Heterogeneous Embedding Propagation for Large-scale E-Commerce User Alignment

Vincent W. Zheng[†], Mo Sha[‡], Yuchen Li[◇], Hongxia Yang[⋆],
Yuan Fang[◇], Zhenjie Zhang[†], Kian-Lee Tan[‡], Kevin Chen-Chuan Chang[*]

[†] Advanced Digital Sciences Center, Singapore {vincent.zheng,zhenjie}@adsc.com.sg
[‡] National University of Singapore, Singpaore {sham,tankl}@comp.nus.edu.sg
[◇] Singapore Management University, Singapore {yuchenli,yfang}@smu.edu.sg
[⋆] Alibaba Group, China yang.yhx@alibaba-inc.com
[*] University of Illinois at Urbana-Champaign, USA kcchang@illinois.edu

***Abstract*—We study the important problem of user alignment in e-commerce: to predict whether two online user identities that access an e-commerce site from different devices belong to one real-world person. As input, we have a set of user activity logs from Taobao and some labeled user identity linkages. User activity logs can be modeled using a heterogeneous interaction graph (HIG), and subsequently the user alignment task can be formulated as a semi-supervised HIG embedding problem. HIG embedding is challenging for two reasons: its heterogeneous nature and the presence of edge features. To address the challenges, we propose a novel Heterogeneous Embedding Propagation (HEP) model. The core idea is to iteratively reconstruct a node's embedding from its heterogeneous neighbors in a weighted manner, and meanwhile propagate its embedding updates from reconstruction loss and/or classification loss to its neighbors. We conduct extensive experiments on large-scale datasets from Taobao, demonstrating that HEP significantly outperforms state-of-the-art baselines often by more than 10% in F-scores.**

***Index Terms*—E-commerce User Alignment, Heterogeneous Interaction Graph, Heterogeneous Embedding Propagation**

I. Introduction

E-commerce user alignment is the task of linking different user identities on an e-commerce site if they belong to one real-world person. A typical scenario faced by Taobao, a leading e-commerce site, is that online users can access Taobao through different devices (*e.g.*, mobile or PC), and they may not log into their accounts. Being able to link these devices to the real-world Taobao user is of great importance: it not only helps the company profile its customers more accurately, but also offers users a seamless shopping experience.

In this paper, we focus on predicting whether a Mobile device ID (MID) and a PC device ID (PID) refer to the same Taobao user. As input, we have *user activity logs* from Taobao. An example of such logs is shown in Table I. Each row is an activity record about which device a user used, which IP she had, which shop she visited, which auction she browsed, and which keywords she searched. Some fields are empty, since an activity may not involve all actions. As output, we predict a binary label for a candidate pair of PID and MID, indicating whether they belong to the same Taobao user.

TABLE I
Examples of e-commerce user activity logs.

Time	User ID	IP	Keyword	Auction	Shop
04/05/2017 16:21:41	PID1	IP2	Keyword1	-	Shop3
05/05/2017 22:16:00	MID3	IP2	Keyword2	Auction1	Shop2
...	...	...	...	...	...

Data model. We choose to model the user activity logs as a *Heterogeneous Interaction Graph* (HIG). Next, we define what an HIG is and explain our motivation.

In our Taobao user alignment task, an HIG is a heterogeneous graph, whose network schema is defined in Fig. 1. Each node refers to an entity in the user activity logs, such as an MID, a PID, an IP, a keyword, an auction or a shop. MID and PID are about users, whereas the other entities are about items. Hence in this paper, we call a PID or an MID as a *user ID*, whereas we call an IP, a keyword, an auction or a shop as an *item ID*. Each edge indicates interactions between two nodes, and we extract a time-aware feature vector to describe such interactions. Particularly, a user-item edge (*e.g.*, PID-shop) indicates how a user browses an item, and its feature vector describes when and how frequently such browsing happens. An item-item edge (*e.g.*, IP-shop) indicates how two items co-occur in the user activity logs, and its feature vector describes when and how frequently such co-occurrence happens. HIG is a generalization to a recent concept of "bipartite interaction graph" [1], which was proposed to model time-dependent interactions between two types of entities (*e.g.*, investors and stocks). HIG has a unique advantage as our data model—its relational structure is suitable for discovering rich semantics of each user and item.

We find that, although there are some alternative data models, they are not as effective as HIG for our user alignment task. For example, we can represent each user ID with a feature vector about when and how frequently she browses

This material is based upon work partially supported by National Research Foundation, Prime Minister's Office, Singapore under its Campus for Research Excellence and Technological Enterprise (CREATE) programme, and Alibaba Group under its Alibaba Innovative Research (AIR) programme. Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the authors and do not necessarily reflect the views of the funding agencies.

978-1-5386-9159-5/18/$31.00 ©2018 IEEE
DOI 10.1109/ICDM.2018.00198

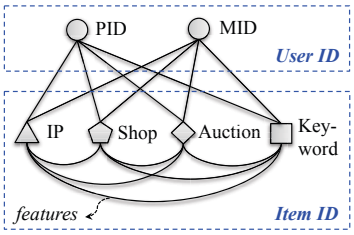

Fig. 1. Schema of HIG.

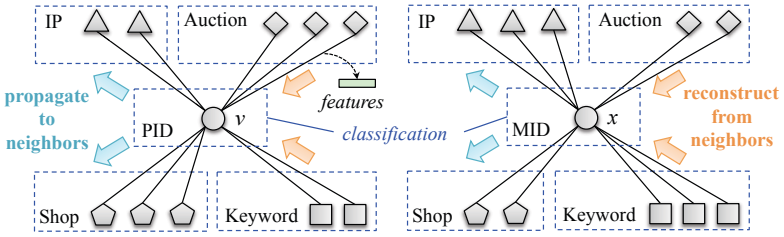

Fig. 2. Overall framework of HEP.

each item, and uses these feature vectors for classification, but this approach leaves out the item interactions. We can also model the item co-occurrence by topic modeling [2], and represent each user ID by her interactions with each item group, but this approach is hard to factor in time information. Finally, we can also model each user as a time series of user-item interactions, and use a recurrent neural network [3] to learn a low-dimensional representation for classification, but this approach still does not fully exploit the user relations as each user's sequence is treated as independent.

Challenges. To leverage the relational structure of HIG, we formulate our user alignment task as a *semi-supervised HIG embedding* problem. We face two challenges below to embed HIG in e-commerce user alignment.

• *Node heterogeneity*. We cannot simply treat the nodes in HIG as homogeneous, as a user clearly has different semantics from an item. Typical graph embedding methods [4], [5] focus on homogeneous graphs.

• *Edge features*. Despite the success of recent heterogeneous graph embedding methods, such as Metapath2vec [6] and HNE [7], they do not consider edges with features.

Technical model. To address the above challenges, we propose a novel *Heterogeneous Embedding Propagation* (HEP) model. As shown in Fig. 2, the core idea is to iteratively "reconstruct" a node's embedding from its heterogeneous neighbors, and "propagate" its embedding updates to its neighbors. To handle node heterogeneity, HEP treats nodes of different types severally. For example, to learn embedding of a PID node v in Fig. 2, HEP first aggregates v's neighbors of each type (*e.g.*, all its IP neighbors), before transforming the aggregation of each node type to reconstruct v's embedding. To leverage edge features, HEP considers each node v as receiving different contributions from its neighbors. When aggregating v's neighbors of the same type, HEP assigns different weights to different neighbors according to their edge features.

Contributions. First, we study a real-world e-commerce user alignment task, and formulate it as a semi-supervised HIG embedding problem. Second, we propose a novel HEP model, which addresses the challenges of node heterogeneity and edge features. Third, we evaluate HEP on two large-scale datasets from Taobao, and the results are promising.

II. Related Work

The user alignment is related to several concepts, including network alignment [8], network anchoring [9], and link prediction [10]. *Network alignment* aims to align nodes across multiple homonegenous networks, such as to link user accounts in two social networks, whereas user alignment on HIG focuses on one network. *Network anchoring* aims to align nodes across multiple heterogeneous networks, such as to link user accounts from a check-in network (*e.g.*, Foursquare) to an online social network (*e.g.*, Twitter) where each network contains heterogeneous nodes such as users, locations and texts. In other words, network anchoring also focuses on multiple networks, thus not applicable to our task either. Finally, *Link prediction* aims to connect two nodes in one network. The state-of-the-art methods employ graph embedding [11], [10], which learns low-dimensional node representations from the network topology and/or content. In general, our user alignment task can be seen as link prediction on a HIG. Next, we shall review the state of the art in graph embedding.

Graph embedding has been a popular graph analytics approach. Early methods were mainly designed to reduce the dimensionality of non-relational data by projecting them into a low-dimensional manifold, as summarized in a recent survey [12]. Recent graph embedding methods use neural networks, with either shallow [13], [4] or deep architectures [14], [15], [7]. DeepWalk [11] and Node2vec [4] try to preserve the first-order graph proximity for node embedding. LINE [16] and SDNE [15] preserve both first and second-order proximity. Compared with the above methods, which focus on homogeneous graphs, our HEP considers a heterogeneous graph.

On heterogeneous graphs, metapath2vec [6] uses predefined meta-path patterns to sample paths, and HNE [7] uses separate deep neural networks to learn embedding for each type of nodes before aggregating them. In contrast, HEP does not require sampling, and considers additional edge features to differentiate heterogeneous edges.

III. Problem Formulation

We first introduce our terminologies and notations, as summarized in Table II.

Definition 1: A **heterogeneous interaction graph** (HIG) is $G = (V, E, C, \phi, F)$, where V is a set of nodes, E is a set of edges, $C = \{c_1, ..., c_{n_1}\}$ is a set of distinct node types, $\phi : V \rightarrow C$ is a node type function, F is a set of

TABLE II
LIST OF COMMONLY USED NOTATIONS.

Notation	Description
G, V, E	Graph G, nodes V, edges E
C, ϕ	Node types C, node typing function $\phi : V \to C$
C', ϕ	Edge types C, edge typing function $\psi : V \times V \to C'$
$F, \mathbf{f}$	Edge feature set F, an edge's feature vector $\mathbf{f}(v,u) \in \mathbb{R}^d$
$\mathcal{D}, m$	Set of training tuples $\mathcal{D}$, with cardinality of m
n_1, n_2	Number of node types n_1 and number of edge types n_2
d, k	Edge feature dimension d and node embedding dimension k
α, β	Trade-off parameters for loss and regularizer
γ	Slack parameter
ζ	Number of negative samples

d-dimensional edge feature vectors, each of which describes interaction between two nodes.

For example, in Fig. 1, we have $C = \{$PID, MID, IP, shop, auction, keyword$\}$. We call PID and MID as user ID's. We call IP, shop, auction and keyword as item ID's. For a node $v \in V$, $\phi(v)$ returns v's node type from C. For an edge $(v,u) \in E$, $\mathbf{f}(v,u) \in F$ returns a d-dimensional vector about the time-dependent interaction features between v and u. For example, if v is a PID and u is an IP, $\mathbf{f}(v,u)$ is a vector describing how many days user v accesses IP u, how many times v accesses u at hours of a day, days of a week, and weekday-vs.-weekend. Similar time-dependent interaction features are extracted for other types of edges. Finally, due to node heterogeneity, the edges are heterogeneous too. For easier discussion later, we denote $\psi(v,u) : V \times V \to C \times C$ as an edge type function. In this paper, we let $\psi(v,u) = \psi(u,v)$, and denote the set of distinct edge types as $C' = \{c'_1, ..., c'_{n_2}\}$. In Fig. 1, we have $C' = \{$PID-IP, PID-shop, PID-auction, PID-keyword, MID-IP, MID-shop, $\ldots\}$.

Problem inputs and outputs. As *input* of our problem, we have a heterogeneous interaction graph G, and a set of training tuples $\mathcal{D} = \{(v_i, u_i, y_i) | i = 1, ..., m, \phi(v_i) = \text{PID}, \phi(u_i) = \text{MID}, y_i \in \{1, -1\}\}$. As *output* of our problem, we wish to learn an embedding vector $\mathbf{h}_v \in \mathbb{R}^k$ for each PID v and an embedding vector $\mathbf{h}_u \in \mathbb{R}^k$ for each MID u. Then, given a pair of (v,u), we can predict the probability of having a link between v and u as

$$P(y_i | v_i, u_i) = \sigma\left(y_i \cdot \mathbf{h}_{v_i}^T W \mathbf{h}_{u_i}\right), \tag{1}$$

where $\sigma(z) = \frac{1}{1+e^{-z}}$ is a sigmoid function. Besides, $W \in \mathbb{R}^{k \times k}$ is a parameter matrix to enforce the bilinear interactions between $\mathbf{h}_v$ and $\mathbf{h}_u$.

Given training data $\mathcal{D}$, we aim to optimize the classification loss by minimizing the negative log-likelihood:

$$L_1 = -\frac{1}{m} \sum_{i=1}^{m} \log P(y_i | v_i, u_i). \tag{2}$$

IV. HEP MODEL

The core idea of HEP is to iteratively "reconstruct" a node's embedding from its heterogeneous neighbors, and "propagate" its embedding update to the neighbors for their own embedding reconstruction later. We use Fig. 2 to illustrate embedding learning for a user ID node v.

Node heterogeneity. In HIG, a user ID (PID or MID) node v is connected with heterogeneous neighbors, whose types can be IP, auction, shop or keyword. We aim to use these neighbors to reconstruct v's embedding. Different types of neighbors indicate different types of interactions; ***e.g.***, an edge between v and one IP indicates how v used this IP, whereas another edge between v and one keyword indicates how v searched that keyword. Due to such different semantics, these different types of edges are not directly comparable. Therefore, to use v's neighbors to reconstruct v's embedding, we have to differentiate the neighbor types. As shown in Fig. 2, HEP tries to aggregate the embedding from each type of neighbors separately (***e.g.***, all the IP neighbors together, and all the keyword neighbors together, *etc*), and then concatenate the outputs from each neighbor type to reconstruct v's embedding. Before we proceed with the reconstruction, there is one more question of how much each neighbor contributes to the reconstruction, which can be addressed by edge features.

Edge features. To quantify how much contribution each neighbor u makes in reconstructing v's embedding, we use the edge features $\mathbf{f}(v,u) \in \mathbb{R}^d$ to compute an edge weight. As neighbor nodes are heterogeneous, we differentiate the edge types. For each edge type $c' \in C'$, we introduce two parameters $\boldsymbol{\lambda}_{c'} \in \mathbb{R}^d$ and $b'_{c'} \in \mathbb{R}$ to compute the weight for each edge with type c'. Formally, for an edge (v,u), whose type is $\psi(v,u)$, its edge weight is defined as

$$s_{v,u} = \sigma\left(\boldsymbol{\lambda}_{\psi(v,u)} \cdot \mathbf{f}(v,u) + b'_{\psi(v,u)}\right). \tag{3}$$

Due to node heterogeneity, $s_{v,u}$ is only comparable to $s_{v,u'}$ if $\phi(u) = \phi(u')$. That is, all the edges of the same type share the same parameters $(\boldsymbol{\lambda}_{c'}, b'_{c'})$.

Reconstruction. We now use v's neighbors and their edge weights $s_{v,u}$'s, to reconstruct v's embedding. Denote N_v as v's neighbors, and $N_v^{(c)} \subset N_v$ as v's type-c neighbors; ***e.g.***, in Fig. 2, if $c = $ IP, $N_v^{(c)}$ is the set of IP neighbors of v. For each type c, We first aggregate v's type-c's neighbors by a weighted average with $s_{v,u}$'s:

$$\tilde{\mathbf{g}}_v^{(c)} = \sum\nolimits_{u \in N_v^{(c)}} \frac{s_{v,u}}{\sum_{u \in N_v^{(c)}} s_{v,u}} \mathbf{h}_u. \tag{4}$$

Given different neighbor types, we concatenate all $\tilde{\mathbf{g}}_v^{(c)}$'s:

$$\tilde{\mathbf{g}}_v = \text{CONCAT}\left(\tilde{\mathbf{g}}_v^{(c_1)}, ..., \tilde{\mathbf{g}}_v^{(c_{n_1})}\right). \tag{5}$$

Finally, we use $\tilde{\mathbf{g}}_v$ to reconstruct v's embedding $\tilde{\mathbf{h}}_v$. To account for node heterogeneity, we introduce two type specific parameters $W'_{\phi(v)} \in \mathbb{R}^{k \times k n_1}$ and $\mathbf{b}''_{\phi(v)} \in \mathbb{R}^k$ for the reconstruction. For each $v \in V$,

$$\tilde{\mathbf{h}}_v = \sigma\left(W'_{\phi(v)} \tilde{\mathbf{g}}_v + \mathbf{b}''_{\phi(v)}\right). \tag{6}$$

Reconstruction loss. Denote $\pi(\tilde{\mathbf{h}}_v, \mathbf{h}_v)$ as the distance between the reconstructed embedding $\tilde{\mathbf{h}}_v$ and the target em-

bedding $\mathbf{h}_v$. Here we adopt the Euclidean distance, although alternatives are possible.

$$\pi(\tilde{\mathbf{h}}_v, \mathbf{h}_v) = \frac{1}{2}\|\tilde{\mathbf{h}}_v - \mathbf{h}_v\|_2^2. \tag{7}$$

Subsequently, we introduce a hinge loss such that the reconstructed embedding $\tilde{\mathbf{h}}_v$ is closer to $\mathbf{h}_v$ than any other $\mathbf{h}_u$. That is, $\forall u \neq v$,

$$\ell(v,u) = \left[\gamma + \pi(\tilde{\mathbf{h}}_v, \mathbf{h}_v) - \pi(\tilde{\mathbf{h}}_v, \mathbf{h}_u)\right]_+, \tag{8}$$

where $[z]_+$ returns z if $z > 0$, and 0 otherwise. $\gamma > 0$ is a slack parameter. To save computation from evaluating every possible $u \in V \backslash v$, we adopt negative sampling [4] to sample a set of u's. Denote DEG_u as the degree of node u. We define the negative sampling probability of a node u by $P_n(u) \propto \text{DEG}_u^{3/4}$. We set the number of negative samples for each node v as ζ. Finally, we have the overall reconstruction loss as

$$L_2 = \frac{1}{|V|}\sum_{v\in V}\sum_{u\sim P_n(u)} \ell(v,u). \tag{9}$$

V. End-to-End Training

Denote $\Theta = \{W, b, \boldsymbol{\lambda}_{c'}, b'_{c'}, W'_c, \mathbf{b}''_c | c \in C, c' \in C'\}$ as the set of parameters. Denote $H = \{\mathbf{h}_v | v \in V\}$ as the set of node embedding. We train Θ and H by minimizing:

$$L(\Theta, H) = L_1 + \alpha L_2 + \beta\Omega(\Theta), \tag{10}$$

where $\alpha > 0$ and $\beta > 0$ are trade-off parameters. Ω is a regularizer (*i.e.*, the sum of each parameter's ℓ_2-norm).

To solve Eq. 10, we adopt an alternate optimization approach to learn Θ and H iteratively, as follows.

Fix Θ, we optimize H. We perform stochastic gradient descent (SGD). For each tuple (v_i, u_i, y_i) in L_1, we have its gradients over H as

$$\frac{\partial L_1}{\partial \mathbf{h}_{x_i}} = -\frac{1}{m}\left[1 - P(y_i|v_i,u_i)\right] y_i W \mathbf{h}_{x_i}, \tag{11}$$

where $x_i \in \{v_i, u_i\}$. For each node v and its negative sample u's, we have their gradients over H as

$$\frac{\partial L_2}{\partial \mathbf{h}_x} = \begin{cases} (-1)^{\delta(x=u)}\frac{1}{|V|}\left(\tilde{\mathbf{h}}_v - \mathbf{h}_x\right), & \text{if } \ell(v,u) > 0; \\ 0, & \text{otherwise,} \end{cases} \tag{12}$$

where $x \in \{v, u\}$. The indicator function $\delta(x = u) = 1$ iff $x = u$. After $\mathbf{h}_v$ and $\mathbf{h}_u$ are updated in one iteration, they will be used to reconstruct the neighbors of v and u in the next iteration, effectivly propagting the gradients of $\mathbf{h}_v$ and $\mathbf{h}_u$.

Fix H, we optimize Θ. We do SGD as well. For each tuple (v_i, u_i, y_i) in L_1, we have its gradients over Θ as

$$\frac{\partial L_1}{\partial W} = -\frac{1}{m}\left[1 - P(y_i|v_i,u_i)\right] y_i \mathbf{h}_{v_i}\mathbf{h}_{u_i}^T, \tag{13}$$

$$\frac{\partial L_1}{\partial b} = -\frac{1}{m}\left[1 - P(y_i|v_i,u_i)\right] y_i. \tag{14}$$

For each edge $(v,x) \in E$, we have their gradients over Θ as

$$\frac{\partial L_2}{\partial \boldsymbol{\lambda}_{\psi(v,x)}} = \left(\frac{\partial L_2}{\partial \tilde{\mathbf{h}}_v}\right)^T \cdot \frac{\partial \tilde{\mathbf{h}}_v}{\partial \tilde{\mathbf{g}}_v^{\phi(x)}} \cdot \frac{\partial \tilde{\mathbf{g}}_v^{\phi(x)}}{\partial s_{v,x}} \cdot s_{v,x}(1 - s_{v,x})\mathbf{f}(v,x), \tag{15}$$

$$\frac{\partial L_2}{\partial b'_{\psi(v,x)}} = \left(\frac{\partial L_2}{\partial \tilde{\mathbf{h}}_v}\right)^T \cdot \frac{\partial \tilde{\mathbf{h}}_v}{\partial \tilde{\mathbf{g}}_v^{\phi(x)}} \cdot \frac{\partial \tilde{\mathbf{g}}_v^{\phi(x)}}{\partial s_{v,x}} \cdot s_{v,x}(1 - s_{v,x}), \tag{16}$$

$$\frac{\partial L_2}{\partial W'_{\phi(v)}} = \frac{\partial L_2}{\partial \tilde{\mathbf{h}}_v} \circ \tilde{\mathbf{h}}_v \circ \left(1 - \tilde{\mathbf{h}}_v\right) \cdot \tilde{\mathbf{g}}_v^T, \tag{17}$$

$$\frac{\partial L_2}{\partial \mathbf{b}''_{\phi(v)}} = \frac{\partial L_2}{\partial \tilde{\mathbf{h}}_v} \circ \tilde{\mathbf{h}}_v \circ \left(1 - \tilde{\mathbf{h}}_v\right), \tag{18}$$

$$\frac{\partial L_2}{\partial \tilde{\mathbf{h}}_v} = \begin{cases} \frac{1}{|V|}\left(\mathbf{h}_u - \mathbf{h}_v\right), & \text{if } \ell(v,u) > 0; \\ 0, & \text{otherwise,} \end{cases} \tag{19}$$

$$\frac{\partial \tilde{\mathbf{h}}_v}{\partial \tilde{\mathbf{g}}_v^{\phi(x)}} = \text{diag}\left(\tilde{\mathbf{h}}_v \circ \left(1 - \tilde{\mathbf{h}}_v\right)\right) \cdot R\left(W'_{\phi(v)}, \phi(x)\right), \tag{20}$$

$$\frac{\partial \tilde{\mathbf{g}}_v^{\phi(x)}}{\partial s_{v,x}} = \frac{\sum_{u\in N_v^{\phi(x)}\backslash\{x\}} s_{v,u}}{\left(\sum_{u\in N_v^{\phi(x)}} s_{v,u}\right)^2}\mathbf{h}_x, \tag{21}$$

where "diag($\mathbf{z}$)" returns a diagonal matrix whose diagonal entries are $\mathbf{z}$, "$\circ$" is an element-wise multiplication, and $R\left(W'_{\phi(v)}, c_i\right)$ returns a sub-matrix of $W'_{\phi(v)}$ from row 1 to row k and from column $(i-1)k+1$ to column ik (*i.e.*, the columns about type c_i).

Finally, we compute gradients for Θ w.r.t. its regularizer $\Omega(\Theta)$. For typical regularizers, such as ℓ_2- or ℓ_1-norm, it is straightforward to compute their gradients $\nabla_\Theta\Omega(\Theta)$. Hence, we skip the details here.

Convergence. Because $L_1 \geq 0$, $L_2 \geq 0$ and $\Omega \geq 0$, we have $L(\Theta, H) \geq 0$. We start with initializing Θ and H, and continuously minimize $L(\Theta, H)$. Since $L(\Theta, H)$ is bounded, we reach convergence eventually.

VI. Experiments

In this section, we study the empirical performance of HEP on the real-world Taobao datasets.

A. Datasets and Settings

Dataset generation. We prepared two datasets, from one week's user activity logs from Taobao in a city of China. We processed these logs in a format as Table I shows. Each record of the logs shows whether it comes from a mobile device (*i.e.*, MID) or a PC device (*i.e.*, PID). We then used Taobao accounts to annotate these records. That is, for each MID or PID, if we observed any Taobao account login , we associated all its activity log records with that Taobao account. Generally, it is possible that one MID/PID is associated with multiple Taobao accounts, if several real world persons logged into Taobao on the same device (*e.g.*, on public computers). In this study, we only kept those devices (and their activity records) which are associated with one Taobao account. Furthermore, we removed the activity records that are not annotated with any Taobao account as well as those associated with only PID or MID.

TABLE III
DATASETS AND LABEL STATISTICS.

	#record	#PID	#MID	#IP	#shop	#auction	#keyword	#pos	#neg
TB-Top	73,394K	204K	53K	277K	1,125K	3,718K	1,317K	147K	10,611K
TB-Rnd	31,202K	99K	46K	167K	495K	1,082K	437K	57K	2,363K

Finally, to ensure data quality, we removed the activity records whose PID or MID are "abnormal" (*e.g.*, per day occurrence in the log exceeds one million, *etc*.). After filtering, we obtained an annotated subset of the original user activity log. In this subset, each record is annotated with one Taobao account, so does each PID or MID. We denote the set of distinct Taobao accounts as $\mathcal{U}$, and the set of annotated records as $\mathcal{X}$. We used $\mathcal{U}$ and $\mathcal{X}$ to prepare two datasets for experiments as follows, which are summarized in Table III.

• **TB-Top**: We selected top 10% "active" Taobao accounts from $\mathcal{U}$ (totaling 45K accounts), and used their corresponding records from $\mathcal{X}$ as our first dataset. Activeness is defined by the account's number of associated activity records.

• **TB-Rnd**: We randomly sampled 10% accounts from $\mathcal{U}$, and used their records from $\mathcal{X}$ as our second dataset.

Labels. For both TB-Top and TB-Rnd, we prepared tuples $\mathcal{D} = \{(v_i, u_i, y_i)\}$, where v_i is a PID, u_i is an MID, and $y_i \in \{+1, -1\}$, as labels for training and evaluation. Specifically, for each dataset, we employed a set of practical rules that are currently used in production at Taobao: each PID and MID in a candidate pair must co-occur with the same IP(s), the same router(s), *etc*. We label a candidate pair as +1 iff both the PID and MID are associated with the same Taobao account. We summarize the label statistics in Table III.

We randomly split about 50% of the tuples in $\mathcal{D}$ for training, 25% for validation and the remaining 25% for testing. We repeated the split for five times, and reported the average results of these data splits for each method.

B. Performance Comparison

Baselines for data model. We first design baselines to validate the choice of HIG as the data model for the user activity log.

• *FEM*: Feature Engineering Method represents each user ID with four feature vectors, each of which describes when and how frequently the user browsed each item ID of a particular type (*i.e.*, IP, shop, auction and keyword). As the resulting feature vectors are very high dimensional and sparse, we further hashed[1] each feature vector into 128 dimensions. After that, to feed the four feature vectors of a PID and those of an MID to a classifier, we chose[2] to compute the cosine similarity between each feature vector for PID and the corresponding one for MID. In the end, we obtained a 4-dimensional vector for binary classification with logistic regression.

[1]https://en.wikipedia.org/wiki/MurmurHash

[2]We also tested other strategies, such as feature concatenation or feature differences between PID and MID, but their performances were not as good.

• *LDA*: Latent Dirichlet Allocation [2] sees each user ID as a "document", and its co-occurred item IDs as "words". For each type of items, we learned a 128-dimensional topic distribution vector for each user ID by LDA. We performed the final classification similar to FEM.

• *GRU*: Gated Recurrent Unit [3] is a recurrent neural network, and it is used to model the sequence of user activity logs for each user ID. To avoid having over-lengthy sequences for each user ID, we discretized time into half-day slots. In each time slot, we used FEM to similarly obtain a 128-dimensional feature vector for each type of item ID, and then concatenated them as inputs for GRU. We fed the last hidden output of a PID's GRU and that of an MID's GRU with a classifier, as defined in Eq. 2, for end-to-end training.

Baselines for technical model. We compare to state-of-the-art baselines to validate the use of HEP as the technical model for addressing node heterogeneity and edge features.

• *Metapath2vec*: we applied Metapath2vec [6] on HIG. Specifically, we first designed a set of meta-path patterns, including "user-item-user", "user-item-item-user" and "user-item-user-item-user", where "user" can be PID or MID, "item" can be IP, shop, auction or keyword. Then, we sampled path instances from HIG for these meta-path patterns, and fed them into Metapath2vec to learn a 128-dimensional embedding for each user ID. Finally, we trained a classifier, as defined in Eq. 2, based on these user ID embedding.

• *EP*: we adapted Embedding Propagation [17] on HIG as well. Specifically, we reconstructed each node's embedding by mean pooling of all its neighbors' embedding without differentiating node heterogeneity or edge weight. As EP is unsupervised, we trained a separate classifier on the learnt user ID embedding, as defined in Eq. 2.

• *HEP-*: we designed a weaker variant of HEP, where edge features are ignored and all edge weights are treated as one.

Hyperparameter settings. We tuned the hyperparameters for all the methods. For logistic regression used in FEM and LDA, we used ℓ_2 regularization with weight 1.0. For LDA, we used Google's implementation[3], and set its $\alpha = 0.4$, $\beta = 0.01$, and number of burn-in iterations to 150. For GRU, we used the TensorFlow implementation. For Metapath2vec, we set the sampled path length as 20, number of paths per node as 20, context window size as 2 and number of negative samples as 64. For EP, we also adopted a similar negative sampling design as HEP (Eq. 9) to evaluate its reconstruction loss. We set the hinge loss slack $\gamma = 0.1$ and number of negative samples as $\zeta = 5$. For both HEP- and HEP, we set $d = 34$, $k = 128$, $\alpha = 0.1$, $\beta = 0.1$, $\gamma = 0.1$ and $\zeta = 5$.

Results and analysis. We report the results of our methods HEP and HEP-, as well as the baselines, in Table IV. Overall, HEP outperforms all competitors in F1 scores. Next, we discuss why each method works or fails in our task.

[3]https://github.com/openbigdatagroup/plda

TABLE IV
COMPARISON WITH BASELINES.

	TB-Top			TB-Rnd		
	Precision	Recall	F1	Precision	Recall	F1
FEM	60.3	3.4	6.4	68.7	1.9	3.7
LDA [2]	70.4	10.6	18.5	68.3	6.1	11.3
GRU [3]	51.8	26.2	34.8	52.6	22.1	31.2
Metapath2vec [6]	1.7	62.9	3.4	2.3	58.7	4.4
EP [17]	34.3	6.7	11.2	35.0	6.1	10.4
HEP- (our variant)	32.9	31.3	32.1	34.7	25.0	29.0
HEP (our model)	36.5	39.2	**37.8**	44.5	40.5	**42.4**

1) Validation of Data Model:

• FEM achieves high precision, but has very low recall on both datasets, as shown in Table IV. This coincides with our intuitions, as discussed in Sect. I. On the one hand, FEM's features are rich to describe the interactions between users and items, leading to high precision. On the other hand, FEM overlooks items with similar semantics (*e.g.*, a keyword "shoes" and an auction about Nike), resulting in low recall.

• LDA improves recall over FEM without compromising precision. The reason is LDA benefits from item co-occurrence through topic modeling. With a better understanding of the item semantics (*i.e.*, similar items are grouped into "topics"), LDA is able to represent each user ID with a more compact feature vector about how the user interacts with different item groups instead of individual items.

• GRU achieves much higher recall than FEM and LDA. The improvement on recall is credited to learning meaningful representations from its input FEM features through neural network. Overall, GRU still has a limitation of not fully exploiting the relations among users and items as GRU treats each user's sequence as independent.

• HEP is better than the above in terms of recall and F1, because it can fully exploit the HIG's relational structure to learn both item-item relations and user-item relations. Besides, HEP is able to explicitly consider the time information encoded in the edge features. On the contrary, GRU only models each user's activity as sequences, losing valuable temporal information (*e.g.*, time of day and day of week).

2) Validation of Technical Model:

• Metapath2vec achieves very low precision and high recall. The high recall is credited to exploiting the relations among users. In particular, we used meta-path patterns to guide the path sampling. These meta-path patterns tend to relate user ID's who browse a similar set of items.

• EP achieves relatively high precision but low recall. The high precision is credited to exploiting both item-item and user-item relations. EP also reconstructs a user ID by all its interacted items, and propagates the embedding over the graph. Hence it is likely to well characterize each user ID and achieve higher precision. However, since EP overlooks node heterogeneity and edge features, it cannot fully leverage the HIG.

• HEP- and HEP both achieve relatively high precision and recall. HEP also outperform all the baselines in terms of F1. Compared with Metapath2vec, our methods are free from path sampling, which is less ideal when there are also edge features. Besides, Metapath2vec still embed users and items in the same space, and thus does not fully account for node heterogeneity. Compared with EP, we are advantageous for modeling node heterogeneity and edge features. Additionally, HEP- and HEP are trained in an end-to-end framework, whereas EP is unsupervised. Finally, HEP outperforms HEP-, demonstrating the need to model edge features.

VII. CONCLUSION

In this paper, we study an important task of e-commerce user alignment, to classify whether user identities across devices are about the same real-world person. We chose to model the user activity logs as a heterogeneous interaction graph (HIG), and formulated the task as a semi-supervised HIG embedding problem. To address the challenges of node heterogeneity and edge features, we proposed a novel heterogeneous embedding propagation (HEP) model. Finally, experiments on Taobao datasets showed that HEP can significantly outperform state of the art in terms of F1 scores.

REFERENCES

[1] Y. Zhang, Y. Xiong, X. Kong, and Y. Zhu, "Learning node embeddings in interaction graphs," in *CIKM*, 2017, pp. 397–406.

[2] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," *JMLR*, vol. 3, pp. 993–1022, 2003.

[3] J. Chung, Ç. Gülçehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *CoRR*, vol. abs/1412.3555, 2014.

[4] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," in *KDD*, 2016, pp. 855–864.

[5] L. F. Ribeiro, P. H. Saverese, and D. R. Figueiredo, "Struc2vec: Learning node representations from structural identity," in *KDD*, 2017, pp. 385–394.

[6] Y. Dong, N. V. Chawla, and A. Swami, "metapath2vec: Scalable representation learning for heterogeneous networks," in *KDD*, 2017, pp. 135–144.

[7] S. Chang, W. Han, J. Tang, G. Qi, C. C. Aggarwal, and T. S. Huang, "Heterogeneous network embedding via deep architectures," in *KDD*, 2015, pp. 119–128.

[8] S. Zhang and H. Tong, "FINAL: fast attributed network alignment," in *KDD*, 2016, pp. 1345–1354.

[9] X. Kong, J. Zhang, and P. S. Yu, "Inferring anchor links across multiple heterogeneous social networks," in *CIKM*, 2013, pp. 179–188.

[10] M. Zhang and Y. Chen, "Weisfeiler-lehman neural machine for link prediction," in *KDD*, 2017, pp. 575–583.

[11] B. Perozzi, R. Al-Rfou, and S. Skiena, "Deepwalk: Online learning of social representations," in *KDD*, 2014, pp. 701–710.

[12] H. Cai, V. W. Zheng, and K. C. Chang, "A comprehensive survey of graph embedding: Problems, techniques and applications," *TKDE*, 2018.

[13] R. Xie, Z. Liu, J. Jia, H. Luan, and M. Sun, "Representation learning of knowledge graphs with entity descriptions," in *AAAI*, 2016, pp. 2659–2665.

[14] M. Niepert, M. Ahmed, and K. Kutzkov, "Learning convolutional neural networks for graphs," in *ICML*, 2016, pp. 2014–2023.

[15] D. Wang, P. Cui, and W. Zhu, "Structural deep network embedding," in *KDD*, 2016, pp. 1225–1234.

[16] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei, "Line: Large-scale information network embedding," in *WWW*, 2015, pp. 1067–1077.

[17] A. García-Durán and M. Niepert, "Learning graph representations with embedding propagation," in *NIPS*, 2017, pp. 5125–5136.