A Semi-Supervised and Inductive Embedding Model for Churn Prediction of Large-Scale Mobile Games

Xi Liu*
Texas A&M University
College Station, TX, USA
xiliu.tamu@gmail.com

Muhe Xie
Samsung Research America
Mountain View, CA, USA
muhexie@gmail.com

Xidao Wen*
University of Pittsburgh
Pittsburgh, PA, USA
xidao.wen@pitt.edu

Rui Chen
Samsung Research America
Mountain View, CA, USA
rui.chen1@samsung.com

Yong Ge
The University of Arizona
Tucson, AZ, USA
yongge@email.arizona.edu

Nick Duffield
Texas A&M University
College Station, TX, USA
duffieldng@tamu.edu

Na Wang
Samsung Research America
Mountain View, CA, USA
na.wang1@samsung.com

***Abstract*—Mobile gaming has emerged as a promising market with billion-dollar revenues. A variety of mobile game platforms and services have been developed around the world. One critical challenge for these platforms and services is to understand user churn behavior in mobile games. Accurate churn prediction will benefit many stakeholders such as game developers, advertisers, and platform operators. In this paper, we present the first large-scale churn prediction solution for mobile games. In view of the common limitations of the state-of-the-art methods built upon traditional machine learning models, we devise a novel semi-supervised and inductive embedding model that jointly learns the prediction function and the embedding function for user-app relationships. We model these two functions by deep neural networks with a unique edge embedding technique that is able to capture both contextual information and relationship dynamics. We also design a novel attributed random walk technique that takes into consideration both topological adjacency and attribute similarities. To evaluate the performance of our solution, we collect real-world data from the Samsung Game Launcher platform that includes tens of thousands of games and hundreds of millions of user-app interactions. The experimental results with this data demonstrate the superiority of our proposed model against existing state-of-the-art methods.**

***Index Terms*—Churn prediction, representation learning, graph embedding, inductive learning, semi-supervised learning, mobile apps**

I. Introduction

With the wide adoption of mobile devices (e.g., smartphones and tablets), mobile gaming has created a billion-dollar market around the globe. According to Newzoo's Global Games Market Report [1], mobile games generated $50.4 billion revenue in 2017. And it is expected that mobile games will generate $72.3 billion revenue in 2020, accounting for more than half of the overall game market. This increasingly vital market has driven software and hardware providers of mobile devices (e.g., Apple, Google and Samsung) to provide integrated mobile game platforms and services for end users, game developers and other stakeholders.

Within these mobile game platforms and services, one particularly vital task is predicting churn in gaming apps. Churn prediction is a long-standing and important task in many traditional business applications [2], the objective of which is to predict the likelihood that a user will stop using a service or product. In our problem setting, the aim is to predict the likelihood that a user will stop using a particular game app in the future. Churn prediction is an important problem for the following reasons. First, the churn rate of a mobile game app is an important business metric to measure the success of the game. Successfully estimating the churn probability of all player-game pairs will allow a game platform to better prioritize its resources for operation and management. Second, by predicting individual churn probabilities, game platforms will be able to devise better marketing strategies to improve user retention. Examples include sending push notifications and providing free items in games to users who are likely to churn. Since, as well known, the acquisition cost for new users is much higher than the retention cost for existing users, successful churn prediction could greatly reduce costs for game developers and platform operators. Third, churn prediction provides direct input to determine the right timing of app recommendation for a game platform. The results of this research will enable testing of the hypothesis that a user is more likely to act on an app recommendation when he/she is about to stop playing other games.

There have been several previous studies [3]–[13] on mobile game churn prediction by using traditional machine learning models (e.g., logistic regression, random forests, Cox regression). However, we observe several major limitations of these studies. First, they were developed for predicting churn of one or a few mobile games; none is capable of handling the churn prediction of large-scale mobile apps and users. For real-world applications, a solution needs to handle tens of thousands of mobile games and hundred of millions of user-app interactions on a daily basis. Second, user-app interaction data often comes with rich contextual information (e.g., WiFi connection status, screen brightness, and audio volume), which has never been

* This work was done during the authors' internship at Samsung Research America

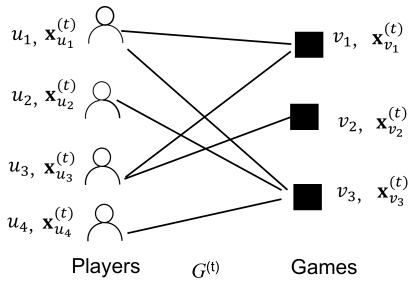


Fig. 1. An example of an attributed bipartite graph for mobile game churn prediction

considered in the existing studies. Third, the existing methods rely on handcrafted features that usually cannot scale well in practice.

To overcome all these limitations, we propose a novel inductive semi-supervised embedding model that *jointly* learns the prediction function and the embedding function for user-game interaction. The user-app interaction data includes detailed information of opens, closes, installs and uninstalls of game apps for each individual user. This data is collected from the Samsung Game Launcher platform[1], which is pre-loaded in most smartphones manufactured by Samsung. We model the interplay between users and games by an attributed bipartite graph and then learn these two functions by deep neural networks with a unique embedding technique that is able to capture both contextual information and dynamic user-game interaction. Our method is fully automatic and can be easily integrated into existing mobile platforms.

Contributions. Our research contributions are as follows:

1) To the best of our knowledge, this paper is the first to develop a solution for churn prediction of large-scale mobile games using hundreds of millions of user-app interaction records. This solution has been tested in Game Launcher, one of the largest commercial mobile gaming platforms. Although the paper mainly applies the proposed solution in mobile game churn prediction, the solution is also applicable to churn prediction in other contexts.

2) We propose a novel semi-supervised and inductive model based on embedding. Our model can capture the dynamics between users and mobile games based on the introduced temporal loss in the formulated objective function. The model is able to embed new users or games not used in training. This is critical for mobile game churn prediction because new games and users continually enter the market.

3) We develop an attributed random walk technique that enables us to sample the contexts of edges in an attributed bipartite graph and that takes into account both topological adjacency and attribute similarities.

4) We conduct a comprehensive experimental evaluation with large-scale real-world data collected from Samsung Game Launcher. The experimental results demonstrate that our model outperforms all state-of-the-art methods with respect to different evaluation metrics for prediction.

[1]https://www.samsung.com/au/support/mobile-devices/how-to-use-game-tools/

TABLE I
NOTATIONS AND DEFINITIONS

Notations	Descriptions or Definitions
$\mathcal{G}^{(t)}$	Attributed graph at time t
$\mathcal{G}^{(t+1)}$	Attributed graph at time $t+1$
$\mathcal{H}^{(t)}$	Historical attributed graphs $\{\mathcal{G}_i\}_{i=0}^{i=t}$
$U^{(t)}$	Set of all player nodes in $\mathcal{G}^{(t)}$
$V^{(t)}$	Set of all game nodes in $\mathcal{G}^{(t)}$
$E^{(t)}$	Set of all edges in $\mathcal{G}^{(t)}$
$e_{uv}^{(t)}$	Indicator of the existence of edge (u,v) in $\mathcal{G}^{(t)}$
$\mathbf{x}_u^{(t)}$	Feature vector of user $u \in U^{(t)}$
$\mathbf{x}_v^{(t)}$	Feature vector of game $v \in V^{(t)}$
$\mathbf{z}_{uv}^{(t)}$	Aggregated feature vector of edge $(u,v) \in E^{(t)}$
n_u, n_v	Numbers of attributes in $\mathbf{x}_u^{(t)}$ and $\mathbf{x}_v^{(t)}$, resp.
d	Number of attributes in $\mathbf{z}_{uv}^{(t)}$
m	Embedding dimension
g	The edge embedding function $g: \mathcal{R}^d \rightarrow \mathcal{R}^m$
f	The churn prediction function $f: \mathcal{R}^m \rightarrow [0,1]$
l_n	Number of prediction hidden layers in Part III
l_p	Number of embedding hidden layers in Part I

II. Problem Formulation

In the context of mobile games, *churn* is defined as a player stopping using a game within a given period (i.e., there is no app usage in the period). The duration T of the period may vary from application to application depending on different business goals. $T = 14$ days and $T = 30$ days are some typical settings used in industry [3, 4, 8]. In this paper, we consider the generic game churn prediction problem without assuming any particular value of T. We note that uninstall is different from churn. Regarding only uninstall as churn would be problematic since there may be a large time gap between cessation of playing and uninstall, if any.

The relationship between players and games can be represented by an attributed bipartite graph as illustrated in Fig. 1, whose two parts correspond to players and games. In the sequel, we use the terms *player* and *user*, and *game* and *app* interchangeably. Let $\mathcal{G}^{(t)}$ be the attributed bipartite graph at time t, the vertex set $U^{(t)}$ denote the set of users and the vertex set $V^{(t)}$ denote the set of games. A player is represented by a node $u \in U^{(t)}$ and a game is represented by a node $v \in V^{(t)}$. Each user u is associated with a feature vector $\mathbf{x}_u^{(t)} \in \mathcal{R}^{n_u}$, where n_u is the size of $\mathbf{x}_u^{(t)}$; each game v is associated with a feature vector $\mathbf{x}_v^{(t)} \in \mathcal{R}^{n_v}$, where n_v is the size of $\mathbf{x}_v^{(t)}$. There is an edge between nodes u and v in $\mathcal{G}^{(t)}$ if player u has played v in the time window $[t+1, t+T]$. The set of edges is denoted by $E^{(t)}$.

Now we are ready to define the mobile game churn prediction problem as follows[2].

Definition 1 (*Mobile game churn prediction*)**.** Consider a collection of attributed bipartite graphs observed from time

[2]This problem definition is indeed generic to be applied to churn prediction in other contexts, for example, general app churn prediction.

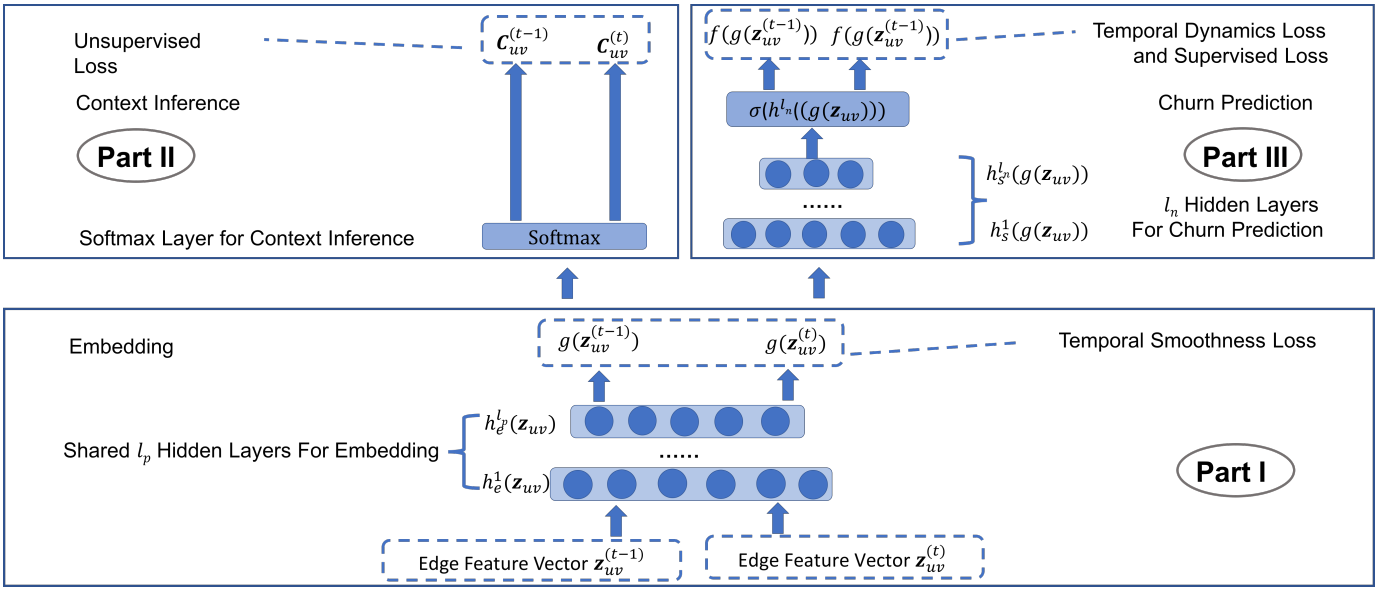


Fig. 2. Deep neural network architecture of the inductive semi-supervised embedding model in training

t_0 to time t ($t > t_0$), which is denoted by $\mathcal{H}^{(t)} = \{\mathcal{G}^{(i)}\}_{i=t_0}^{i=t}$. Let $e_{uv}^{(i)}$ be the indicator of the existence of edge (u, v) in $\mathcal{G}^{(i)}$. For any edge (u, v) in $\mathcal{G}^{(t)}$, predict the probability $Pr(e_{uv}^{(t+1)} = 0 | e_{uv}^{(t)} = 1, \mathcal{H}^{(t)})$, that is, the probability that (u, v) disappears in $\mathcal{G}^{(t+1)}$.

The notations and their descriptions are listed in TableI.

III. Methods

A. Overview of Our Solution

Most existing works [3]–[13] rely on traditional machine learning models to solve the churn prediction problem, which suffer from several major limitations identified in Section I. In view of the recent progress in deep learning and graph embedding, a natural promising direction is to adopt graph embedding frameworks for churn prediction. However, we face several key technical challenges that have not been addressed by existing research: (1) All existing methods are transductive, and thus cannot produce embeddings for new player-game pairs. (2) Existing methods are either purely supervised or unsupervised, and thus do not take full advantage of relevance between embedding and a task. (3) Existing methods are node-centric, and thus are not directly applicable to edge related tasks. (4) Existing methods only handle a static graph and do not incorporate graph dynamics in embedding. In the mobile game industry, players and games, however, change very quickly. There are new players, new games, and new player-game relationships every day.

In addressing these challenges, we propose a novel inductive semi-supervised embedding model in dynamic graphs that jointly learns the prediction function f and the embedding function g. The prediction function f and the embedding function g are learned by deep neural networks (DNN). The architecture of the proposed DNN is presented in Fig. 2. The DNN consists of three parts. Part I is responsible for producing embedding feature vectors $g(\mathbf{z}_{uv}^{(t)})$ from raw edge feature vectors $\mathbf{z}_{uv}^{(t)} \in \mathcal{R}^d$, where d is the size of raw edge feature vectors. To learn the probability of churn, we need to construct a feature vector for each (u, v) with $e_{uv}^{(t)} = 1$. However, it is impractical to calculate features for all possible edges which may appear in the prediction period because the number of possible edges is large, which is $\mathcal{O}(|U^{(t)}| \cdot |V^{(t)}|)$. Instead, we construct the feature vector $\mathbf{z}_{uv}^{(t)}$ of (u, v) from attribute-wise cosine similarity aggregation of $\mathbf{x}_u(t)$ and $\mathbf{x}_v(t)$. As such, all $\mathbf{z}_{uv}^{(t)}$ can readily computed from $|U^{(t)}| + |V^{(t)}|$ node features, where $|\cdot|$ denotes the cardinality of a set.

Part II is in charge of inferring contexts from embedding feature vectors. Here the context of an edge refers to the edges that are similar to and co-occur with the edge under some graph sampling strategy, for example, random walk. Part I and Part II make up the *unsupervised* component of our model. They are jointly trained by minimizing the error due to incorrect context inference and inconsistency with temporal smoothness (see Section III-C for an explanation of temporal smoothness). Part I and Part II are trained in an inductive and edge-centric way. In contrast to transductive node embedding that learns a distinct embedding vector for each node, our idea is to learn an embedding function that generalizes to any *unseen* edges as long as their feature vector is available. Since producing contexts of edges has not be previously studied, we propose a novel attributed random walk to sample similar edges as contexts (see Section III-D for details).

Part III fulfills the supervised churn prediction task from embedding feature vectors. Part III forms the *supervised* component of the proposed model, which is trained by minimizing the error of incorrect churn predictions. The supervised component and unsupervised component are simultaneously trained in a single objective function. Traditional unsupervised embedding techniques are not designed in a task-specific way

and hence are not able to incorporate task-specific information to improve performance. In contrast, in our model Part III and Part II share the common hidden layers in Part I, and therefore they are latently coupled with each other. This helps the embedding align with the supervised prediction task.

Part I and Part III both consider graph dynamics in training. Part I handles graph dynamics by requiring the embeddings of the same edge at two consecutive timestamps to stay close. Part III handles graph dynamics by requiring the churn probabilities of the same edge at two consecutive timestamps to follow a decaying pattern.

The objective function of our model is composed of four parts:

$$L := L_S + \alpha L_U + \beta L_T + \gamma L_R. \tag{1}$$

L_S denotes the supervised loss due to incorrect predictions and will be discussed in Section III-B1. L_U denotes unsupervised loss, which comes from failures of context inference and will be addressed in Section III-B2. L_T is the temporal loss that consists of two parts: temporal smoothness and temporal dynamics, and will be explained in Section III-C. L_R presented in Section III-B3 is the regularization term, and (α, β, γ) are trade-off weights.

B. Static Loss Functions

1) Supervised Loss Function L_S: The supervised loss function L_S is designed for Part III. Let $h_s^k(g(\mathbf{z}_{uv}^{(t)})) = \phi(W_s^k h_s^{k-1}(g(\mathbf{z}_{uv}^{(t)})) + b^k)$ be the k-th hidden layer for churn prediction (referred to as *prediction hidden layer* in the sequel), where W^k and b^k are the weights and biases in the k-th prediction hidden layer, and $\phi(\cdot)$ is a non-linear activation function. We model the churn prediction function f by l_n such layers in Part III. Then the prediction output layer can be represented by:

$$\begin{aligned} f(g(\mathbf{z}_{uv}^{(t)})) &:= Pr(e_{uv}^{(t+1)} = 0 | e_{uv}^{(t)} = 1, \mathcal{H}^{(t)}) \\ &:= \sigma(h_s^{l_n}(g(\mathbf{z}_{uv}^{(t)}))) \\ &:= \frac{\exp\left(h_s^{l_n}(g(\mathbf{z}_{uv}^{(t)}))^T w_s\right)}{1 + \exp\left(h_s^{l_n}(g(\mathbf{z}_{uv}^{(t)}))^T w_s\right)} \end{aligned} \tag{2}$$

where $\sigma(\cdot)$ is the sigmoid function and w_s is the sigmoid weights vector that combines the output from the last hidden layer to predict churn. Now we can define the supervised loss L_S as follows:

$$\begin{aligned} L_S = \frac{1}{L} \sum_{i=0}^{t-1} \sum_{(u,v)\in E^{(i)}} \delta_{uv}^{(i+1)} \Big(1 - e_{uv}^{(i+1)} - \\ \frac{\exp\left(h_s^{l_n}(g(\mathbf{z}_{uv}^{(i)}))^T w_s\right)}{1 + \exp\left(h_s^{l_n}(g(\mathbf{z}_{uv}^{(i)}))^T w_s\right)}\Big)^2, \end{aligned} \tag{3}$$

where L is the number of training examples and $\delta_{uv}^{(i+1)}$ is a censoring indicator and will be discussed in Section III-C.

2) Unsupervised Loss Function L_U: The unsupervised loss function L_U is devised for Part II, which guides to embed the handcrafted features $\mathbf{z}_{uv}^{(t)} \in \mathcal{R}^d$ into a latent space $g(\mathbf{z}_{uv}^{(t)}) \in \mathcal{R}^m$, where m is the size of the latent space. Denote the k-th hidden layer for embedding (referred to as *embedding hidden layer* in the sequel) by $h_e^k(\mathbf{z}_{uv}^{(t)}) = \phi(W_e^k h_e^{k-1}(\mathbf{z}_{uv}^{(t)}) + b_e^k)$, where W_e^k and b_e^k are the weights and biases in the k-th embedding hidden layer. We use the l_p layers in Part I to represent the embedding function g, and the embedding output layer can be represented by:

$$g(\mathbf{z}_{uv}^{(t)}) := h_e^{l_p}(\mathbf{z}_{uv}^{(t)}). \tag{4}$$

We can define the unsupervised loss function as follows:

$$L_U = -\sum_{i=t_0}^{t} \sum_{(u,v)\in E^{(i)}} \sum_{(u',v')\in C_{uv}^{(i)}} \delta_{uv}^{(i)} \log\Big(\Pr\left((u',v') | g(\mathbf{z}_{uv}^{(i)})\right)\Big), \tag{5}$$

where $C_{uv}^{(i)}$ denotes the context (i.e., contextual edges) of (u, v) in $\mathcal{G}^{(i)}$. The contextual edges $C_{uv}^{(i)}$ are obtained by attributed random walk on the bipartite graph, which will be discussed in Section III-D.

The likelihood of having a contextual edge (u', v') of (u, v) conditional on the embedding of (u, v) is:

$$Pr\big((u',v') | g(\mathbf{z}_{uv}^{(i)})\big) = \frac{\exp\left(g(\mathbf{z}_{uv}^{(i)})^T w_{u'v'}\right)}{\sum_{(u^*,v^*)} \exp\left(g(\mathbf{z}_{uv}^{(i)})^T w_{u^*v^*}\right)}, \tag{6}$$

where $w_{u^*v^*}$ and $w_{u'v'}$ are the vectors of weights for edges (u^*, v^*) and (u', v') in the softmax layer, respectively. The denominator is computed by negative sampling [14]. Although the embedding function is learned by training a context inference task, it is still considered as "unsupervised" because the contexts are calculated by sampling on the attributed graph, which is independent of any supervised learning task [15].

The objective function in Equation (1) contains both the supervised loss function and the unsupervised loss function. Thus the embedding hidden layers are jointly trained with prediction hidden layers. Compared to traditional embedding methods, the semi-supervised approach makes the embedding more suitable for the prediction task.

Note that the target of the embedding process is to learn a mapping function from the feature space to the embedding space, instead of directly learning the embeddings. Thus, the input for the embedding hidden layers only contains attributes. A new edge can be embedded for churn prediction as long as we can observe its attributes. This indicates that the proposed approach is inductive. It is able to produce the embedding for an edge that is not used in training.

3) Regularization Loss L_R: Regularization loss is introduced mainly to avoid over-fitting. The weights for regularization consists of $\{\{W_e^i, b_e^i\}_{i=1}^{l_p}, \{W_s^i, b_s^i\}_{i=1}^{l_n}, w_s\}$. Therefore the regularization part can be expressed as:

$$\begin{aligned} L_R \quad := \quad & \lambda_0 \sum_{i=1}^{l_p} \|W_e^i\|_2^2 + \lambda_1 \sum_{i=1}^{l_p} \|b_e^i\|_2^2 + \lambda_2 \sum_{i=1}^{l_n} \|W_s^i\|_2^2 \\ & + \lambda_3 \sum_{i=1}^{l_n} \|b_s^k\|_2^2 + \lambda_4 \|w_s\|_2^2, \end{aligned} \tag{7}$$

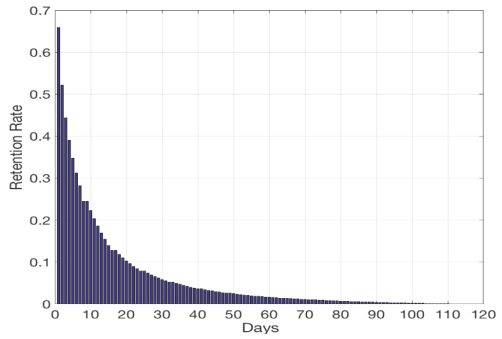


Fig. 3. Average mobile game retention rate as a function of time

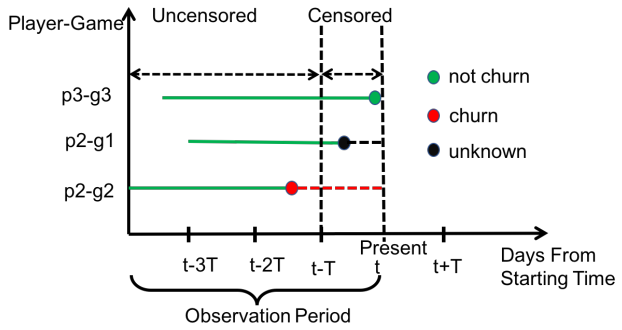


Fig. 4. An illustration of censored data by three randomly sampled player-game pairs

where $\{\lambda_i\}_{i=0}^4$ are trade-off weights on different regularization terms.

C. Temporal Loss Function

Temporal loss refers to the loss related to graph dynamics. Different from existing works [14]–[16], the proposed embedding model takes into account graph dynamics. Understanding the temporal dynamics of attributed bipartite graphs is crucial to precisely model the churn behavior. We make several key observations of the temporal dynamics, which help to achieve good prediction performance in practical settings.

Observation 1. For a given user-game play relationship, the longer the relationship exists, the more likely the user is to churn the game.

This is because the content of a mobile game is usually somewhat fixed. Players can easily lose interests after going through all contents and passing all levels in the game, let alone many players churn before passing all levels. With more days of play, their initial interests in the game are gradually effacing. Indeed, 71% of all mobile app users churn within 90 days [17]. The churn rate of mobile games is even higher. We plot the average retention rate of mobile games as a function of time, based on Game Launcher data in Fig. 3. It shows that 95% of user-game play relationships end after 40 days, which well justifies our observation. We formally state Observation 1 below.

$$f(g(\mathbf{z}_{uv}^{(i)})) \lessapprox f(g(\mathbf{z}_{uv}^{(i+1)})) \ \forall 0 \leq i \leq t-1, (u,v) \in D^{(i)}, \tag{8}$$

where $D^{(i)} = \{(u,v) : (u,v) \in E^{(i)} \cap E^{(i+1)}\}$ and $\lessapprox$ represents "almost always smaller than". We refer to this observation as *temporal dynamics* in the following discussion.

The second observation we make is stated in Observation 2.

Observation 2. For a given user-game play relationship denoted by an edge in the attributed bipartite graph, its context usually evolves slowly at two consecutive timestamps.

This observation is also derived from the real-world data. Due to space limitation, we omit the figure here. It follows that the topology and attribute values of the attributed bipartite graph mostly evolve smoothly at two consecutive timestamps, resulting in similar contexts for a given edge at consecutive timestamps. Therefore, its embeddings at these two timestamps should also be close, that is,

$$g(\mathbf{z}_{uv}^{(i+1)}) \approxeq g(\mathbf{z}_{uv}^{(i)}) \ \forall 0 \leq i \leq t-1, (u,v) \in D^{(i)}, \tag{9}$$

where $\approxeq$ represents "almost always equal to". We refer to this observation as *temporal smoothness* in the later discussion.

The final observation is:

Observation 3. By definition, churn in nature introduces right censoring to the training dataset.

The problem of right censoring has been widely studied [18] and is illustrated in Fig. 4. The observation period refers to some time duration in history. Suppose we are at time t and the observation period is from time t_0 to time t. Data for training and testing all come from the observation period. Since the label of a player-game pair at a specific timestamp is determined by their interaction in the next T time duration, the labels of some player-game pairs in the last T duration could be *unknown*. For instance, the last observation of the pair of player 2 and game 1 (denoted by p2-g1) was in the last T time duration, and therefore the labels after that time are unknown. This is known as right censoring. In contrast, the pair p3-g3 has play records every day in the last T days, and it is not censored during the observation period.

Considering the existence of censored instances, we introduce a binary indicator $\delta_{uv}^{(i)}$ to indicate whether an edge (u,v) is censored at timestamp i. $\delta_{uv}^{(i)} = 0$ if (u,v) is censored; $\delta_{uv}^{(i)} = 1$ otherwise. Then Inequality(8) needs to be updated by

$$f(g(\mathbf{z}_{uv}^{(i)})) \lessapprox f(g(\mathbf{z}_{uv}^{(i+1)})) \ \forall \ 0 \leq i \leq t_{uv}-1, (u,v) \in D^{(i)} \tag{10}$$

$$f(g(\mathbf{z}_{uv}^{(t_{uv})}) \lessapprox f(g(\mathbf{z}_{uv}^{(i)})) \ \forall \ t_{uv} \leq i \leq t, (u,v) \in E^{(t_{uv})},$$

where t_{uv} denotes the timestamp when the edge (u,v) was observed, i.e., $t_{uv} = \max\{i : \delta_{uv}^{(i)} = 1\}$. The update reflects the fact that after timestamp t_{uv} the label of (u,v) becomes unknown. Since the existence of edge (u,v) after

t_{uv} is unknown, it is more reasonable to just require that Inequality (10) holds *pairwise* between time point t_{uv} and all time points after t_{uv}. Therefore the temporal loss can be expressed as:

$$L_T := \sum_{i=t_0}^{t-1} \sum_{(u,v)\in D^{(i)}} \Big\{ \|g(\mathbf{z}_{uv}^{(i+1)}) - g(\mathbf{z}_{uv}^{(i)})\|_2 + \big[\mathbb{1}(\delta_{uv}^{(i+1)} = 1) f(g(\mathbf{z}_{uv}^{(i)})) + \mathbb{1}(\delta_{uv}^{(i+1)} = 0) f(g(\mathbf{z}_{uv}^{(t_{uv})})) - f(g(\mathbf{z}_{uv}^{(i+1)}))\big]_+ \Big\}, \tag{11}$$

where $\mathbb{1}(A)$ denotes the indicator function for event A and $[x]_+ = \max\{x, 0\}$. The first term corresponds to the temporal smoothness and the second term corresponds to the temporal dynamics. Taking Equations (2) and (4) into Equation (11), we have the temporal loss L_T as follows.

$$L_T := \sum_{i=0}^{t-1} \sum_{(u,v)\in D^{(i)}} \Big\{ \|h_e^{l_p}(\mathbf{z}_{uv}^{(i+1)}) - h_e^{l_p}(\mathbf{z}_{uv}^{(i)})\|_2 + \big[\mathbb{1}(\delta_{uv}^{(i+1)} = 1) \frac{\exp\left(h_s^{l_n}(g(\mathbf{z}_{uv}^{(i)}))^T w_s\right)}{1 + \exp\left(h_s^{l_n}(g(\mathbf{z}_{uv}^{(i)}))^T w_s\right)} + \mathbb{1}(\delta_{uv}(i+1) = 0) \frac{\exp\left(h_s^{l_n}(g(\mathbf{z}_{uv}^{(t_{uv})}))^T w_s\right)}{1 + \exp\left(h_s^{l_n}(g(\mathbf{z}_{uv}^{(t_{uv})}))^T w_s\right)} - \frac{\exp\left(h_s^{l_n}(g(\mathbf{z}_{uv}^{(i+1)}))^T w_s\right)}{1 + \exp\left(h_s^{l_n}(g(\mathbf{z}_{uv}^{(i+1)}))^T w_s\right)} \big]_+ \Big\}. \tag{12}$$

D. Context Generation by Attributed Random Walk

All above discussion assumes the availability of contexts of edges. There have been a series of node-centric research on graph embedding proposing to apply random walk to sample contexts [19]. These methods are normally topology-based. In our problem setting, our goal is to embed an *edge*, not a node. In this case, a simple topology-based random walk may return two adjacent edges having the same player or the same game while totally ignoring the similarity of the other end. This is undesirable. In contrast, attributed random walk measures such similarity by attributes and allows to transit to similar nodes even if they are not connected.

To this end, we propose a novel attributed random walk technique that takes into account both topological adjacency and attribute similarities to make the transition decision of the walk. Fig. 5 gives an illustrative example of attributed random walk in an attributed bipartite graph. For clarity, we omit the time index in the following discussion. The *solid line* indicates that there exists an edge in the attributed bipartite graph. We denote the type of node o by $type(o)$. A node's type can be of value either player or game. The *dashed directed* lines do not exist in the original attributed graph but may be considered as transitions by our attributed random walk due to attribute similarities.

It is time-consuming to calculate pairwise similarities between a node and all other nodes with the same type. For this reason, we add augmented edges for a proportion of the same-type nodes. For example, the augmented edges of node o_1 are $\{(o_1, o) : sim(o_1, o) > 1 - \epsilon, type(o_1) = type(o)\}$, where $0 < \epsilon < 1$ is a filtering parameter and:

$$sim(o_1, o) := \frac{\mathbf{x}_{o_1} \cdot \mathbf{x}_o}{\|\mathbf{x}_{o_1}\| \ \|\mathbf{x}_o\|}. \tag{13}$$

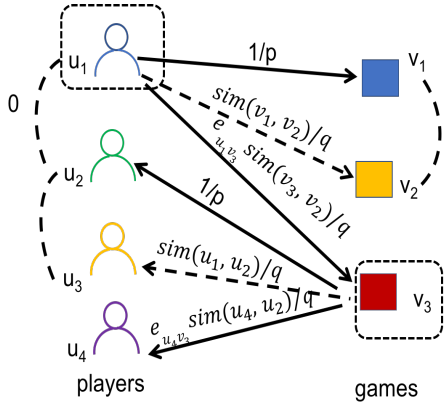


Fig. 5. An illustrative example of attributed random walk in an attributed bipartite graph

The *dashed undirected* lines in Fig. 5 represent the added augmented edges between the nodes and their similar same-typed nodes.

Consider a random walker that just traversed edge (v_1, u_1) in Fig. 5 and now resides at node u_1. Now the walker needs to decide which node to transit to. Since attribute similarities matter in this walk, the walker cannot just evaluate those nodes that are neighbors of u_1 as suggested in [19]. The walker needs to evaluate all nodes of the same type within the two-hop neighborhood of v_1. We denote the one-hop same-typed adjacent nodes of v_1 by $N_1(v_1) = \{v : d(v_1, v) = 1, type(v) = type(v_1)\}$, where $d(v_1, v)$ is the length of shortest path between v_1 and v. For example, $v_2 \in N_1$ (due to the augmented edge between v_2 and v_1). Similarly, we denote the set of two-hop same-typed neighbor nodes of v_1 by $N_2(v_1) = \{v : d(v_1, v) = 2, type(v) = type(v_1)\}$. Then the transition probability in attributed random walk can be calculated as follows:

$$p(o|u_1) = \begin{cases} 0, & \text{if } type(o) \neq type(v_1) \\ \frac{1}{p}, & \text{if } o = v_1 \\ \frac{sim(v_1, o)}{q}, & \text{if } o \in N_1(v_1) \\ \frac{sim(v_1, o) e_{u_1 o}}{q}, & \text{if } o \in N_2(v_1) \end{cases} \tag{14}$$

where p and q are normalization constants used to control the walk strategy. Recall that $e_{u_1 o} = 1$ if there is an edge between u_1 and o. The attributed walk in an attributed bipartite graph walks through different types of nodes repeatedly. It enlarges the probability that any two consecutive edges in a path are similar, and thus the probability that the whole set of edges on the attributed random walk path are similar. As an example, suppose we are at u_1 from v_1 as shown in Fig. (5). Since v_1 and v_2 have high similarities, the walker may transit to v_2 and produce a path with edges (v_1, u_1) and (u_1, v_2) although u_1 is not connected to v_2 in the bipartite graph.

TABLE II
DATASET STATISTICS

Dataset	# of users	# of games	# of play records
USA	15,000	19,705	76,468,301
Korea	25,000	18,470	106,544,313

Our proposed method shares some similarities with the idea of [20]. However, in their work attribute similarities have no influence on which nodes a walker goes through. In our proposed method, when choosing the next node to visit, there is a nonzero probability to choose those that are not connected to the present node but share similar attributes to the previous node. For those that are connected to the present node, the probability to choose from them is weighted by the similarity between the descendant and the ancestor nodes. In this way, an edge can be the context of another even when they are not adjacent but similar in both ends.

IV. Experimental Evaluation

In this section, we conduct a comprehensive experimental evaluation over the large-scale real data collected from the Samsung Game Launcher platform. We compare our semi-supervised model (referred to as `SS` in the sequel) with the following state-of-the-art models for mobile game churn prediction:

- `LR`: the logistic regression based solution used in [5, 9, 13]
- `RS`: the supervised variant of our model, in which the loss function contains only the supervised component and the regularization term
- `DT`: a decision tree based solution
- `RF`: the random forests based solution used in [5, 13]
- `SVM`: the SVM based solution used in [5, 9].

In the experiments, we consider the churn duration $T = 14$, but again the proposed solution is not restricted to any particular choice of T.

A. Dataset and Feature/Label Construction

Two anonymous datasets were collected *independently* from the Samsung Game Launcher platform within a 4-months period (from August 1st, 2017 to November 30th, 2017) with users' consent, one from the users in *USA* and the other from the users in *Korea*. We summarize the key statistics of these two datasets in Table II.

The collected data contains three major types of information: (1) play history, (2) game profiles, and (3) user information. Each play record in the play history contains the anonymous user id, the game package name, and the timestamp of play. It is also accompanied with rich contextual information, such as WiFi connection status, screen brightness, audio volume, etc. Game profiles are collected from different game stores, which include features like genre, developer, number of downloads, rating values, number of ratings, etc. User information contains the device model, region, OS version, etc. However, data within games (e.g., levels) is not available due to privacy concerns.

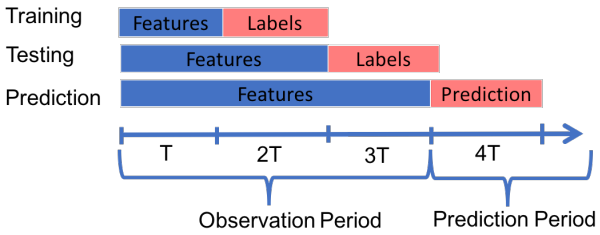


Fig. 6. Feature and label generation process

Features and labels need to be generated with care in order to avoid data leakage. The generation process is illustrated in Fig. 6. Labels and features are taken from disjoint periods to avoid data leakage. The model is trained based on features and labels within the observation period. Labels are whether a player churns a game on that day. Features are constructed from historical data before the day to be predicted. The training set and testing set are split by label days in chronological order [21], for example, taking the labeled data in the first 2/3 of the observation period as the training set and the remaining 1/3 as the testing set. This is also to ensure that there is a time difference between the testing set and the training set.

B. Experimental Settings

We tune the hyperparameters, including learning rate, batch size, regularization terms, number of layers and number of neurons per layer, based on the model performance on the testing datasets. A grid search on those parameters is performed and the combination yielding the best performance is chosen. The regularization parameters $\{\lambda_i\}_{i=0}^{4}$ are all set to be 1. α, β, and γ in Equation (1) are chosen to be 0.02, 0.01, 1e-5, respectively. ϵ, p, and q discussed in Section III-D are set to be 1, 1, and 0.05, respectively. The parameters used for training the deep neural network are summarized below.

Parameter settings of the *USA* dataset:

- Player feature dimension: 10,042
- Game feature dimension: 10,042
- Player-game feature dimension: 30
- Learning rate: the initial value is 0.017 and decay by $\eta = \eta_0/(1 + k/2)$, where k is the number of epochs
- Number of neurons: input layers 30, embedding layers 50, output layers 380K
- Number of epochs: 6-8
- Batch size: 1,024
- Context number per user-game pair: 4
- Optimizer: Adam method [22]
- Activation function: rectified linear unit (ReLU)

Parameter settings of the *Korea* dataset:

- Player feature dimension: 10,042
- Game feature dimension: 10,042
- Player-game feature dimension: 30
- Learning rate: the initial value is 0.019 and decay by $\eta = \eta_0/(1 + k/2)$, where k is the number of epochs
- Number of neurons: input layers 30, embedding layers 50, output layers 632K

TABLE III
PERFORMANCE OF DIFFERENT PREDICTION MODELS ON *USA*

Model	AUC	Recall	Precision
SS	**0.82**	**0.78**	0.32
RS	0.77	0.75	0.27
LR	0.66	0.38	0.26
DT	0.59	0.28	0.32
RF	0.75	0.31	**0.41**
SVM	0.61	0.78	0.18

TABLE IV
PERFORMANCE OF DIFFERENT PREDICTION MODELS ON *Korea*

Model	AUC	Recall	Precision
SS	**0.82**	**0.70**	0.34
RS	0.76	0.70	0.25
LR	0.67	0.59	0.21
DT	0.58	0.26	0.30
RF	0.73	0.27	**0.42**
SVM	0.63	0.67	0.18

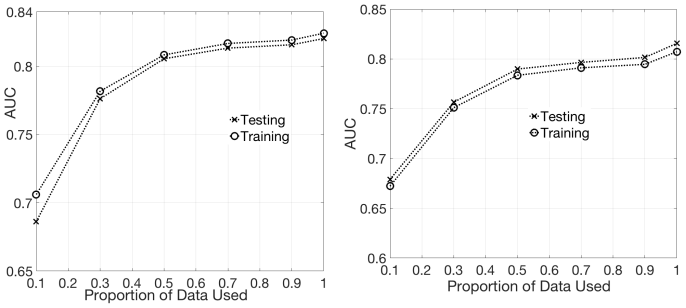
- Number of epochs: 8-12
- Batch size: 4,096
- Context number per user-game pair: 4
- Optimizer: Adam method [22]
- Activation function: rectified linear unit (ReLU)

C. Experimental Results

We use three widely-used evaluation metrics to compare the performance of different models. The most important metric with respect to the business goals is *the area under the ROC curve* (AUC). Following previous studies [5, 9, 13], we also consider *precision* and *recall*. Accuracy is not used because our data is imbalanced with around 85% negative instances in the *USA* dataset and 86% negative instances in the *Korea* dataset.

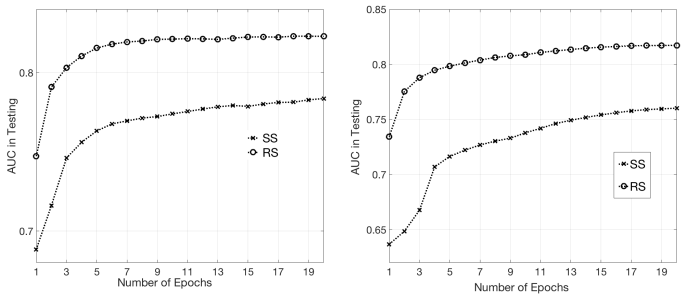
We report the main experimental results in Table III and Table IV. It can be observed that in general our model achieves the best AUC and recall on the two testing datasets. Our model outperforms all single models (i.e., LR, DT and SVM) in terms of all the three metrics. In particular, it is worth mentioning that SVM achieves a high recall at the cost of a very low precision. This is because it makes a large number of false positives. This fact makes it less useful for business decision making. Compared with the ensemble method RF, which has been considered so far the best method in the field, our model still achieves 34% AUC improvement and 250% recall improvement. RF achieves the highest precision on the dataset. However, we point out that this number is actually misleading because it can easily overfit and only recognize a small proportion of churn labels. Meanwhile, its recall is poor, making it difficult to meet business requirements of churn prediction (e.g., targeted promotion campaigns). We also carefully compare the AUC of SS in training and testing in Fig 7. Since the curves are very close, it can be learned that our model is neither overfitting nor underfitting.

The performance difference between SS and RS justifies the benefits of incorporating unsupervised loss and temporal loss in the objective function. We provide a further comparison between SS and RS with respect to the number of epochs in Fig. 8. Both models take 5-7 epochs to reach relatively stable performance. We observe that SS outperforms RS in general under a different number of epochs and for both Korea users and USA users.



(a) AUC on *USA* (b) AUC on *Korea*

Fig. 7. Comparison of AUC for SS in training and testing

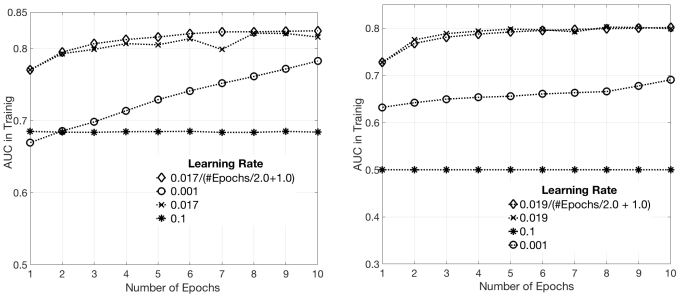


(a) AUC on *USA* (b) AUC on *Korea*

Fig. 8. AUC comparison between SS and RS under different numbers of epochs

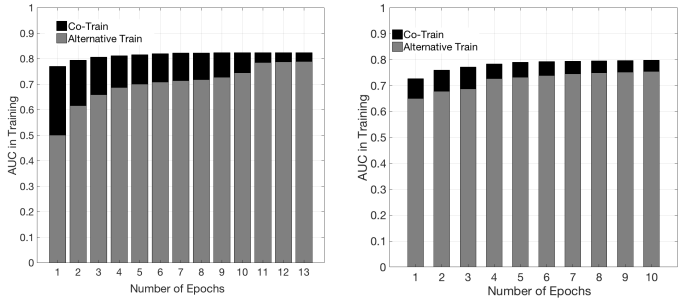
Since the architecture of our DNN is novel and unique (e.g., contain both supervised outputs and unsupervised outputs), we expose more details on how we choose parameters and train the model. A comparison of AUC in training under different learning rates is given in Fig. 9. It can be observed that the choice of the learning rate greatly influences the model performance after the initial epoch. We experimentally find that 0.1 is too large for the learning rate, which makes the step in gradient descent too large to find a good minimum and that 0.001 is too small, making it converge very slowly to the optimal point. Therefore we experimentally test learning rates between 0.1 and 0.001 and find that $0.017/(\#\text{Epochs}/2+1)$ works best for training on *USA* while $0.019/(\#\text{Epochs}/2+1)$ works best for training on *Korea*.

For the supervised component and unsupervised component, we try two different training methods: *co-train* and *alternative train*. Co-train means that we simultaneously train the supervised loss function and the unsupervised loss function; alternative train means that we alternately train the unsupervised component with the unsupervised loss function and the supervised component with the supervised loss function. This is a widely-used training method for similar structures [14, 15].



(a) AUC on *USA* (b) AUC on *Korea*

Fig. 9. Comparison of AUC for SS in training under different learning rates



(a) AUC on *USA* (b) AUC on *Korea*

Fig. 10. Comparison of AUC for SS in training under different training methods

It is interesting to observe that in general co-train outperforms alternative train in terms of AUC under a different number of epochs as shown in Fig. 10. Therefore, we choose co-train as the final training methods in our experiments.

V. Related Work

In this section, we review two categories of existing research that are relevant to this paper. The first category includes the existing works for (game) churn prediction. The early works [3]–[5, 7]–[9] are based on more traditional machine learning models, such as logistic regression, random forests, SVM, naive Bayes, etc., and are experimentally evaluated on extremely small numbers of games (i.e., less than five). As shown in our experiments, their performance on large-scale real data with tens of thousands of mobile games and hundred of millions of user-app interactions is generally not satisfactory. In addition, we also observe scalability issues when they are applied to large-scale data. Some recent research has started to use more advanced models. [6, 10, 12] propose to use survival model for churn prediction, in which churn probability is modeled as a function of playtime. Kim et al. [13] achieve good performance by using convolution neural networks (CNN) and long short-term memory networks (LSTM). There are also several recent deep-learning-based studies [23]–[25] for non-game churn prediction problems, which report better performance. This motivates us to employ deep neural network models. While being a generic solution, our model is able to accommodate the unique characteristics of mobile gaming. We provide a comparison between all existing works and ours in Table V.

TABLE V
Comparison between this paper and existing works in data size and key techniques

Paper	Data size	Key techniques
[3]	5 games 50 thousand users	Decision tree, naive Bayes
[4]	2 games 10 thousand users	Hidden Markov Model combined with a single layer neural network
[6, 10]	1 game 3 thousand users	Survival ensembles
[12]	1 game 1 thousand users	Survival model
[7]	1 game 10 thousand users	Hidden Markov Model
[5]	2 games 1 thousand users	SVM, decision tree, logistic regression
[8]	1 game 130 thousand users	Heuristic decision tree
[9]	3 games 60 thousand users	Logistic regression, decision tree, SVM
[13]	3 games 200 thousand users	Logistic regression, random forests, CNN, LSTM
Ours	**40,000** games 40 thousand users	Deep attributed edge embedding

The second category contains recent works on graph embedding. Graph embedding automates the entire process of feature engineering by casting feature extraction as a representation learning problem. It frees models from human bottleneck introduced by handcrafted features and is able to utilize the full richness of data. However, most existing works such as Node2Vec [19], Deep Walk [26] and LINE [27] are *node-centric* embedding. When it comes to game churn prediction, the entities to be embedded are edges that represent the relationship between players and games. Very limited work has studied edge embedding. Abu-El-Haija et al. [28] propose to model edge embedding as a function of node embedding, in which the two ends of an edge are first embedded and then passed into a deep neural network with edge embedding as output. This method embeds edges in an indirect way and does not take into account any attribute information. In addition, these models [19, 26, 27] all belong to the transductive framework, in which embedding cannot be generated if an object has never appeared in training. However, in our problem new users and new games appear continuously; new relationships between existing users and games may form anytime in the future. Therefore, for game churn prediction the capability of handling new edges is indispensable.

Several very recent works have proposed the idea of inductive graph embedding [14]–[16], inspired by which we propose our inductive edge embedding model for churn prediction. Our model improves these works in several major ways. First, our embedded features are learned in a *semi-supervised* manner, where the supervised component is for churn prediction while the unsupervised component is for context recovery. Compared to unsupervised methods, embedding features learned

in a semi-supervised way have been shown to achieve better performance [15]. Second, our model captures graph dynamics by imposing temporal loss to the embeddings of the same edge in consecutive graph snapshots. Unlike all existing works, where embeddings represent structures or attribute information, embeddings in our model are designed to simultaneously capture contexts and graph dynamics. To our best knowledge, our model is the first to achieve such benefits.

VI. Conclusion

Churn prediction of mobile games is a vital research and business problem that is backed up by a billion-dollar market. In this paper, we proposed a novel inductive semi-supervised embedding model for large-scale game churn prediction. Our model jointly learns a prediction function and an edge embedding function that can automatically map handcrafted features to latent features. The contexts of an edge are sampled by a novel attributed random walk technique, in which both topological adjacency and attribute similarities are considered. We modeled the prediction function and the embedding function by deep neural networks, where the embedding component is designed to capture both contextual information and relationship dynamics. We compared our model with several state-of-the-art baseline methods on large-scale real-world data, which is collected from the Samsung Game Launcher platform. The experimental results clearly demonstrate the effectiveness of the proposed model.

Although the paper focuses on mobile game churn prediction, the proposed method is not restricted to this specific problem and also works for more general problems. Since the inputs of our model only contain the attributes of objects and their relationships at different timestamps, the proposed model can be generalized to fulfill any churn prediction task where the underlying data can be modeled in a similar way, for instance, customer disengagement prediction in membership business (e.g., Apple Music, Costco, and insurance companies) and interest group unsubscription prediction in social networks (e.g., Facebook, Meetup, and Google+).

References

[1] NewZoo. (2017) New gaming boom: Newzoo ups its 2017 global games market estimate to $116.0bn growing to $143.5bn in 2020. [Online]. Available: https://newzoo.com/insights/articles/new-gaming-boom-newzoo-ups-its-2017-global-games-market-estimate\-to-116-0bn-growing-to-143-5bn-in-2020/

[2] M. Zaki, D. Kandeil, A. Neely, and J. R. McColl-Kennedy, "The fallacy of the net promoter score: Customer loyalty predictive model," *Cambridge Service Alliance*, pp. 1–25, 2016.

[3] F. Hadiji, R. Sifa, A. Drachen, C. Thurau, K. Kersting, and C. Bauckhage, "Predicting player churn in the wild," in *Proceedings of IEEE Conference on Computational Intelligence and Games (CIG)*, 2014.

[4] J. Runge, P. Gao, F. Garcin, and B. Faltings, "Churn prediction for high-value players in casual social games," in *Proceedings of IEEE Conference on Computational Intelligence and Games (CIG)*, 2014.

[5] H. Xie, S. Devlin, D. Kudenko, and P. Cowling, "Predicting player disengagement and first purchase with event-frequency based data representation," in *Proceedings of IEEE Conference on Computational Intelligence and Games (CIG)*, 2015.

[6] Á. Periáñez, A. Saas, A. Guitart, and C. Magne, "Churn prediction in mobile social games: towards a complete assessment using survival ensembles," in *Proceedings of IEEE International Conference on Data Science and Advanced Analytics (DSAA)*, 2016.

[7] M. Tamassia, W. Raffe, R. Sifa, A. Drachen, F. Zambetta, and M. Hitchens, "Predicting player churn in destiny: A hidden markov models approach to predicting player departure in a major online game," in *Proceedings of IEEE Conference on Computational Intelligence and Games (CIG)*, 2016.

[8] A. Drachen, E. T. Lundquist, Y. Kung, P. Rao, R. Sifa, J. Runge, and D. Klabjan, "Rapid prediction of player retention in free-to-play mobile games," in *Proceedings of AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE)*, 2016.

[9] H. Xie, S. Devlin, and D. Kudenko, "Predicting disengagement in free-to-play games with highly biased data," in *Proceedings of AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE)*, 2016.

[10] P. Bertens, A. Guitart, and Á. Periáñez, "Games and big data: A scalable multi-dimensional churn prediction model," in *Proceedings of IEEE Conference on Computational Intelligence and Games (CIG)*, 2017.

[11] M. Viljanen, A. Airola, A.-M. Majanoja, J. Heikkonen, and T. Pahikkala, "Measuring player retention and monetization using the mean cumulative function," *arXiv preprint arXiv:1709.06737*, 2017.

[12] M. Viljanen, A. Airola, J. Heikkonen, and T. Pahikkala, "Playtime measurement with survival analysis," *IEEE Transactions on Games*, vol. 10, no. 2, pp. 128–138, 2017.

[13] S. Kim, D. Choi, E. Lee, and W. Rhee, "Churn prediction of mobile and online casual games using play log data," *PloS one*, vol. 12, no. 7, pp. 1–19, 2017.

[14] J. Liang, P. Jacobs, and S. Parthasarathy, "Seano: semi-supervised embedding in attributed networks with outliers," in *Proceedings of SIAM International Conference on Data Mining (SDM)*, 2018.

[15] Z. Yang, W. W. Cohen, and R. Salakhutdinov, "Revisiting semi-supervised learning with graph embeddings," in *Proceedings of International Conference on Machine Learning (ICML)*, 2016.

[16] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Proceedings of Conference on Neural Information Processing Systems (NIPS)*, 2017.

[17] Localytics. (2018) Mobile apps: Whats a good retention rate? [Online]. Available: http://info.localytics.com/blog/mobile-apps-whats-a-good-retention-rate

[18] M. C. Wu and R. J. Carroll, "Estimation and comparison of changes in the presence of informative right censoring by modeling the censoring process," *Biometrics*, vol. 44, no. 1, pp. 175–188, 1988.

[19] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," in *Proceedings of ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (SIGKDD)*, 2016.

[20] N. K. Ahmed, R. A. Rossi, R. Zhou, J. B. Lee, X. Kong, T. L. Willke, and H. Eldardiry, "A framework for generalizing graph-based representation learning methods," *arXiv preprint arXiv:1709.04596*, 2017.

[21] B. P. Chamberlain, A. Cardoso, C. H. Liu, R. Pagliari, and M. P. Deisenroth, "Customer lifetime value prediction using embeddings," in *Proceedings of ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (SIGKDD)*, 2017.

[22] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proceedings of International Conference on Learning Representations (ICLR)*, 2015.

[23] A. Wangperawong, C. Brun, O. Laudy, and R. Pavasuthipaisit, "Churn analysis using deep convolutional neural networks and autoencoders," *arXiv preprint arXiv:1604.05377*, 2016.

[24] V. Umayaparvathi and K. Iyakutti, "Automated feature selection and churn prediction using deep learning models," *International Research Journal of Engineering and Technology (IRJET)*, vol. 4, no. 3, pp. 1846–1854, 2017.

[25] H. Martins, "Predicting user churn on streaming services using recurrent neural networks," Master's thesis, KTH Royal Institute of Technology, 2017.

[26] B. Perozzi, R. Al-Rfou, and S. Skiena, "Deepwalk: Online learning of social representations," in *Proceedings of ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (SIGKDD)*, 2014.

[27] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei, "Line: Large-scale information network embedding," in *Proceedings of International Conference on World Wide Web (WWW)*, 2015.

[28] S. Abu-El-Haija, B. Perozzi, and R. Al-Rfou, "Learning edge representations via low-rank asymmetric projections," in *Proceedings of ACM International Conference on Information and Knowledge Management (CIKM)*, 2017.