IEEE Open Journal of the
Industrial Electronics Society

Received 28 May 2020; revised 22 June 2020 and 20 July 2020; accepted 25 July 2020. Date of publication 28 July 2020;
date of current version 17 August 2020. The review of this article was arranged by Associate Editor Dr. M. Manic.

Digital Object Identifier 10.1109/OJIES.2020.3012627

A DNN-Based Cross-Domain Recommender System for Alleviating Cold-Start Problem in E-Commerce

HANXIN WANG[1], DAICHI AMAGATA[1] (Member, IEEE), TAKUYA MAKEAWA[1] (Member, IEEE), TAKAHIRO HARA[1] (Senior Member, IEEE), NIU HAO[2], KEI YONEKAWA[2], AND MORI KUROKAWA[2]

[1] Osaka University, Suita 565-0871, Japan
[2] KDDI Research, Inc., Fujimino 356-0003, Japan

CORRESPONDING AUTHOR: DAICHI AMAGATA (e-mail: amagata.daichi@ist.osaka-u.ac.jp)

This work was supported by JST CREST under Grant J181401085.

ABSTRACT Many applications use recommender systems to predict user preferences, improve user experience, and increase the amount of sales. However, because of the cold-start problem, it is not easy to recommend items to new users accurately. Recommendation performance degrades in the case of users with little interaction, in particular latent users who have never used the service. To alleviate the cold-start problem, we develop a framework that combines an online shopping domain with information from an Ads platform. Our framework employs deep learning to build a cross-domain recommender system based on shared users in these two domains. This is the first attempt that models users based on shared users in online shopping and Ads domains for solving the user-cold start problem. We apply Word2Vec to turn textual information on users and items into latent vectors as their representations. The experimental results show the effectiveness of deep neural approaches with knowledge transferred from another domain for the cold-start problem. Textual information may contain useless information, and Word2Vec cannot capture some structural and semantic correlations between different users. Therefore, we propose R-metapath2Vec to enhance user modeling and use the Stacking model to integrate these two kinds of user representations. The experimental results demonstrate the effectiveness of our integration model: our framework can recommend products to users of another domain through Ads distribution in a more accurate level.

INDEX TERMS Cross-domain recommender system, deep neural networks, e-commerce, heterogeneous information network.

I. INTRODUCTION

It is hard for online shopping users to explore more than thousands items and make a better choice from them in a limited time. Hence, users prefer to get suggestions of items they may buy from online vendors. Recommender systems, which can help users find items quickly, have been playing an important role in e-commerce. For example, Wang *et al.* [1] showed that recommendation contributes to both revenue and traffic in Taobao,[1] which is the largest online consumer-to-consumer platform in China.

The key part of personalized recommender systems, which model users' preferences for items based on their past interactions, such as purchase records, is collaborative filtering (CF) [2]. It is based on a simple assumption that, if users have interacted with some items in the past, they are likely to interact with other similar items in the future. One of the most popular CF techniques is matrix factorization (MF) [3], [4], which projects users and items into a shared latent space by factorizing the user-item interaction matrix. Further predictions are then made by calculating the inner product of their latent vectors. The item-based CF is also a famous method [2] which calculates the item-item similarity by using user-item interactions. Although classical MF and item-based CF are

[1] [Online]. Available: https://www.taobao.com/

employed in both academia and industry, it is well known that recommendation performance can be hindered by the inner product and the underlying representation (e.g. one-hot encoding). Specifically, inner product, which simply calculates the linear combination of latent features, may not be able to capture the complexity of user-item interaction. One-hot encoding incurs sparsity problems because of its high dimension, and it cannot accurately reflect the relationships between users and items.

Some recent works have applied deep neural networks (DNNs) to recommendation tasks. With their powerful abilities to model a high level of non-linearity, [5] and [6] proposed methods that leverage a multi-layer feed-forward neural network to learn the user-item interaction function. The pioneer work on recommender systems, which used neural networks to learn the interaction function, proposed a two-layer Restricted Boltzmann Machine (RBM) to model users' explicit ratings on items [7]. Inspired by the RBM model, many recommendation models [5], [8], [9] used the multi-layer perceptron (MLP) as neural extensions to add nonlinear transformation to traditional approaches. The work in [8] replaced the inner product with a multi-layer feed-forward neural network and proposed neural network matrix factorization (NNMF). In [9], Guo *et al.* seamlessly integrated the factorization machine (FM) [10] and MLP, rendering an end-to-end model called DeepFM. For object representation, inspired by Word2Vec [11], which learns the distributed representations of words and phrases with the skip-gram model, [12] proposed Item2Vec to learn the dense embedding for items from item sequences with which users have interacted. Several works [1], [13] integrated graph embedding methods and side information (e.g. categories of items, brands of items) to improve recommendation performance and alleviate the cold-start problem. However, these methods still suffer from the cold-start problem, which is a significant challenge in the recommendation field. The cold-start problem means that accurate recommendation for users with very little historical information is difficult. In particular, recommending items to new users who have never used the service before is hard.

Efforts have been made to improve the effectiveness of recommender systems for new users. A simple strategy is to recommend the most popular items. However, this strategy is not a personalized recommendation method, so it is highly dependent on trends and cannot provide recommendation according to users' preferences. Another strategy is to recommend items based on user's profile, such as gender, age, and income [14]. Unfortunately, the profile information on new users is not always available in e-commerce, because such information requires their additional efforts (inputs).

It is important to note that users usually participate in different services simultaneously. An effective solution for the cold-start problem is to transfer the knowledge from relevant domains to the target domain and build a cross-domain recommender system [15]. Some relevant works [16], [17] combined a traditional CF method with social network analysis to integrate user-item interactions from the recommendation domain and user-user relationships from the social domain. However, it is impractical to collect sufficient social information of users in e-commerce, because connections between e-commerce services and social network services are not guaranteed. Fortunately, as online shopping websites are always combined with digital advertising (Ads) platforms, it is not difficult to obtain a large number of browsing records of users in the Ads domain who have also purchased items on the online shopping website. It is therefore possible to transfer the domain knowledge from the Ads domain to the online shopping domain, which effectively deals with the cold-start problem in e-commerce. We use *bridge users* to denote users who have both purchase records in the online shopping domain and browsing records in the Ads domain.

Motivated by the above observations, we develop a framework specifically designed for alleviating the cold-start problem in e-commerce. We build a cross-domain recommender system by introducing side information from the Ads domain. Moreover, we use Word2Vec [11] to represent users by using their browsing records in the Ads platform, so that new users can be linked with old users of the online shopping website. For modeling the user-item interaction function in the online shopping domain, we train a deep neural model based on bridge users' purchase records to improve the recommendation performance for new users with sufficient browsing records in the Ads domain. We conduct extensive experiments on a real-world dataset and test different kinds of collaborative filtering models. The experimental results show the effectiveness of integrating deep learning models and side information from other domains in alleviating the cold-start problem.

The preliminary version of this paper [18] has the above contents. We find that there are still two issues with modeling user behaviors by Word2Vec: (1) We may lose important information on the browsing records, as some websites block web scraping and some websites contain only useless information, e.g., *404 Not Found* error, and (2) Word2Vec cannot represent some semantic relationships between users. Therefore, we further devise a representation learning method based on Metapath2Vec [19]. This is used to improve heterogeneous network mining tasks, e.g., node classification, to generate the latent representation of users. We combine the representation learned by Word2Vec and Metapath2Vec. Our experimental results demonstrate the effectiveness of integrating the two kinds of representations.

The rest of this paper is organized as follows. We introduce some preliminaries of our work in Section II. We present our framework in Section III, and the experimental results are presented in Section IV. Finally, the conclusion and future work are given in Section V.

II. PRELIMINARY

This section defines the terms used in this paper.

Domain. A domain is defined as a particular field of thought, activity, or interest. This is illustrated as a set of entities sharing certain characteristics that can be exploited by recommender systems. These characteristics can be item

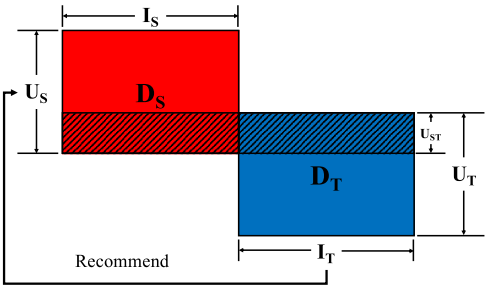


FIGURE 1. Cross-domain recommendation and bridge users.

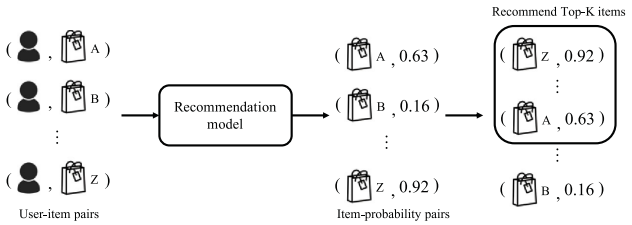


FIGURE 2. Schematic illustration of ranking prediction.

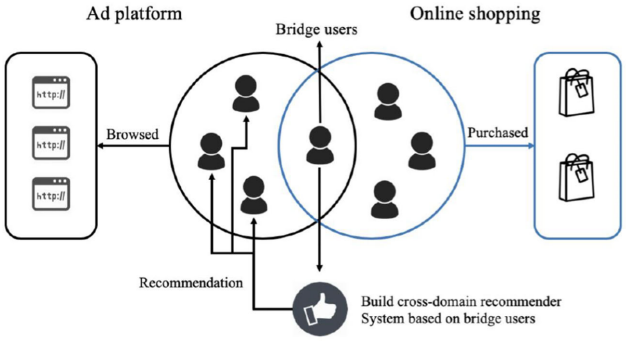


FIGURE 3. Overview of the framework.

attributes, user profiles, and ratings. In the literature on cross-domain recommendation, researchers have considered the distinct notion of a domain. In [17], Wang *et al.* treated a touristic application as one domain and a social networkins service such as Twitter[2] as another domain. [20] viewed news reading records as one domain and user-book ratings on Amazon as the second domain. In this paper, we define the Ads platform as the source domain D_S and an online shopping website as the target domain D_T.

Bridge Users. Given a set of users in the target domain (i.e., U_T) and a set of users in the source domain (i.e., U_S), the bridge users are defined as the users overlapping between the target and source domains, as shown in Figure 1. A set of bridge users U_{ST} is $U_S \cap U_T$. In our work, bridge users are simultaneously involved in both the Ads domain and the online shopping domain. They act as a bridge to propagate user-item interaction across the domains.

Cross-domain Recommendation. Different from a single domain recommendation, cross-domain recommendation considers data from multiple domains. In this paper, we consider two domains. The task of cross-domain recommendation in our work is illustrated in Figure 1, where I_T and I_S denote the set of items in D_T and D_S, respectively. The task is to recommend items I_T to non-bridge users in D_S (i.e., $U_S - U_{ST}$), who are characterized as new users of D_T. Let A_T and A_S be a set of associated attributes of users and items in D_T and D_S, respectively. The user-item interaction is denoted as $Y = \{y_{ui}\}$, which is a binary implicit feedback. Formally, our task is defined as:

- Input: an online shopping domain with $\{U_T, I_T, Y, A_T\}$, an Ads domain with $\{U_S, A_S\}$, and U_{ST} where $U_{ST} \neq \varnothing$.
- Output: a personalized ranking function $f : (U_S, I_T) \rightarrow \mathbb{R}$ that maps each pair of users and items, i.e., (u, i), where u is the user of the Ads domain and i is the item in the online shopping domain, to a real value.

Ranking Prediction. The evaluation of recommender systems is typically categorized into two broad types: *rating prediction* and *ranking prediction*. Rating prediction is commonly associated with explicit feedback and the root mean square error (RMSE) as its objective function, while ranking prediction is a useful approach to implicit feedback such as purchase and click and is more suitable for our work. A schematic illustration of ranking prediction is shown in Figure 2. For each user, besides the item with which user has interacted, i.e., the target item, ranking prediction considers all other items with which the user has not interacted, and then uses the recommendation model to predict the probability of future interaction between the user and items. Finally, it ranks items based on the predicted results and evaluates the recommendation accuracy by the hit position of the target item. In practice, the recommender system recommends the top-K items in the ranking list to the user.

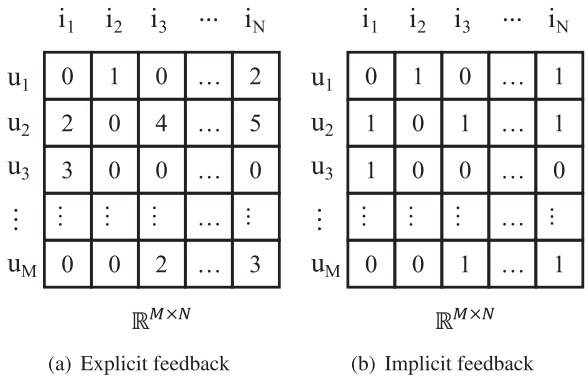
III. OUR FRAMEWORK

To start with, we briefly introduce an overview of the framework in Figure 3. We represent users by their browsing records on the Ads platform, without using any user information, such as the user profile in the online shopping domain. For item representation, we utilize the textual information of each item, including the title, sub-title, and other textual description. After generating the representations of users and items, we train the state-of-the-art neural collaborative filtering model on bridge users of these two domains. We can give accurate recommendations to users who have no interaction with the online shopping domain but with sufficient browsing records in the Ads domain.

A. LEARNING FROM IMPLICIT FEEDBACK

The input of recommender systems is broadly categorized into two different types: *explicit feedback* and *implicit feedback*. High quality explicit feedback, including the numerical values that indicate user preferences, is the most convenient input

[2][Online]. Available: https://twitter.com/



(a) Explicit feedback (b) Implicit feedback

FIGURE 4. Interaction matrices of explicit and implicit feedback.

for recommendation tasks. For example, Netflix and Yelp let users express their preferences by a 5-star rating system. The rating is between 1 star (bad experience) and 5 stars (top-quality). The corresponding user-item interaction matrix is formalized as in Figure 4(a). However, explicit feedback is not always available. In many situations, therefore, recommender systems infer user preferences from implicit feedback, e.g., purchase records, browsing records, or even mouse click records.

In our work, we build a cross-domain recommender system on a real-world dataset that contains both online shopping and Ads data. The e-commerce dataset contains purchase records and item information, such as title, textual description, and category, whereas the other contains the browsing records in the Ads domain. Browsing records are stored in the Ads platform when a user accesses the web page where the Ads are distributed. The real-world dataset also includes an ID mapping table of bridge users between the online shopping domain and the Ads domain. The recommendation model is learned based on the purchase records, which is a kind of implicit feedback that is defined as

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

where y_{ui} represents whether a user u has purchased an item i. This is the entry in the user-item interaction matrix $\mathbb{R}^{M\times N}$ with M users and N items, as shown in Figure 4. It is important to note that a value of 1 for y_{ui} does not necessarily mean that the user u actually likes the item i. It indicates only that an interaction has happened between u and i, or reflects that u is interested in i. For instance, some items may have been purchased because they are on a flash sale, or perhaps the user is disappointed with the item but he/she does not give extra feedback after shopping. Besides, different from the ratings in explicit feedback, $y_{ui} = 0$ in implicit feedback does not necessarily mean that u does not want to purchase this item. We see that implicit feedback is inherently noisy for inferring user preferences for items, and it naturally lacks negative feedback.

To handle the problem of recommendation with implicit feedback, many works formulated it as predicting the scores

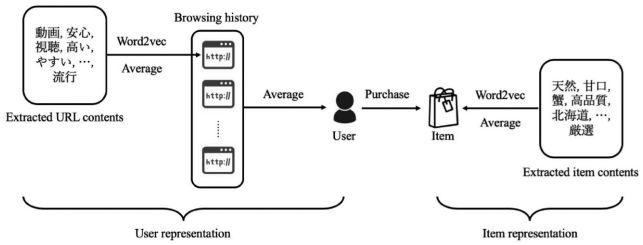


FIGURE 5. Process for user and item representations.

of unobserved y_{ui} in the user-item interaction matrix $\mathbb{R}^{M\times N}$, which are used for ranking the items [21]–[24]. Therefore, the problem can be seen as learning $\hat{y_{ui}} = f(u, i|\theta)$ by optimizing an objective function, where $\hat{y_{ui}}$ is the predicted result of y_{ui}, θ is a parameter of the predictive model, and f is an interaction function that represents the relationships between users and items. For the objective function, we follow works [22], [23] to use the point-wise loss. This aims at minimizing the cross entropy loss between $\hat{y_{ui}}$ and y_{ui}. For the negative feedback problem, we sample k negative instances from the unobserved entries of user u for each user-item pair (u, i) to deal with the absence of negative feedback [5], [22]. This is because treating all unobserved entries $y_{ui} = 0$ as negative data is time-consuming for model training.

B. REPRESENTATION OF USERS AND ITEMS

1) WORD2VEC APPROACH

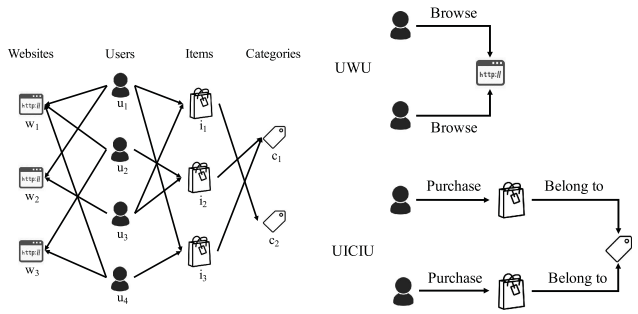
Our task is to recommend items on the online shopping website to new users of this domain with sufficient browsing records on the Ads platform. Therefore, we represent users only by corresponding browsing records in the Ads platform, instead of information such as user profiles in the online shopping domain. This is because we can generalize our recommender system to users in other domains. An overview of creation of the user and item representations is shown in Figure 5. In this section, we take the creation of user representation as an example to illustrate the whole process.

The first step is to collect the textual information, which can reflect the preferences and characteristics of bridge users, from the Ads domain. However, there is a challenge in the web scraping phase. As there are millions of websites nowadays and different websites are usually constructed based on different structures or templates, it is unrealistic to run a specialized crawler for each website to collect the contents. Therefore, we extract textual information only from the HTML tags, such as *title*, *keywords*, and *description*. For most websites, these contents are general and representative. To further reduce the computational cost and save time in this step, we collect all textual information at the domain level of each URL.

After obtaining a set of textual contents for each browsing record, we extract only the nouns, verbs, and adjective, to filter noises, because there are many meaningless words such as stop words. In the next step, we use the Word2Vec model to turn each word into its corresponding vector. The Word2Vec

model we have applied in our task is trained based on the skip-gram with negative sampling (SGNS), which is a neural word embedding method [11]. By this model, the latent representations of words can carry semantic meanings. Finally, we represent each browsing record as the averaged vector of all the word vectors that belong to the website. Similarly, for each user, we average the vectors of his/her browsing records as his/her representation.

We adopt a similar process for item representation. To create item representation, we use textual information such as title, subtitle, and description of each item and turn them into vectors by Word2Vec.



(a) Example of HIN (b) Examples of metapath

FIGURE 6. The examples of HIN and metapath.

2) ENHANCING USER MODEL

Although using Word2Vec for user representation enables knowledge to be transferred from bridge users to new users in the Ads domain, it has two issues:

- Because some websites block web scraping or contain only useless information such as *404 Not Found*, we may lose important information on these browsing records.
- It is difficult for Word2Vec to capture some semantic relationships, e.g., co-view relationships between users.

To solve these issues and further improve the recommendation performance, we use the Metapath2Vec method [19]. This method can capture both the semantic and structural correlations between different types of objects, so it is a good choice for user representation learning. Metapath2Vec maps the word-context concept in a corpus into a network, and it utilizes metapath-based random walks to generate node sequences and applies a heterogeneous skip-gram to learn the latent representation of a node v in the heterogeneous information network (HIN). However, because there is a great difference between the numbers of different types of nodes (e.g. users and websites) in our task, the heterogeneous skip-gram will be biased toward nodes with a dominant number, and it cannot learn effective representations for other types of nodes in the node sequences. Therefore, we propose a revised version of Metapath2Vec, *R-metapath2Vec*, which leverages the metapath-based random walks and the traditional skip-gram model to enhance user modeling. For ease of presentation, we first define heterogeneous information network and metapath. After that, we introduce R-metapath2Vec.

Heterogeneous Information Network (HIN) is a kind of information network that contains multiple types of nodes and multiple types of links. It is a graph $G = (V, E)$ with a node set V and a link set E. A node $v \in V$ and a link $e \in E$ are associated with a node type mapping function $\phi(v) : V \to T_V$ and a link type mapping function $\psi(e) : E \to T_E$, respectively. Let T_V and T_E respectively denote the sets of node and link types, where $|T_V| + |T_E| > 2$.

Take the information network of the Ads platform and the online shopping website as an example. As shown in Figure 6(a), the HIN consists of multiple types of nodes, such as bridge user (U), website (W), item (I), and category of items (C). There are also different types of links between nodes to represent different relationships. For example, a user-item link indicates the purchase behavior, and a user-website link indicates that the user viewed the website.

Metapath. A metapath p is defined based on a HIN $G = (V, E)$. Let VT and $T_V = \{VT_i\}_{i=1}^{m}$ denote the type of node and the whole set of node types, respectively. Similarly, we use R and $T_R = \{R_i\}_{i=1}^{n}$ to denote the different types of relations and the whole relation set, respectively. The metapath is formally defined as $p = VT_1 \xrightarrow{R_1} VT_2 \xrightarrow{R_2} VT_3 \cdots \xrightarrow{R_l} VT_{l+1}$. This represents a composite relation $R = R_1 \odot R_2 \odot R_3 \odot \cdots \odot R_{l+1}$ between the nodes with type VT_1 and VT_{l+1}, where $\odot$ is the composite operator.

As shown in Figure 6(b), the metapath *User-Website-User* (UWU) indicates the co-browse relationship between these two users, meaning that they have browsed the same website. Moreover, *User-Item-Category-Item-User* (UICIU) path indicates that these two users have purchased items belonging to the same category. Accordingly, different metapaths usually represent different relations between nodes in HIN and convey different semantics.

R-metapath2Vec. Given a HIN $G = (V, E)$ and a metapath $p = VT_1 \xrightarrow{R_1} VT_2 \xrightarrow{R_2} VT_3 \cdots \xrightarrow{R_l} VT_{l+1}$, the transition probability in step t is defined as:

$$P(v_{t+1}|v_t, p) = \begin{cases} \frac{1}{|N^{R_{t+1}}(v_t)|}, & (\text{if } (v_{t+1}, v_t) \in E, \phi(v_{t+1}) = VT_{t+1}) \\ 0, & (\text{otherwise}) \end{cases}$$

where $N^{R_{t+1}}(v_t)$ is a set of neighbors of a node v_t with the type VT_{t+1}. A random walk follows the pattern of the metapath until it reaches the pre-defined length. The metapath-based random walk makes sure the semantic relationships between different nodes, and the structural features of the network can be incorporated into a skip-gram for representation learning.

Figure 7 represents the heterogeneous information network of our cross-domain recommender system. The metapath *User-Item-Category-Item-User* represents the relationship that two users have purchased the items belonging to the same category. In addition, the metapath *User-Device-User* indicates the relationship that two users have used smartphones of the same brand (e.g., iPhone, Samsung, etc.), which can

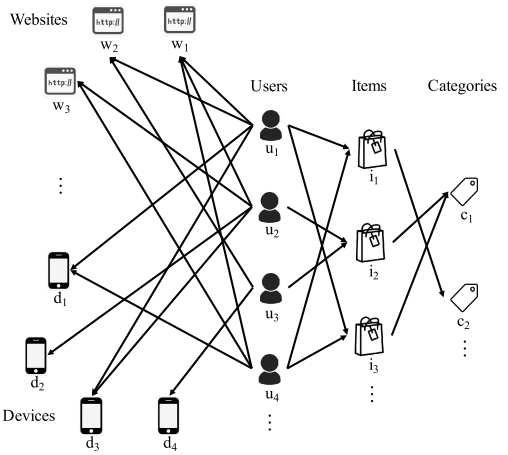


FIGURE 7. HIN of our cross-domain recommender system.

be parsed from the *User-agent* information in the Ads domain. As a specific example, we take *Website-User-Website (WUW)*, which indicates that these websites are browsed by the same user. We can generate a node sequence s by starting with the website node w_2, then the walker will move toward the user node u_1 which is linked to w_2. Following the pattern of WUW, the sequence s is generated as $w_2 \longrightarrow u_1 \longrightarrow w_1 \longrightarrow u_4 \longrightarrow w_3 \cdots \longrightarrow w_1$.

Objects such as websites, smartphones, and browsers may also be interacted with many other new users, and these objects are easier to transfer from bridge users to new users. Because of the imbalance between the numbers of users and the other objects, it is difficult to learn effective representations simultaneously for all types of nodes in the sequences. Therefore, we select only metapaths starting with the related object type and followed by the user type, and then we filter the nodes of the user type out of the node sequences. For example, the sequence s will become $w_2 \longrightarrow w_1 \longrightarrow w_3 \cdots \longrightarrow w_1$ in this way. We treat these node sequences as input of the skip-gram model to learn the latent representations of users. An overview of R-metapath2Vec is illustrated in Figure 8, and the pseudo code of R-metapath2Vec is listed in Algorithm 1. Although applying R-metapath2Vec to item embedding may further improve the recommendation performance, we focus on user representation enhancement in this paper, and we leave it as a future work.

C. GENERALIZATION OF NEUMF

Neural matrix factorization (NeuMF) is a state-of-the-art neural collaborative filtering model that integrates linear generalized matrix factorization (GMF) and non-linear multi-layer perceptron (MLP). We introduce these components before presenting the details of the NeuMF model. Finally, we provide the generalized version of NeuMF.

GMF is a model generalized from matrix factorization (MF), which is one of the most popular models in the recommendation field and has been investigated extensively in

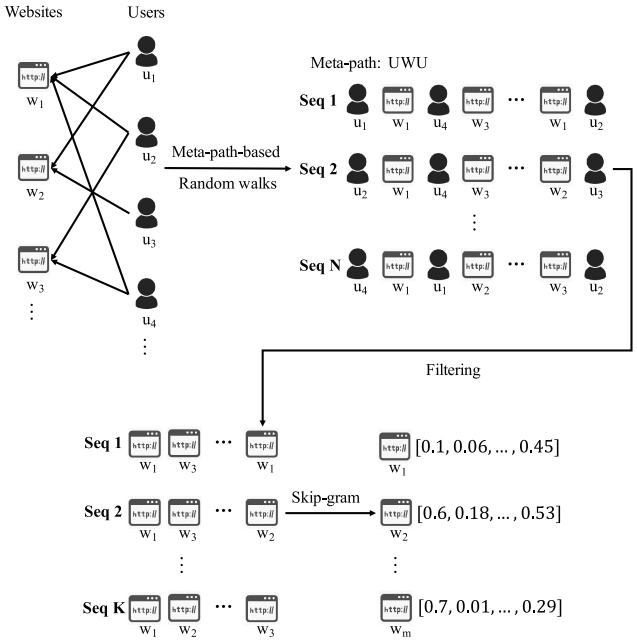


FIGURE 8. Overview of R-metapath2Vec.

Algorithm 1: R-metapath2Vec Algorithm.

Input: A HIN $G = (V, E)$, a metapath p, a set of nodes with starting type V_t, the number of walks per node w, the length of each walk l, the embedding dimension d, the size of context window k

Output: The latent representation of nodes $X \in \mathbb{R}^{|V_t| \times d}$

Initialize X and the list for storing node sequences S
for $i = 1$ *to* w **do**
  for *each* $v \in V_t$ **do**
    $mp[1] = v$
    for $j = 2$ *to* l **do**
      Move to node u based on $P(v_{t+1}|v_t, p)$
      $mp[j] = u$
    $k = 1$
    for $j = 1$ *to* $ceiling(\frac{l}{2})$ **do**
      $s[j] = mp[k]$
      $k_{new} = k_{old} + 2$
    Add s to list S
for *each* $s \in S$ **do**
  for $i = 1$ *to* $ceiling(\frac{l}{2})$ **do**
    $v = s[i]$
    for $j = max(0, i - k)$ *to* $min(i + k, ceiling(\frac{l}{2}))$ *and* $i \neq j$ **do**
      $c = s[j]$
      Update X by (v, c) according to skip-gram

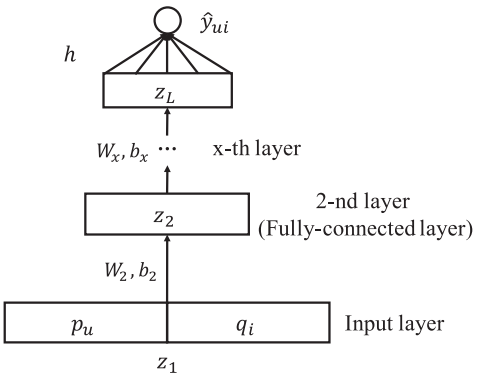


FIGURE 9. Structure of MLP.

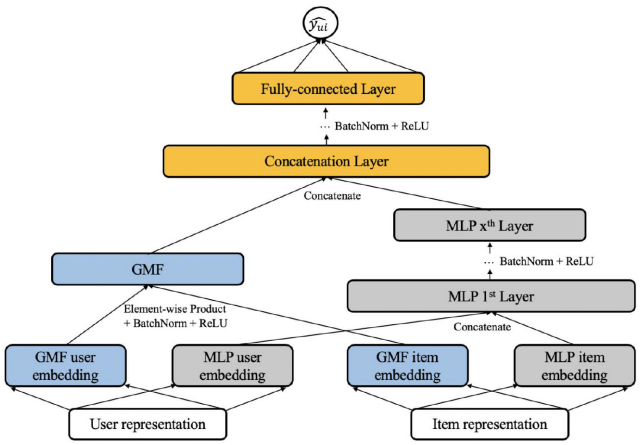


FIGURE 10. Structure of generalized NeuMF.

literature. In general, MF takes the one-hot encoding representations of user ID and item ID as input, learns the latent vectors, then calculates their inner product (predicted score). The interaction function of MF is

$$\hat{\mathbf{y}_{ui}} = a_{out}(\mathbf{h}^T(\mathbf{p}_u \odot \mathbf{q}_i)),$$

where $\mathbf{p}_u$ and $\mathbf{q}_i$ are the latent vectors of user u and item i, respectively. Note that $\odot$ is the element-wise product of vectors, i.e., the inner product. In addition, a_{out} and $\mathbf{h}$ are an identity function and a uniform vector with all elements as 1, respectively. If we generalize a_{out} to other activation functions like *sigmoid* or *tanh*, and set $\mathbf{h}$ as a sequence of weights that can be learned from data, MF can be easily generalized to a more powerful and expressive version, GMF.

MLP is a multi-layer feed-forward neural network and can learn the complex and non-linear interactions between users and items, rather than only uses the linear combination of element-wise products. Recommendation models always adopt two pathways, i.e., user representation and item representation, to model the interaction between users and items. It is hence intuitive to combine the characteristics of users and items by concatenating their representations as one latent vector [5], [25]. The MLP model under [5] is defined as:

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

where $\mathbf{W}_x$, $\mathbf{b}_x$, and a_x are a weight matrix, a bias vector, and an activation function for the x-th layer of MLP, respectively.

The structure of MLP for the recommendation task is shown in Figure 9. As for the activation function $\mathbf{a}_x$, there are many available choices, such as *sigmoid*, hyperbolic tangent (*tanh*) and rectified linear unit (*ReLU*). According to a work on activation function [26], the sigmoid function restricts the value of each neuron to be in (0, 1). This may limit the performance of the neural network, and it can easily suffer from saturation, incurring the vanishing gradient problem, i.e., stopping network learning. The tanh function seems a better choice than sigmoid, but it alleviates the problems of sigmoid to some extent [27]. As shown in [28], ReLU has been proved to be non-saturated, well-suited for sparse data, and has a higher convergence speed. It can also reduce the overfitting problem. Therefore, we select ReLU as the activation function in our work.

NeuMF. Figure 10 shows the structure of generalized NeuMF. The NeuMF model combines GMF and MLP in their last hidden layers. Such a design allows NeuMF to better model the complex user-item interaction by enabling GMF and MLP to mutually reinforce each other. The authors in [5] allowed GMF and MLP to learn separate latent embeddings, 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 task by using the representations of users and items as input, while [5] uses only the one-hot encoding representation of user/item ID as input. The interaction function of our NeuMF is as follows:

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

where $\mathbf{p}_u^G$ and $\mathbf{p}_u^M$ are the user embedding for GMF and MLP, respectively. Similarly, $\mathbf{q}_i^G$ and $\mathbf{q}_i^M$ are the item embedding for these two parts, and $\hat{y}_{ui}$ is the predicted score of NeuMF. We add a fully-connected layer after the concatenation layer to provide NeuMF with the ability to learn more complex user-item interactions. Then we apply batch normalization for each layer to overcome the hardness of convergence that may still be caused by ReLU, and to speed up training [29]. For all the models mentioned above, the objective function to minimize

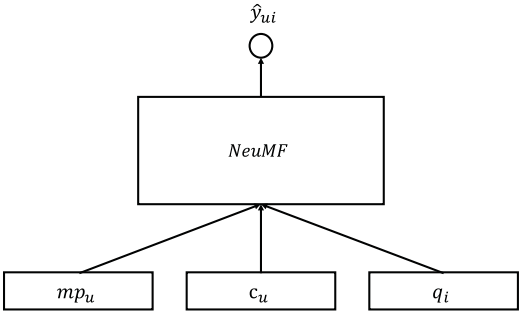


FIGURE 11. Three-ways integration model.

for the recommendation task is:

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

where $\mathcal{Y}$ is the set of observed interactions and $\mathcal{Y}^-$ is the set of negative samples, which are uniformly sampled from the unobserved interactions for each training instance. This objective function is known as *logloss* and it can be optimized by stochastic gradient descent (SGD).

D. INTEGRATION OF WORD2VEC AND R-METAPATH2VEC

So far we have developed two kinds of user representation methods, Word2Vec, which captures the semantic features from the textual information, and R-metapath2Vec, which reflects both the semantic and structural correlations between different types of nodes in the HIN. Then we have a question: how can we integrate these methods so that they can reinforce each other to further improve the recommendation performance for the cold-start users?

The simplest solution is to add these two representations together as the embedding vector of a new user. However, simply adding two vectors may introduce extra noisy signals w.r.t. user preferences. It also implies that the vectors from these two methods must be of the same size, but this may fail to achieve the best performance when the optimal embedding size of the two methods varies a lot. We therefore propose two methods based on the NeuMF model, *three-ways integration* and *stacking*.

Three-ways Integration. Since NeuMF adopts two pathways to model users and items, it is intuitive to extend the pathway of user modeling as two different branches, which take the Word2Vec user representation and R-metapath2Vec user representation as inputs, respectively. Three-ways integration can model the interactions between item latent features and two different user representations simultaneously. That is, it can utilize both the features learned by Word2Vec and R-metapath2Vec in recommendation tasks.

The structure of the three-ways integration model is shown in Figure 11, and the interaction function is similar to NeuMF:

$$\mathbf{z}_{GMF} = \mathbf{mp}_u^G \odot \mathbf{c}_u^G \odot \mathbf{q}_i^G,$$

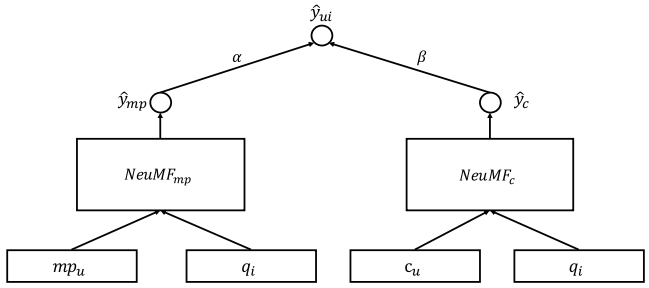


FIGURE 12. Stacking integration model.

$$\mathbf{z}_{MLP} = a_L\left(\mathbf{W}_L^T(a_{L-1}(\ldots a_2(\mathbf{W}_2^T\begin{bmatrix}\mathbf{mp}_u^M\\ \mathbf{c}_u^M\\ \mathbf{q}_i^M\end{bmatrix}+\mathbf{b}_2)\ldots))+\mathbf{b}_L\right),$$

$$\hat{y}_{ui} = \sigma\left(\mathbf{h}^T\cdot\begin{bmatrix}\mathbf{z}_{GMF}\\ \mathbf{z}_{MLP}\end{bmatrix}\right),$$

where $\mathbf{mp}_u^G$, $\mathbf{c}_u^G$, and $\mathbf{q}_i^G$ are the R-metapath2Vec user embedding, Word2Vec user embedding, and item embedding for GMF, respectively. Similarly, $\mathbf{mp}_u^M$, $\mathbf{c}_u^M$, and $\mathbf{q}_i^M$ are the corresponding latent embeddings for MLP part. Moreover, $\hat{y}_{ui}$ refers to the predicted score of three-ways integration model. The objective function is set as *logloss* and is optimized by SGD.

Stacking is a model-level integration method for the combination of Word2Vec representation and R-metapath2Vec representation. Different from three-ways integration, stacking trains two NeuMF models simultaneously and integrates the outputs of the NeuMF models given two weights α and β, which can be learned automatically from the data. An image of stacking is illustrated in Figure 12. NeuMF$_{mp}$ (left side) aims at capturing the user-item relationships based on metapath features, whereas NeuMF$_c$ (right side) utilizes the textual features for interaction modeling. The final interaction result $\hat{y}_{ui}$ is composed of the outputs of the two parts, and the importance of each part is dependent on the weights α and β. The interaction function is formalized as:

$$\hat{y}_{ui} = \alpha\hat{y}_{mp} + \beta\hat{y}_c$$

where $\hat{y}_{mp}$ and $\hat{y}_c$ are the outputs of NeuMF$_{mp}$ and NeuMF$_c$, respectively. We omit the specific definition of $\hat{y}_{mp}$ and $\hat{y}_c$ here, because they are the same as the interaction function of NeuMF.

As initialization plays an important role in the convergence and performance of deep learning models [30], we further propose a stacking integration model with *pre-training*. Specifically, we first train NeuMF$_{mp}$ and NeuMF$_c$ separately with random initializations until convergence, then we combine the pre-trained models by setting $\alpha = \beta = 0.5$. The objective function is set as *logloss* and is optimized by SGD.

IV. EXPERIMENT

We conduct experiments to demonstrate the effectiveness of both our proposed framework and the refined user representation method.

TABLE I. Statistics of the Original, Validation, and Test Sets (Year is 2017)

Dataset	#interactions	#users	#items	period
Original	156,287	14,659	18,511	5-11 to 9-30
Validation	12,410	9,937	1,539	9-11 to 9-17
Test	14,443	10,273	2,369	9-18 to 9-30

A. SETTING

Dataset. We used an Ads platform dataset that contains the browsing records of users from *5-11-2017* to *9-30-2017* and an online shopping dataset with the purchase records of users, i.e., interactions, in the same period. We divided purchase records into three parts: interactions from *5-11-2017* to *9-10-2017* for training, interactions from *9-11-2017* to *9-17-2017* for validation, and interactions from *9-18-2017* to *9-30-2017* for testing. To avoid using future information to predict the past, we utilized the browsing records only from *5-11-2017* to *9-10-2017* for user representation learning. The original interaction dataset is very large but highly sparse, i.e., many users have less than a few purchase records. We hence retained only users with at least 5 interactions (purchases) as the training set. It results in a subset of the original dataset, and it includes 14,659 users, 18,511 items, and 156,287 interactions. Moreover, to evaluate the recommendation performance on cold-start users (i.e., new users), we removed all the purchase records of users who are in the training set from the validation and test sets. The statistics of these datasets are summarized in Table I.

Evaluation Criteria. We followed the common strategy of top-K recommendation in [5], [20] to rank the target item in each interaction with items that have not been interacted by the user. Because it is too time-consuming to rank all items for every user, the strategy that randomly samples I items with which the user has not interacted and ranks the target item among the $(I + 1)$ items has been widely adopted by a series of works on recommendation [5], [6], [20], [31], [32]. The items in our online shopping domain exist in a short period of time, and we found that there are on average about 1,500 items for sale per day from *5-11-2017* to *9-10-2017*. Due to this fact, we randomly sampled 1,499 items that have no interaction with the user, and ranked the target item among 1,500 items.

The performance of a ranked list was evaluated by *Hit Ratio* (HR) and *Normalized Discounted Cumulative Gain* (NDCG) [33]. Given an interaction that contains the user and the target item, the recommendation model produces a ranked list of items. If the target item appears in the recommended list, which consists of K items, it is viewed as a hit. Therefore, HR is:

$$HR@K = \frac{\text{Number of hits}@K}{|T|}$$

where $|T|$ denotes the number of interactions in the test set.

To address the problem that HR does not consider the *rank* of the target item, NDCG accounts for hit position by assigning higher scores to hits at top ranks:

$$NDCG@K = \sum_{i=1}^{K} \frac{2^{r_i} - 1}{\log_2(i+1)}$$

where r_i is the graded relevance of the target item at position i: $r_i = 1$ if the target item is ranked in the i-th position, otherwise $r_i = 0$.

Without special mention, we truncated the ranked list with K from 5 to 10 for evaluation in our work, because recommendation lists in practice are usually short. We calculated both metrics for each interaction and show the average score.

Evaluation Methods. To justify the effectiveness of our proposals, we evaluated the following methods (the first four methods are based on existing approaches while the remaining ones are our approaches):

- **Cosine Similarity** is a simple and standard method of modeling user-item relationships, and it has been widely adopted as the final step for interaction function in recommendation models. This provides the K most similar items, where similarity is calculated by Cosine similarity. (This method does not use cross-domain knowledge.)
- **ItemPop** is a method that ranks items based on their popularity. In our case, the popularity is the the number of interactions of items that are for sale. It is a non-personalized method, but the performance is usually used as a baseline. (This method also does not use cross-domain knowledge.)
- **GMF** [5] is a generalized version of the traditional MF. It is a shallow neural network model that learns user-item interactions by applying an activation function to the linear combination of the element-wise product of the input vectors. It first accepts only user IDs and item IDs as one-hot vectors. These vectors are transformed to dense vectors, and then they are fed into a neural architecture to obtain prediction scores of user-item pairs.
- **DMF** [6] is a state-of-the-art deep matrix factorization model. It first create an implicit user-item matrix. A DNN accepts this matrix as its input to generate high-dimensional user vectors, which consider rating for each item, and high-dimensional item vectors, which consider their rating across all users. Finally, these user and item vectors are projected into a low-dimensional latent space. The similarity of a user and an item is calculated by the cosine similarity.
  We followed the setting in [6] to build a two-layer DMF and tuned hyper-parameters to produce the best recommendation performance.
- **NeuMF**$_c$ is the generalized version of the NeuMF model applied in our work. We use NeuMF$_c$ to denote NeuMF in our work unless otherwise stated.
- **NeuMF**$_{mp}$ takes the user representation generated by R-metapath2Vec, instead of Word2Vec in NeuMF.
- **Three-ways** is an integration model that extends the two-pathway NeuMF$_c$ by adding an extra pathway to model the features captured in R-metapath2Vec.

TABLE II. Performance of HR@K (Percentage Display)

K	5	6	7	8	9	10
NeuMF	**13.77**	**17.73**	**20.95**	**23.17**	**24.82**	**26.17**
DMF	12.39	14.71	16.77	18.69	20.43	21.82
GMF	6.00	7.13	8.03	9.04	10.01	10.90
ItemPop	1.46	8.14	8.14	8.14	8.14	8.14

TABLE III. Performance of NDCG@K

K	5	6	7	8	9	10
NeuMF	0.067	**0.081**	**0.091**	**0.098**	**0.103**	**0.107**
DMF	**0.073**	**0.081**	0.089	0.094	0.099	0.103
GMF	0.036	0.040	0.043	0.046	0.049	0.052
ItemPop	0.006	0.030	0.030	0.030	0.030	0.030

- **Stacking** is a model-level integration method for combining Word2Vec and R-metapath2Vec representations.

Implementation Details. All the neural network models were implemented based on PyTorch.[3] For the training of each model, we randomly sampled four negative samples for each positive interaction. Parameters to be trained were randomly initialized from Gaussian distribution $\mathcal{N}(0, 0.01^2)$. The optimizer was SGD with a batch size of 256, and we tested a 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]. As for structure of MLP part, we used a tower pattern, which halves the layer size for each successive higher layer. To determine the best hyper-parameters for each model, we tuned hyper-parameters based on the validation set. Moreover, we built the skip-gram model for R-metapath2Vec based on Gensim.[4] We set 3 as the size of the context window, 5 as the number of negative samples, and 0.1 as the learning rate to train the model until convergence.

B. PERFORMANCE COMPARISON

We report the recommendation performances of NeuMF$_c$, DMF, GMF, ItemPop, and Cosine Similarity. Table II shows the performance of HR@K, while Table III shows the performance of NDCG@K. Because of the weak performance of Cosine Similarity, it is omitted to better highlight the performance difference of the other methods.

We see that ItemPop, a non-personalized model, performs poorly in recommending items to new users. This indicates the necessity of representing users by modeling their personalized preferences. GMF outperforms ItemPop except HR@6 and HR@7, and it has a great improvement in NDCG@K compared with ItemPop. This result demonstrates the usefulness of transferring the knowledge from other domains for alleviating the cold-start problem. For the models based on deep learning, DMF and NeuMF$_c$ consistently give better performance in both metrics than the other methods. For HR@10, NeuMF$_c$ achieves improvements of up to 15.27% and 18.02% compared with GMF and ItemPop, respectively. For NDCG@10, the performance of NeuMF$_c$ is about 2 times better than that of GMF and 3.64 times better than that of ItemPop. This confirms the effectiveness of applying neural approaches with knowledge introduced from another domain.

We observe that NeuMF$_c$ outperforms DMF in HR@K. This result shows that NeuMF$_c$ adds more target items into top-K list, and DMF puts less items at a higher position in the ranked list. Although NeuMF$_c$ performs slightly worse than DMF in NDCG@5, NeuMF$_c$ outperforms DMF in f NDCG@K when K $>$ 6. For example, it achieves a 4.35% improvement in HR@10 and performs 1.04 times better than DMF in NDCG@10. The experimental results suggest that NeuMF$_c$ is better at modeling the overall user-item interactions. That is, our user and item presentation developed in Section III-B functions well, and our model learns their vectors so that the inner products between users and their preferred items are high. Also, the validity and effectiveness of our method are clear, because Tables II and III show that the accuracy our approach is higher than those of the existing approaches.

TABLE IV. Performance of HR@K (Percentage Display)

K	5	6	7	8	9	10
NeuMF$_c$	42.38	45.21	47.53	49.62	51.69	53.25
NeuMF$_{mp}$	42.56	45.26	47.43	49.67	51.71	53.56
Three-ways	41.51	43.66	45.79	47.77	49.68	51.42
Stacking	**44.05**	**46.79**	**49.17**	**51.27**	**53.27**	**55.30**

TABLE V. Performance of NDCG@K

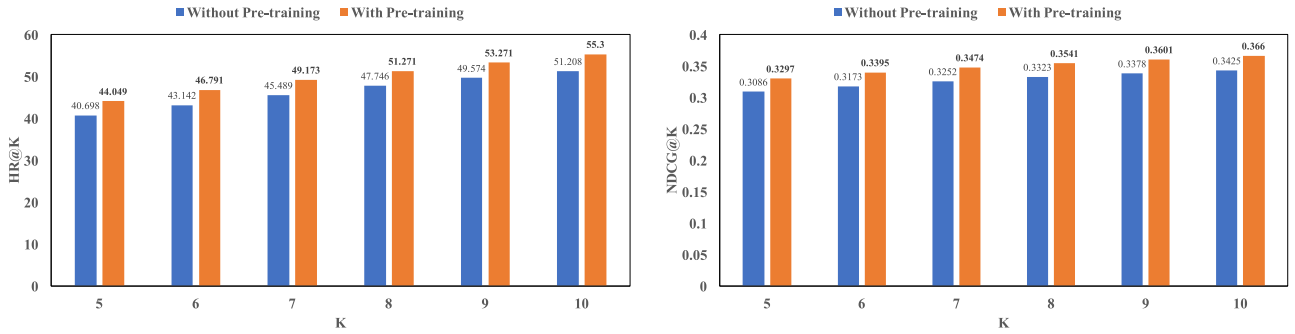
K	5	6	7	8	9	10
NeuMF$_c$	0.322	0.332	0.340	0.347	0.353	0.357
NeuMF$_{mp}$	0.316	0.326	0.333	0.340	0.346	0.351
Three-ways	0.325	0.332	0.340	0.346	0.352	0.357
Stacking	**0.330**	**0.340**	**0.347**	**0.354**	**0.360**	**0.366**

C. EFFECTIVENESS OF INTEGRATION

We next evaluate the recommendation performances of NeuMF$_c$, NeuMF$_{mp}$, Three-ways, and Stacking, to demonstrate the effectiveness of integrating Word2Vec and R-metapath2Vec. Stacking here refers to the version of Stacking with pre-training, and we set $\alpha = \beta = 0.5$. The performances w.r.t. HR@K and NDCG@K are respectively shown in Table IV and V.

The overall results of NeuMF$_{mp}$ and NeuMF$_c$ are almost the same. This verifies that Word2Vec and R-metapath2Vec have their own strengths in user modeling. Therefore, our strategy, which integrates these representations, is suitable for improving recommendation performance. Three-ways, on the other hand, does not perform as well as we expected. We see that Three-ways achieves a performance close to those of NeuMF$_{mp}$ and NeuMF$_c$ in NDCG@K, but it is outperformed by these non-integrated models in HR@K. This result indicates the importance of applying an appropriate integration method to integrate different kinds of user latent embeddings. It also shows that a simple vector concatenation cannot reinforce Word2Vec and R-metapath2Vec. Stacking achieves the best performance in both metrics for all the values of

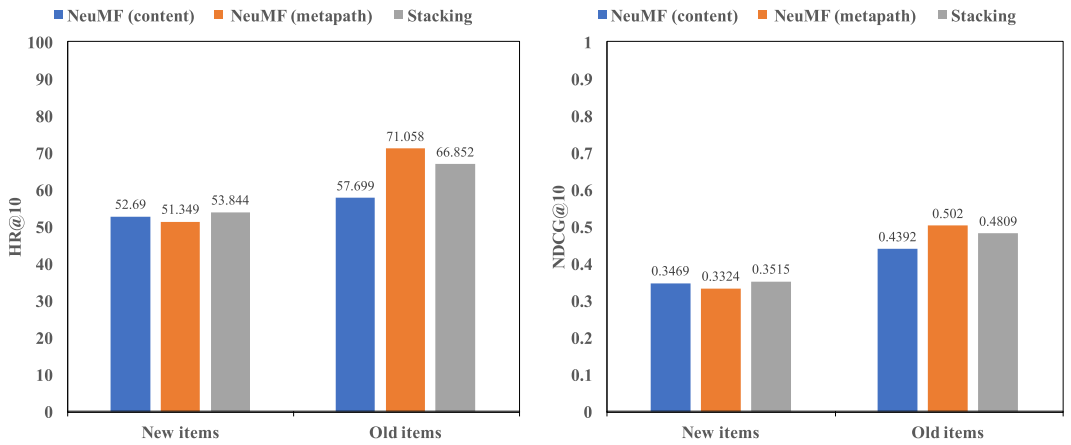
[3][Online]. Available: https://pytorch.org
[4][Online]. Available: https://radimrehurek.com/gensim/



(a) HR@K of Stacking with and without pre-training

(b) NDCG@K of Stacking with and without pre-training

FIGURE 13. Comparison between Stacking with and without pre-training. The results confirm that the pre-training version can recommend the proper item more accurately than the version of without pre-training.



(a) HR@10

(b) NDCG@10

FIGURE 14. Performance of HR@10 and NDCG@10 w.r.t. the new items and old items. Stacking shows the best performance for new items, thereby it is better to use Stacking for cold-start *items*.

K. For example, HR@10 of Stacking achieves about 1.75% improvement compared with that of NeuMF$_{mp}$. This demonstrates the effectiveness of our integration of Word2Vec and R-metapath2Vec in Stacking.

D. UTILITY OF PRE-TRAINING

We compare the performance of two versions of Stacking, with and without pre-training, to demonstrate the utility of pre-training. For Stacking with pre-training, we trained NeuMF$_c$ and NeuMF$_{mp}$ separately and integrated them by setting $\alpha = \beta = 0.5$. For Stacking without pre-training, we used SGD to learn the whole model with random initialization.

The experimental results of HR@K and NDCG@K are shown in Figure 13(a) and 13(b), respectively. We can see that Stacking with pre-training consistently achieves better performance than the version of without pre-training. In particular, the improvement of Stacking with pre-training in HR@10 is about 4.1%. For NDCG@10, Stacking with pre-training performs about 1.07 times better. The results justify the usefulness of pre-training. Training α and β automatically by model is generally a good way to determine the best values. However, learning the whole model without pre-training seems to weaken each component.

TABLE VI. Statistics of the New and Old Items

Dataset	#items	#interactions
New items	2,166	12,826
Old items	203	1,617

E. RECOMMENDATION ON NEW/OLD ITEMS

We further investigate the recommendation performances of NeuMF$_c$, NeuMF$_{mp}$, and Stacking (with pre-training) on new items and old items. New items are defined as items that are not included in the training and validation sets. Table VI illustrates the statistics of the new and old item datasets. The number of new items is much larger than the number of old items, and there are nearly eight times more interactions of new items than of old items. These characteristics indicate that users of our online shopping domain prefer to purchase

new items. The ability to provide accurate recommendations for new items is therefore a critical factor in improving the overall performance.

We evaluated the performance of HR@10 and NDCG@10 w.r.t. new and old items, and the result is shown in Figure 14. For old items, NeuMF$_{mp}$ achieves the best performance in both HR@10 and NDCG@10. In particular, NeuMF$_{mp}$ improves HR@10 for old items by about 13.4% compared with NeuMF$_c$. Stacking is also slightly outperformed by NeuMF$_{mp}$ on old items. However, Stacking performs the best for both metrics on new items, which are the main part of the whole interaction dataset. This result proves our assumption that providing accurate recommendations for new items has a great influence on the overall performance.

V. CONCLUSION

In this work, we developed a framework that utilizes the knowledge in an Ads platform to improve the recommendation performance for cold-start users of an online shopping domain. We turned textual information of users and items into latent vectors by Word2Vec as their representations. Then we applied different kinds of collaborative filtering models to build a cross-domain recommender system and conducted extensive experiments to evaluate their recommendation performances for new users. The experimental results demonstrate the effectiveness of transferring the knowledge from other domains in alleviating the cold-start problem. As the models based on deep learning outperform other shallow models and non-personalized methods, we can also find the usefulness of applying neural approaches with knowledge introduced from another domain.

To overcome the drawbacks of Word2Vec and further improve the recommendation performance, we proposed R-metapath2Vec and integrated these two kinds of user representations by different kinds of integration models, Three-ways and Stacking. We further conducted extensive experiments to compare the recommendation performances between integration models and non-integrated models such as NeuMF$_c$ and NeuMF$_{mp}$. The experimental results show the effectiveness of our Stacking integration model.

In future, we plan to collect textual information for user representation at the URL level and apply TF-IDF [34] to give different weights to different URLs and words. This is because it can reflect the importance of each entry and distinguish different users more accurately. Because the items on our online shopping website change over a period of time, we also plan to further explore the impact of time on the recommendation performance of our methods.

REFERENCES

[1] J. Wang, P. Huang, H. Zhao, Z. Zhang, B. Zhao, and D. L. Lee, "Billion-scale commodity embedding for e-commerce recommendation in alibaba," in *Proc. ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2018, pp. 839–848.

[2] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, "Item-based collaborative filtering recommendation algorithms," in *Proc. Int. Conf. World Wide Web*, 2001, pp. 285–295.

[3] Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," *Computer*, vol. 42, no. 8, pp. 30–37, 2009.

[4] R. Salakhutdinov and A. Mnih, "Probabilistic matrix factorization," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2007, pp. 1257–1264.

[5] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T.-S. Chua, "Neural collaborative filtering," in *Proc. Int. Conf. World Wide Web*, 2017, pp. 173–182.

[6] H.-J. Xue, X.-Y. Dai, J. Zhang, S. Huang, and J. Chen, "Deep matrix factorization models for recommender systems," in *Proc. Int. Joint Conf. Artif. Intell.*, 2017, pp. 3203–3209.

[7] R. Salakhutdinov, A. Mnih, and G. Hinton, "Restricted boltzmann machines for collaborative filtering," in *Proc. Int. Conf. Mach. Learn.*, 2007, pp. 791–798.

[8] G. K. Dziugaite and D. M. Roy, "Neural network matrix factorization," 2015, *arXiv:1511.06443*.

[9] H. Guo, R. Tang, Y. Ye, Z. Li, and X. He, "Deepfm: A factorization-machine based neural network for CTR prediction," in *Proc. Int. Joint Conf. Artif. Intell.*, 2017, pp. 1725–1731.

[10] S. Rendle, "Factorization machines," in *Proc. IEEE Int. Conf. Data Mining*, 2010, pp. 995–1000.

[11] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2013, pp. 3111–3119.

[12] O. Barkan and N. Koenigstein, "Item2vec: Neural item embedding for collaborative filtering," in *Proc. IEEE Int. Workshop Mach. Learn. Signal Process.*, 2016, pp. 1–6.

[13] H. Zhao, Q. Yao, J. Li, Y. Song, and D. L. Lee, "Meta-graph based recommendation fusion over heterogeneous information networks," in *Proc. ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2017, pp. 635–644.

[14] I. Fernández-Tobías, M. Braunhofer, M. Elahi, F. Ricci, and I. Cantador, "Alleviating the new user problem in collaborative filtering by exploiting personality information," *User Model. User-Adapted Interact.*, vol. 26, no. 2/3, pp. 221–255, 2016.

[15] P. Cremonesi, A. Tripodi, and R. Turrin, "Cross-domain recommender systems," in *Proc. IEEE Int. Conf. Data Mining Workshops*, 2011, pp. 496–503.

[16] L. Hu, L. Cao, J. Cao, Z. Gu, G. Xu, and J. Wang, "Improving the quality of recommendations for users and items in the tail of distribution," *ACM Trans. Inf. Syst.*, vol. 35, no. 3, pp. 25:1–25:37, 2017.

[17] X. Wang, X. He, L. Nie, and T.-S. Chua, "Item silk road: Recommending items from information domains to social users," in *Proc. Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2017, pp. 185–194.

[18] H. Wang *et al.*, "Preliminary investigation of alleviating user cold-start problem in e-commerce with deep cross-domain recommender system," in *Proc. Companion World Wide Web Conf.*, 2019, pp. 398–403.

[19] Y. Dong, N. V. Chawla, and A. Swami, "metapath2vec: Scalable representation learning for heterogeneous networks," in *Proc. ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2017, pp. 135–144.

[20] G. Hu, Y. Zhang, and Q. Yang, "Conet: Collaborative cross networks for cross-domain recommendation," in *Proc. ACM Int. Conf. Inf. Knowl. Manage.*, 2018, pp. 667–676.

[21] Y. Wu, C. DuBois, A. X. Zheng, and M. Ester, "Collaborative denoising auto-encoders for top-n recommender systems," in *Proc. ACM Int. Conf. Web Search Data Mining*, 2016, pp. 153–162.

[22] X. He, H. Zhang, M.-Y. Kan, and T.-S. Chua, "Fast matrix factorization for online recommendation with implicit feedback," in *Proc. Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2016, pp. 549–558.

[23] Y. Hu, Y. Koren, and C. Volinsky, "Collaborative filtering for implicit feedback datasets," in *Proc. IEEE Int. Conf. Data Mining*, 2008, pp. 263–272.

[24] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme, "Bpr: Bayesian personalized ranking from implicit feedback," in *Proc. Conf. Uncertainty Artif. Intell.*, 2009, pp. 452–461.

[25] N. Srivastava and R. R. Salakhutdinov, "Multimodal learning with deep boltzmann machines," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2012, pp. 2222–2230.

[26] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proc. Int. Conf. Artif. Intell. Statist.*, 2010, pp. 249–256.

[27] Y. LeCun *et al.*, "Backpropagation applied to handwritten zip code recognition," *Neural Comput.*, vol. 1, no. 4, pp. 541–551, 1989.
[28] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," in *Proc. Int. Conf. Artif. Intell. Statist.*, 2011, pp. 315–323.
[29] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proc. Int. Conf. Mach. Learn.*, 2015, pp. 448–456.
[30] D. Erhan, Y. Bengio, A. Courville, P.-A. Manzagol, P. Vincent, and S. Bengio, "Why does unsupervised pre-training help deep learning?" *J. Mach. Learn. Res.*, vol. 11, pp. 625–660, 2010.
[31] A. M. Elkahky, Y. Song, and X. He, "A multi-view deep learning approach for cross domain user modeling in recommendation systems," in *Proc. Int. Conf. World Wide Web*, 2015, pp. 278–288.
[32] Y. Koren, "Factorization meets the neighborhood: A multifaceted collaborative filtering model," in *Proc. ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2008, pp. 426–434.
[33] K. Järvelin and J. Kekäläinen, "Cumulated gain-based evaluation of ir techniques," *ACM Trans. Inf. Syst.*, vol. 20, no. 4, pp. 422–446, 2002.
[34] G. Salton and M. J. McGill, "Introduction to modern information retrieval," 1986.

HANXIN WANG received the B.E. and M.S. degrees from Sun Yat-sen University in 2017 and Osaka University in 2020, respectively. His research interest includes AI and machine learning technologies.

DAICHI AMAGATA (Member, IEEE) received the B.E., M.Sc., and the Ph.D. degrees from Osaka University, in 2012, 2014, and 2015, respectively. He is an Assistant Professor with the Department of Multimedia Engineering Graduate School of Information Science and Technology Osaka University, Osaka, Japan. His research interests include distributed and parallel query processing and data monitoring over stream environments. He is a member of IEEE.

TAKUYA MAKEAWA (Member, IEEE) was born in 1980. He received the Ph.D. in information science and technology from Osaka University. He is an Associate Professor at Osaka University, Japan. His research interests include ubiquitous computing and mobile sensing. He is a member of ACM.

TAKAHIRO HARA (Senior Member, IEEE) received the B.E., M.E., and Dr.E. degrees in information systems engineering from Osaka University, Osaka, Japan, in 1995, 1997, and 2000, respectively. Currently, he is a Full Professor of the Department of Multimedia Engineering, Osaka University. His research interests include distributed databases, peer-to-peer systems, mobile networks, and mobile computing systems. He is a distinguished scientist of ACM and a member of three other learned societies.

NIU HAO received the Ph.D. degree from the Department of Information and Communication Engineering, University of Tokyo, Japan, in 2016. He is currently a Researcher with KDDI Research, Inc. His research interests include machine learning, data mining and Internet of Things.

KEI YONEKAWA received the B.S. and M.S. degrees from the Department of Electrical Engineering, University of Tokyo, in 2012 and 2014, respectively. In 2014, he joined KDDI Corporation, where he was engaged in the operation of the infrastructure of cloud service. In 2015, he joined KDDI Research, Inc., where he is currently a Researcher. His research interests include machine learning, data mining, and big data application.

MORI KUROKAWA received the Master's degree from the Faculty of Science and Technology, Keio University, Japan, in 2007. He is currently a Research Manager in KDDI Research, Inc. His research interests include machine learning and data mining.