF) is a state-of-the-art neural CF model that ensembles linear generalized matrix factorization (GMF) and non-linear multi-layer perceptron (MLP). Here, we first briefly introduce these two important components of the NeuMF, and then present a generalized version of NeuMF.

GMF. The GMF model is generalized from MF, which is the most popular model in the recommendation field. In general, MF takes the one-hot encoding of user ID and item ID as input, learns the latent embedding vectors, and finally calculates the inner product of the user's and item's latent vectors as the predicted score. The interaction function of MF can be defined as

[2]There is a problem with this method. It will fail to collect some important information that can distinguish different users more accurately. Although solving this problem might further improve the performance, we leave it as a future work.

[3]http://taku910.github.io/mecab/

$$y_{ui} = a_{out}(\mathbf{h}^T(\mathbf{p}_u \odot \mathbf{q}_i)), \quad (2)$$

where $\mathbf{p}_u$ denotes the user latent vector, $\mathbf{q}_i$ denotes the item latent vector, and $\odot$ denotes the element-wise product of the vectors. Also, a_{out} and $\mathbf{h}$ denote an identity function and a uniform vector with all elements as 1, respectively. If we generalize a_{out} to other activation functions such as sigmoid, and set $\mathbf{h}$ as weights that can be learnt from data, the MF model can be easily generalized to the more powerful and expressive GMF model.

MLP. The MLP is a multi-layer feed-forward neural network which is used to learn interactions between users and items. It is a more flexible model with a higher level of non-linearity than the GMF, which can only model user-item interactions linearly from the inner-product of two vectors. The MLP model under [4] is defined as

$$\begin{aligned}
\mathbf{z}_1 &= \begin{bmatrix} \mathbf{p}_u \\ \mathbf{q}_i \end{bmatrix}, \\
\mathbf{z}_2 &= a_2(\mathbf{W}_2^T \mathbf{z}_1 + \mathbf{b}_2), \\
&\ldots\ldots \\
\mathbf{z}_L &= a_L(\mathbf{W}_L^T \mathbf{z}_{L-1} + \mathbf{b}_L), \\
\hat{y}_{ui} &= \sigma(\mathbf{h}^T \cdot \mathbf{z}_L)
\end{aligned} \quad (3)$$

where $\mathbf{p}_u$ denotes the user latent vector, and $\mathbf{q}_i$ denotes the item latent vector, as in GMF. Furthermore, $\mathbf{W}_x$, $\mathbf{b}_x$ and $\mathbf{a}_x$ denote the weight matrix, the bias vector and the activation function for the x-th layer of MLP, respectively. For the $\mathbf{a}_x$, there are several available choices, such as sigmoid, tanh, and ReLU. As shown in [2], the sigmoid function can easily suffer saturation, which results in the vanishing gradient problem and halts the learning of the network. The tanh function alleviates the problems of the sigmoid function to some extent [12]. The ReLU has been proven to be non-saturated, well-suited for sparse data, and is capable of a higher convergence speed to speed up training. It can also reduce the overfitting problem of models [3]. Thus, we select ReLU as the activation function in our work, similar to several other works [4, 6, 27].

NeuMF. Figure 3 shows the structure of generalized NeuMF in our work. NeuMF combines GMF and MLP in their last hidden layers, for better modeling of user-item interactions by enabling them to mutually reinforce each other. As in [4], we allow GMF and MLP to learn separate embeddings under NeuMF, which is more flexible and expressive than sharing the same embedding between these two components. We adopt this setting and generalize NeuMF to fit our problem by using the representation of users and items described in Section 3.2 as input, in contrast to [4], which only uses the one-hot encoding of user (item) ID as input. The interaction function of generalized NeuMF is defined as follows:

$$\begin{aligned}
\mathbf{z}_{GMF} &= \mathbf{p}_u^G \odot \mathbf{q}_i^G, \\
\mathbf{z}_{MLP} &= a_L(\mathbf{W}_L^T(a_{L-1}(\ldots a_2(\mathbf{W}_2^T \begin{bmatrix} \mathbf{p}_u^M \\ \mathbf{q}_i^M \end{bmatrix} + \mathbf{b}_2)\ldots)) + \mathbf{b}_L), \\
\hat{y}_{ui} &= \sigma(\mathbf{h}^T \cdot \begin{bmatrix} \mathbf{z}_{GMF} \\ \mathbf{z}_{MLP} \end{bmatrix}),
\end{aligned} \quad (4)$$

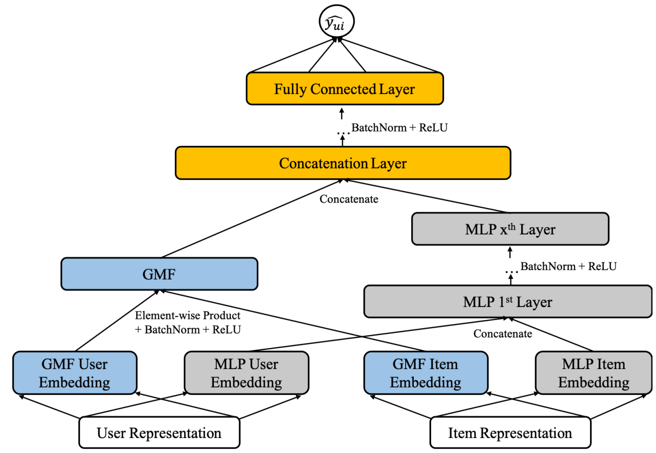

Figure 3: Structure of generalized NeuMF

where $\mathbf{p}_u^G$ and $\mathbf{p}_u^M$ denote the user embedding for GMF and MLP, respectively. Similarly, $\mathbf{q}_i^G$ and $\mathbf{q}_i^M$ denote the item embedding for these two components, and $\hat{y}_{ui}$ is the predicted result of NeuMF. We add a fully-connected layer after the concatenation layer to give NeuMF the ability to learn more complex user-item interactions. Then we apply batch normalization for each layer, in order to overcome the hard-to-converge problem, which might still be caused by ReLU, and speed up training [8].

For all the models described above, the objective function is defined as

$$L = -\sum_{(u,i)\in\mathcal{Y}\cup\mathcal{Y}^-} y_{ui}\log\hat{y}_{ui} + (1-y_{ui})\log(1-\hat{y}_{ui}), \quad (5)$$

where $\mathcal{Y}$ denotes the set of observed interactions and $\mathcal{Y}^-$ denotes the set of negative samples. This objective function is known as *logloss*.

4 EXPERIMENT

In this section, we conduct experiments to evaluate the performance of NeuMF in alleviating the user cold-start problem. Then we compare the NeuMF model with several baseline recommendation algorithms, to demonstrate the effectiveness of deep cross-domain recommender systems.

We first describe the setting of the real-world dataset that we used for training and evaluation. Next, we present the evaluation criteria, briefly review the baseline methods and the implementation details of all the methods that have been used in the experiments. Finally, we summarize the recommendation performance of the different methods, and discuss the results.

4.1 Experimental Settings

Dataset. As noted in Section 3.1, we used an ad platform dataset with users' browsing histories, and an online shopping dataset with users' purchase records from May 11th, 2017 to September 30th, 2017. We divide the purchase records into three parts: May 11th, 2017 to September 10th, 2017 for training, September 11th, 2017

Table 1: Statistics of the training set

Dataset	#interactions	#users	#items	Sparsity(%)
Training	156,287	14,659	18,511	99.942

Table 2: Statistics of the validation set and test set

Dataset	#interactions	#users	#items
Validation	12,410	9,937	1,539
Test	14,443	10,273	2,369

to September 17th, 2017 for validation, September 18th, 2017 to September 30th, 2017 for testing, and only use browsing histories from May 11th, 2017 to September 10th, 2017 for user modeling to avoid using future information to predict the past. As the original data is highly sparse, we only select users with at least five interactions (purchases) as a training set. The statistics are summarized in Table 1. In order to evaluate the recommendation performance for only new users, we filtered all the purchase records of users in the training set out of the validation and test sets. The statistics of these two datasets are shown in Table 2.

Evaluation Criteria. To evaluate the item recommendation performance, we follow the common strategy of top-K recommendation in [4, 6], to rank the target item of each interaction relative to items that the user did not interact with. Given that the items in our online shopping domain are only presented for a limited period of time, we determine that there are on average, about 1,500 items for sale each day from May 11th, 2017 to September 10th, 2017. In light of this, we randomly sample 1,499 items having no interaction with the user, and rank the 1,500 items, in contrast to [4, 6] which only sampled 99 items with no interaction for ranking. The performance of a ranked list can be evaluated by the *Hit Ratio* (HR) and *Normalized Discounted Cumulative Gain* (NDCG) [9]. The HR intuitively measures whether the target item is presented in the top-K list, while the NDCG accounts for the hit position by assigning higher scores to hits in the top ranks. As the online shopping site in our work always provide a list of recommended items with length between 5 to 10, we set K as integers from 5 to 10 for evaluation, calculate both metrics for each interaction, and present the average score.

Baselines. We compare the NeuMF employed in our work with various different methods:

- **Cosine Similarity.** This is the standard method to model user-item relationship, and is widely adopted as the final step in recommendation models.

- **ItemPop.** This is a method which ranks items based on their popularity. In our case, we judge the popularity by the sales amount of products that are still in sales. It is a non-personalized method which can be applied to cold-start users.

- **GMF [4].** As noted in Section 3.3, GMF is a generalized version of traditional MF. It is a shallow neural network model that learns user-item interactions as applying activation function to the linear combination of the element-wise product of the input vectors, which is more expressive than traditional MF.

- **DMF [27].** DMF is a state-of-the-art deep matrix factorization model that projects users and items into a low-dimensional latent space by DNN, and calculates the cosine similarity of these two vectors as the predicted result. We follow the result in [27] to build a DMF with two layers and tune the other hyper-parameters to produce the best performance.

Implementation Details. All the neural network models are implemented based on PyTorch[4]. For all the models except Cosine Similarity and ItemPop, we randomly sampled four negative samples for each positive instance. The parameters to be trained are randomly initialized from a Gaussian distribution $\mathcal{N}(0, 0.01^2)$. The optimizer is SGD with the batch size of 256, and we tested the learning rate of [0.001, 0.005, 0.01]. We also tested the number of neurons in the last hidden layer of [16,32,64,100]. For the structure of MLP model, we used a tower pattern, which halves the layer size for each successive higher layer. To optimize the hyper-parameters for each model, we tuned them based on the validation set.

4.2 Experimental Results

In this section, we report the recommendation performance of the various methods, and discuss the experimental results. Table 3 shows the performance of HR@K with K varied from 5 to 10. Table 4 shows the performance of NDCG@K in the same situation. Due to the weak performance of Cosine Similarity, it is omitted to better highlight the performance of other methods.

According to the experimental results, the shallow GMF model outperforms ItemPop in all the situations except HR@6 and HR@7, particularly for the great improvement in NDCG. We can see that deep-learning-based models, NeuMF and DMF, consistently give better performance in both metrics than the other methods, for all the values of K, with improvement up to 15.267% and 18.023% in HR@10 compared to the best shallow GMF model and ItemPop respectively. For NDCG@10, the performance of the best deep-learning-based model NeuMF is about 2.08 times better than the GMF and 3.64 times better than ItemPop. This shows the effectiveness of neural approaches in alleviating the user cold-start problem using knowledge introduced from another domain.

We can also find that NeuMF outperforms DMF and achieves 4.348% improvement in HR@10 and 1.04 times better than DMF in NDCG@10. It shows NeuMF is better at modeling the overall user-item interactions and can rank more target items into the top-10 ranked list than DMF, which is more practical because it is general to recommend a list of items to users. The results of NDCG@K also show this fact, as DMF still gives better performance than NeuMF in NDCG@5 and NDCG@6, while it has already underperformed NeuMF in HR@5 and HR@6.

5 CONCLUSION AND FUTURE WORK

In this work, we transferred information from an ad platform to an online shopping domain, based on bridge users of these two domains. Then we applied different kinds of CF models to build

[4] https://pytorch.org

Table 3: Performance of HR@K

K	5	6	7	8	9	10
NeuMF	**13.771%**	**17.725%**	**20.951%**	**23.167%**	**24.815%**	**26.165%**
DMF	12.394%	14.706%	16.769%	18.694%	20.425%	21.817%
GMF	6.003%	7.125%	8.032%	9.036%	10.005%	10.898%
ItemPop	1.461%	8.142%	8.142%	8.142%	8.142%	8.142%

Table 4: Performance of NDCG@K

K	5	6	7	8	9	10
NeuMF	0.0666	0.0807	**0.0914**	**0.0984**	**0.1034**	**0.1073**
DMF	**0.0728**	**0.081**	0.0879	0.094	0.0992	0.1032
GMF	0.0359	0.0399	0.0429	0.046	0.049	0.0515
ItemPop	0.0057	0.0295	0.0295	0.0295	0.0295	0.0295

cross-domain recommender systems, and evaluated their recommendation performance with respect to the new users of the online shopping domain. The experimental results show that deep-learning-based models far outperform other shallow models for CF, in alleviating the user cold-start problem. Moreover, NeuMF could model overall user-item interactions better than DMF in HR@K when K is between 5 and 10, which shows that the model combines linearity and non-linearity is more powerful than the pure non-linear model in the performance of our task.

In future, we will study different kinds of user-modeling methods, such as applying TF-IDF [18] to give different weights to different URLs and words, or directly converting URLs into vectors to represent users [22], in order to further improve the performance of our framework. In addition, we plan to further explore the impact of time on the recommendation performance, and investigate performance of these methods with respect to new items in the online shopping domain. Moreover, we are particularly interested in building cross-domain recommender systems through transfer learning methods like domain adaptation [10].

ACKNOWLEDGMENTS

This research is partially supported by JST CREST Grant Number J181401085.

REFERENCES

[1] Paul Covington, Jay Adams, and Emre Sargin. 2016. Deep Neural Networks for YouTube Recommendations. In *RecSys*. 191–198.
[2] Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *AISTATS*. 249–256.
[3] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. 2011. Deep sparse rectifier neural networks. In *AISTATS*. 315–323.
[4] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural collaborative filtering. In *WWW*. 173–182.
[5] Xiangnan He, Hanwang Zhang, Min-Yen Kan, and Tat-Seng Chua. 2016. Fast matrix factorization for online recommendation with implicit feedback. In *SIGIR*. 549–558.
[6] Guangneng Hu, Yu Zhang, and Qiang Yang. 2018. CoNet: Collaborative Cross Networks for Cross-Domain Recommendation. In *CIKM*. 667–676.
[7] Yifan Hu, Yehuda Koren, and Chris Volinsky. 2008. Collaborative filtering for implicit feedback datasets. In *ICDM*. 263–272.
[8] Sergey Ioffe and Christian Szegedy. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv:1502.03167* (2015).
[9] Kalervo Järvelin and Jaana Kekäläinen. 2002. Cumulated gain-based evaluation of IR techniques. *TOIS* 20, 4 (2002), 422–446.
[10] Heishiro Kanagawa, Hayato Kobayashi, Nobuyuki Shimizu, Yukihiro Tagami, and Taiji Suzuki. 2018. Cross-domain Recommendation via Deep Domain Adaptation. *arXiv:1803.03018* (2018).
[11] Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix factorization techniques for recommender systems. *Computer* 8 (2009), 30–37.
[12] Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. 1989. Backpropagation applied to handwritten zip code recognition. *Neural computation* 1, 4 (1989), 541–551.
[13] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *NIPS*. 3111–3119.
[14] Andriy Mnih and Ruslan R Salakhutdinov. 2008. Probabilistic matrix factorization. In *NIPS*. 1257–1264.
[15] Shumpei Okura, Yukihiro Tagami, Shingo Ono, and Akira Tajima. 2017. Embedding-based news recommendation for millions of users. In *KDD*. 1933–1942.
[16] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian personalized ranking from implicit feedback. In *UAI*. 452–461.
[17] Ruslan Salakhutdinov, Andriy Mnih, and Geoffrey Hinton. 2007. Restricted Boltzmann machines for collaborative filtering. In *ICML*. 791–798.
[18] Gerard Salton and Michael J McGill. 1986. Introduction to modern information retrieval. (1986).
[19] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. 2001. Item-based collaborative filtering recommendation algorithms. In *WWW*. 285–295.
[20] Suvash Sedhain, Aditya Krishna Menon, Scott Sanner, and Lexing Xie. 2015. Autorec: Autoencoders meet collaborative filtering. In *WWW*. 111–112.
[21] Florian Strub and Jeremie Mary. 2015. Collaborative filtering with stacked denoising autoencoders and sparse inputs. In *NIPS workshop on machine learning for eCommerce*.
[22] Yukihiro Tagami, Hayato Kobayashi, Shingo Ono, and Akira Tajima. 2018. Representation Learning for Users' Web Browsing Sequences. *IEICE Transactions on Information and Systems* 101, 7 (2018), 1870–1879.
[23] Aaron Van den Oord, Sander Dieleman, and Benjamin Schrauwen. 2013. Deep content-based music recommendation. In *NIPS*. 2643–2651.
[24] Hao Wang, Naiyan Wang, and Dit-Yan Yeung. 2015. Collaborative deep learning for recommender systems. In *KDD*. 1235–1244.
[25] Xiang Wang, Xiangnan He, Liqiang Nie, and Tat-Seng Chua. 2017. Item silk road: Recommending items from information domains to social users. In *SIGIR*. ACM, 185–194.
[26] Yao Wu, Christopher DuBois, Alice X Zheng, and Martin Ester. 2016. Collaborative denoising auto-encoders for top-n recommender systems. In *WSDM*. 153–162.
[27] Hong-Jian Xue, Xinyu Dai, Jianbing Zhang, Shujian Huang, and Jiajun Chen. 2017. Deep Matrix Factorization Models for Recommender Systems. In *IJCAI*. 3203–3209.
[28] Wenhui Yu, Huidi Zhang, Xiangnan He, Xu Chen, Li Xiong, and Zheng Qin. 2018. Aesthetic-based clothing recommendation. In *WWW*. 649–658.