Continuous-Time Dynamic Graph Learning via Neural Interaction Processes

Xiaofu Chang
Ant Group, China
xiaofu.cxf@antgroup.com

Xuqin Liu
Ant Group, China
xuqin.lxq@antgroup.com

Jianfeng Wen
Ant Group, China
sylvain.wjf@antgroup.com

Shuang Li
Harvard University, United States
shuangli@harvard.edu

Yanming Fang
Ant Group, China
yanming.fym@antgroup.com

Le Song
Ant Group, China
le.song@antgroup.com

Yuan Qi
Ant Group, China
yuan.qi@antgroup.com

ABSTRACT

Dynamic graphs such as the user-item interactions graphs and financial transaction networks are ubiquitous nowadays. While numerous representation learning methods for static graphs have been proposed, the study of dynamic graphs is still in its infancy. A main challenge of modeling dynamic graphs is how to effectively encode temporal and structural information into *nonlinear* and compact dynamic embeddings. To achieve this, we propose a principled graph-neural-based approach to learn *continuous-time* dynamic embeddings. We first define a *temporal dependency interaction graph*(*TDIG*) that is induced from sequences of interaction data. Based on the topology of this *TDIG*, we develop a *dynamic message passing neural network* named **TDIG-MPNN**, which can capture the fine-grained *global* and *local* information on *TDIG*. In addition, to enhance the quality of *continuous-time* dynamic embeddings, a novel *selection mechanism* comprised of two successive steps, i.e., *co-attention* and *gating*, is applied before the above **TDIG-MPNN** layer to adjust the importance of the nodes by considering *high-order correlation* between interactive nodes' k-depth neighbors on *TDIG*. Finally, we cast our learning problem in the framework of temporal point processes (TPPs) where we use **TDIG-MPNN** to design a neural intensity function for the dynamic interaction processes. Our model achieves superior performance over alternatives on temporal interaction prediction (including *tranductive* and *inductive* tasks) on multiple datasets.

KEYWORDS

time point process; continuous-time dynamic embedding; graph neural network; dynamic graph

CIKM '20, October 19–23, 2020, Virtual Event, Ireland

ACM ISBN 978-1-4503-6859-9/20/10…$15.00
https://doi.org/10.1145/3340531.3411946

ACM Reference Format:
Xiaofu Chang, Xuqin Liu, Jianfeng Wen, Shuang Li, Yanming Fang, Le Song, and Yuan Qi. 2020. Continuous-Time Dynamic Graph Learning via Neural Interaction Processes . In *Proceedings of the 29th ACM International Conference on Information and Knowledge Management (CIKM '20), October 19–23, 2020, Virtual Event, Ireland*. ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3340531.3411946

1 INTRODUCTION

Representation learning over graph data has become a core machine learning task with wide applications including e-commerce, finance, social networks, and bioinformatics. Various neural graph representations such as [6, 7, 10, 17, 23, 24] have been proposed to learn from *static* graphs with fixed topology and these methods have been successfully used for downstream tasks (e.g., classification). Dynamic graphs where nodes and edges can change over time, however, are less studied. Usually the designed graph learning methods are strongly related to the way of graph construction, therefore the methods tailored for *static* graphs construction cannot be directly applied to *dynamic* graphs well.

To handle time-evolving graphs, one can approximate it by a sequence of snapshot graphs, each of which includes all interactions that occur during a user-specified time interval, as shown in [5, 16, 18, 20, 25]. This treatment discretizes the time and reduces the power of modeling fine-grained temporal topological information. It is also challenging to specify an appropriate aggregation granularity. To remedy these problems, Nguyen [15] proposed a continuous-time dynamic network embedding method (CTDNE) which adds a temporal constraint of sampling orders for random walks, but it can't model irregular time intervals which convey important information for analyzing dynamic interaction behaviors, e.g., a user who used to use a game app begins to open it less frequently, which means the user may have gradually lost interest in this app. Also, CTDNE is a transductive method that can't handle unseen nodes. Temporal point processes (TPPs) [3] which treat event times (or time intervals) as random variables provide an elegant way to model fine-grained dynamics. DeepCoevolve proposed by Dai [2] and its extension [21, 22] applied TPPs to model dynamic graphs. They used the coupled recurrent neural networks (RNNs) to learn dynamic embeddings and define different neural

intensity functions of TPPs to characterize dynamic processes. Recently, JODIE [11] also employs the coupled RNNs to generate dynamic embeddings but it is trained by predicting the next embedding directly instead of modeling the intensity. The main drawback of the existing coupled-RNNs-based methods is: regarding to the current interaction(nodes), they treat all the historical interaction information equally and utilize vanilla coupled RNNs to compress them into compact representations as dynamic embeddings of the current nodes. However, the learned embeddings are noisy and non-informative since not all history information are equally important or relevant for the current interaction. In addition, vanilla RNN is often hard to train and suffers from the vanishing gradient problem, which again hurts the quality of the learned embeddings.

Our work falls into the category of learning *continuous-time* dynamic network embeddings by the TPPs framework. Inspired by previous work, in this paper, we present a powerful graph-neural-based approach to learn informative embeddings over the time-evolving graph, from which an neural intensity function is designed to model the occurrence rate of an interaction. Specifically, our contributions are summarized as follows:

- We formally define a time-evolving dynamic graph as a *temporal dependency interaction graph*(TDIG) induced by a sequence of temporal cascades of interactions, and it forms the basis for our later modeling. Compared with the coarse approximation of time-evolving graphs using snapshots, our treatment maintains the fine-grained temporal topological information. Unlike treatment in DeepCoevol and JODIE, our method considers all the relevant interactive nodes to construct the graph and maintains more fine-grained evolution information.
- According to the topology of $TDIG$, We further propose a novel deep learning architecture for the representation learning, where a *dynamic message passing neural network*(**TDIG-MPNN**) is given to compute dynamic embeddings. Specifically, the **TDIG-MPNN** includes a *heterogeneous* graph attention($TDIG$-HGAN) network and a recurrent graph neural network($TDIG$-RGNN), which can capture both *local* and *global* graph information and thus generate informative node representations.
- Inspired by the fact that historical interactions (nodes) will also have varying importance due to their distinct contents, we propose a plug-in module applied before **TDIG-MPNN**, named *selection mechanism*, which allows the network to selectively emphasize informative nodes and suppress irrelevant ones. To be concrete, a *coattention* operation on interactive nodes' k-depth neighbors(history) of $TDIG$ is first performed to capture *high-order correlation*, based on which *content-awared gate functions* are then learned to adjust the importance of nodes. To our best knowledge, this is the first attempt which explores *high-order* proximity to enhance quality of dynamic representations in learning *continuous-time* graph embeddings.
- Based on the learned dynamic embedddings, we design an implicit neural intensity function which explores *non-linear* additive and multiplicative interaction relations and then employ it to characterize our *neural interaction processes* over the $TDIG$. Our method can also incorporate attributes of nodes and interactions, making it *inductive* for unseen nodes and their interactions.

2 RELATED WORK

Static Graph Embeddings. Inspired by the Skip-gram [14] for word embedding, several network embedding methods such as DeepWalk[17] and Node2vec[6] based on random walks on graphs have been proposed. Methods proposed in [7, 10, 23] are a recent class of methods which extend convolutions from spatial domains to graph-structured domains. Moreover they are inductive methods which can generate node embeddings for previously unseen data by leveraging node feature information. The above methods are designed for static graphs with fixed topological structures.

Dynamic Graph Embeddings. As for dynamic graphs with the varying number of nodes and edges (e.g., following new friends on social networks), a branch of methods convert them to a sequence of snapshots [5, 16, 20], but it is difficult to specify the appropriate aggregation granularity. To model dynamic graphs in continuous time, CTDNE [15] adds temporal order constraints on random walk sampling. Temporal point processes (TPPs) is another alternative to model graphs in continuous time. DeepCoevolve[2] used two coupled RNNs over evolving networks to define a neural intensity function in point processes, which allows the model to capture fine grained temporal dynamics of interactions. KnowEvolve[21] extended DeepCoevolve to reason on dynamic knowledge graphs. In DyRep[22], both association and communication interactions are modeled by a two time-scale TPPs. A recent method JODIE [11] also employs the coupled RNNs to generate dynamic embeddings but trained by predicting the next embedding directly instead of modeling the intensity. ***In this work***, unlike previous methods that use the coupled vanilla RNNs and have limited model capacity, our *dynamic message passing neural network*(**TDIG-MPNN**) can generate more expressive and informative embedding representations.

Attention Mechanisms On Graph. Graph Attention Network[23] employed a *self-attention* mechanism to specify different weights to nodes in a *one-hop* neighbors in static graphs. Given an observed interaction between two nodes, DyRep [22] first applied a *self-attention* operation on a node's *one-hop* neighbors on its association graph and then used the attended representation (combined with time information and previous state) to update the dynamic embedding of the other node evolved in this interaction. ***In this work***, distinct from the above methods that only use the limited *one-hop* information, our ***selection mechanism*** comprised of two successive steps, i.e., ***coattention*** and ***gating***, can exploit correlation information of *high-order* temporal neighbors(i.e., k-depth sub-graphs) to obtain enhanced dynamic representation.

3 TEMPORAL POINT PROCESSES

We are interested in modeling the dynamics of the events (i.e., interactions) localized at irregular timestamps $\{t_1, t_2, \dots\}$. Temporal point process (TPP) provides an elegant tool for modeling these discrete events without discretizing the time horizon. By definition, a TPP is a random process whose realization consists of an ordered sequence of events, i.e., $\mathcal{H}_t := \{t_1, \dots, t_i \mid t_i < t\}$, where $\mathcal{H}_t$ is the history up to time t. Let $N(t, t+dt)$ represent the number of events falling within the time window $[t, t+dt)$, and the *conditional intensity* is denoted as

$$\lambda(t \mid \mathcal{H}_t)dt = \mathbb{E}[N(t, t+dt) \mid \mathcal{H}_t]. \tag{1}$$

The TPP can be fully characterized by $\lambda(t|\mathcal{H}_t)$, which explicitly captures how the current event occurrence rate is influenced by history. According to survival analysis [1], the *conditional intensity* has an intrinsic relation with the *conditional density* of the inter-event times, i.e., $p(t\,|\,\mathcal{H}_{t_i}) = \lambda(t|\,\mathcal{H}_t)\exp\left\{-\int_{t_i}^{t}\lambda(s|\,\mathcal{H}_s)ds\right\}$, where t_i is the last point before t, and the exponential part is the survival function which is the conditional probability that no event happens before t. Use the chain rule and we obtain the joint likelihood for a realization of $\{t_1,\ldots,t_n\}$ within $[0,T]$ as

$$\mathcal{L} = \exp\left\{-\int_0^T \lambda(s|\mathcal{H}_s)ds\right\}\prod_{i=1}^{n}\lambda(t_i|\mathcal{H}_{t_i}). \tag{2}$$

By specifying the function forms of $\lambda\,(t|\mathcal{H}_t)$, we can capture various dependency structures in these discrete events. For instance, the Poisson processes [9], corresponds to $\lambda\,(t|H_t)$ is set to be a non-negative constant with an assumption that the process is stationary and events in history are independent of each other. However, the specific parametric assumptions are too limited to reflect the reality and restrict the expressive power of the TPPs. Several vanilla coupled RNNs-based methods [2, 21, 22] for dynamic graph have been proposed to learn general representations of nonlinear dependency over histories, based on which different intensity functions are designed.

4 TEMPORAL DEPENDENCY INTERACTION GRAPH

The way we define or treat a time-evolving graph is vital for model performance. One can discretize a time-evolving graph into several snapshots (the static graph is a special case with only one snapshot). However, model performance will degrade in such a treatment due to the information loss of fine-grained time-dependent structure and the exact timestamps (or time intervals), which can be critical for real-world scenes. To avoid coarse approximation, in this section, we present a *temporal dependency interaction graph* that can maintain fine-grained evolution information and will act as a backbone upon which our model is designed.

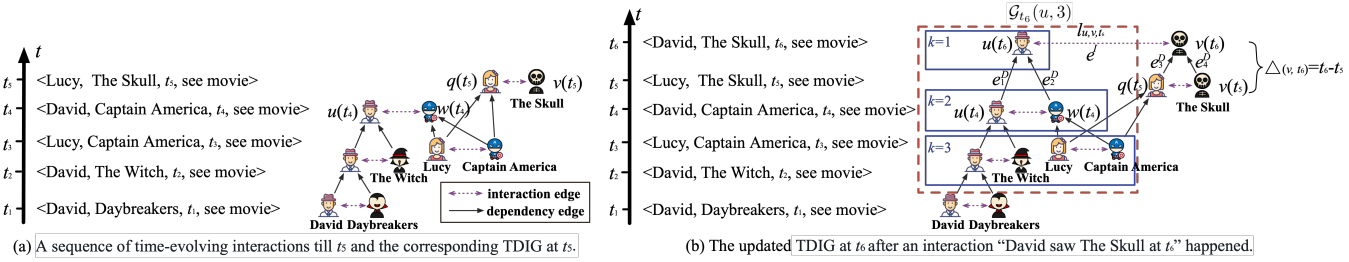
A *temporal dependency interaction graph* $\mathcal{G}_t = (\mathcal{V}_t, \mathcal{E}_t)$ (*TDIG*) consists of a node set $\mathcal{V}_t$ and an edge set $\mathcal{E}_t$ associated with time $t \in \mathbb{R}^+$ and is induced from a sequence of chronological interactions until time t. *Note that* we record the timestamp t of each interaction and thus consider the finest granularity graph construction. Formally, we denote an interaction occurred at time t as $l_{u,v,t}$, where nodes $u, v \in \mathcal{V}_t$ are the two parties associated with this interaction. Denote node u at time t in $l_{u,v,t}$ as $u(t)$ and the two nodes associated with u's last interaction at time t^{u-} (just before time t) as $u^1(t^{u-})$ and $u^2(t^{u-})$ respectively. For convenience, $u^1(t^{u-})$ is denoted as node u itself at time t^{u-} and $u^2(t^{u-})$ is denoted as the other party who interacted with node u at time t^{u-}. Denote $\mathbf{x}_{u(t)} \in \mathbb{R}^d$ as the node feature (i.e., attribute information or id embedding) for the node u at time t.

Our *TDIG* is a *heterogeneous* time-evolving graph. The node set $\mathcal{V}_t$ *in this paper* contains two types of nodes (correspond to the two parties) involved in the temporal interactions. For example, on e-commerce platforms, the two types of nodes are users and clicked items; in financial transaction network, the two types of nodes are payees and payers. *Note that in some cases there are more than two node types involved in temporal interactions and we can assign different parameters to distinguish them, which will be left for future work.* As for the edge set $\mathcal{E}_t$, two types of edges are considered, which will be triggered if an interaction $l_{u,v,t}$ happens between the two associated nodes u and v. One is the *interaction edge* (*denoted as I*) that links the two interactive nodes at time t, i.e, $e^I = (u(t), v(t)) \in \mathcal{E}_t$. The context feature of the $l_{u,v,t}$ (*if available*) is attached with the e^I as an edge feature, denoted as f^I. The other one is the *dependency edge* (*denoted as D*) that links the current node $u(t)$ and the two corresponding nodes $u^1(t^{u-})$ and $u^2(t^{u-})$ respectively in its last interactions at t^{u-}, i.e, $e_0^D = (u^1(t^{u-}), u(t))$ and $e_1^D = (u^2(t^{u-}), u(t)) \in \mathcal{E}_t$, and this implies that the node $u(t)$ is affected by its last interaction. Similarly, we have $e_2^D = (v^1(t^{v-}), v(t))$ and $e_3^D = (v^2(t^{v-}), v(t)) \in \mathcal{E}_t$ for the other node $v(t)$. The *dependency edge* represents the evolution and the causality between temporal relevant nodes. The time interval (denoted as $\Delta_{(u,t)} = t - t^u$) between two adjacent interactions is attached as the dependency edge feature, denoted as f^D. From the definition of $\mathcal{E}_t$, we can know that each node $u(t)$ has three neighbor nodes on $\mathcal{G}_t$, i,e, one interactive node $v(t)$ and two temporal dependency nodes $u^1(t^{u-})$ and $u^2(t^{u-})$.

As illustrated in Fig.1 (a), given a collection of user-movie interactions at different timestamps, the users and movies form a *TDIG*, where users and movie are two types of nodes representing the two parties involved in time-evolving interactions. The purple dash arrows represent the *interaction* edges and the black arrows indicate the *dependency* edges. In Fig.1 (b), when an interaction "*David saw the movie The Skull at* t_6" occurred, a two-way *interaction* edge between the node $u(t_6)$ (*the David at* t_6) and the node $v(t_6)$ (*the Skull at* t_6), i.e., e^I=($u(t_6)$,$v(t_6)$) $\in \mathcal{E}_t$, is added to the $\mathcal{G}_{t_5}$. Four new directed *dependency* edges, including e_0^D=($u(t_4)$,$u(t_6)$), e_1^D=($w(t_4)$,$u(t_6)$), e_2^D=($q(t_5)$,$v(t_6)$) and e_3^D=($v(t_5)$,$v(t_6)$)$\in \mathcal{E}_t$, also form the top part of the $\mathcal{G}_{t_5}$. Obviously, this illustrates *an incremental and a fine-grained way* for constructing $\mathcal{G}_{t_6}$ based on $\mathcal{G}_{t_5}$.

All the edges are directed and this represents how information is propagated over the *TDIG*. Whenever an interaction $l_{u,v,t}$ occurred, the states (embeddings) of the two associated nodes $u(t)$ and $v(t)$ will be updated by *message passing* from their neighbours on the *TDIG*. In Fig.1 (b), we aggregate neighbor information from the dependency nodes $u(t_4)$, $w(t_4)$ and the current interactive node $v(t_6)$ into node $u(t_6)$. In practice, when we aggregate information from dependency nodes to update states of the nodes u and v in an observed interaction $l_{u,v,t}$, we only consider the max k-depth dependency sub-graphs of u and v for computation efficiency. The max k-depth *TDIG* sub-graphs related to the nodes u and v are denoted as $\mathcal{G}_t(u,k)$ and $\mathcal{G}_t(v,k)$ respectively. As illustrated in Fig.1 (b), the red dotted square denotes the 3-depth *TDIG* sub-graphs for the node $u(t_6)$ as $\mathcal{G}_{t_6}(u,3)$.

Relation to previous work. The graph construction in the DeepCoevolveve and JODIE[2, 11] differs from our *TDIG* in that the nodes $u(t)$ and $v(t)$ associated with the $l_{u,v,t}$ are only connected by these two nodes themselves before the time t(i.e., they use a different way to construct the temporal graph edges and they don't provide a formal definition about their graph), which means when updating the states, only previous states of these two nodes themselves are considered, not including the states of the other parties



(a) A sequence of time-evolving interactions till t_5 and the corresponding TDIG at t_5.

(b) The updated TDIG at t_6 after an interaction "David saw The Skull at t_6" happened.

Figure 1: (a) An example of a sequence of five chronological interactions and the corresponding TDIG at t_5. The black arrows indicate the dependency edges between adjacent relevant interactions and the two-way purple dashed arrows represent interaction edges. All the edges are directed representing how messages are passed on the TDIG. (b) After a new interaction l_{u,v,t_6} is observed at time t_6, the corresponding TDIG is updated with one newly-added two-way interaction edge e^I and the four newly-added directed dependency edges, i.e., e_0^D, e_1^D, e_2^D and e_3^D. The red dotted square denotes the 3-depth TDIG sub-graph $\mathcal{G}_{t_6}(u, k=3)$ for the node u at t_6 and the $\Delta_{(v,t_6)}$ represents the time interval(as the edge feature for the edge e_3^D and e_4^D) between l_{u,v,t_6} and the last interaction *"Lucy saw the Skull movie at* t_5*"* where the node v was involved.

that had interaction with these two nodes just before the time t. In contrast, our construction of *TDIG* considers all the relevant interactive nodes and maintains more fine-grained evolution information. The differences between the two graph constructions naturally lead to different model designs.

5 PROPOSED APPROACH

Given the structure of *TDIG*, in this section, we start with describing our *dynamic graph message passing neural network* (**TDIG-MPNN**) that can generate informative *continuous-time* dynamic embeddings. A novel *selection mechanism*, applied before **TDIG-MPNN**, is introduced to enhance expressive power of the dynamic embeddings, as illustrated in Fig 2. Lastly, we learn the model parameters by leveraging the theory of temporal point processes (TPPs) of interactions, i.e, we characterize the generative process of a new interaction at time t by designing a neural intensity function based on the dynamic embeddings of *TDIG* just before time t.

5.1 TDIG-MPNN

Our work aims to learn *continuous-time* dynamic embeddings which can encode *non-linear* and *fine-grained* temporal and structural information well. Inspired by the general GNN framework[4] that computes nodes' embeddings by aggregating messages from nodes' neighborhoods, we propose our dynamic graph message passing neural network named **TDIG-MPNN**. The **TDIG-MPNN** is a joint model consisting of *TDIG*-informed *heterogeneous* graph attention network(TDIG-HGAN) and *TDIG*-informed recurrent graph neural network(TDIG-RGNN). The TDIG-HGAN takes into consideration *local* temporal and structural characteristics by gathering messages from neighborhoods of nodes on *TDIG*, then the TDIG-RGNN further integrates the TDIG-HGAN output by recurrence in time to capture the *global* (*long-term dependencies*) information over *TDIG*. By this way, the proposed **TDIG-MPNN** can generate an informative dynamic embedding for each node that captures both the *local* and *global* graph information. We will elaborate on them as follows.

5.1.1 TDIG-HGAN. According to the topology of $\mathcal{G}_t$, a *TDIG*-informed *heterogeneous* graph attention network (TDIG-HGAN) is designed to capture the *local* temporal and structural characteristic of the node $u(t)$ (*associated with the* $l_{u,v,t}$) using the following equations:

$$
\begin{aligned}
F_{\text{msg}}(w, u(t)) &= \text{ReLU}(\mathbf{W}_{\text{msg}}^{F_{\text{e-type}}(w,u(t))}[\mathbf{x}_w; \mathbf{x}_{u(t)}; f_{(w,u(t))}^{F_{\text{e-type}}(w,u(t))}]^\top) \\
\alpha(w, u(t)) &= \sigma(\mathbf{W}_{\alpha}^{F_{\text{e-type}}(w,u(t))}([\mathbf{x}_w; \mathbf{x}_{u(t)}; f_{(w,u(t))}^{F_{\text{e-type}}(w,u(t))}]^\top)) \\
F_{\text{agg}}(u(t)) &= \sum\nolimits_{w \in \mathcal{N}(u(t))} \alpha(w, u(t)) \odot F_{\text{msg}}(w, u(t)) \\
\mathbf{h}_{u(t)}^c &= \text{ReLU}(F_{\text{agg}}(u(t)))
\end{aligned} \tag{3}
$$

where $\sigma(\cdot)$ is the sigmoid function, $\odot$ represents the Hadamard product and [;] means a concatenation operation. $F_{\text{msg}}(w, u(t))$ is a function for computing a *local* message from the node w to the node $u(t)$, which is modeled as a non-linear transformation with the rectified linear unit (ReLU) applied on the concatenation of the two node features $\mathbf{x}_w$ and $\mathbf{x}_{u(t)}$ and their corresponding edge feature $f_{(w,u(t))}^{F_{\text{e-type}}(w,u(t))}$ where the function $F_{\text{e-type}}(w, u(t))$ maps the edge $(w, u(t))$ to its corresponding edge type, i.e., I and D. Due to the *heterogeneity* of *TDIG*, messages from different kinds of neighbors should be treated differently by using the *(edge-type)-specific* transformation matrix $\mathbf{W}_{\text{msg}}^{F_{\text{e-type}}(w,u(t))}$. Considering different roles (importances) of different type of *local* messages, we employ the function $F_{\text{agg}}(u(t))$ to aggregate different *local* messages from neighborhoods of the node $u(t)$, i.e., $\mathcal{N}(u(t))$, using attention mechanism. The $\alpha(w, u(t)) \in \mathbb{R}$ acts as an edge gate considering both the two nodes features and the corresponding edge-type feature(context feature or temporal intervals). Similarly, we utilize the weight parameter $\mathbf{W}_{\alpha}^{F_{\text{e-type}}(w,u(t))}$ for different-type *local* messages when computing $\alpha(w, u(t))$. After performing a ReLU activation on the local aggregated information, we obtain the local embedding $\mathbf{h}_{u(t)}^c \in \mathbb{R}^d$ that encodes the *local* temporal and structural information of the node $u(t)$. We next pass $\mathbf{h}_{u(t)}^c$ to the TDIG-RGNN layer.

5.1.2 TDIG-RGNN. Due to the recursive structure of our *TDIG*, we devise a *TDIG*-informed recurrent graph neural network(TDIG-RGNN) to capture *global* information on *TDIG* by integrating the the *local* embedding $\mathbf{h}_{u(t)}^c$ in a recurrent manner. The memory cell

combined with the gate mechanisms in a chain-structured Long Short-Term Memory LSTM[8] provide the ability to model *long-term* dependencies. In light of this, our TDIG-RGNN extends a chain-structured LSTM to model *long-term* message passing on $TDIG$. Specifically, given the $l_{u,v,t}$, the TDIG-RGNN models information propagation on the $\mathcal{G}_t$ as follows:

$$\begin{aligned}
\mathbf{z}_{u(t)} &= \sigma\left(\mathbf{W}_z\bar{\mathbf{h}}^c_{u(t)} + \sum_{i=1}^{2}\mathbf{R}_{z_i}\mathbf{h}^r_{u^i(t^{u-})} + \mathbf{b}_z\right)\\
\mathbf{o}_{u(t)} &= \sigma\left(\mathbf{W}_o\bar{\mathbf{h}}^c_{u(t)} + \sum_{i=1}^{2}\mathbf{R}_{o_i}\mathbf{h}^r_{u^i(t^{u-})} + \mathbf{b}_o\right)\\
\mathbf{s}_{u(t)} &= \tanh\left(\mathbf{W}_s\bar{\mathbf{h}}^c_{u(t)} + \sum_{i=1}^{2}\mathbf{R}_{s_i}\mathbf{h}^r_{u^i(t^{u-})} + \mathbf{b}_s\right)\\
\mathbf{g}^i_{u(t)} &= \sigma\left(\mathbf{W}_{g_i}\bar{\mathbf{h}}^c_{u(t)} + \mathbf{R}_{g_i}\mathbf{h}^r_{u^i(t^{u-})} + \mathbf{M}_{g_i}\Delta_{(u,t)} + \mathbf{b}_{g_i}\right)\\
\mathbf{c}_{u(t)} &= \mathbf{z}_{u(t)}\odot\mathbf{s}_{u(t)} + \sum_{i=1}^{2}\mathbf{c}_{u^i(t^{u-})}\odot\mathbf{g}^i_{u(t)}\\
\mathbf{h}^r_{u(t)} &= \mathbf{o}_{u(t)}\odot\tanh\left(\mathbf{c}_{u(t)}\right)
\end{aligned} \tag{4}$$

where $\sigma(\cdot)$ is the sigmoid function, tanh is the hyperbolic tangent function, $\odot$ represents the Hadamard product, $\mathbf{W}_z$, $\mathbf{W}_o$, $\mathbf{W}_s$, $\mathbf{W}_{f_i}$, $\mathbf{W}_{g_i}$, $\mathbf{R}_{z_i}$, $\mathbf{R}_{o_i}$, $\mathbf{R}_{s_i}$, $\mathbf{R}_{f_i}$, $\mathbf{R}_{g_i}$ are the weight parameters and $\mathbf{b}_z$, $\mathbf{b}_o$, $\mathbf{b}_s$, $\mathbf{b}_{f_i}$, $\mathbf{b}_{g_i}$ are the bias vectors to be learned during training. The $\bar{\mathbf{h}}^c_{u(t)}$ is a concatenation of the *local embedding* $\mathbf{h}^c_{u(t)}$ and $\mathbf{h}^c_{v(t)}$ as an input for the TDIG-RGNN layer. Since each node $u(t)$ has two dependency nodes $u^1(t^{u-})$ and $u^2(t^{u-})$ on $TDIG$, we let the *update gate* $\mathbf{s}_{u(t)}$, the *input gate* $\mathbf{z}_{u(t)}$ and the *output gate* $\mathbf{o}_{u(t)}$ depend on the two predecessors' dynamic embeddings $\mathbf{h}^r_{u^1(t^{u-})}$ and $\mathbf{h}^r_{u^2(t^{u-})}$, in addition to the local message $\bar{\mathbf{h}}^c_{u(t)}$. We also introduce a *temporal and structural gate* $\mathbf{g}^i_{u(t)}$ for the corresponding dependency edge $e^D_i = (u^i(t^{u-}), u(t))$ where $i = 1, 2$ to control how much the dependent temporal and structural message is passed to the current memory cells $\mathbf{c}_{u(t)}$ through the corresponding dependency path. *Unlike* LSTM that only processes equally spaced chain-structured series, the proposed gate $\mathbf{g}^i_{u(t)}$ can consider both irregular time intervals and different dependency nodes information to decide which dependency path can convey more informative message for the current input. The memory cell $\mathbf{c}_{u(t)}$ is responsible for keeping track of the *long-term* dependencies (*global* information) over the $TDIG$, based on which *the updated dynamic embedding* $\mathbf{h}^r_{u(t)} \in \mathbb{R}^d$ can encode *global* information on $TDIG$.

To reduce the computational cost in practice, we only use the information stored in the sub-graph $\mathcal{G}_t(u,k)$ to update $\mathbf{h}^r_{u(t)}$, where k is a hyper-parameter.

5.2 Selection Mechanism

While **TDIG-MPNN** is capable of modeling the *non-linear temporal* and *structural* information on $TDIG$, we further propose a plug-in module named *selection mechanism* applied before the **TDIG-MPNN** layer, which will increase the expressive power of the final dynamic embedding $\mathbf{h}^r_{u(t)}$. Intuitively not all the historical nodes (interactions) are equally-related to the current interaction $l_{u,v,t}$. If we can filter out irrelevant nodes and keep the relevant ones for the given $l_{u,v,t}$, then the obtained embeddings $\mathbf{h}^r_{u(t)}$ and $\mathbf{h}^r_{v(t)}$ by the **TDIG-MPNN** layer could be more informative. Motivated by this, to enhance the quality of the dynamic embedding, our *selection mechanism* will try to adaptively emphasize informative nodes and suppress irrelevant ones by considering their ***high-order*** *inter-dependencies* between neighbor nodes of $u(t)$ and $v(t)$ on $TDIG$. For computation efficiency, we consider k-depth historical nodes in $\mathcal{G}_t(u,k)$ and $\mathcal{G}_t(v,k)$. **Note that** the proposed *temporal* and *structural* gate $g^i_{u(t)}$(where $i = 1, 2$) in last subsection can also help to select informative messages from the corresponding dependency nodes, but it only considers information from the historical nodes of $u(t)$ itself, i.e., $\mathcal{G}_t(u,k)$. The proposed *selection mechanism* can explore extra relevant information from the sub-graph $\mathcal{G}_t(v,k)$, which further helps to yield more expressive dynamic embeddings.

Our *selection mechanism* includes two successive steps, the **coattention** and the **gating**. Specifically,

- The *coattention* operation is first performed to measure the *information correlation* between the two sub-graphs $\mathcal{G}_t(u,k)$ and $\mathcal{G}_t(v,k)$:

$$\mathbf{Q} = \tanh\left(\mathbf{X}_u^\top\mathbf{W}_Q\mathbf{X}_v\right) \tag{5}$$

  where $\mathbf{X}_u = \left[\mathbf{x}_{a_1}, \ldots, \mathbf{x}_{a_i}, \ldots, \mathbf{x}_{a_m}\right]$ is packed with the content of the nodes $\{a_i\}_{i=1,\ldots,m}$ in the $\mathcal{G}_t(u,k)$, $\mathbf{X}_v = [\mathbf{x}_{b_1}, \ldots, \mathbf{x}_{b_i}, \ldots, \mathbf{x}_{b_n}]$ is packed with the content of the nodes $\{b_i\}_{i=1,\ldots,n}$ in the $\mathcal{G}_t(v,k)$, m and n are the counts of nodes in the two corresponding sub-graphs. Here the $\mathbf{Q}$ is a *coattention affinity matrix* which captures the *inter-dependencies* information between the sub-graphs $\mathcal{G}_t(u,k)$ and $\mathcal{G}_t(v,k)$. The $\mathbf{W}_Q \in \mathbb{R}^{d\times d}$ is a parameter matrix. Based on the *coattention affinity matrix* $\mathbf{Q}$, we compute the two global context embeddings $\mathbf{p}_u$ and $\mathbf{p}_v$ for the two sub-graphs $\mathcal{G}_t(u,k)$ and $\mathcal{G}_t(v,k)$ respectively as follows:

$$\mathbf{p}_u = \mathbf{X}_u\text{SoftMax}\left(\text{Max}_{colwise}\,\mathbf{Q}\right) \tag{6}$$

$$\mathbf{p}_v = \mathbf{X}_v\text{SoftMax}\left(\text{Max}_{colwise}(\mathbf{Q}^\top)\right) \tag{7}$$

  where the Max represents a *max-pooling* operation that is used to extract the most relevant information between the two corresponding sub-graphs $\mathcal{G}_t(u,k)$ and $\mathcal{G}_t(v,k)$. Based on the most relevant information, we obtain the embeddings $\mathbf{p}_u$ and $\mathbf{p}_v$ as the weighted summation of their own historical nodes information, i.e., $\mathbf{X}_u$ and $\mathbf{X}_v$, respectively. By this way, the embeddings $\mathbf{p}_u$ and $\mathbf{p}_v$ not only contain global information of their own corresponding sub-graphs, but also include information from the sub-graph of the other party regarding to this interaction $l_{u,v,t}$.
- Next, the two *content-awared gate functions* are learned to adaptively adjust the importance of the nodes in $\mathcal{G}_t(u,k)$ and $\mathcal{G}_t(v,k)$, respectively.

$$g_u(\mathbf{p}_u, \mathbf{x}_{a_i}) = \sigma(\mathbf{W}_p\mathbf{p}_u + \mathbf{W}_h\mathbf{x}_{a_i}) \tag{8}$$

$$\bar{\mathbf{x}}_{a_i} = g_u(\mathbf{p}_u, \mathbf{x}_{a_i})\odot\mathbf{x}_{a_i} \tag{9}$$

$$g_v(\mathbf{p}_v, \mathbf{x}_{b_i}) = \sigma(\mathbf{W}_p\mathbf{p}_v + \mathbf{W}_h\mathbf{x}_{b_i}) \tag{10}$$

$$\bar{\mathbf{x}}_{b_i} = g_v(\mathbf{p}_v, \mathbf{x}_{b_i})\odot\mathbf{x}_{b_i} \tag{11}$$

  where $\mathbf{W}_p \in \mathbb{R}^{d\times d}$ and $\mathbf{W}_h \in \mathbb{R}^{d\times d}$ are weight parameters. $\bar{\mathbf{x}}_{a_i}$ and $\bar{\mathbf{x}}_{b_i}$ are new representations after using the two corresponding gate functions $g_u(\mathbf{p}_u, \mathbf{x}_{a_i})$ and $g_v(\mathbf{p}_v, \mathbf{x}_{b_i})$ respectively, which is illustrated in Figure 2.

5.3 Comparison to other models

The state-of-the-art *continuous-time* embedding methods JODIE and DeepCoevolve are the most related to our work. However, our

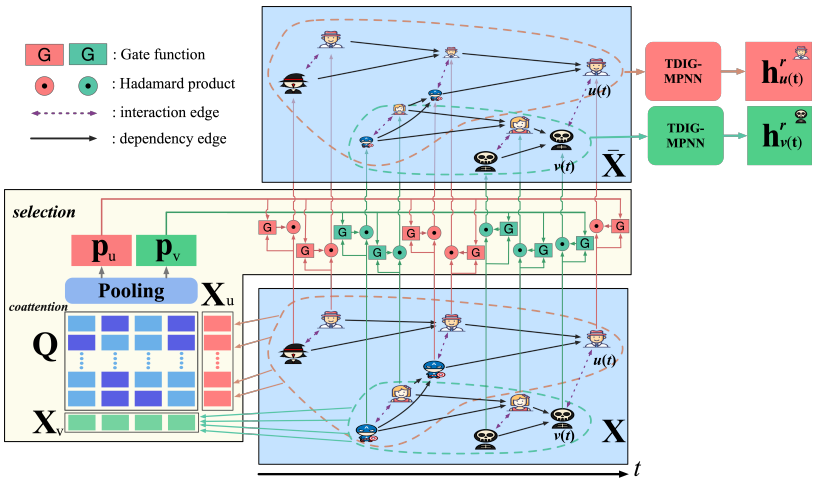


Figure 2: Selection Mechanism: To enhance the quality of dynamic embeddings $h^r_{u(t)}$ and $h^r_{v(t)}$, a *selection mechanism* is employed beforehand to adjust the importance of the historical nodes(in contents) by considering their *inter-dependencies* between historical nodes of $u(t)$ and $v(t)$. For illustration purpose, the sizes of the history nodes (in the figure) are adjusted by the *selection mechanism* before they are fed into the TDIG-MPNN layer.

model is significantly different from them in following aspects: **First**, as introduced in section 4, our treatment for dynamic graph *TDIG* can maintain more fine-grained interactive evolution information than the ones used in JODIE and DeepCoevolveve; **Secondly**, our *TDIG*-specific graph message passing neural network, i.e., **TDIG-MPNN**, can explore both *local* and *global* information on *TDIG* to generate informative embeddings, while the compared methods JODIE and DeepCoevolve employ vanilla RNNs to model co-evolution on their graph. The vanilla RNNs usually have limited model capacity and suffer from the vanishing gradient problems, which may lead to less informative embeddings; **Thirdly**, our *selection mechanism* can help enhance the quality of embeddings by considering their ***high-order*** *inter-dependencies* between neighbor nodes of $u(t)$ and $v(t)$ on *TDIG* but DeepCoevolve and JODIE can't.In addition, the proposed model has potentionally good interpretability for the learned dynamic embedding by leaveraging the *temporal and structural* gate $g^i_{u(t)}$, where $i = 1, 2$, which is a big advantage over the other *continuous-time* methods. Please see case study in Section 6.6.

5.4 Model Optimization

5.4.1 Neural Interaction Processes. We cast our learning problem as modeling multi-dimensional TPPs of temporal interactions. According to the Section 3, the key step is to design a conditional intensity function(as defined in Equation (1)) that models the inter-event time of interactions occurring between nodes. Specifically, our conditional intensity of *neural interaction process* is constructed on the basis of the dynamic node embeddings that encode the *non-linear* temporal and structural information on *TDIG*:

$$\lambda^{u,v}\left(t|\mathcal{H}^{u,v}_t\right) = \text{SoftPlus}\left(\mathbf{w}^\top_{add}(\bar{\boldsymbol{h}}^r_{u(t^{u-})} + \bar{\boldsymbol{h}}^r_{v(t^{v-})}) + \mathbf{w}^\top_{mul}(\bar{\boldsymbol{h}}^r_{u(t^{u-})} \odot \bar{\boldsymbol{h}}^r_{v(t^{v-})}) + b^{u,v}_\lambda\right) \quad (12)$$

where $\mathcal{H}^{u,v}_t = \mathcal{G}_{t^{u-}}(u^1,k) \cup \mathcal{G}_{t^{u-}}(u^2,k) \cup \mathcal{G}_{t^{v-}}(v^1,k) \cup \mathcal{G}_{t^{v-}}(v^2,k)$ represents the corresponding graph-structured histories for predicting an interaction between nodes u and v at some time t, $\bar{\boldsymbol{h}}^r_{u(t^{u-})} = \left[\mathbf{h}^r_{u^1(t^{u-})}; \mathbf{h}^r_{u^2(t^{u-})}\right]$ and $\bar{\boldsymbol{h}}^r_{v(t^{v-})} = \left[\mathbf{h}^r_{v^1(t^{v-})}; \mathbf{h}^r_{v^2(t^{v-})}\right]$, where all $\mathbf{h}_{u^1(t^{u-})}$, $\mathbf{h}_{u^2(t^{u-})}$, $\mathbf{h}_{v^1(t^{v-})}$, and $\mathbf{h}_{v^2(t^{v-})}$ are computed as in Equation. (4). The parameters $\mathbf{w}_{add} \in \mathbb{R}^{2d}$, $\mathbf{w}_{mul} \in \mathbb{R}^{2d}$ and the scalar $b^{u,v}_\lambda \in \mathbb{R}$. The $b^{u,v}_\lambda$ can be viewed as a base intensity for the occurrence of interaction between u and v. The SoftPlus function is applied to ensure the non-negativity of the intensity function. Our intensity function is constant about the time variable t over the period between the successive relevant interactions since the dynamic embeddings are only updated whenever an interaction happens in this paper. **It's also noteworthy that** since one node's state depends on not only this node's previous embedding but also the embedding of the paired node involved in the previous interaction, we use four node embeddings, $\mathbf{h}_{u^1(t^{u-})}$, $\mathbf{h}_{u^2(t^{u-})}$, $\mathbf{h}_{v^1(t^{v-})}$ and $\mathbf{h}_{v^2(t^{v-})}$, to infer the occurrence rate of the u, v paired interaction at t, *which is different* from DeepCoevolve [2] and its extension where only previous states of u and v before time t are used. We explore both the additive and multiplicative relations between $\bar{\boldsymbol{h}}^r_{u(t^{u-})}$ and $\bar{\boldsymbol{h}}^r_{v(t^{v-})}$ to capture rich interactive patterns.

5.4.2 Objectives and Training. Given a list of interactions as $O = \{(u_i, v_i, t_i)\}_{i=1}^N$ observed in a time window $[0, T]$, we can learn the model by minimizing the negative joint log-likelihood as follows: $\mathcal{L} = -\sum_i^N \log p^{u_i,v_i}\left(t_i|\mathcal{H}^{u_i,v_i}_{t_i}\right)$ where $p^{u_i,v_i}(t_i|\mathcal{H}^{u_i,v_i}_{t_i})$ represents the probability of formalizing an interaction between u_i and v_i at time t_i given the graph-structured history $\mathcal{H}^{u_i,v_i}_{t_i}$. We further have $\mathcal{L} = -\sum_i^N \log \lambda^{u_i,v_i}\left(t_i|\mathcal{H}^{u_i,v_i}_{t_i}\right) + \int_0^T \sum_{u,v} \lambda^{u,v}(t)dt$. To accelerate the evaluation of the survival term that requires enumerating all (u,v) pairs, we employ the similar negative sampling approaches used in DeepCoevolve [2].

Our method can be easily parallelized. Following the training pipeline of GraphSage [7], we propose a batch-training algorithm. Given an interaction $l_{u,v,t}$, we sample a tree rooted at each node $u(t)$ or $v(t)$ by recursively expanding the root node's *dependency* neighborhood by k steps, i.e., $\mathcal{G}_t(u,k)$ or $\mathcal{G}_t(v,k)$. For each sampled paired trees, i.e., $\mathcal{G}_t(u,k)$ and $\mathcal{G}_t(v,k)$, we compute the root nodes's dynamic representations, i.e., $h^r_{u(t)}$ and $h^r_{v(t)}$, by **TDIG-MPNN** combined with our *selection mechanism*.

Since each node $u(t)$ has two dependency nodes on $TDIG$, we have 2^k nodes in the sampled tree which is much smaller than that in GraphSage [7] where the total number is r^k and the number of sampled neighbors r for each layer is usually greater than 10. The overall complexity of training on $\mathcal{G}_t$ is linearly proportional to $2^k * N$ where N is the number of temporal interactions(until the time t) that are used to construct $\mathcal{G}_t$.

Due to the incremental construction of $TDIG$ and the recursive computation for dynamic embeddings, we can naturally train our model in an incremental way on the new updated parts on $TDIG$, which is more efficient than static methods like GraphSage where all the nodes embeddings will be re-computed once new nodes and edges are added.

6 EXPERIMENTS

We evaluate our proposed model for **temporal interaction prediction** on several diverse interaction datasets, including

(*a*) **CollegeMsg** [12], which consists of message sending activities on a social network at the University of California, Irvine.

(*b*) **Amazon** [13], which is composed of commodity rating data from amazon users. We focus on **Clothing** and **Cellphone** items.

(*c*) **LastFM** [12], which consists of who-listens-to-which song interactions.

(*d*) **Huabei Trades** [1,2], which is about transaction records processed by Huabei during August 2018 and each transaction has 11 context features (e.g., buyer types, seller types, purchased items' categories and trading platform).

For the datasets **CollegeMsg**, **Amazon** and **LastFM**, they have no attribute and context features and are used for **Transductive Task**. For the **Huabei Trade** dataset, there are 41.69% new nodes (i.e., unobserved in training and only appear in the test set) and we leverage its context features for the **Inductive Task**. Our datasets are *diverse* in terms of the number of interactions, the number of each party in interactions, the number of new interactions and time duration. Detailed statistics of our datasets are given in Table 1.

6.1 Baselines and Evaluation Metrics

Our baselines include **Static Models** and **Dynamic Models**:

(*i*) For Static Models, we consider **GraphSage** [7] including four aggregators: GCN, MEAN, MAX and LSTM. A graph self-attention aggregator GAT [23] is also implemented based on GraphSage Framework. For convenience, we report the best results among these five aggregators denoted as **GraphSage***.

(*ii*) For Dynamic Models, *in this paper*, we focus on *continuous-time* embedding methods: **(a) CTDNE** [15] is an extension to deep walk with time constraint for sampling orders; **(b)** Among TPP-based methods, for example, **DeepCoevolve** [2], **KnowEvolve** [21] and **DyREP** [22], we only select **DeepCoevolve** [2] for comparison since the other two methods require special data information and we only focus on general interaction datasets in this paper; **(c) JODIE**[11] is a state-of-art method that models interaction processes by predicting the next interaction embedding directly instead of modeling the intensity function. In addition, two recent snapshot-based dynamic methods **DynGEM**[5] and **DySAT**[18] are also considered.

Given an interaction (u,v,t), each method outputs the u's preference scores over all the items in test. We sort scores in a descending order and record v' s rank. We report the average ranks for all interactions in test data, which is denoted as **Mean Rank**. We also report the **Hit@10**: the proportion of times that a test tuple appears in the top 10.

6.2 Experimental Setting

For **Transductive Task**, since no attributes features are provided, we use node id embedding as inputs. For all models, the size of id embedding is set as 64. For **Inductive Task**, we utilize the 11 types of given context features as inputs. The graph embedding size (hidden) is also set to be 64 for all methods in this task.

We use the Adam optimizer in training, and set the batch size to be 512 with a grid-search for learning rate in {0.01, 0.001, 0.0005, 0.0001}. For our proposed method, we consider the depth k of sub-graphs to be {1, 2, 3, 4, 5} respectively. For **GraphSage**, the maximum number of 1/2/3/4/5-hop neighbor nodes is set to be 25/10/10/10/10. For snapshot methods, we search the number of snapshots in {1,5,10,15} for all datasets. All models are trained for at most 30 epochs with an early-stopping if no improvement for 5 consecutive epochs. The number of negative sampling is set to be 10 for all methods except for the **DynGEM** and **JODIE** that don't need it.

We split each dataset orderly into training/validation/test sets by 6/2/2, separately, except for the dataset **LastFM**. The dataset **LastFM** is the largest dataset among datasets used by **JODIE**. For fair comparison with **JODIE**, we follow the same data split in **JODIE** by 8/1/1.

6.3 Overall Performances

Table 2 summarizes the overall performances of all methods in terms of **Mean Rank(MR)** and **Hit@10** in **Transductive** and **Inductive** tasks, respectively. On the whole, our proposed method consistently beats all second-best baselines(underlined in Table 2) over five different datasets in two tasks in terms of **MR** by 42.41%, 7.93%, 17.58%, 29.78% and 35.84% respectively. As for **Hit@10** performance, our method nearly beat all the baseline methods except for the datatset **Amazon-Clothing**.

On the **Transductive Task**, the **CTDNE** method performs worst across all the datasets since the generated embedding is static and can't be updated across validation and test datasets. Meanwhile we can see that there is no consistent winner among baselines and all the methods perform relatively better on the two datasets **CollegeMsg** and **LastFM**. It should be noted that static methods **GraphSage*** perform competitive with dynamic baselines on these

[1] https://www.alipay.com/

[2] The data used in this research were all processed by data abstraction and data encryption, and the researchers were unable to restore the original data.

Table 1: Dataset Statistics.

Dataset	CollegeMsg	Clothing	Cellphone	LastFM	Huabei
Task	Transductive	Transductive	Transductive	Transductive	Inductive
#User	999	10109	11934	1000	22657
#Item	1352	17404	8316	1000	7115
#Interactions	48,771	67,002	74,963	1,293,103	86,596
#New Node(tst)	0.0%	0.0%	0.0%	13.11%	41.69%
#New Interaction(tst)	76.82%	100%	100%	51.71%	96.77%
Repetition	67.28%	0.0%	0.0%	88.01%	36.15%
Duration (day)	193.63	3657.00	3882.00	1586.89	30.00
Time Interval (day)	1.14	111.16	78.60	0.80	1.19

Table 2: Overall Comparison for Transductive and Inductive tasks.

Dataset	**CollegeMsg**		**Clothing**		**Cellphone**		**LastFM**		**Huabei**	
Task	Transductive		Transductive		Transductive		Transductive		Inductive	
Metrics	MR↓	Hit@10↑	MR↓	Hit@10↑	MR↓	Hit@10↑	MR↓	Hit@10↑	MR↓	Hit@10↑
CTDNE[15]	604.86	1.91	8675.35	0.01	4145.52	0.12	479.70	1.29	-	-
GraphSage*[7][23]	270.06	11.01	7162.13	0.16	2697.59	0.57	263.18	10.90	3069.62	1.52
DynGEM[5]	338.59	3.42	6235.55	**0.82**	2062.82	3.62	472.55	3.58	-	-
DeepCoevolve[2]	447.21	2.14	7086.37	0.21	2397.34	1.57	492.08	1.12	2676.67	6.55
DySAT[18]	358.17	2.67	7875.76	0.32	3050.42	2.02	287.22	4.71	2317.98	13.15
JODIE[11]	283.41	24.75	8011.22	0.46	3255.49	2.43	296.49	27.86	1917.02	3.62
Our method w/o Selection	165.33	34.87	6371.61	0.37	2055.63	3.67	200.04	28.03	1402.56	25.91
Our method	**155.52**	**38.97**	**5740.90**	0.57	**1700.15**	**3.87**	**184.81**	**31.00**	**1229.99**	**32.60**

two datasets **CollegeMsg** and **LastFM**. These phenomenons could be explained that there are many *repetitive interactions*, i.e., 67.28% in **CollegeMsg** and 88.01% in **LastFM** and *repetitive information* makes recurring interaction predicted easily, especially for static graph methods which can make full use of structural information. In contrast, all the methods perform relatively worse on the two **Amazon** datasets since *no repetitive information*(i.e.,100% cold-started interactions in test dataset) can be utilized. As for comparison with dynamic methods, our work outperforms **DeepCoevolve** and **JODIE** which use the coupled vanilla RNNs to update dynamic states. Such improvements might be attributed to two factors: *(a)* the proposed **TDIG-MPNN** is more powerful for capturing *non-linear* temporal and structural information than vanilla RNN. *(b)* the *selection mechanism* can select more relevant history information to enhance the quality of embeddings rather than treat all past history information equally like in **DeepCoevolve** and **JODIE**, which may lead to noisy embeddings. The reason that **JODIE** performs worse in two **Amazon** datasets could be that **JODIE** imposes a temporal regularizer to enforce smoothness of the dynamic nodes embeddings in two adjacent interactions, but it may fail when the datasets have large time intervals(like **Amazon** data shown in Table 1), which might cause a node's two adjacent states to change drastically. Our proposed method outperforms **DynGEM** and **DySAT** because our method can capture *fine-grained* temporal and structural information corresponding to our novel graph construction *TDIG* while the methods **DynGEM** and **DySAT** only capture coarse dynamics based on their snapshots. Moreover, it's non-trivial to specify the appropriate aggregation granularity in diverse scenes.

As for **Huabei** on **Inductive Task**, we don't provide results for **CTDNE** and **DynGEM** since they are transductive and can't leverage attributes information. The GraphSage* performs worst since static embeddings can't be evolved for new interactions where there are 96.77% *unseen interactions*. Again, our proposed method outperforms the other dynamic methods.

6.4 Ablation Study

In this part, we conduct ablation study for *selection mechanism*. The **Mean Rank** and **Hit@10** are also given for our method without using *selection mechanism* in Table 2. We observe that without *selection mechanism*, the performances are degraded for all the datasets. This can be explained that treating historical interactions equally might lead to generate noise dynamic embeddings which contain irrelevant information. Meanwhile, it is noteworthy that there is a relatively a big drop on datasets with less repetitive information(e.g. **Amazon**). According to our definition of $TDIG$, if the construction of $TDIG$ is based on less repetitive interactions, then $TDIG$ will be sparse and has many disjoint sub-graphs. The proposed *selection mechanism* provides a connection between two disjoint sub-graphs by considering their inter-dependencies in contents, which may assist information propagation and alleviate sparsity problems. Therefore, without *selection mechanism*, information propagation

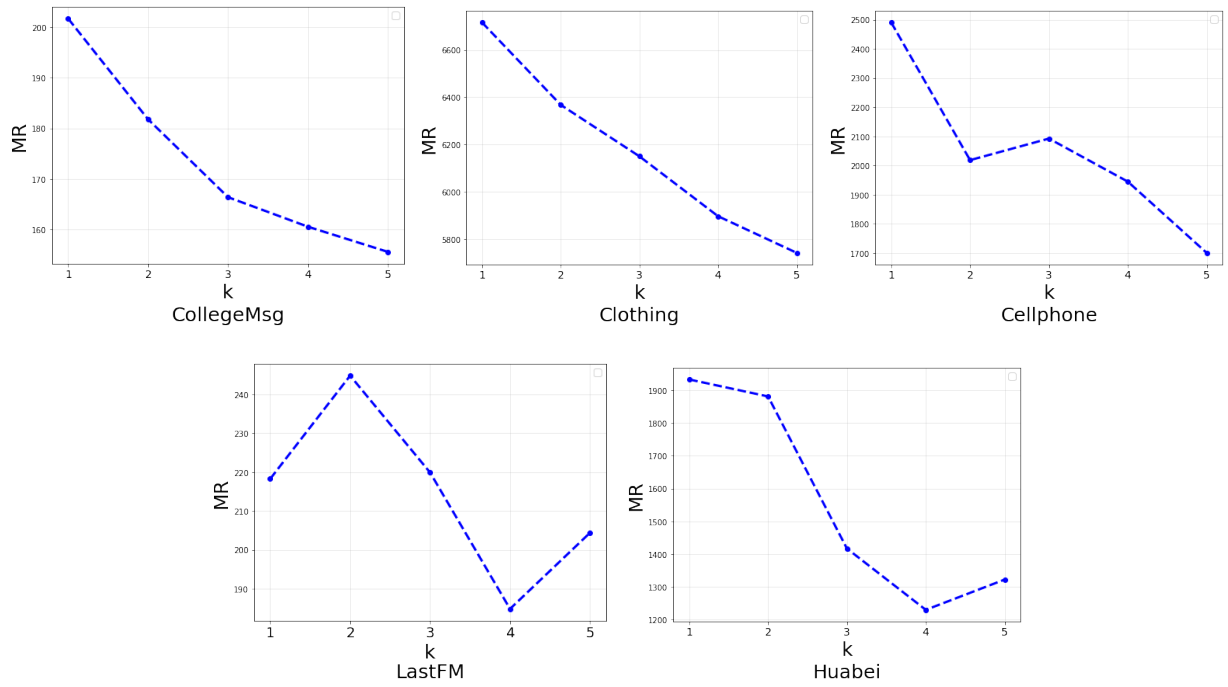


Figure 3: Mean Rank of our method with different depths in five datasets

between the two disjoint sub-graphs $\mathcal{G}_t(u,k)$ and $\mathcal{G}_t(v,k)$ associated with the $l_{u,v,t}$ will be blocked, which may lead to less informative dynamic embeddings especially for scenes that have less repetitive interactions.

6.5 Effect of Sub-graph Depth

We vary the depth of sub-graphs(i.e., $\mathcal{G}_t(u,k)$ and $\mathcal{G}_t(v,k)$) to investigate its impact on our model performances as shown in Fig 3. In particular, we search the depth k in the range of {1, 2, 3, 4, 5}. Obviously, increasing the depth has a positive impact on model performances across all the datasets. We attribute this improvement to the usage of more history information which benefits extraction of temporal structure information using **TDIG-MPNN**. Moreover, when the depth increases, our *selection mechanism* can also exploit more high-order information between the two corresponding sub-graphs $\mathcal{G}_t(u,k)$ and $\mathcal{G}_t(v,k)$ to adjust importances of nodes and then enhance the learnt dynamic embeddings.

6.6 Case Study

To verify the ability of our proposed model which can encode *non-linear* temporal structure information effectively, we randomly select an interaction $l_{u_{1807},v_{367},t}$ from Amazon-Clothing test set and visualize message passing on $\mathcal{G}_t(u_{1807},4) \cup \mathcal{G}_t(v_{367},4)$, which is based on the vector magnitudes of *temporal and structural* gate $g^i_{u(t)}$, where $i = 1,2$. The larger value means that more information from dependency nodes flows into the current nodes. As shown in Figure 4, two message paths colored with darker red, i.e., $u_{133} \rightarrow v_{115} \rightarrow v_{115} \rightarrow u_{1807}$ and $u_{133} \rightarrow v_{367} \rightarrow v_{367}$, convey

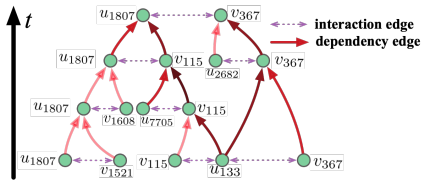


Figure 4: Real Example from Amazon-Clothing for illustrating informative dynamic representation.

more information into the two nodes u_{1807} and v_{367} involved in $l_{u_{1807},v_{367},t}$. This could be explained that v_{115} was reviewed by both u_{1807} and u_{133}, which means they may have common interests, so u_{1807} might be interested in v_{367} which was reviewed by u_{133}. The above illustration provides an evidence that the obtained embeddings can encode temporal structure information well. Meanwhile, this also provides a way for interpretability which is a big advantage over the other *continuous-time* methods.

7 CONCLUSIONS

In this paper, We proposed a *dynamic message passing neural network* (**TDIG-MPNN**) combined with our novel *selection mechanism* to learn informative dynamic graph representations, which is based on our fine-grained graph construction *TDIG*. Experimental results demonstrated the superior performances of our methods, which are consistent on several datasets. Besides, we provide good model explainability via visualizing the learned *temporal and structural gate*. In the future, we will explore additional information(or

knowledge) as used by **KnowEvolve** [21] and **DyREP** [22] to learn dynamic and semantic representations. Meanwhile, how to model long-dependencies on dynamic graph is still a challenging problem. We will consider **Memory Network** [19] to capture *long-term* information on $TDIG$.

8 ACKNOWLEDGEMENT

The authors would like to thank all the anonymous reviewers for their insightful comments.

REFERENCES

[1] Odd Aalen, Ornulf Borgan, and Hakon Gjessing. 2008. *Survival and event history analysis: a process point of view.* Springer Science & Business Media.
[2] Hanjun Dai, Yichen Wang, Rakshit Trivedi, and Le Song. 2016. Deep coevolutionary network: Embedding user and item features for recommendation. *arXiv preprint arXiv:1609.03675* (2016).
[3] Daryl J Daley and David Vere-Jones. 2007. *An introduction to the theory of point processes: volume II: general theory and structure.* Springer Science & Business Media.
[4] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. 2017. Neural message passing for quantum chemistry. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, 1263–1272.
[5] Palash Goyal, Nitin Kamra, Xinran He, and Yan Liu. 2018. Dyngem: Deep embedding method for dynamic graphs. *arXiv preprint arXiv:1805.11273* (2018).
[6] Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable Feature Learning for Networks. *knowledge discovery and data mining* (2016), 855–864.
[7] Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems*. 1024–1034.
[8] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9, 8 (1997), 1735–1780.
[9] John Frank Charles Kingman. 2005. P oisson Processes. *Encyclopedia of biostatistics* 6 (2005).
[10] Thomas N Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. *international conference on learning representations* (2017).
[11] Srijan Kumar, Xikun Zhang, and Jure Leskovec. 2019. Predicting dynamic embedding trajectory in temporal interaction networks. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, 1269–1278.
[12] Jure Leskovec and Andrej Krevl. 2014. SNAP Datasets: Stanford Large Network Dataset Collection. http://snap.stanford.edu/data.
[13] Julian McAuley, Christopher Targett, Qinfeng Shi, and Anton Van Den Hengel. 2015. Image-based recommendations on styles and substitutes. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 43–52.
[14] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*. 3111–3119.
[15] Giang Hoang Nguyen, John Boaz Lee, Ryan A. Rossi, Nesreen K. Ahmed, Eunyee Koh, and Sungchul Kim. 2018. Continuous-Time Dynamic Network Embeddings. In *Companion of the The Web Conference 2018 on The Web Conference 2018, WWW 2018, Lyon , France, April 23-27, 2018*. 969–976. https://doi.org/10.1145/3184558.3191526
[16] Aldo Pareja, Giacomo Domeniconi, Jie Chen, Tengfei Ma, Toyotaro Suzumura, Hiroki Kanezashi, Tim Kaler, Tao B. Schardl, and Charles E. Leiserson. 2020. EvolveGCN: Evolving Graph Convolutional Networks for Dynamic Graphs. In *Proceedings of the Thirty-Fourth AAAI Conference on Artificial Intelligence*.
[17] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 701–710.
[18] Aravind Sankar, Yanhong Wu, Liang Gou, Wei Zhang, and Hao Yang. 2020. DySAT: Deep Neural Representation Learning on Dynamic Graphs via Self-Attention Networks. In *Proceedings of the 13th International Conference on Web Search and Data Mining*. 519–527.
[19] Sainbayar Sukhbaatar, Jason Weston, Rob Fergus, et al. 2015. End-to-end memory networks. In *Advances in neural information processing systems*. 2440–2448.
[20] Aynaz Taheri, Kevin Gimpel, and Tanya Berger-Wolf. 2019. Learning to Represent the Evolution of Dynamic Graphs with Recurrent Models. In *Companion Proceedings of The 2019 World Wide Web Conference*. ACM, 301–307.
[21] Rakshit Trivedi, Hanjun Dai, Yichen Wang, and Le Song. 2017. Know-evolve: Deep temporal reasoning for dynamic knowledge graphs. *arXiv preprint arXiv:1705.05742* (2017).
[22] Rakshit Trivedi, Mehrdad Farajtabar, Prasenjeet Biswal, and Hongyuan Zha. 2019. DyRep: Learning Representations over Dynamic Graphs. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. https://openreview.net/forum?id=HyePrhR5KX
[23] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2017. Graph attention networks. *arXiv preprint arXiv:1710.10903* (2017).
[24] Daixin Wang, Peng Cui, and Wenwu Zhu. 2016. Structural Deep Network Embedding. (2016), 1225–1234.
[25] Lekui Zhou, Yang Yang, Xiang Ren, Fei Wu, and Yueting Zhuang. 2018. Dynamic network embedding by modeling triadic closure process. In *Thirty-Second AAAI Conference on Artificial Intelligence*.