Time-aware Graph Embedding: A temporal smoothness and task-oriented approach

Yonghui Xu, Shengjie Sun, Yuan Miao, Dong Yang, Xiaonan Meng,
Yi Hu, Ke Wang, Hengjie Song, and Chuanyan Miao*

Abstract—Knowledge graph embedding, which aims to learn the low-dimensional representations of entities and relationships, has attracted considerable research efforts recently. However, most knowledge graph embedding methods focus on the structural relationships in fixed triples while ignoring the temporal information. Currently, existing time-aware graph embedding methods only focus on the factual plausibility, while ignoring the temporal smoothness which models the interactions between a fact and its contexts, and thus can capture fine-granularity temporal relationships. This leads to the limited performance of embedding related applications. To solve this problem, this paper presents a Robustly Time-aware Graph Embedding (RTGE) method by incorporating temporal smoothness. Two major innovations of our paper are presented here. At first, RTGE integrates a measure of temporal smoothness in the learning process of the time-aware graph embedding. Via the proposed additional smoothing factor, RTGE can preserve both structural information and evolutionary patterns of a given graph. Secondly, RTGE provides a general task-oriented negative sampling strategy associated with temporally-aware information, which further improves the adaptive ability of the proposed algorithm and plays an essential role in obtaining superior performance in various tasks. Extensive experiments conducted on multiple benchmark tasks show that RTGE can increase performance in entity/relationship/temporal scoping prediction tasks.

Index Terms—Knowledge Graph, Graph Embedding, Temporal Information, Temporal Smoothness

✦

1 INTRODUCTION

With the rapid growth of Knowledge Graph (KG) construction, YAGO [1], Wiki [2], Freebase [3], DBpedia [4], NELL [5] and many other Knowledge Bases (KB) have been created for many real-world applications, *i.e.,* semantic parsing [6], named entity disambiguation [7], information extraction [8], recommender systems [9] and question answering [10]. In order to provide an effective and efficient way to solve the graph analytics problem in these applications, *i.e.,* relation extraction [11], entity classification [12], link prediction [13], entity resolution [14], and Graph Embedding (GE) methods [15] have been proposed. The key idea of GE is to map components (*i.e.,* head entity, tail entity, and relation of the triple $< head-entity, relation, tail-entity >$) or sub-graph of a KG onto a low dimensional space in which the graph information is preserved. By representing a sub-graph (or head entity, tail entity, relation) as a low dimensional space vector, graph analytics can be conducted accurately and efficiently.

Current research on knowledge graph embedding [16, 17, 18, 19] has mainly concerned graphs with fixed triples [20]. However, in real-life scenarios, graphs, like social graphs in Twitter, citation graphs in DBLP [21], are time-variant, and many relations are only valid for a certain period of time. In these applications, the underlying graph structure keeps on changing continuously over time. *i.e.,* in social graphs [22], when a new user registers onto the

graph or friendship is established between two users, a new entity/relation appear in the graph. However, when a user cancels the account or a friendship breaks down, previously established entity/relation disappears. The entity/relation representation of the users should be updated accordingly, such that the learned entity/relation can reflect the temporal evolution of their social relationship. Similarly, due to the frequent publication of a new graph that cites existing technology, the citation graph [23] of scientific papers is continuously enriched. As a result, the influence of the article, and sometimes even the classification, has changed over time. The node embedding needs to be updated to reflect this change. In financial networks [24], transactions are naturally timestamped. If a user is a victim of credit card fraud, or the user's account is involved in money laundering, the characteristics of the user's account may change due to the nature of the transaction involved. In these cases, early detection of such changes is critical to improving law enforcement efficiency and reducing financial institution losses. These unique characteristics of the time-aware graph make traditional graph embedding methods fail to work since static graph embedding methods completely ignore the time-varying information of the graph and cannot capture the evolutionary patterns of the given time-aware graph. Therefore, how to design an embedding method to models the interactions between a fact and its contexts for a time-aware graph is critical.

To embed the temporal information [25] of the graph in the learning model while maintaining the inherent structural information of the given time-aware graph, an obvious way is to slice the graph into different time bins [26]. Then the embeddings can be learned on these bins separately. Although these kinds of models take into account the temporal

---

• *Yonghui Xu, and Chuanyan Miao are with the Nanyang Technological University in Singapore. E-mail:{ASCYMiao}@ntu.edu.sg. Shengjie Sun, Dong Yang, Xiaonan Meng, and Yi Hu are with the Alibaba Group.Yuan Miao is with the Victoria University, Australia. Ke Wang is with the Simon Fraser University, Canada. Hengjie Song is with the South China University of Technology, China.*

information of the graph in the embedding process, they cannot explicitly model the temporally-aware information. This is because these kinds of models are fit on different time bins independently, and cannot share statistical strength between two adjacent time bins. What is worse, such a model trained independently in a fixed time bin cannot remain robust when the structure of the graph changes drastically at a specific time. Suffering from the defects explained above, the current research work on temporal embedding is rather limited. It is necessary to design a graph embedding algorithm that can fit the current graph structure well, and simultaneously it does not deviate too dramatically from recent history.

Previous studies [20, 26] about graph embedding always use the same sampling strategy for different specific tasks, *i.e., head/tail entity prediction task, relationship prediction task.*

- The (head/tail) entity prediction task: we observe the relation and (tail/head) entity in the triple $< head - entity, relation, tail - entity >$, and predict the head/tail entity.
- The relationship prediction task: we observe the head entity and tail entity in the triple $< head - entity, relation, tail - entity >$, and predict the relation according to the learned model.

The triples in the knowledge graph represent real facts and can only be used as positive samples in the process of training the model. To achieve better performance in different specific tasks, we need to construct negative samples based on the triples in the knowledge graph. Most existing sampling strategies obtain negative samples by replacing the head or tail entities of the triples. This kind of negative sampling strategy can obviously improve the performance of the model in the (head/tail) entity prediction task. However, these strategies would not explicitly replace relationships in the triples. This may cause positive and negative samples for the relationship to be unbalanced. Potentially imbalanced data may cause the model learned by the embedding algorithm to be unfavorable for relational prediction tasks. Furthermore, errors introduced by relational embeddings may be passed on to head or tail entity embeddings. This makes graph embedding suffer from the limitations associated with a biased sampling strategy. As a result, how to design a sampling strategy for specific tasks is crucial for graph embedding.

To address the aforementioned issues, we propose a robustly time-aware graph embedding algorithm to encode temporal information in the learned embeddings directly. Particularly, RTGE slices the temporally-scoped input knowledge graph into multiple static subgraphs in which each subgraph corresponds to a timestamp. And then RTGE projects the entities and the relations of each subgraph onto temporally aware hyperplanes. We define a temporal smoothness between hyperplanes of adjacent time steps. By maintaining the temporal smoothness, we expect RTGE can avoid the hyperplanes deviate too dramatically from recent history. Moreover, we propose a task-oriented negative sampling strategy. By performing negative sampling in a balanced manner for both entities and relationships, we hope to obtain training triples with balanced positive and negative samples of entities and relationships. So that the

learned entity embedding and relationship embedding can be applied to a variety of head entity/tail entity/relation prediction tasks. We highlight our contributions as follows:

- Different from previous time-aware graph embedding methods (*i.e.,* t-TransE, HyTE), which learn hyperplanes of adjacent time steps independently, RTGE attempts to maintain the temporal smoothness between hyperplanes of adjacent time steps. Thus, RTGE can model the evolution of KGs more accurately and obtain better performance in the applications.
- Unlike the existing graph embedding algorithms which only perform negative sampling on entities in different tasks, we designed a task-oriented negative sampling strategy in time-aware graph embedding, which can do negative sampling for both entities and relationships in different tasks. The newly proposed strategy can well avoid the problem of sampling data imbalance caused by biased sampling.

We organized the rest of this paper as follows. In the next section, we review the related work. Section 3 presents the problem and our proposed method and how we format the method into an optimization problem in detail. Experimental results on benchmark datasets are reported in Section 4. Finally, Section 5 concludes this paper and discuss future work.

2 RELATED WORKS

Knowledge graph embedding has been an active research area for the past couple of years. Various graph embedding methods [27] have been put forward. Among the existing GE methods, approaches related to our study can be summarized into three categories: *static graph embedding*, *dynamic/incremental graph embedding*, and *time-aware graph embedding*.

2.1 Static Graph Embedding Methods

As a static graph embedding method, TransE [20] proposes an energy-based model for entities embeddings, by requiring the tail entity embedding to be close to the head entity embedding plus a vector corresponding to the relationship. Despite TransE is efficient and straightforward, it has flaws in dealing with reflexive / one-to-many/many-to-one / many-to-many relations. Different from TransE [28], TransH models a relation as a hyperplane together with a translation operation on it. By utilizing the relation-specific hyperplanes, TransH overcomes the flaws of TransE in dealing with reflexive / one-to-many/many-to-one / many-to-many relations. Unlike TransE and TransH which put both entities and relations within the same semantic space, TransR [29] build entity and relation embeddings in separate entity space and relation spaces. In this way, TransR can avoid the problem of insufficient common space in modeling. TransD [30] as an improvement of TransR proposes representing a named symbol object (entity and relation) by two vectors. The first vector represents an entity (or relation), the other vector is used to construct a mapping matrix for each entity-relation pair. By considering the diversity of entities and relations in the process of the construction mapping

matrix, TransD encodes more discriminative information and obtains better results than TransR.

Recently, much effort has been invested in neural embedding for a static graph. For instance, DistMult [31] presents a general neural network framework for multi-relational representation learning. In particular, By utilizing bilinear objective for relation representations, DistMult captures compositional semantics of relations. Besides, DistMult successfully extracts Horn rules that involve compositional reasoning. Instead of using embeddings containing real numbers as DistMult, ComplEx [32] discuss and demonstrate the capabilities of complex embeddings. In this way, ComplEx well avoids overfitting problems caused by the explosion of parameters in existing embedded models when dealing with symmetrical/antisymmetric relationships. DistMult and ComplEx learn less powerful features than deep, multi-layer models since they only focus on shallow, fast models that can scale to large graphs. To solve the problem of shallow architectures, and the overfitting problem of fully connected deep architectures, ConvE [33] proposes to use parameter efficient, fast operators which can be composed into deep networks. Recently, more and more TransE-based or neural network-based graph embedding algorithms (*i.e.,*TranSparse [34], TransF [35]) are proposed for static graph embedding, and they generate effective results on static graph data sets. However, none of the above methods try to combine the temporal information to explore the evolutionary pattern of the knowledge graph.

2.2 Dynamic / Incremental Graph Embedding Methods

Dynamic/incremental KG embedding aims to learn embedding in an online fashion when the KG is frequently updated. For instance, [36] proposes puTransE (Parallel Universe TransE), an online and robust adaptation of TransE. To capture the temporal information for each edge in the knowledge graphs, [37] presents a novel deep evolutionary knowledge network, Know-Evolve which learns nonlinearly evolving entity representations over time. Based on Generalized SVD Decomposition and matrix perturbation theory, [38] dynamically updates the node representation of the dynamic network while retaining high-order similarity. When the network structure changes at the next moment, [38] can quickly and effectively update the representation of the node. Recently, [39] extends graph convolutional network, GCN [40] to the dynamic setting by utilizing a recurrent mechanism to update the parameters. In this way, [39] expects to capture the dynamism of the graphs. Unlike the dynamic graph neural network algorithms which require to retrain a model or wait for convergence, [41] develops new approaches to the problems of streaming graph embedding, by only updating the representations of a small proportion of vertices. In this way, [41] has low space and time complexity to generate latent representations for new vertices under specified iteration rounds. [42] proposes to initialize node embeddings with respect to the static graph. Then the initial node embeddings are aligned at different time points and eventually adapted for the specific task with a joint optimization. To model complex and nonlinearly evolving dynamic processes of the dynamic graph, [43] proposes a deep temporal point process model based on specially designed temporally attentive representation network. By this method, [43] learn to encode structural-temporal information over the graph into low dimensional representations.

2.3 Time-aware graph Embedding Methods

The method proposed in this paper, RTGE, is a typical time-aware graph embedding method. Different from dynamic/incremental graph embedding, time-aware graph embedding, which tries to learn the evolving patterns of a graph and incorporate time information into embedding learning and learns embedding in an offline fashion. For instance, based on TransE, t-TransE [44] provides a link prediction method by using temporal order constraints to model transformation between time-sensitive relations. In the embedding process, t-TransE enforces the embeddings to be temporally consistent. Similarly, HolE [45] earlier attempts to consider such temporal information for graph embedding. To incorporate the valid time of facts, [46] proposes a time-aware graph embedding approach with a joint time-aware inference model using temporal consistency information as constraints. In this way, [46] expects to be more accurate concerning various temporal constraints. Unlike translation based embedding methods, for example, TransE, TransH, and TransR, which ignores the time information of the graph and learns the embedding representation by defining a global margin-based loss function over the data. [47] proposes to encode temporal information of the graph by adaptively adjusting the optimal margin over time.

The most related work to our study is the hyperplane-based temporally aware knowledge graph embedding method (HyTE) proposed by [26]. By investigating different hyperplanes to represent different time (i.e., segregate the embedding space into different time zones by these hyperplanes), HyTE attempts to encode temporal information directly in the learned embeddings. Note that, although the basic ideas behind HyTE and our proposed RTGE are similar, i.e., to learn hyperplanes for different time bins, RTGE differs from HyTE in two aspects: 1) In HyTE, the hyperplanes of adjacent time intervals are independent of each other. On the contrary, in RTGE, we introduce the concept of timing smoothing to optimize and learn the hyperplanes of adjacent time intervals jointly. In this way, RTGE can avoid the problem of missing timing associations between embedded spaces caused by independent learning of hyperplanes of adjacent time intervals. 2) HyTE only uses a negative sampling strategy based on randomly replacing the head or tail entities in the negative sampling process. RTGE adds relation-based negative sampling on the basis of HyTE. In this way, RTGE can avoid the problem of an imbalance of positive and negative samples due to the lack of relationship-based negative sampling.

3 TIME-AWARE GRAPH EMBEDDING MODEL

3.1 Problem Statement

Let $\mathcal{G} = \{< \widehat{h}_i, \widehat{\ell}_i, \widehat{\zeta}_i, t_s^i, t_e^i > | 1 \leq i \leq N\}$ denotes a time-aware graph, where $< \widehat{h}_i, \widehat{\ell}_i, \widehat{\zeta}_i >$ indicates the triple of the graph. $\widehat{h}_i$ denotes a head entity, $\widehat{\ell}_i$ denotes a tail entity and $\widehat{\zeta}_i$ denotes a relation. t_s^i and t_e^i indicate the start and end

timestamps of the fact represented by triplet $< \widehat{h}_i, \widehat{\ell}_i, \widehat{\zeta}_i >$, respectively. t_s^i is the start timestamps of the fact, and t_e^i is the end timestamps of the fact. For series of given timestamp, $t \in 1, 2, \ldots, T$, the time-aware graph $\mathcal{G}$ can be split into multiple static graphs $\mathcal{G}_1 \ldots \mathcal{G}_T$, and each graph consists of several triples that are valid in the corresponding timestamp, e.g., knowledge graph $\mathcal{G}$ can be denoted by,

$$\mathcal{G} = \mathcal{G}_1 \cup \mathcal{G}_2 \cup \ldots \mathcal{G}_t \cdots \cup \mathcal{G}_T \tag{1}$$

where $t \in [1, T]$. Here, we denote the embeddings of head entity $\widehat{h}_i$, tail entity $\widehat{\zeta}_i$ and relation $\widehat{\ell}_i$ by h_i, ζ_i and ℓ_i, respectively. Time-aware graph embedding aims to learn $h_i \in \mathcal{R}^{d \times 1}$, $\zeta_i \in \mathcal{R}^{d \times 1}$ and $\ell_i \in \mathcal{R}^{d \times 1}$ for each entity and relation, and the appropriate mapping functions $\Gamma_t, t \in [1, T]$ to meet the requirements of the following three tasks.

- Head prediction task: for an incomplete fact $< ?, \widehat{\ell}_i, \widehat{\zeta}_i >$ at t, Γ_t can predict the head entity $\widehat{h}_i$.
- Relation prediction task: for an incomplete fact $< \widehat{h}_i, ?, \widehat{\zeta}_i >$ at t, Γ_t can predict the relation $\widehat{\ell}_i$.
- Tail prediction task: for an incomplete fact $< \widehat{h}_i, \widehat{\ell}_i, ? >$ at t, Γ_t can predict the tail entity $\widehat{\zeta}_i$.
- Temporal scoping prediction task: for a fact $< \widehat{h}_i, \widehat{\ell}_i, \widehat{\zeta}_i >$, Γ_t can predict the temporal scoping of this fact.

3.2 Modeling Embeddings Over Time

This section deduces how temporal information can be integrated into graph embedding based on TransE. The basic idea of TransE is that it models relationships as translations in the embedding space, and enforces the embedding of the tail entity $\widehat{\zeta}_i$ to be close to the embedding of the head entity $\widehat{h}_i$ plus some vector that depends on the relationship $\widehat{\ell}_i$, (i.e., $h_i + \ell_i \approx \zeta_i$) when the triplet $< \widehat{h}_i, \widehat{\ell}_i, \widehat{\zeta}_i >$ holds. Obviously, if the triplet $< \widehat{h}_i, \widehat{\ell}_i, \widehat{\zeta}_i >$ does not hold, TransE enforces $h_i + \ell_i$ to be far away from ζ_i. Based on this idea, TransE can obtain the embeddings by minimizing a margin-based ranking loss over the whole training set [20].

TransE provides a basic framework to embed entity and relation in the semantic space. Though it works well on irreflexive and one-to-one relations for a static graph, it has problems to deal with reflexive or many-to-one/one-to-many/many-to-many relations for a time-aware graph. For instance, fact: "*Jone lives in Beijing in 2018*", can be described with a triple as (2).

$$< \widehat{h}_i : John, \widehat{\ell}_i : lives-in, \widehat{\zeta}_i : Beijing, t_s^i : 2018, t_e^i : 2018 > \tag{2}$$
$$< \widehat{h}_j : John, \widehat{\ell}_j : lives-in, \widehat{\zeta}_j : Singapore, t_s^j : 2019, t_e^j : 2019 > \tag{3}$$

For another fact: "*Jone lives in Singapore in 2019*", we can describe it as (3). Since: "*Jone moved from Beijing to Singapore in 2019*", the triple (2) and the triple (3) have the same head entity and relationship but has different tail entities. If the time information is not considered, TransE mandates $h_i + \ell_i = \zeta_i$ and $h_j + \ell_j = \zeta_j$. Since $h_i = h_j$ and $\ell_i = \ell_j$, TransE will deduce a wrong conclusion $\zeta_i = \zeta_j$.

In order to avoid the above problem, temporal aware hyperplane can be utilized to segregate the embedding space into different time zones. With the help of temporal

aware hyperplane at time t, the representation of the triple valid at time t will be projected onto hyperplane $w_t \in \mathcal{R}^{d \times 1}$ as follows,

$$\mathcal{Q}_t(h_i) = h_i - (w_t^\top h_i) w_t \tag{4}$$

$$\mathcal{Q}_t(\ell_i) = \ell_i - (w_t^\top \ell_i) w_t \tag{5}$$

$$\mathcal{Q}_t(\zeta_i) = \zeta_i - (w_t^\top \zeta_i) w_t \tag{6}$$

where $\mathcal{Q}_t(h_i)$, $\mathcal{Q}_t(\ell_i)$ and $\mathcal{Q}_t(\zeta_i)$ denote the projection of the head entity $\widehat{h}_i$, the relationship $\widehat{\ell}_i$, and the tail entity $\widehat{\zeta}_i$ on the hyperplane w_t, respectively. With this approach, triples with the same head entity and the same relationship at different times will be projected into different subspaces, and the tail entities of these triples will be represented as different embeddings in different subspaces. Thus, the many-to-one problem caused by time is avoided.

Following the strategy adopted in previous method, HyTE, we learn the embeddings in (4), (5) and (6) by minimizing a margin-based ranking loss over the training set,

$$\arg \min_{W, h, \ell, \zeta} g(W, h, \ell, \zeta) = \tag{7}$$
$$\sum_{t=1}^{T} \sum_{s_i^+}^{S_t^+} \sum_{s_j^-}^{S_t^-} \max \left(\mathcal{L}(s_i^+) + \gamma - \mathcal{L}(s_j^-), 0 \right)$$

where $W = [w_1, w_2, \ldots, w_T]$. s_i^+ indicates the fact $< \widehat{h}_i, \widehat{\ell}_i, \widehat{\zeta}_i >$ which is valid during timestamp t. s_i^+ can be denoted as,

$$s_i^+ = < h_i, \ell_i, \zeta_i > \in S_t^+. \tag{8}$$

s_j^- indicates the negative fact which is not valid during timestamp t. s_j^- can be generated by replacing the head entity or the tail entity of a valid fact $< \widehat{h}_j, \widehat{\ell}_j, \widehat{\zeta}_j >$. If we use a negative head entity, s_j^- can be defined as,

$$s_j^- = < h_j', \ell_j, \zeta_j > \in S_t^-. \tag{9}$$

For a negative tail entity, s_j^- can be defined as,

$$s_j^- = < h_j, \ell_j, \zeta_j' > \in S_t^-. \tag{10}$$

$\mathcal{L}$ in (7) indicates the loss corresponding to the projection of triples on w_t. For instance,

$$\mathcal{L}(s_i^+) = \mathcal{L}(h_i, \ell_i, \zeta_i) = \| \mathcal{Q}_t(h_i) + \mathcal{Q}_t(\ell_i) - \mathcal{Q}_t(\zeta_i) \| \tag{11}$$

3.3 Temporal Smoothness

According to the introduction in the previous section, the optimal W, h, ℓ, ζ can be solved by minimizing (7). However, as can be seen from (8), the model (7) is only the total sum of the margin-based ranking criterion on each subgraph. For the fixed h, ℓ, ζ, the w_t learning process corresponding to each timestamp is independent of other timestamps. This result in the model learned by minimizing (7) only containing subgraph structure information at different timestamps, but not the evolutionary pattern of the global graph.

To explain this problem more clearly, Fig. 1 shows an illustration of a time-aware graph and its relevant embedding evolution over time. As the figure shows, h^0, ζ^0, and ℓ^0 indicates the embedding of node 1, node 2, and
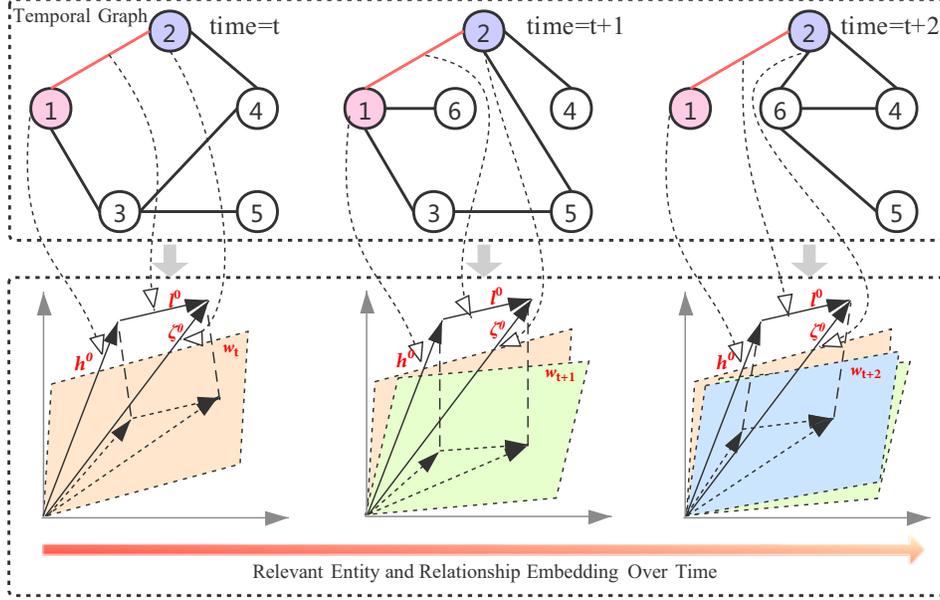
Fig. 1: Illustration of time-aware graph and its relevant embedding evolution over time.

the associated relationship at time 0, respectively. ω_t, ω_{t+1} and ω_{t+2} represent the optimal hyperplane at time t, $t+1$ and $t+2$, respectively. From the figure, we can see that there are differences in the optimal hyperplanes at different times, and there is a close relationship between the optimal hyperplanes at adjacent times. This example validates the necessity of introducing temporal smoothness in the time-aware graph embedding model.

Actually, a graph may not change much in a short period of time, so the embedded space should not change too much. Inspired by this factor, we propose constraining the variation between hyperplanes at adjacent timestamps while accumulating the margin-based ranking criterion of each subgraph. Therefore, we define the temporal smoothness by minimizing the Euclidean distance between hyperplanes in adjacent timestamps. Formally, the corresponding loss function is,

$$g(W)_{smooth} = \sum_{t=1}^{T-1} \|w_{t+1} - w_t\|_2 \tag{12}$$

The smoothness for RTGE defined in (12) aims to request the coordinates of an entity or a relationship to be very close to the average coordinates of its neighbors over adjacent time. For example, a person may experience the following four events at different moments,

Born $(t_1) \longrightarrow$ *Go to school* $(t_2) \longrightarrow$ *Graduate from a school* $(t_3) \longrightarrow$ *Go to a company to work* $(t_4) \longrightarrow$ *Death* (t_5)

The four events mentioned above naturally have a time sequence. For example, "*Go to school*" cannot occur before "*Born*". "*Go to a company to work*" cannot happen after "*Death*". When learning the representation of $t_1 \ldots t_5$, if the constraints of temporal smoothness are lacking, t_1 may be farther away from t_2 and t_3, but closer to t_5. This may cause the model to observe "*Go to a company to work*" at the previous moment and predict "*Born*" at the next moment. This prediction is clearly contrary to the facts. Therefore,

it is necessary to keep the timing smooth constraint in the process of learning time embedding.

3.4 Task-Oriented Negative Sampling

Another problem with the model (7) occurs during the negative sampling process. This section analyzes the advantages and disadvantages of existing negative sampling strategies in different tasks, and proposes a task-oriented negative sampling strategy based on this analysis. Note that (7) can obtain a large number of negative sample triples by randomly replacing the head or tail entities in the training triples. By introducing a negative sample into the loss function in (7), the discriminating ability of the model concerning the head or tail entity can be significantly improved, thereby improving the predicted performance of the learned embeddings for head or tail entity prediction task.

By comparing (8), (9) and (10), we can find that the negative sampling of the relationship is not shown in the negative sampling strategy of (7). Although some negative samples about relationships can be obtained in negative entity sampling, in the absence of independent relationship negative sampling, the balance of positive and negative samples is difficult to guarantee. In this case, the performance of the model (7) in relational prediction tasks is limited. This problem is even more serious when the number of relationships is large, or the relationship has a high degree of similarity.

Based on the above analysis, the existing negative sampling strategy is more suitable for entity prediction tasks, but not suitable for relational prediction tasks. To improve the robustness and adaptability of graph embedding algorithms in different tasks, we propose a task-oriented negative sampling strategy that considers the negative sampling of the relationship while considering the negative sampling

of the head and tail entities. The specific loss function and the sampling method are as follows,

$$g_{task}(W, h, \ell, \zeta) = \tag{13}$$

$$\sum_{t=1}^{T} \sum_{s_i^+}^{S_t^+} \sum_{s_{j,e}^-, s_{k,r}^-}^{S_{t,e}^-, S_{t,r}^-} \max \left[2\mathcal{L}(s_i^+) + \gamma - \mathcal{L}(s_{j,e}^-) - \beta\mathcal{L}(s_{k,r}^-), 0 \right]$$

where $g_{task}(W, h, \ell, \zeta)$ is the loss function corresponding to specific task. S_t^+ indicates the positive triple set for training,

$$S_t^+ = \{< h_i, \ell_i, \zeta_i > |$$
$$< h_i, \ell_i, \zeta_i > \in \mathcal{G}_t, i \in [1, |\mathcal{G}_t|]\} \tag{14}$$

$S_{t,e}^-$ indicates the sampled negative triple set for training which is generated by replacing the head or tail entity in S_t^+,

$$S_{t,e}^- = \{< h_i^{'}, \ell_i, \zeta_i >, < h_i, \ell_i, \zeta_i^{'} > |$$
$$< h_i, \ell_i, \zeta_i > \in \mathcal{G}_t, < h_i^{'}, \ell_i, \zeta_i > \in \overline{\mathcal{G}_t},$$
$$< h_i, \ell_i, \zeta_i^{'} > \in \overline{\mathcal{G}_t}\} \tag{15}$$

$S_{t,r}^-$ indicates the sampled negative triple set for training which is generated by replacing the relation in S_t^+,

$$S_{t,r}^- = \{< h_i, \ell_i^{'}, \zeta_i > |$$
$$< h_i, \ell_i, \zeta_i > \in \mathcal{G}_t, < h_i, \ell_i^{'}, \zeta_i > \in \overline{\mathcal{G}_t}\} \tag{16}$$

$|\mathcal{G}_t|$ indicates the size of $\mathcal{G}_t$. $\overline{\mathcal{G}_t}$ denotes the complement of $\mathcal{G}_t$, $\overline{\mathcal{G}_t} \cup \mathcal{G}_t = \mathcal{G}$ and $\overline{\mathcal{G}_t} \cap \mathcal{G}_t = \emptyset$.

$\mathcal{L}(s_{j,e}^-)$ in (13) represents the loss function associated with the entity negative triple, and $\mathcal{L}(s_{k,r}^-)$ represents the loss function associated with the relationship negative triple. Compared to (8), (13) replaces the loss $\mathcal{L}(s_j^-)$ associated with the head entity and the tail entity negative triple in (8) with a linear combination of $\mathcal{L}(s_{j,e}^-)$ and $\mathcal{L}(s_{k,r}^-)$. Following consideration of the difference between the magnitude of the entity and the relationship, excessive use of negative triples about relationships may affect the ability of the model to discriminate on the entity. To avoid this problem, we add a parameter β for $\mathcal{L}(s_{k,r}^-)$ to adjust its weight. In this way, we introduce the discriminant characteristics of the relationship in the model (13), so that the model can be applied to both the prediction task of the entity and the prediction task of relationship.

3.5 Model Learning

By combining (12) and (13), we derive an overall optimization approach for the following unified optimization problem, which is constructed by using $\mathcal{J}(W, h, \ell, \zeta)$ as a general loss function,

$$\arg \min_{W, h, \ell, \zeta} \mathcal{J}(W, h, \ell, \zeta) = \tag{17}$$
$$\alpha * g_{smooth}(W) + g_{task}(W, h, \ell, \zeta)$$
$$s.t. \quad \|h_i\|_2 = 1, \|\ell_i\|_2 = 1, \|\zeta_i\|_2 = 1, i \in [1, N].$$
$$\|w_t\|_2 = 1, t \in [1, T].$$

where α is a tradeoff parameter. In this section, we derive approaches to solve the optimization problems constructed

in (17). Firstly, we convert the optimization problem to an unconstrained one,

$$\arg \min_{W, h, \ell, \zeta} \mathcal{J}(W, h, \ell, \zeta) = \tag{18}$$
$$\alpha * g_{smooth}(W) + g_{task}(W, h, \ell, \zeta) +$$
$$\xi \sum_{t=1}^{T} (\|w_t\|_2 - 1)^2 + \xi \sum_{i=1}^{N} [(\|h_i\|_2 - 1)^2$$
$$+ (\|\ell_i\|_2 - 1)^2 + (\|\zeta_i\|_2 - 1)^2],$$

where ξ is a tradeoff parameter. Then, we propose an alternating optimization algorithm to learn W, h, ℓ and ζ alternatively and iteratively. To be specific, at the ρ-th iteration, we first fix the matrix h, ℓ and ζ and update the value of each w_t in W using gradient descent based on the following rule,

$$(w_t)_{\rho+1} = (w_t)_\rho - \psi \frac{\partial \mathcal{J}(W, h, \ell, \zeta)}{\partial w_t} \tag{19}$$

where ψ indicates the learning rate, and

$$\frac{\partial \mathcal{J}}{\partial w_t} = \alpha \left(\frac{w_t - w_{t+1}}{\|w_{t+1} - w_t\|_2} + \frac{w_t - w_{t-1}}{\|w_t - w_{t-1}\|_2} \right)$$
$$+ \sum_{s_i^+}^{S_t^+} \sum_{s_{j,e}^-, s_{k,r}^-}^{S_{t,e}^-, S_{t,r}^-} [2\nabla_{w_t}\mathcal{L}(s_i^+) - \nabla_{w_t}\mathcal{L}(s_{j,e}^-)$$
$$- \beta\nabla_{w_t}\mathcal{L}(s_{k,r}^-)]_{\dagger} + 2\xi(\|w_t\|_2 - 1)w_t,$$

where $[\ldots]_{\dagger}$ in (20) is an indication function. If $2\mathcal{L}(s_i^+) - \mathcal{L}(s_{j,e}^-) - \beta\mathcal{L}(s_{k,r}^-) > 0$ then $[x]_{\dagger} = x$, otherwise $[x]_{\dagger} = 0$.

After updating the value of W, we then alternatively fix W and update h, ℓ and ζ respectively, based on the following rule,

$$(h_u)_{\rho+1} = (h_u)_\rho - \psi \frac{\partial \mathcal{J}(W, h, \ell, \zeta)}{\partial h_u} \tag{20}$$

$$(\ell_u)_{\rho+1} = (\ell_u)_\rho - \psi \frac{\partial \mathcal{J}(W, h, \ell, \zeta)}{\partial \ell_u} \tag{21}$$

$$(\zeta_u)_{\rho+1} = (\zeta_u)_\rho - \psi \frac{\partial \mathcal{J}(W, h, \ell, \zeta)}{\partial \zeta_u} \tag{22}$$

where

$$\frac{\partial \mathcal{J}}{\partial h_u} = \sum_{t=1}^{T} \sum_{s_i^+}^{S_t^+} \sum_{s_{j,e}^-, s_{k,r}^-}^{S_{t,e}^-, S_{t,r}^-} [2\nabla_{h_u}\mathcal{L}(s_i^+) - \nabla_{h_u}\mathcal{L}(s_{j,e}^-)$$
$$- \beta\nabla_{h_u}\mathcal{L}(s_{k,r}^-)]_{\dagger} + 2\xi(\|h_u\|_2 - 1)h_u,$$

$$\frac{\partial \mathcal{J}}{\partial \ell_u} = \sum_{t=1}^{T} \sum_{s_i^+}^{S_t^+} \sum_{s_{j,e}^-, s_{k,r}^-}^{S_{t,e}^-, S_{t,r}^-} [2\nabla_{\ell_u}\mathcal{L}(s_i^+) - \nabla_{\ell_u}\mathcal{L}(s_{j,e}^-)$$
$$- \beta\nabla_{\ell_u}\mathcal{L}(s_{k,r}^-)]_{\dagger} + 2\xi(\|\ell_u\|_2 - 1)\ell_u,$$

$$\frac{\partial \mathcal{J}}{\partial \zeta_u} = \sum_{t=1}^{T} \sum_{s_i^+}^{S_t^+} \sum_{s_{j,e}^-, s_{k,r}^-}^{S_{t,e}^-, S_{t,r}^-} [2\nabla_{\zeta_u}\mathcal{L}(s_i^+) - \nabla_{\zeta_u}\mathcal{L}(s_{j,e}^-)$$
$$- \beta\nabla_{\zeta_u}\mathcal{L}(s_{k,r}^-)]_{\dagger} + 2\xi(\|\zeta_u\|_2 - 1)\zeta_u.$$

Algorithm 1 RTGE

Result: Output W, h, ℓ and ζ

Initialization W, h, ℓ and ζ, maximum number of iterations κ, threshold ϵ.

while $t \leq T$ **do**

    Sampling entity negative tripe set S_t^+ based on (15).

    Sampling relation negative tripe set S_r^+ based on (16).

end

while *the number of iteration* $\leq \kappa$ *or* $\mathcal{J}(W, h, \ell, \zeta)$ *in (17) does not converge* **do**

    if W *does not converge* **then**

        Update the value of each w_t in W by (19).

    end

    if h, ℓ *or* ζ *do not converge* **then**

        Update head entity embedding h by (20).

        Update relation embedding ℓ by (21).

        Update tail entity embedding ζ by (22).

    end

end

TABLE 1: Characteristic information of the Wikidata12K Data set and YAGO11K Data set.

Datasets	#Entity	#Relations	Train	Valid	Test	Period (year)
Wikidata12K	12,554	24	32.5k	4k	4k	1320 - 2019
YAGO11K	10,623	10	16.4k	2k	2k	1900 - 2017

We alternatingly and iteratively update W, h, ℓ and ζ until the change in values of the objective function $\mathcal{J}$ is less than a threshold ϵ. Considering that the possible combination of negative triples is enormous, we sample several negative triples for each training triplet. The negative triples are randomly sampled and include three groups, (M negative head entity samples, M negative relationship samples, and M negative tail entity samples).

3.6 Computational Complexity

In this section, we analyze the computational complexity of our proposed RTGE. The total time spent is mainly determined by the time complexity of computing $\mathcal{J}$'s gradients concerning W, h, ℓ and ζ. The computational cost for computing the gradient of $\mathcal{J}$ with respect W, h, ℓ and ζ is $O(CT\kappa d)$, where C indicates the number of constraint pairs consisting of positive and negative sample triples, and d is the dimensionality of each embedding. Moreover, the computational cost caused by other operations, in Algorithm 1 is not more than $O(CT\kappa d)$. Therefore, the overall computational complexity of Algorithm 1 is $O(CT\kappa d)$.

4 EXPERIMENTAL RESULTS

This section conducts experiments to evaluate RTGE [1] and demonstrate its advantages through comparative study.

---

1. We will open-source code and data sets upon the publishing of this paper.

4.1 Data Sets with Time-aware Information

We conduct extensive experiments on two famous benchmark datasets, the Wikidata12k data set and the YAGO11k data set. YAGO11k is drawn from YAGO3 [48] in which some temporally associated facts have meta-facts as (factID, occurSince, start-time), (factID, occurSince, end-time). The total number of time annotated facts in YAGO3 containing both occursSince and occursUntil is 722,494. We choose the top 10 most frequent temporally rich relations of YAGO3 according to the preprocessing method proposed by [48]. To handle sparsity and ensures healthy connectivity within the graph, we recursively delete the edges in the subgraph that contain only one mention containing the entity. Finally, we obtain a purely time-aware graph. We process the Wikidata [2] dataset according to a similar method of processing YAGO3, and obtain Wikidata12k. Different from YAGO11k, we select the top 24 frequent temporally rich relations for Wikidata12k.

The time annotations in YAGO11k and Wikidata12k includes the year, month, and day information. Following the setting in HyTE, we drop the month and data information. To distribute the time annotations in the KG uniformly, we club the less frequent year mentions into the same time interval by applying a minimum threshold of 300 triples per interval during construction. For the years with high frequency, we club them into individual intervals. Then we treat timestamps as 61 and 78 different intervals for YAGO and Wikidata, respectively. Statistics of the Wikidata12k and the YAGO11k are summarized in Table 1.

Although WordNet [49] and Freebase [3] are accessible knowledge graph datasets for static knowledge graph research, we do not test on these two datasets. This is because the algorithm proposed in this paper is aimed at the problem of knowledge graph embedding with time information, and the current version of these data sets lacks time information.

4.2 Entity, Relation, and Temporal Scoping Prediction Tasks

To verify the performance of RTGE, we conduct four types of tasks on each time-aware dataset (including head entity prediction, tail entity prediction, relationship prediction, and temporal scoping prediction). The settings for these tasks are as described in the problem statement section. For a given test triplet with missing entities or relationships, we use formula (1) to calculate the loss of all potential entities or relationships with the test triplet under this formula. We sort the potential entities or relationships in ascending order according to the loss value. For evaluation, we select the ranking of real entities or relationships corresponding to the triples. For a given test triple with missing time, we project the entities and relationships in the triple to all potential time hyperplanes. We calculate the loss of all potential time and test triples under this formula according to formula (1), and sort the potential time according to the loss value. For evaluation, we select the real-time ranking corresponding to the triple. In these tasks, we follow a time agnostic negative sampling procedure to generate negative samples. For instance, with a given tail and head query

TABLE 2: Experimental results on Wikidata12K data set. ↓ means the lower the better. ↑ means the higher the better.

Metric	Mean Rank ↓	Mean Rank ↓	Mean Rank ↓	Hits@10(%) ↑	Hits@10(%) ↑	Hits@1(%) ↑
Task	head	tail	relation	head	tail	relation
Trans-E [20]	740	520	1.35	6	11	88.4
TransH [28]	648	423	1.4	11.8	23.7	88.1
DistMult [31]	635	531	-	19.6	26.6	-
ComplEx [32]	706	551	-	11.8	23.7	-
ConvE [33]	355	241	-	25.5	33.4	-
HolE [45]	808	734	2.23	12.3	25	83.96
t-TransE [44]	413	283	1.97	14.5	24.5	74.2
HyTE [26]	237	179	**1.13**	25	41.6	**92.6**
RTGE-n ($d = 128$)	**201**	**141**	**1.13**	**29.7**	**44.6**	92.6
RTGE-s ($d = 128$)	**210**	**148**	**1.11**	**29.5**	**43.4**	92.6
RTGE ($d = 128$)	**183**	**127**	**1.09**	**29.9**	**44.6**	92.8
RTGE ($d = 256$)	**153**	**91**	**1.08**	**32.6**	**49.7**	93.5

TABLE 3: Experimental results on YAGO11K data set. ↓ means the lower the better. ↑ means the higher the better.

Metric	Mean Rank ↓	Mean Rank ↓	Mean Rank ↓	Hits@10(%) ↑	Hits@10(%) ↑	Hits@1(%) ↑
Task	head	tail	relation	head	tail	relation
Trans-E [20]	2020	504	1.7	1.2	4.4	78.4
TransH [28]	1808	354	1.53	1.5	5.8	76.1
DistMult [31]	1550	616	-	17.3	31.4	-
ComplEx [32]	1758	603	-	19.2	35.3	-
ConvE [33]	1464	702	-	19.2	33.5	-
HolE [45]	1953	1828	2.57	13.7	29.4	69.3
t-TransE [44]	1692	292	1.66	1.3	6.2	75.5
HyTE [26]	1069	107	1.23	16	38.4	81.2
RTGE-n ($d = 128$)	**854**	113	**1.22**	**20.52**	**40.9**	83.9
RTGE-s ($d = 128$)	**891**	118	1.24	**20.1**	**39.1**	82.8
RTGE ($d = 128$)	**799**	110	**1.15**	**20.1**	**40.9**	88.2
RTGE ($d = 256$)	**725**	**105**	**1.11**	**21.4**	**41.5**	88.9

term, we randomly replace a tail or head entity such that newly generated triple is not observed.

4.3 Baselines

We compare RTGE with a set of state-of-the-art graph embedding algorithms.

- Firstly, considering that our RTGE is a kind of translation-style approach, we compare with Trans-E and TransH, which are two translation-based graph embedding algorithm.
- Secondly, noting that the outstanding performance of graph neural network algorithms in graph embedding has received increasing attention from researchers in recent years, we include DistMult, ComplEx, and ConvE as baselines for comparison.
- Thirdly, considering that we have motivated the study by using time-aware information, whereas HolE, t-TransE, and HyTE are known as an effective time-aware graph embedding algorithm for entity or relation prediction tasks, we include they as baselines for comparison.
- Regarding our proposed RTGE, to further investigate the impact of the negative relation sampling and the temporal smoothness to the overall performance, we denote by RTGE-s a reduction of RTGE only using temporal smoothness and randomly selecting the negative entity sampling. We denote by RTGE-n a reduction of RTGE only using the negative relation

sampling and learning all the hyperplanes independently.

We use similar evaluation metrics as traditional knowledge graph embedding methods for time-aware knowledge graph embedding methods. RTGE utilizes equation (11) to calculate the loss corresponding to triples formed by each potential head entity and the observed tail entity and relationship. We calculate and record the ranking of the loss for the real head entity after sorting all losses. Then, we report the mean rank, Hits@1, Hits@2,..., and Hits@10. Other prediction tasks use the same method to evaluate the performance of the algorithms.

4.4 Qualitative Results

Table 2 and Table 3 reports the experimental results on Wikidata12K and YAGO11K data sets. We compare and analyze the performance differences of the proposed RTGE and benchmark algorithms from the following five aspects. In order to show the performance of RTGE on two benchmark datasets in more detail, we report the experimental results of RTGE on Wikidata12K and YAGO11K data sets concerning different Hits@X in Figure. 2. It can be seen from the figure that when X is smaller, the algorithm is less effective in the entity prediction task. In the entity prediction task, the smaller X is not enough to reflect the overall performance of the algorithm. As X gradually increases to 10, the effect of the algorithm in the entity prediction task is significantly improved. A larger X can better reflect the effect of the
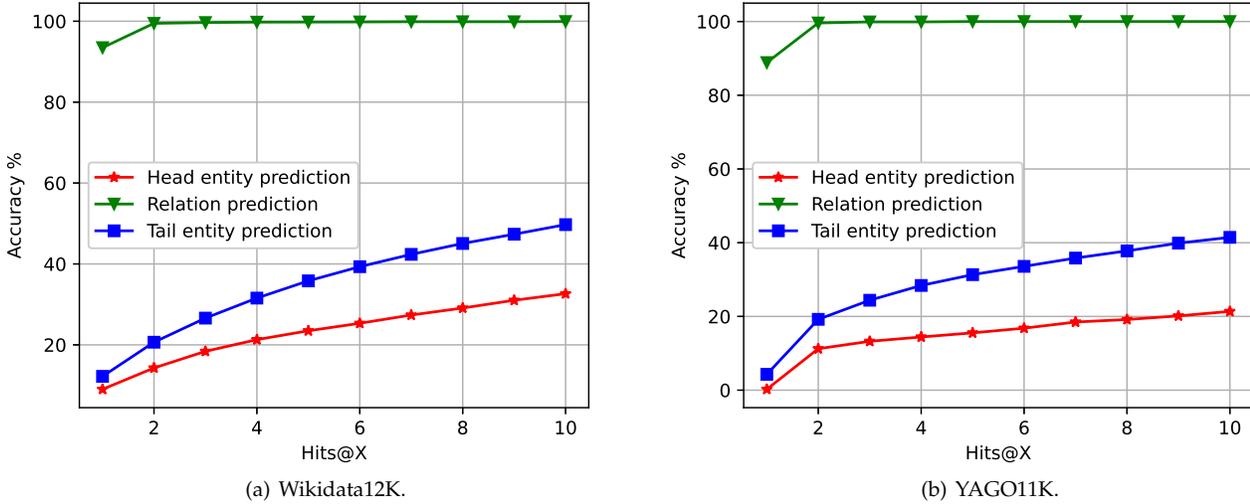
(a) Wikidata12K.

(b) YAGO11K.

Fig. 2: Experimental results of RTGE on Wikidata12K and YAGO11K data sets with respect to different Hits@X.
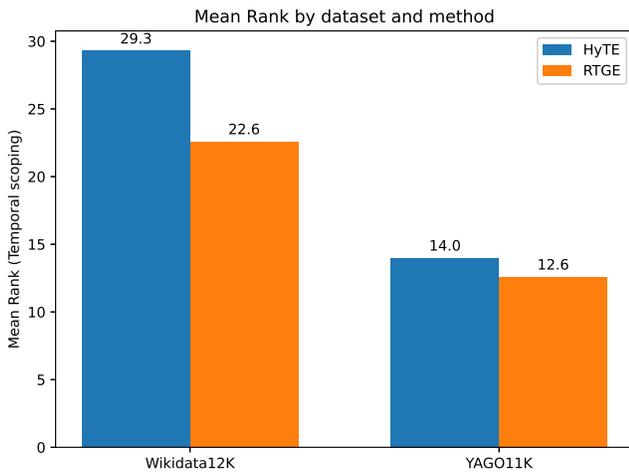

Fig. 3: The predicted Mean Rank (lower the better) for temporal scoping.

algorithm. In contrast, in the relationship prediction task, when X is larger, the accuracy of the algorithm always approaches 100%. This is not conducive to the performance of the comparison algorithm. Therefore, a smaller X is more suitable for the relationship prediction task.

4.4.1 Comparative Analysis with Translation-based algorithms

From the Table 2 and Table 3, we can observe that the average performances of Trans-E and TransH on all the three tasks are worse than the performance of RTGE. This is because Trans-E and TransH only learn one embedding for each entity or relationship at different time intervals. However, as time changes, new entities or relationships are added to the knowledge graph, and old entities or relationships disappear. This means that the structure of the knowledge graph is constantly changing at different time intervals. It is insufficient to rely on a fixed embedding to express entities or relationships at different times in time.

4.4.2 Comparative Analysis with Graph Neural Network-based algorithms

Table 2 and Table 3 show that the three graph-based neural network algorithms DistMult, ComplEx, and ConvE have significantly better execution results than the translation-based algorithms Trans-E and TransH. However, due to the lack of modeling of time information, even if DistMult, ComplEx, and ConvE use more complex neural network models, the effect is still no better than RTGE. These results indicate that the static knowledge graph embedding algorithm has inherent flaws when processing time-aware knowledge graphs.

4.4.3 Comparative Analysis with Time-aware Graph Embedding algorithms

Despite t-TransE and HyTE both take into account the time factor and try to use temporal information during model transformation or projected-time translation, their performance still does not exceed RTGE-s and RTGE. This is because RTGE not only considers the graph structure information at different timestamps but also consider the association between graph structures between adjacent timestamps. Due to this advantage, RTGE learns the evolution of time-aware graphs over time more accurately, thus predicting entities or relationships more accurately at a future moment.

4.4.4 Verification on Temporal Smoothness and Task-Oriented Negative Sampling

By comparing RTGE-s and HyTE in Table 2 and Table 3, we can verify the effectiveness of temporal smoothness. From the experimental results in the two tables, it can be seen that compared to HyTE, RTGE-s achieved a better result in most test groups. This is because, compared to HyTE, RTGE-s introduces temporal smoothness terms in the embedding model by constraining the variation between hyperplanes at adjacent timestamps. In this way, RTGE-s avoids the challenge of the anomaly data to the model and indirectly improves the robustness of the graph embedding algorithm. Therefore, RTGE-s achieve better performance than HyTE.
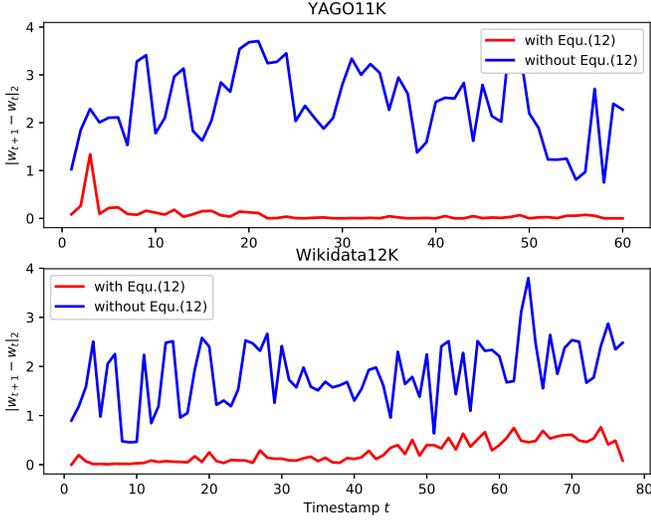
Fig. 4: Statistics of w_t based on time embeddings obtained after training RTGE with / without equation (12).

By comparing RTGE-n and HyTE in Table 2 and Table 3, we can verify the effectiveness of task-oriented negative sampling. From the figues, we can find that RTGE-n not only outperforms HyTE in relational prediction tasks but also outperforms HyTE in entity prediction tasks. This is because, compared to HyTE, RTGE-n introduces negative sampling triplets for the relationship in the embedding model, which significantly improves the discriminative ability of the model for different relationships, and it further enhances the performance of the embedding model in the entity prediction task.

4.4.5 Temporal Scoping Prediction Result

Fig. 3 shows the experimental results of temporal scoping prediction. It can be seen from the figure that the predicted mean rank for temporal scoping of RTGE is lower than in HyTE. This is because HyTE does not use a timing smoothing mechanism, which is not conducive to learning time embedding to maintain timing information. Hence, HyTE is not conducive to temporal scoping for accurately predicting events. On the contrary, RTGE introduces a temporal smoothness mechanism, which can well avoid this problem encountered in HyTE. Fig. 4 shows the $\|w_{t+1} - w_t\|_2$ based on time embeddings which are obtained after training RTGE with or without equation (12). It can be seen from the figure that in the w_t's learned by RTGE with equation (12), the change of w_t at the neighboring moment is relatively smooth, and the change range is relatively small, which is benefited from the timing smoothing mechanism of RTGE. However, in the w_t's learned by RTGE without equation (12), the variation range of w_t at the neighboring moment is large and irregular, which is not conducive to maintaining the consistency of the embedded timing.

In order to show the time embedding learned by RTGE and HyTE more intuitively, Fig. 5 shows the 2-d PCA projection of time embeddings, which are obtained after training RTGE for the temporal scoping task. We observe that the time representation after training RTGE is forming natural clusters in chronological order. However, the time representation after training HyTE is more evenly

distributed in the figure. Only a small number of adjacent time representations come together. This indirectly verifies that the time embedding learned by the RTGE model can effectively retain the time sequence information in the time-aware knowledge graph.

4.5 Parameter Tuning

This section shows the results of RTGE's parameter tuning experiments, including the trade-off parameter β, the number of negative sampling triples m, the trade-off parameter α, ξ, the learning rate ψ, the margin γ, embedding dimension d. In our experiments, ξ, and ψ are empirically set to 1 and 0.0001 on data sets. In order to shorten the parameter tuning time, we set $d = 128$ in the adjustment experiment of parameter m, α, β and γ.

4.5.1 Sensitivity Study on m

Fig. 6(a) shows the sensitivity study result of m. As can be seen from the figure, when m is small, the performance of RTGE is poor. For example, on Wikidata12K dataset, when $m = 1$, the mean rank of RTGE on the head entity prediction task is 400. As m increases, the performance of RTGE continues to improve. When $m = 5$, the mean rank of RTGE on the head entity prediction task is lower than 200. This result indicates that the appropriate increase in the number of negative sampling triples will help to improve the discriminative ability of RTGE. Besides, it can be observed that with the continuous increase of m, the improvement of RTGE performance is decreasing. This result indicates that too large m cannot continue to significantly improve the performance of RTGE. Too large m will bring more training samples, which will significantly increase the time required for training. Given these observations, we set $m = 5$.

4.5.2 Sensitivity Study on α

Fig. 6(b) provides the sensitivity study result of α. It can be seen from the figure that the sensitivity of α on different data sets and different tasks varies greatly. For example, in the test of the Wikidata12K dataset, a smaller α is beneficial to improve the performance of RTGE in the entity prediction task. In contrast, in relation prediction, RTGE tends to choose a larger α. In view of these results, we choose different α's for RTGE on different datasets and different tasks. For example, on the Wikidata12K dataset, we choose $\alpha = 0.1$ for the entity prediction task, and $\alpha = 100$ for the relation prediction task. On the YAGO11K dataset, we choose $\alpha = 100$ for all tasks.

4.5.3 Sensitivity Study on β

Fig. 6(c) reports the potential impacts imposed by different numbers of trade-off parameter, β, to the mean rank of RTGE. From Fig. 6(c), we can observe that when the epoch is small, the mean rank of RTGE on the three tasks is more significant. As epoch increases, the mean rank of RTGE gradually decreased and stabilized in most test groups. Moreover, when β is less than or equal to 0.01, the mean rank of RTGE can get the smallest value at the maximum epoch. These results indicate that an oversized β increases the performance of RTGE on Wikidata12K, which is not

(a) Wikidata12K (RTGE).

(b) YAGO11K (RTGE).

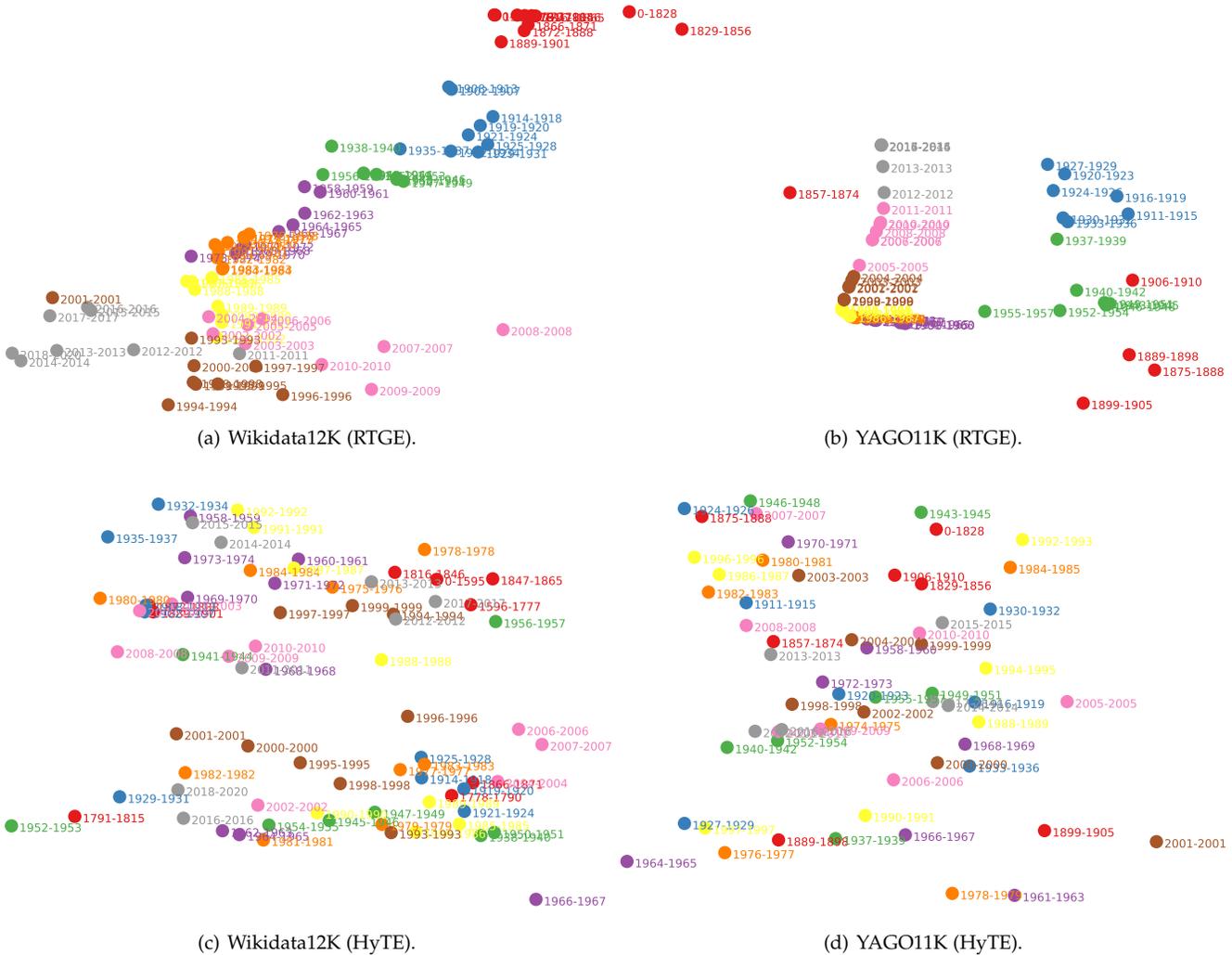(c) Wikidata12K (HyTE).

(d) YAGO11K (HyTE).

Fig. 5: The figure illustrates 2-d PCA projection of time embeddings which are obtained after training RTGE and HyTE for temporal scoping task. Dots of the same color indicate adjacent time embeddings.

conducive to prediction. Therefore, we set the number of the trade-off parameter β to 0.01 for Wikidata12K.

4.5.4 Sensitivity Study on γ

To analyze the effect of different values of γ on the prediction performance of RTGE, the sensitivity study experiment is conducted on entity and relation prediction tasks where γ is selected from 0.1 to 100. Fig. 6(d) shows the experimental results. We observe that, on the Wikidata12K dataset, the entity prediction task tends to choose $\gamma = 10$. And $\gamma = 10$ or $\gamma = 0.1$ is more suitable for the relation prediction task. Different the result on Wikidata12K, $\gamma = 100$ is more suitable for the head entity prediction task on YAGO11K. Furthermore, $\gamma = 0.1$ is the best choice for the tail entity and relation prediction tasks on YAGO11K.

4.5.5 Sensitivity Study on d

To find a suitable dimension of the embedding, we tested the performance of RTGE on a benchmark dataset using different dimensions of embeddings. The experimental results are shown in Fig. 7. From the figure, we can find an obvious rule. As the epoch increases, the mean rank of the RTGE embedded with different dimensions is decreasing.

Higher-dimensional embeddings can make RTGE's mean rank drop faster than lower-dimensional embeddings, and they can always help to reach the lowest mean rank. In particular, the performance of RTGE in multiple tasks of two data sets is optimal when $d = 256$. This is because higher-dimensional embeddings can describe more detailed graph structure information than lower-dimensional embeddings. Therefore, higher-dimensional embeddings can achieve the best result. However, as the dimensions continue to increase, the training and testing of the algorithm bring more time loss, which can be verified from the section 3.

5 CONCLUSION

We propose a robustly time-aware graph embedding method. In contrast with other state-of-the-art methods, the main characteristics of our approach are two folds. Firstly, RTGE proposes incorporating temporal smoothness into the learning framework of embedding. By taking into account the temporal smoothness between hyperplanes of adjacent time steps, RTGE can more efficiently obtain the evolution of KGs and avoid the interpretation of the wrong conclusion. Secondly, RTGE proposes a task-oriented sampling
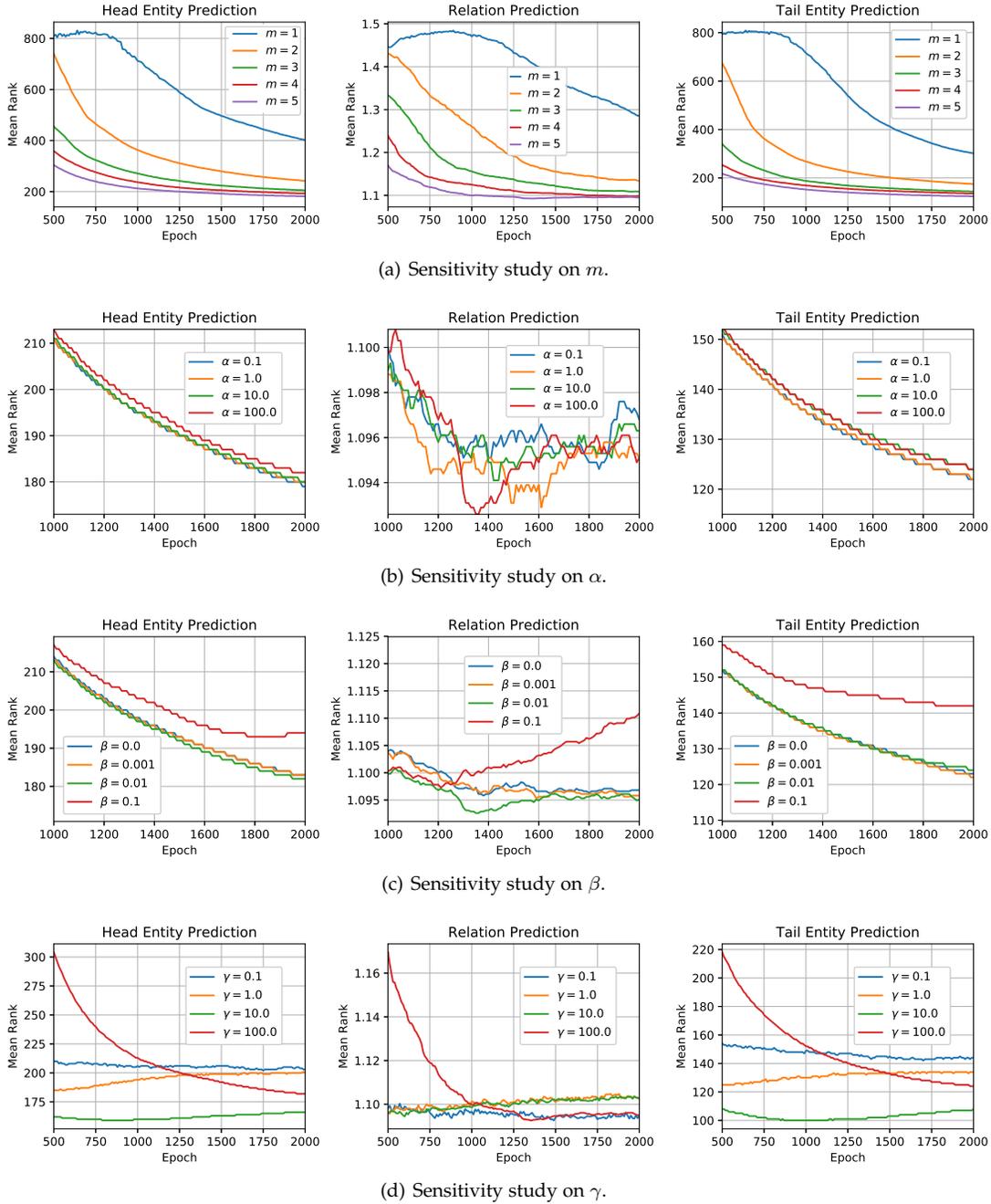
(a) Sensitivity study on m.

(b) Sensitivity study on α.

(c) Sensitivity study on β.

(d) Sensitivity study on γ.

Fig. 6: Sensitivity study of m, α, β and γ on Wikidata12K.

strategy for different tasks that can dynamically adjust the negative sampling ratio of entities and relationships for the characteristics of different tasks. The expanded negative sampling strategy provides RTGE with a stronger adaptive ability to separate tasks. Experiments on Wikidata12K data set and YAGO11K data set verify the superiority of RTGE over other state-of-the-art baseline methods. Although the algorithm proposed in this paper has achieved good results on the benchmark data set, RTGE still uses the TransH-like embedding principle to learn time-based hyperplanes at independent intervals. Compared to TransH, the embedding principles mentioned in TransF and ConvE have better performance. If the embedding principles like TransF and ConvE are introduced into RTGE, it is possible to improve the performance of the algorithm further. This is our future work.

REFERENCES

[1] F. M. Suchanek, G. Kasneci, and G. Weikum, "Yago: a core of semantic knowledge," in *Proceedings of the 16th international conference on World Wide Web*.   ACM, 2007, pp. 697–706.

[2] J. Leblay and M. W. Chekol, "Deriving validity time in knowledge graph," in *Companion Proceedings of the The Web Conference 2018*, ser. WWW '18.   Republic and Canton of Geneva, Switzerland: International World Wide Web Conferences Steering Committee, 2018, pp. 1771–1776. [Online]. Available: https://doi.org/10.1145/3184558.3191639
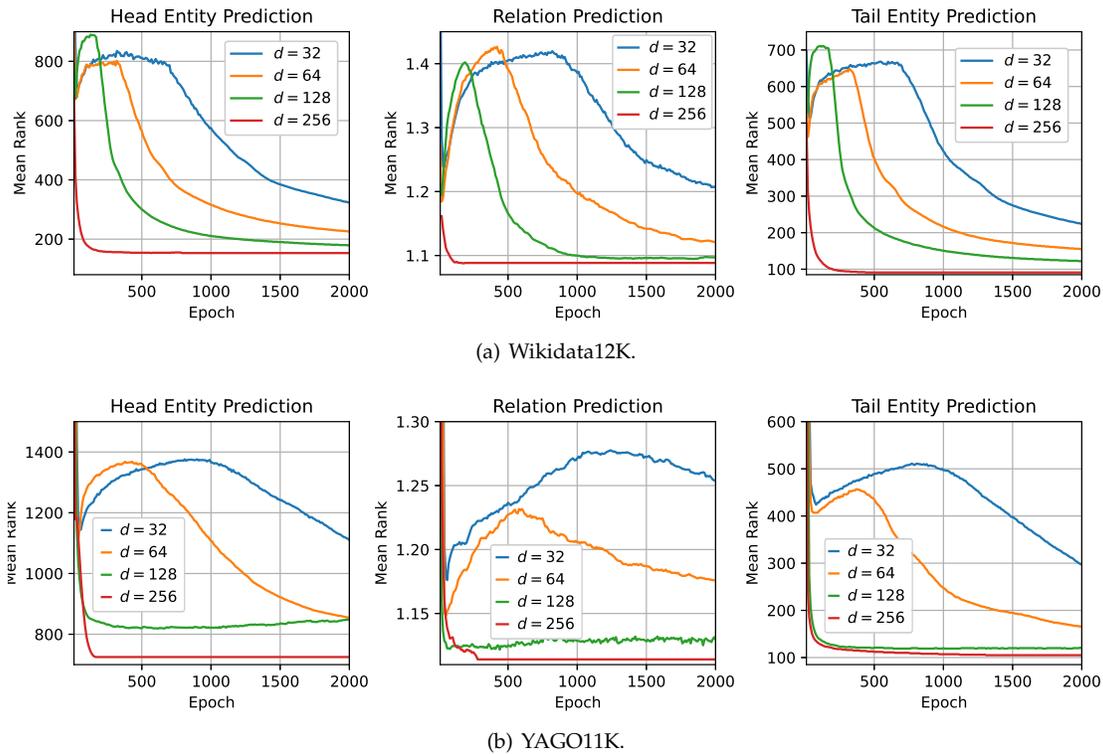
(a) Wikidata12K.

(b) YAGO11K.

Fig. 7: Sensitivity study of d.

[3] K. Bollacker, C. Evans, P. Paritosh, T. Sturge, and J. Taylor, "Freebase: a collaboratively created graph database for structuring human knowledge," in *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, 2008, pp. 1247–1250.

[4] J. Lehmann, R. Isele, M. Jakob, A. Jentzsch, D. Kontokostas, P. N. Mendes, S. Hellmann, M. Morsey, P. Van Kleef, S. Auer *et al.*, "Dbpedia–a large-scale, multilingual knowledge base extracted from wikipedia," *Semantic Web*, vol. 6, no. 2, pp. 167–195, 2015.

[5] A. Carlson, J. Betteridge, B. Kisiel, B. Settles, E. R. Hruschka Jr, and T. M. Mitchell, "Toward an architecture for never-ending language learning." in *AAAI*, vol. 5. Atlanta, 2010, p. 3.

[6] J. Berant, A. Chou, R. Frostig, and P. Liang, "Semantic parsing on freebase from question-answer pairs," in *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, 2013, pp. 1533–1544.

[7] D. Damljanovic and K. Bontcheva, "Named entity disambiguation using linked data," in *Proceedings of the 9th Extended Semantic Web Conference*, 2012, pp. 231–240.

[8] J. Daiber, M. Jakob, C. Hokamp, and P. N. Mendes, "Improving efficiency and accuracy in multilingual entity extraction," in *Proceedings of the 9th International Conference on Semantic Systems*. ACM, 2013, pp. 121–124.

[9] R. Yin, K. Li, G. Zhang, and J. Lu, "A deeper graph neural network for recommender systems," *Knowledge-Based Systems*, vol. 185, p. 105020, 2019.

[10] Y. Zhang, H. Dai, Z. Kozareva, A. J. Smola, and L. Song, "Variational reasoning for question answering with knowledge graph," in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[11] J. Weston, A. Bordes, O. Yakhnenko, and N. Usunier, "Connecting language and knowledge bases with embedding models for relation extraction," *arXiv preprint arXiv:1307.7973*, 2013.

[12] Q. Wang, Z. Mao, B. Wang, and L. Guo, "Knowledge graph embedding: A survey of approaches and applications," *IEEE Transactions on Knowledge and Data Engineering*, vol. 29, no. 12, pp. 2724–2743, 2017.

[13] X. Du, J. Yan, and H. Zha, "Joint link prediction and network alignment via cross-graph embedding," in *Proceedings of the 28th International Joint Conference on Artificial Intelligence*. AAAI Press, 2019, pp. 2251–2257.

[14] M. Nickel, V. Tresp, and H.-P. Kriegel, "A three-way model for collective learning on multi-relational data." in *ICML*, vol. 11, 2011, pp. 809–816.

[15] N. Guan, D. Song, and L. Liao, "Knowledge graph embedding with concepts," *Knowledge-Based Systems*, vol. 164, pp. 38–44, 2019.

[16] P. Cui, X. Wang, J. Pei, and W. Zhu, "A survey on network embedding," *IEEE Transactions on Knowledge and Data Engineering*, vol. 31, no. 5, pp. 833–852, 2018.

[17] P. Goyal and E. Ferrara, "Graph embedding techniques, applications, and performance: A survey," *Knowledge-Based Systems*, vol. 151, pp. 78–94, 2018.

[18] H. Cai, V. W. Zheng, and K. C.-C. Chang, "A comprehensive survey of graph embedding: Problems, techniques, and applications," *IEEE Transactions on Knowledge and Data Engineering*, vol. 30, no. 9, pp. 1616–1637, 2018.

[19] X. Huang, Q. Song, Y. Li, and X. Hu, "Graph recurrent networks with attributed random walks," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, ser.

KDD '19.   New York, NY, USA: ACM, 2019, pp. 732–740. [Online]. Available: http://doi.acm.org/10.1145/3292500.3330941

[20] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, and O. Yakhnenko, "Translating embeddings for modeling multi-relational data," in *Advances in neural information processing systems*, 2013, pp. 2787–2795.

[21] J. Yang and J. Leskovec, "Defining and evaluating network communities based on ground-truth," *Knowledge and Information Systems*, vol. 42, no. 1, pp. 181–213, 2015.

[22] Y. Li, J. Fan, Y. Wang, and K.-L. Tan, "Influence maximization on social graphs: A survey," *IEEE Transactions on Knowledge and Data Engineering*, vol. 30, no. 10, pp. 1852–1872, 2018.

[23] A. Lauscher, K. Eckert, L. Galke, A. Scherp, S. T. R. Rizvi, S. Ahmed, A. Dengel, P. Zumstein, and A. Klein, "Linked open citation database: enabling libraries to contribute to an open and interconnected citation graph," in *Proceedings of the 18th ACM/IEEE on Joint Conference on Digital Libraries*.   ACM, 2018, pp. 109–118.

[24] P. Gai and S. Kapadia, "Networks and systemic risk in the financial system," *Oxford Review of Economic Policy*, vol. 35, no. 4, pp. 586–613, 2019.

[25] P. Goyal, S. R. Chhetri, and A. Canedo, "dyngraph2vec: Capturing network dynamics using dynamic graph representation learning," *Knowledge-Based Systems*, vol. 187, p. 104816, 2020.

[26] S. S. Dasgupta, S. N. Ray, and P. Talukdar, "Hyte: Hyperplane-based temporally aware knowledge graph embedding," in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, 2018, pp. 2001–2011.

[27] R. Liao, Z. Zhao, R. Urtasun, and R. S. Zemel, "Lanczos-net: Multi-scale deep graph convolutional networks," *arXiv preprint arXiv:1901.01484*, 2019.

[28] Z. Wang, J. Zhang, J. Feng, and Z. Chen, "Knowledge graph embedding by translating on hyperplanes." in *AAAI*, vol. 14, 2014, pp. 1112–1119.

[29] Y. Lin, Z. Liu, M. Sun, Y. Liu, and X. Zhu, "Learning entity and relation embeddings for knowledge graph completion." in *AAAI*, vol. 15, 2015, pp. 2181–2187.

[30] G. Ji, S. He, L. Xu, K. Liu, and J. Zhao, "Knowledge graph embedding via dynamic mapping matrix," in *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, vol. 1, 2015, pp. 687–696.

[31] B. Yang, W. Yih, X. He, J. Gao, and L. Deng, "Embedding entities and relations for learning and inference in knowledge bases," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, Y. Bengio and Y. LeCun, Eds., 2015. [Online]. Available: http://arxiv.org/abs/1412.6575

[32] T. Trouillon, J. Welbl, S. Riedel, É. Gaussier, and G. Bouchard, "Complex embeddings for simple link prediction."   International Conference on Machine Learning (ICML), 2016.

[33] T. Dettmers, M. Pasquale, S. Pontus, and S. Riedel, "Convolutional 2d knowledge graph embeddings," in *Proceedings of the 32th AAAI Conference on Artificial Intelligence*, February 2018, pp. 1811–1818. [Online]. Available: https://arxiv.org/abs/1707.01476

[34] G. Ji, K. Liu, S. He, and J. Zhao, "Knowledge graph completion with adaptive sparse transfer matrix." in *AAAI*, 2016, pp. 985–991.

[35] J. Feng, M. Huang, M. Wang, M. Zhou, Y. Hao, and X. Zhu, "Knowledge graph embedding by flexible translation." in *KR*, 2016, pp. 557–560.

[36] Y. Tay, A. T. Luu, and S. C. Hui, "Non-parametric estimation of multiple embeddings for link prediction on dynamic knowledge graphs," in *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[37] R. Trivedi, H. Dai, Y. Wang, and L. Song, "Know-evolve: Deep temporal reasoning for dynamic knowledge graphs," in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*.   JMLR. org, 2017, pp. 3462–3471.

[38] D. Zhu, P. Cui, Z. Zhang, J. Pei, and W. Zhu, "High-order proximity preserved embedding for dynamic networks," *IEEE Transactions on Knowledge and Data Engineering*, vol. 30, no. 11, pp. 2134–2144, 2018.

[39] A. Pareja, G. Domeniconi, J. Chen, T. Ma, T. Suzumura, H. Kanezashi, T. Kaler, and C. E. Leisersen, "Evolvegcn: Evolving graph convolutional networks for dynamic graphs," *arXiv preprint arXiv:1902.10191*, 2019.

[40] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *arXiv preprint arXiv:1609.02907*, 2016.

[41] P. Liu, L. Zhang, and J. A. Gulla, "Real-time social recommendation based on graph embedding and temporal context," *International Journal of Human-Computer Studies*, vol. 121, pp. 58–72, 2019.

[42] U. Singer, I. Guy, and K. Radinsky, "Node embedding over temporal graphs," *arXiv preprint arXiv:1903.08889*, 2019.

[43] R. Trivedi, M. Farajtabar, P. Biswal, and H. Zha, "Dyrep: Learning representations over dynamic graphs," in *International Conference on Learning Representations*, 2019.

[44] T. Jiang, T. Liu, T. Ge, L. Sha, S. Li, B. Chang, and Z. Sui, "Encoding temporal information for time-aware link prediction," in *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, 2016, pp. 2350–2354.

[45] M. Nickel, L. Rosasco, and T. Poggio, "Holographic embeddings of knowledge graphs," in *Thirtieth Aaai conference on artificial intelligence*, 2016.

[46] T. Jiang, T. Liu, T. Ge, L. Sha, B. Chang, S. Li, and Z. Sui, "Towards time-aware knowledge graph completion," in *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, 2016, pp. 1715–1724.

[47] Y. Jia, Y. Wang, X. Jin, H. Lin, and X. Cheng, "Knowledge graph embedding: A locally and temporally adaptive translation-based approach," *ACM Transactions on the Web (TWEB)*, vol. 12, no. 2, pp. 1–33, 2017.

[48] F. Mahdisoltani, J. Biega, and F. M. Suchanek, "Yago3: A knowledge base from multilingual wikipedias," 2013.

[49] G. A. Miller, "Wordnet: a lexical database for english," *Communications of the ACM*, vol. 38, no. 11, pp. 39–41, 1995.