TIME-AWARE RELATIONAL GRAPH ATTENTION NETWORK FOR TEMPORAL KNOWLEDGE GRAPH EMBEDDINGS

Anonymous authors
Paper under double-blind review

ABSTRACT

Embedding-based representation learning approaches for knowledge graphs (KGs) have been mostly designed for static data. However, many KGs involve temporal data, which creates the need for new representation learning approaches that can characterize and reason over time. In this work, we propose a Time-aware Relational Graph ATtention Network (TR-GAT) for temporal knowledge graph (TKG) embeddings, in which the initial feature of each entity is represented by fusing its embedding and the embeddings of its connected relations and timestamps as well as its neighboring entities. Different from the existing temporal GNN models which discretize temporal graphs into multiple snapshots, we treat timestamps as properties of links between entities. To further incorporate relation and time information into the graph structures, we utilize a self-attention mechanism which specifies different weights to different nodes according to the corresponding link features, i.e., embeddings of the relevant relations and timestamps within one neighborhood. Experimental results show that our approach achieves state-of-the-art performances regarding TKG completion and entity alignment tasks on several well-established TKG datasets due to the effective and efficient integration of time information.

1 INTRODUCTION

Knowledge graphs (KGs) abstract knowledge from the real world into a complex network graph consisting of billions of triples. Each triple is denoted as (e_s, r, e_o), where e_s is the subject entity, e_o is the object entity, and r is the relation between the entities. Large-scale KGs like DBpedia Lehmann et al. (2015) have been widely used in many AI applications, e.g., question answering, search, and natural language processing. Oftentimes, a single KG is far from complete and cannot support these applications with sufficient facts. Therefore, two fundamental KG tasks have been proposed: (1) **KG completion**, a.k.a **link prediction**, which aims to predict the missing entities for incomplete facts in a single KG; (2) **entity alignment**, which aims to align equivalent entity pairs referring to the same real-word object across multiple KGs. To address the above challenges, KG embedding (KGE) approaches are leveraged to map entities and relations in KGs into a low-dimensional vector space and measure probabilities of triples and similarities between entities based on their embeddings.

Recently, studies of the temporal dynamics in knowledge graphs have intrigued increasing interests. Some temporal KGs (TKGs) including Wikidata Erxleben et al. (2014), YAGO Mahdisoltani et al. (2013), and event-based datasets like ICEWS Lautenschlager et al. (2015) and GDELT Leetaru & Schrodt (2013) store billions of temporal facts, e.g., the triple (*Biden*, *PresidentOf*, *USA*) is valid only from Jan. 2021. Such triples attached with time information are represented as quadruples, shaped like (e_s, r, e_o, τ), where τ denotes the timestamp. Traditional KGE approaches disregard time information, leading to ineffective link predictions for temporal queries, e.g., (*?*, *PresidentOf*, *USA*, *2015*). Thus, a growing number of studies Leblay & Chekol (2018); García-Durán et al. (2018); Xu et al. (2019; 2020a); Lacroix et al. (2020) in recent years pay attention to new temporal KGE (TKGE) approaches for TKG completion task.

KGE approaches Sun et al. (2017; 2018) are widely used for entity alignment between KGs. Especially, GNN-based Wang et al. (2018); Cao et al. (2019); Mao et al. (2020a) entity alignment

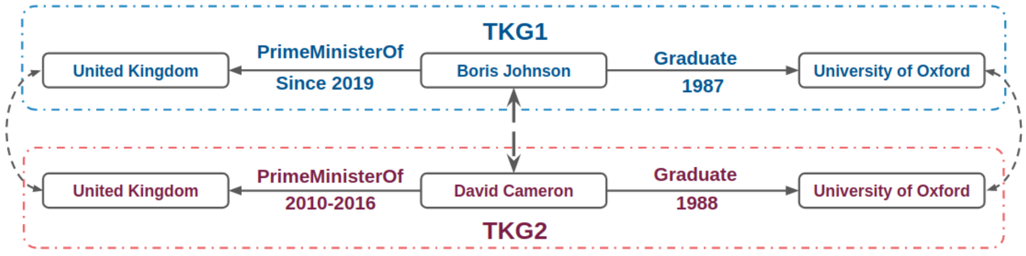


Figure 1: Illustration of the limitation of the existing time-agnostic entity align approaches.

approaches achieve great success. However, to the best of our knowledge, none of the existing embedding-based entity alignment approaches considers time information, leaving much room for improving their performance on TKGs. Taking the case in Figure 1 as an example, given two unaligned entities, Boris Johnson and David Cameron which exist in two TKGs respectively, time-agnostic approaches are likely to ignore time information and wrongly recognize these two entities as the same person in the real world due to the homogeneity of their neighborhood information.

To tackle the above issues, we propose a Time-aware Relational Graph ATtention network (TR-GAT) for TKG embedding (TKGE) where the initial feature of each entity is represented by fusing its embedding and the embeddings of its connected relations and timestamps as well as its neighboring entities. To further incorporate relation and time information into the graph structures, we utilize a self-attention mechanism which specifies different weights to different nodes in a neighborhood with the corresponding link features, i.e., embeddings of the relevant relations and timestamps.

We build and use multiple datasets extracted from ICEWS, GDELT, Wikidata and YAGO to evaluate our proposed approaches. The main contributions of this paper are listed as follows:

- We propose a novel GNN-based approach which can model temporal relational graphs with a time-aware self-attention mechanism, adapt well to datasets where timestamps are represented in the various forms: time points or time intervals, and perform different TKG learning tasks.
- Existing temporal GNN models typically discretize a temporal graphs into multiple static snapshots of higher sparsity and utilize a combination of GNNs and recurrent architectures which oftentimes suffers from long training time. Differently, our framework could take advantage of efficient training by treating timestamps as attentional properties of links between nodes. This idea could potentially be used for other temporal graph representation learning.
- Experiments show that our approaches achieve state-of-the-art results regarding TKG completion and remarkably outperform several strong entity alignment baseline models on several well-established TKG datasets. To the best of our knowledge, there is no previous work to integrate time information into entity alignment approaches.

2 Related Work

Graph Neural Network (GNN) has become increasingly popular in many areas, including social networks and KGs Schlichtkrull et al. (2018), due to its ability to model non-Euclidean space. Graph Convolutional Network (GCN) Kipf & Welling (2016) is an extension of GNN, which generates node-level embeddings by aggregating information from the nodes' neighborhoods. Furthermore, Graph Attention Network (GAT) Veličković et al. (2017) employs a self-attention mechanism to calculate the hidden representations of each entity by attending over its neighbors. With the success of these GNN models in the static setting, we approach further practical scenarios where the graph temporally evolves. Existing approaches Chen et al. (2018); Manessi et al. (2020); Pareja et al. (2020) generally discretize a temporal graph into multiple static snapshots in a timeline and utilize a combination of GNNs and recurrent architectures (e.g., LSTM), whereby the former digest graph information and the latter handle dynamism.

KG embedding (KGE) aims to embed entities and relations in a KG into a low-dimensional vector space for representation learning. A lot of KGE models have been proposed to perform KG completion task by using a single embedding-lookup layer as an encoder and defining a score function as a decoder Bordes et al. (2013); Yang et al. (2014); Trouillon et al. (2016); Nayyeri et al. (2020); Xu et al. (2020b). While such shallow models might suffer from insufficient expressiveness Dettmers et al. (2018), GNN-based approaches Schlichtkrull et al. (2018); Vashishth et al. (2020) have been proposed to leverage multi-hop information around entities by using GNN layers as encoders. With the development of TKGs, TKGE draws increasing attention. Some recent works extend static KGE models to the temporal domain. Such approaches employ embedding methods with a shallow encoder and design time-sensitive decoding function Leblay & Chekol (2018); García-Durán et al. (2018); Xu et al. (2019; 2020a); Lacroix et al. (2020); Sadeghian et al. (2021). Another line of TKGE work including RE-NET Jin et al. (2019) and TeMP Wu et al. (2020) uses GNNs to capture intra-graph neighborhood information, which is combined with temporal recurrence. RE-NET uses a combination of R-GCN Schlichtkrull et al. (2018) and multi-level RNN for the task of graph extrapolation (i.e., inferring the next timestep in a sequence). TeMP also combines R-GCN with a GRU model or a temporal transformer as an encoder to incorporate both structural information and historical information into entity representation.

Entity alignment (EA) is to find equivalent entities between multiple KGs. Recent studies Chen et al. (2016); Sun et al. (2017; 2018) have found that KG embedding can also improve the performance on the entity alignment task. Among them, many embedding-based EA approaches introduce GNNs into entity alignment task, which is originated with the ability to model global information of graphs Wang et al. (2018); Wu et al. (2019b); Cao et al. (2019); Mao et al. (2020a;b). To the best of our knowledge, there is not any existing work to study entity alignment between TKGs. And the current GNN-based TKGE approaches are not compatible with the EA setting.

3 Problem Formulation

Formally, a TKG is represented as $\mathcal{G} = (\mathcal{E}, \mathcal{R}, \mathcal{T}, \mathcal{Q})$ where $\mathcal{E}$, $\mathcal{R}$, $\mathcal{T}$ and $\mathcal{Q}$ are the sets of entities, relations, timestamps and quadruples, respectively. **TKG completion** is a fundamental task for TKGE García-Durán et al. (2018). Given a KG, it aims to predict the object entity e_o given $(e_s, r, ?, \tau)$ or predict the subject entity e_s given $(?, r, e_o, \tau)$ where $e_o, e_s \in \mathcal{E}$, $r \in \mathcal{R}$ and $\tau \in \mathcal{T}$.

Let $\mathcal{G}_1 = (\mathcal{E}_1, \mathcal{R}_1, \mathcal{T}_1, \mathcal{Q}_1)$ and $\mathcal{G}_2 = (\mathcal{E}_2, \mathcal{R}_2, \mathcal{T}_2, \mathcal{Q}_2)$ be two TKGs, $\mathcal{S} = \{(e_{i1}, e_{i2}) | e_{i1} \in \mathcal{E}_1, e_{i2} \in \mathcal{E}_2\}$ be a set of pre-aligned entity pairs between $\mathcal{G}_1$ and $\mathcal{G}_2$. Since timestamps in most TKGs are presented in similar formats with Arabic numerals and thus could be be easily aligned by manually uniforming their formats, a unified time set $\mathcal{T}^* = \mathcal{T}_1 \cup \mathcal{T}_2$ can be constructed and two TKGs can be renewed as $\mathcal{G}_1 = (\mathcal{E}_1, \mathcal{R}_1, \mathcal{T}^*, \mathcal{Q}_1)$ and $\mathcal{G}_2 = (\mathcal{E}_2, \mathcal{R}_2, \mathcal{T}^*, \mathcal{Q}_2)$ sharing the same set of timestamps. The task of **time-aware entity alignment** aims to find new aligned entity pairs between $\mathcal{G}_1$ and $\mathcal{G}_2$ based on the alignment seeds $\mathcal{S}$ and prior knowledge of time information $\mathcal{T}^*$.

4 Method

Our proposed approach can be separated two substructures: an encoder based on TR-GAT and task-oriented decoders. Specifically, the TR-GAT encoder mainly includes two parts: a time-aware entity representation and a time-aware self-attention mechanism. Figure 2 depicts the overall framework of our proposed approaches and the architecture of the time-aware self-attention mechanism. We elaborate on details of the proposed approach in this section.

4.1 Time-Aware Entity Representation

Time information τ in a temporal fact (e_s, r, e_o, τ) can be represented in various forms, e.g., time points, begin or end time and time intervals. A time interval is shaped like $[\tau_b, \tau_e]$ where τ_b and τ_e denote the actual begin time and end time of the fact, respectively. A time point can be represented as $[\tau_b, \tau_e]$ where $\tau_b = \tau_e$. Noteworthily, we represent a begin or end time as $[\tau_b, \tau_0]$ or $[\tau_0, \tau_e]$ where $\tau_0 \in \mathcal{T}^*$ is the first time step in the time set denoting the unknown time information. A fact without known time information can be denoted as $(e_s, r, e_o, [\tau_0, \tau_0])$ to deal with heterogeneous temporal knowledge bases where a significant amount of relations might be non-temporal.

In order to integrate relation direction, we create a reverse relation r^{-1} for each relation r and extend the relation set $\mathcal{R} = \{r_0, r_1, \cdots, r_{|\mathcal{R}|-1}\} \rightarrow \{r_0, r_0^{-1}, \cdots, r_{|\mathcal{R}|-1}, r_{|\mathcal{R}|-1}^{-1}\}$. And each fact

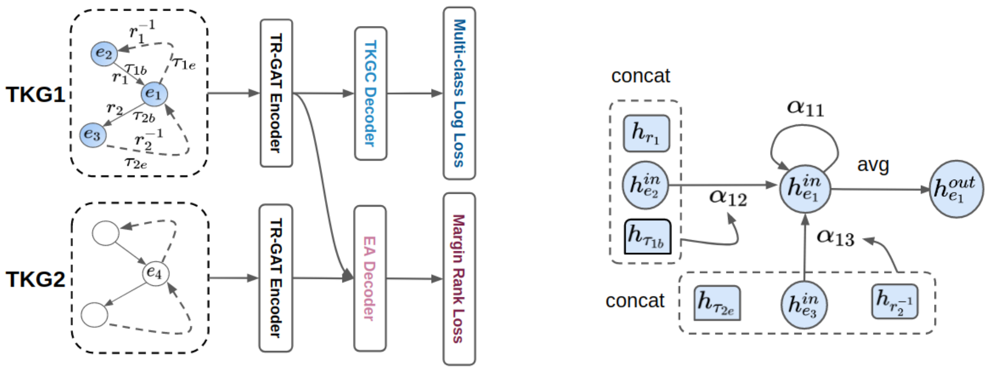


Figure 2: **Left:** Depiction of the framework of TR-GAT models for TKG completion and time-aware EA, where dashed arrows denote created reverse links. **Right:** An illustration of the time-aware self-attention mechanism by the node e_1, where curved arrows denote attention computations corresponding to its neighboring nodes.

$(e_s, r, e_o, [\tau_b, \tau_e])$ is decomposed into two quadruples (e_s, r, e_o, τ_b) and $(e_o, r^{-1}, e_s, \tau_e)$ to handle the begin and the end of the relation, respectively.

We map all of entities, relations (including reverse relations) and time steps in TKGs into a vector space $\mathbb{R}^d$ where d denotes the dimension of the vector space. Embeddings of the entity e_i, relation r_j, time step τ_m are denoted as $h_{e_i}, h_{r_j}, h_{\tau_m} \in \mathbb{R}^d$, respectively. To enforce both relation information and time information into the entity representation, we first average the embeddings of each entity and its neighboring entities and then concatenate the average entity embedding with the features of the inward links in the entity's neighborhood. In the case of Figure 2, the inward links in the neighborhood of the entity e_1 include $(e_2, r_1, e_1, \tau_{1b})$ and $(e_3, r_2^{-1}, e_1, \tau_{2e})$ in which e_1 performs as the object entity. As shown in Figure 2, the features of the inward links are represented as the average embeddings of the involved relations and time steps. The complete time-aware entity representation $h_{e_i}^{in}$ of an entity e_i can be formulated as,

$$h_{e_i}^{in} = \left[\frac{1}{|\mathcal{N}_i^e| + 1} \sum_{e_j \in \mathcal{N}_i^e \cup \{e_i\}} h_{e_j} \,||\, \frac{1}{|\mathcal{N}_i^r|} \sum_{r_j \in \mathcal{N}_i^r} h_{r_j} \,||\, \frac{1}{|\mathcal{N}_i^\tau|} \sum_{\tau_j \in \mathcal{N}_i^\tau} h_{\tau_j} \right], \tag{1}$$

where $\mathcal{N}_i^e$ is the set of neighboring entities of e_i, $\mathcal{N}_i^r$ and $\mathcal{N}_i^\tau$ are sets of relations and time steps which connect inwardly to e_i. $\overline{h_{e_i}}$, $\overline{h_{r_i}}$ and $\overline{h_{\tau_i}}$ denote the averages of the involved entity embeddings, relation embeddings and time embeddings, respectively. $||$ denotes the concatenation operator.

4.2 Time-Aware Self-Attention

Time-aware self-attention aims to integrate both time and relation information into the generated entity representation by assigning different weights to different neighboring nodes according to the time and relation features of inward links between nodes. We define the weighted importance $\beta_{i,j}$ of neighboring entity e_j to e_i as follows,

$$\beta_{i,j} = \omega^T \left[h_{e_i}^{in} || h_{e_j}^{in} || \sum_{r_m \in \mathcal{L}_{ij}^r} \frac{h_{r_m}}{|\mathcal{L}_{ij}^r|} || \sum_{\tau_m \in \mathcal{L}_{ij}^\tau} \frac{h_{\tau_m}}{|\mathcal{L}_{ij}^\tau|} \right], \tag{2}$$

where $\omega \in \mathbb{R}^{8d}$ is a shared attention weight vector, $\mathcal{L}_{ij}^r$ and $\mathcal{L}_{ij}^\tau$ denote the sets of relations and time steps in the links from e_j to e_i, respectively.

Following GAT Veličković et al. (2017), we define the normalized element $\alpha_{i,j}$ representing the connectivity from entity e_i to e_j with a LeakyReLU activation function (in which the negative input slope α = 0.2) as follows,

$$\alpha_{i,j} = \frac{\exp(\text{LeakyReLU}(\beta_{i,j}))}{\sum_{e_m \in \mathcal{N}_i^e \cup \{e_i\}} \exp(\text{LeakyReLU}(\beta_{i,m}))}, \tag{3}$$

We take the original time-aware entity representations as the input features of entities in the first hidden layer. And the output features $h_{e_i}^{out}$ are obtained with a linear combination of the input features of neighboring entities and a nonlinear ReLU activation function $\sigma(\cdot)$, i.e.,

$$h_{e_i}^{out} = \sigma\left(\frac{1}{M}\sum_{m=1}^{M}\left[\sum_{e_j \in \mathcal{N}_i^e \cup \{e_i\}} \alpha_{i,j}^m h_{e_j}^{in}\right]\right). \tag{4}$$

Same as GAT, we utilize the averaging multi-head attention to stabilize the learning process of self-attention. M denotes the number of attention heads and $\alpha_{i,j}^m$ are normalized attention coefficients computed by the m-th attention mechanism. It is noteworthy that a TR-GAT attention head has a lower time complexity compared to a single GAT attention head. Please see more details about the time complexities of GAT and TR-GAT in Appendix A.1.

4.3 TR-GAT ENCODER

A TR-GAT encoder comprises of an input layer which generates the time-aware entity features and one or several time-aware attentional layers.

Let the the l-th attentional layer's output features of entity e_i as $h_{e_i}^{out(l)}$. A cross-layer representation is employed to capture multi-hoop neighboring information in the previous work Mao et al. (2020a) by concatenating output features of different layers. Similarly, we define the final entity representation z_{e_i} of e_i as

$$z_{e_i} = [h_{e_i}^{out(0)} || h_{e_i}^{out(1)} || \cdots || h_{e_i}^{out(L)}], \tag{5}$$

where L is the number of attention layers and $h_{e_i}^{out(0)} = h_{e_i}^{in}$ are the input entity features.

To reduce the storage complexity to linear in the number of entities and links and enable the execution of TR-GAT models on larger TKG datasets, we are able to produce a version of the TR-GAT encoder that leverages sparse matrix operations.

4.4 TEMPORAL KNOWLEDGE GRAPH COMPLETION MODEL

To perform TKG completion, we first use an encoder based on TR-GAT layers to integrate the intra-graph neighborhood information into entity representations z_{e_i}.

Let $\phi(\cdot)$ denote the score for a quadruple (e_s, r, e_o, τ) and let TKGC denote any proper decoding function for TKG completion, e.g., TComplEx Lacroix et al. (2020). The score for the quadruple is defined as follows:

$$\phi(e_s, r, e_o, \tau) = \text{TKGC}(z_{e_s}, z_r, z_{e_o}, z_\tau), \tag{6}$$

where z_{e_s} and z_{e_o} are entity representations of the subject and object, z_r and z_τ are learned embeddings of the relation r and the timestamp τ. In this work, we use the score function defined for TComplEx Lacroix et al. (2020), which is one of the most recent TKGE models. Note that we first map z_{e_s}, z_{e_s}, z_{e_o}, z_τ from the real vector space $\mathbb{R}^d$ to the complex vector space $\mathbb{C}^k$ by dimidiating the embedding dimension, i.e., $k = d/2$.

To train our TKG completion model using this score function, the model parameters are learned using gradient-based optimization in mini-batches. For each mini-batch $\mathcal{Q}_b \subseteq \mathcal{Q}$, we follow the setting used for TComplEx to adopt N3 regularization for embeddings and temporal smoothness and define the full multiclass log-softmax loss function as follows,

$$\begin{aligned}\mathcal{L} =& \frac{1}{b}\sum_{(e_s,r,e_o,\tau)\in\mathcal{Q}_b}\Big[-\log\Big(\frac{\exp(\phi(e_s,r,e_o,\tau))}{\sum_{e'_o\in\mathcal{E}}\exp(\phi(e_s,r,e'_o,\tau))}\Big) - \log\Big(\frac{\exp(\phi(e_s,r,e_o,\tau))}{\sum_{e'_s\in\mathcal{E}}\exp(\phi(e'_s,r,e_o,\tau))}\Big) \\ &+ \lambda_b(||z_{e_s}||_3^3 + ||z_r||_3^3 + ||z_{e_o}||_3^3 + ||z_\tau||_3^3)\Big] + \lambda_\tau\sum_{i=1}^{||\mathcal{T}||-1}||z_{\tau_{i+1}} - z_{\tau_i}||_3^3,\end{aligned} \tag{7}$$

where b denotes the batch size, λ_b denotes the N3 regularization weight, and λ_τ denotes the coefficient of the temporal smoothness regularizer which is used to promote that the neighboring timestamps have close representations.

4.5 Time-aware Entity Alignment Model

Time-aware EA model embeds two TKGs into a unified vector space by pushing the seed alignments of entities together. In this work, the time-aware EA model consists of a TR-GAT encoder and a translational decoding function which measures similarities between entity representations.

Entity alignments are predicted based on the distances between the final output features of entities from two KGs. For two entities $e_i \in \mathcal{E}_1$ and $e_j \in \mathcal{E}_2$ from different sources, we use L1 distance to measure the distance between them as follows,

$$d(e_i, e_j) = ||z_{e_i} - z_{e_j}||_1, \tag{8}$$

A margin rank loss is used as the optimization objective of the entity align model, i.e.,

$$\mathcal{L} = \sum_{(e_i,e_j)\in\mathcal{S}} \sum_{(e_i,e'_j),(e'_i,e_j)\in\mathcal{S}'} \left[\sigma(d(e_i,e_j)+\lambda-d(e_i,e'_j))+\sigma(d(e_i,e_j)+\lambda-d(e'_i,e_j))\right], \tag{9}$$

where λ denotes the margin, $\mathcal{S}'$ is the set of generated negative entity pairs, $e'_i \in \mathcal{E}_1$ and $e'_j \in \mathcal{E}_2$ are the negative entities of e_i and e_j, respectively. Negative entities are sampled randomly and an Ada optimizer is used to minimize the loss function. During testing, we adopt CSLS Conneau et al. (2017) as the distance metric to measure similarities between entity embeddings.

We compare the total numbers of trainable parameters in our time-aware EA approach with several existing EA models. As shown in Table 1, compared to parameter-efficient translational entity align models like MTransE, TR-GAT uses additional parameters only for reverse relation embeddings, time embeddings and attention weight vectors.

EA Methods	Intra-graph	Relation	Time	Number of Parameter
MTransE Chen et al. (2016)		✓		$k(\lvert\mathcal{E}_1\rvert + \lvert\mathcal{E}_2\rvert + \lvert\mathcal{R}_1\rvert + \lvert\mathcal{R}_2\rvert)$
JAPE Sun et al. (2017)		✓		$k(\lvert\mathcal{E}_1\rvert + \lvert\mathcal{E}_2\rvert + \lvert\mathcal{R}_1\rvert + \lvert\mathcal{R}_2\rvert)$
BootEA Sun et al. (2018)		✓		$k(\lvert\mathcal{E}_1\rvert + \lvert\mathcal{E}_2\rvert + \lvert\mathcal{R}_1\rvert + \lvert\mathcal{R}_2\rvert)$
GCN-Align Wang et al. (2018)	✓			$k(\lvert\mathcal{E}_1\rvert + \lvert\mathcal{E}_2\rvert) + 2k^2$
MRAEA Mao et al. (2020a)	✓	✓		$k(\lvert\mathcal{E}_1\rvert + \lvert\mathcal{E}_2\rvert + 2\lvert\mathcal{R}_1\rvert + 2\lvert\mathcal{R}_2\rvert) + 3kML$
RREA Mao et al. (2020b)	✓	✓		$k(\lvert\mathcal{E}_1\rvert + \lvert\mathcal{E}_2\rvert + 2\lvert\mathcal{R}_1\rvert + 2\lvert\mathcal{R}_2\rvert) + 3kL$
TR-GAT	✓	✓	✓	$k(\lvert\mathcal{E}_1\rvert + \lvert\mathcal{E}_2\rvert + 2\lvert\mathcal{R}_1\rvert + 2\lvert\mathcal{R}_2\rvert + \lvert\mathcal{T}^*\rvert) + 4kML$

Table 1: Comparison of our EA method with other EA methods. Note that $k = d$ when embeddings are real-valued. Overall, TR-GAT is most comprehensive.

5 Experiments and Results

5.1 Datasets

TKG completion datasets We evaluate our model on three popular benchmarks for TKG completion, namely **ICEWS14**, **ICEWS05-15**, and **Yago15K** The first two datasets are subsets of Integrated Crisis Early Warning System (ICEWS), which is a very popular knowledge graph used by the community. ICEWS14 is collected from 01/01/2014 to 12/31/2014, while ICEWS15-05 is the subset occurring between 01/01/2005 and 12/31/2015. It is worth mentioning that each fact in ICEWS datasets involves a time point as its timestamp. The YAGO15K dataset García-Durán et al. (2018) is a modification of FB15K Bordes et al. (2013) with the additional timestamps from YAGO Mahdisoltani et al. (2013). Different from ICEWS datasets, YAGO15K is a hybrid dataset where a part of facts are non-temporal and timestamps in YAGO15K are represented as start time or end time, like *"occurSince 2000"* or *"occurUntil 2000"*. See more details of TKGC datasets in Appendix A.2.

Time-aware EA datasets Existing EA benchmarks are mostly extracted from FreeBase Bollacker et al. (2008), DBpedia Lehmann et al. (2015), Wikidata Erxleben et al. (2014) and YAGO. As we mentioned, the latter two knowledge bases contains millions of temporal facts. However, the existing EA benchmarks do not include time information attached to temporal facts in Wikidata and YAGO. Thus, we create five time-aware EA datasets in this work for evaluation. We build two datasets **DICEWS-1K** and **DICEWS-200** from ICEWS05-15 in the similar way to the construction of DFB datasets Zhu et al. (2017). The only difference between DICEWS-1K and DDICEWS-20000 is the proportion of alignment seed $\mathcal{S}$. We extract three datasets **YAGO-WIKI50K-5K**,

YAGO-WIKI50K-1K and **YAGO-WIKI20K** from Wikidata and YAGO. The first two datasets contain about 50,000 entity pairs, among which 5,000 and 1,000 entity pairs are taken as pre-known alignment seeds, and each fact in both datasets is temporal. Meanwhile, YAGO-WIKI20K is a hybrid dataset with 20,000 entity pairs. It is noteworthy that timestamps in YAGO-WIKI datasets are represented in various forms, e.g., time points, start or end time, time intervals. Statics of time-aware EA datasets are listed in Table 2. Please see Appendix A.3 for more details of the above datasets.

Dataset	$\|\mathcal{E}_1\|$	$\|\mathcal{E}_2\|$	$\|\mathcal{R}_1\|$	$\|\mathcal{R}_2\|$	$\|\mathcal{T}^*\|$	$\|\mathcal{Q}_1\|$	$\|\mathcal{Q}_2\|$	$\|\mathcal{P}\|$	$\|\mathcal{S}\|$
DICEWS-1K/200	9,517	9,537	247	246	4,017	307,552	307,553	8,566	1,000/200
YAGO-WIKI50K-5K/1K	49,629	49,222	11	30	245	221,050	317,814	49,172	5,000/1,000
YAGO-WIKI20K	19,493	19,929	32	130	405	83,583	142,568	19,462	400

Table 2: Statistics of time-aware EA datasets. $|\mathcal{P}|$ denotes the total number of reference entity pairs.

5.2 Experimental Setup

We perform TKG completion and time-aware EA as ranking tasks based on scores of factual quadruples and similarities between entity embeddings, and use Mean Reciprocal Rank (MRR) and Hits@N (N=1, 3, 10) as evaluation metrics. Following the previous work García-Durán et al. (2018); Xu et al. (2019; 2020a); Lacroix et al. (2020); Wu et al. (2020), a time-aware filtering setting is used for TKG completion to avoid possibly flawed evaluation.

For TKG completion, we compare our model against several state-of-the-art TKGE methods, including TTransE Leblay & Chekol (2018), HyTE Dasgupta et al. (2018), TA-TransE, TA-DistMult García-Durán et al. (2018), DE-SimplE Goel et al. (2020), TComplEx($\times$10), TNTComplEx Lacroix et al. (2020)($\times$10), TeMP Wu et al. (2020) and ChronoR Sadeghian et al. (2021). We mainly reuse the results reported in literature.

For entity alignment between TKGs, we compare our model with three strong translational baseline models and four state-of-the-art GNN-based models including MTransE Chen et al. (2016), JAPE Sun et al. (2017), AlignE Sun et al. (2018), GCN-Align Wang et al. (2018), MuGNN Cao et al. (2019), MRAEA Mao et al. (2020a) and RREA Mao et al. (2020b). We choose AlignE instead of BootEA since we do not use iterative learning for other models including our proposed models. Due to the deficiency of attribute information, we do not select attribute-aware EA models, e.g., AttrE Trisedya et al. (2019) or AttrGCN Liu et al. (2020), and use the SE (Structural Embedding) variants of JAPE and GCN-Align as baseline models. Since we do not use entity names as enhancement information for our proposed models, EA models using textual information like HGCN and RDGCN Wu et al. (2019a;b) are also excluded from the baseline. Except that the experiments of MTransE is implemented based on OpenEA framework Sun et al. (2020), all experiments of baseline models are implemented based on their resource codes. To verify the effect of the incorporation of time information on the EA performance of TR-GAT, we additionally implement a time-unaware variant of TR-GAT as an EA baseline model, which takes all time steps $\tau_i \in \mathcal{T}^*$ as unknown time information τ_0 and is denoted as TU-GAT.

Implementation details of our models and baseline models can be found in Appendix A.4. Datasets and source codes are submitted as supplementary materials for reproducibility.

5.3 TKG Completion Results

Main Results The TKG completion results on three datasets are reported in Table 3. Compared to a shallow baseline model, TComplEx which has the same decoding function as our TKG completion model, TR-GAT achieves significant improvements on ICEWS-14 and ICEWS05-15 regarding all metrics. On YAGO15K, TR-GAT also outperforms TComplEx across most metrics except Hits@10. Overall, TR-GAT achieves the state-of-the-art results on ICEWS14 and YAGO15K regarding MRR, Hits@1, Hits@3. On ICEWS05-15, TR-GAT achieves the best Hits@1 and also shows a competitive performance regarding other metrics. Two of the most recent TKGE models, TeMP-GRU and TeMP-SA which use combinations of R-GCN and temporal recurrent models based on GRU and temporal

	ICEWS14				ICEWS05-15				YAGO15K			
	MRR	Hits@1	Hits@3	Hits@10	MRR	Hits@1	Hits@3	Hits@10	MRR	Hits@1	Hits@3	Hits@10
TTransE	.255	.047	-	.601	.271	.084	-	.616	.321	.230	-	.510
HyTE	.297	.108	.416	.655	.316	.116	.445	.681	-	-	-	-
TA-TransE	.275	.095	-	.625	.299	.096	-	.668	.321	.231	-	.512
TA-DistMult	.477	.363	-	.686	.474	.346	-	.728	.291	.216	-	.476
DE-SimplE	.526	.418	.592	.725	.513	.392	.578	.748	-	-	-	-
TComplEx($\times$10)	.610	.530	.660	.770	.660	.590	.710	.800	.360	.280	.380	**.540**
TNTComplEx($\times$10)	.620	.520	.660	.760	.670	.590	.710	.810	**.370**	.290	.390	**.540**
TeMP-GRU	.601	.478	.681	.828	**.691**	.566	.782	**.917**	-	-	-	-
TeMP-SA	.607	.484	**.684**	.840	.680	.553	**.769**	.913	-	-	-	-
ChronoR	.625	.547	.669	.773	.675	**.596**	.723	.820	.366	**.292**	.379	.538
TR-GAT	**.637**	**.556**	**.684**	.790	.673	**.596**	.720	.816	**.370**	**.292**	**.391**	.525

Table 3: TKG completion results on ICEWS14, ICEWS05-15 and YAGO15K. Dashes: results are not reported in the respective literature. The best results among all models are written bold

transformer, have better performance on ICEWS05-15 than TR-GAT regarding MRR, Hits@3 and Hits@10 but also suffer from longer training time. Although TeMP models use lower-dimensioanl embeddings than TComplEx, TNTComplEx, ChronoR and ours, the training processes of TeMP-GRU and TeMP-SA on ICEWS05-15 take about 49 and 52 minutes per epoch using a single GTX Titan X GPU, while it averagely takes about 17 minutes for TR-GAT to complete a training epoch on ICEWS05-15 with the same device.

5.4 Time-aware Entity Alignment Results

Models	**DICEWS-1K**			**DICEWS-200**			**YAGO-WIKI50K-5K**			**YAGO-WIKI50K-1K**		
	MRR	Hits@1	Hits@10	MRR	Hits@1	Hits@10	MRR	Hits@1	Hits@10	MRR	Hits@1	Hits@10
MTransE	.150	.101	.241	.104	.067	.175	.322	.242	.477	.033	.012	.067
JAPE	.198	.144	.298	.138	.098	.210	.345	.271	.488	.157	.101	.262
AlignE	.593	.508	.751	.303	.222	.457	.800	.756	.883	.618	.565	.714
GCN-Align	.291	.204	.466	.231	.165	.363	.581	.512	.711	.279	.217	.398
MuGNN	.617	.525	.794	.412	.367	.583	.808	.762	.890	.632	.589	.733
MRAEA	.745	.675	.870	.564	.476	.733	.848	.806	.913	.685	.623	.801
RREA	.780	.722	.883	.719	.659	.824	.868	.828	.938	.753	.696	.859
TU-GAT	.748	.681	.870	.576	.489	.739	.815	.767	.902	.672	.607	.795
TR-GAT	**.900**	**.876**	**.942**	**.849**	**.815**	**.909**	**.903**	**.871**	**.959**	**.799**	**.748**	**.891**
Improv.	20.3%	28.6%	7.9%	47.4%	66.7%	23.0%	9.7%	14.9%	6.3%	18.9%	23.2%	12.1%

Table 4: Entity alignment results on ICEWS and YAGO-WIKI50K datasets. Improv. indicates the improvement achieved by TR-GAT against its time-unaware variant TU-GAT. The best results among all models are written bold.

Main Results Table 4 shows the entity alignment results of our proposed models and all baselines on ICEWS and YAGO-WIKI50K datasets. It can be shown that TR-GAT remarkably outperforms all baseline models on four TKG datasets across all metrics. Compared to RREA which achieves the best results among than all baseline models, TR-GAT obtains the improvement of 23.7%, 21.3%, 7.5% and 5.2% regarding Hits@1 on four TKG datasets, respectively.

Robustness To study the effect of the incorporation of time information, we test the time-unaware variant of TR-GAT, TU-GAT. As shown in Table 4, TU-GAT outperforms most baseline models other than RREA since it uses a static relational attention mechanism to capture relation information between entities, including relation types and relation directions. It can be observed that the improvements of TR-GAT against TU-GAT are more significant on datasets with less seeds. Specifically, TR-GAT improves Hits@1 by 66.7% and 23.2% regarding Hits@1 on DICEWS-200 and YAGO-WIKI50K-1K while the improvements on DICEWS-1K and YAGO-WIKI50K-5K are 28.6% and 14.9%. Similar results can be observed between TR-GAT and RREA.

In practice, it is quite important for an entity alignment model to maintain a good performance with few pre-aligned entities since such prior knowledge is difficult to be obtained. Meanwhile, time information from different digital resources can be easily aligned because time information is mostly recorded with Arabic numerals and represented in similar formats which can be uniformed. Thus, we conduct a study of the robustness of our models against the numbers of alignment seeds in Appendix A.5. By incorporating time information, TR-GAT shows the better robustness and the lower sensitivity on the number of pre-aligned entities, compared to time-unaware models.

Sensitivity We also conduct a study on the prediction accuracy of aligned entities which have different time sensitivity. As mentioned in Section 5.1, we generate a hybrid dataset YAGO-WIKI20K where 17.5% of YAGO facts and 36.6% of Wikidata facts are non-temporal. We divide all testing entity pairs in this dataset into two categories based on their sensitivity to time information, i.e., **highly time-sensitive** entity pairs and **lowly time-sensitive** entity pairs. Time sensitivity s_i of a single entity e_i is defined as the ratio of the number of its time-aware connected links in which $\tau \neq \tau_0$ over the total number of all links $\mathcal{L}_i$ within its neighborhood, i.e.,

$$s_i = 1 - |\mathcal{L}_i^{\tau_0}|/|\mathcal{L}_i|, \tag{10}$$

where $\mathcal{L}_i^{\tau_0}$ denotes the set of time-unaware links connecting e_i. Given an entity pair (e_{i1}, e_{i2}) between $\mathcal{G}_1$ and $\mathcal{G}_2$, we call them as a higly time-sensitive entity pair if $s_{i1} \geqslant 0.5$ and $s_{i2} \geqslant 0.5$. Otherwise, they are lowly time-sensitive.

Among 19,062 testing entity pairs of YAGO-WIKI20K, 6,898 of them are highly time-sensitive and others are lowly time-sensitive according to the above definitions. The entity alignment results of TR-GAT and TU-GAT on the highly time-sensitive test set and the lowly time-sensitive test set are reported in Table 4. It can be shown that TR-GAT and TU-GAT have close performance on entity alignment for lowly time-sensitive entity pairs while TR-GAT remarkably outperforms TU-GAT on the highly time-sensitive test set. In other words, the effect of incorporation of time information are more significant when testing entity pairs are more time-sensitive.

	Highly Time-Sensitive			**Lowly Time-Sensitive**			**In Total**		
	MRR	Hits@1	Hits@10	MRR	Hits@1	Hits@10	MRR	Hits@1	Hits@10
TR-GAT	.805	.797	.892	.331	.284	.419	.503	.470	.590
TU-GAT	.700	.639	.818	.314	.264	.411	.454	.400	.558

Table 5: Entity alignment results on different test sets of YAGO-WIKI20K.

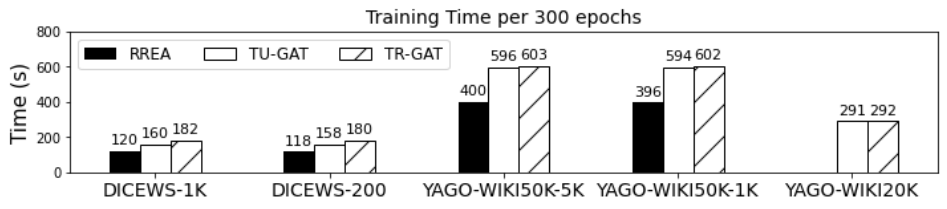


Figure 3: Training time per 300 epochs on different datasets.

Training Time As shown in Figure 3, the processing of the additional time information does not excessively increase the training time for TR-GAT, compared to RREA and TU-GAT. Since we set the maximum number of epochs as 6000, the training processes of our proposed models on different datasets can be completed within a couple of hours on a single GeForce GTX Titan X GPU.

6 Conclusion

In this paper, we studied the TKGE learning and proposed TR-GAT to remedy the problems of using attention mechanisms and graph neural networks to learn time-aware relational graphs in a time-efficient way. We presented an end-to-end framework, which uses a TR-GAT encoder to learn entity representations with the integration of time and relation information and different task-oriented decoding functions to perform time-aware entity alignment and TKG completion. Our experiments showed that the proposed method obtained superior performance for time-aware entity alignment and competitive results for TKG completion, and adapted well to TKG datasets where timestamps are represented inthe various forms: time points, start time or end time, and time intervals. To the best of our knowledge, there is no previous literature to perform entity alignment between KGs using a time-aware embedding-based approach.

REFERENCES

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pp. 1247–1250, 2008.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. Translating embeddings for modeling multi-relational data. In *Advances in Neural Information Processing Systems*, pp. 2787–2795, 2013.

Yixin Cao, Zhiyuan Liu, Chengjiang Li, Juanzi Li, and Tat-Seng Chua. Multi-channel graph neural network for entity alignment. *arXiv preprint arXiv:1908.09898*, 2019.

Jinyin Chen, Xuanheng Xu, Yangyang Wu, and Haibin Zheng. Gc-lstm: Graph convolution embedded lstm for dynamic link prediction. *arXiv preprint arXiv:1812.04206*, 2018.

Muhao Chen, Yingtao Tian, Mohan Yang, and Carlo Zaniolo. Multilingual knowledge graph embeddings for cross-lingual knowledge alignment. *arXiv preprint arXiv:1611.03954*, 2016.

Alexis Conneau, Guillaume Lample, Marc'Aurelio Ranzato, Ludovic Denoyer, and Hervé Jégou. Word translation without parallel data. *arXiv preprint arXiv:1710.04087*, 2017.

Shib Sankar Dasgupta, Swayambhu Nath Ray, and Partha Talukdar. Hyte: Hyperplane-based temporally aware knowledge graph embedding. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 2001–2011, 2018.

Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. Convolutional 2d knowledge graph embeddings. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

Fredo Erxleben, Michael Günther, Markus Krötzsch, Julian Mendez, and Denny Vrandečić. Introducing wikidata to the linked data web. In *International Semantic Web Conference*, pp. 50–65. Springer, 2014.

Alberto García-Durán, Sebastijan Dumančić, and Mathias Niepert. Learning sequence encoders for temporal knowledge graph completion. *arXiv preprint arXiv:1809.03202*, 2018.

Rishab Goel, Seyed Mehran Kazemi, Marcus Brubaker, and Pascal Poupart. Diachronic embedding for temporal knowledge graph completion. In *AAAI*, 2020.

Woojeong Jin, Changlin Zhang, Pedro Szekely, and Xiang Ren. Recurrent event network for reasoning over temporal knowledge graphs. *arXiv preprint arXiv:1904.05530*, 2019.

Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

Timothée Lacroix, Guillaume Obozinski, and Nicolas Usunier. Tensor decompositions for temporal knowledge base completion. *arXiv preprint arXiv:2004.04926*, 2020.

Jennifer Lautenschlager, Steve Shellman, and Michael Ward. Icews event aggregations, 2015. URL `https://doi.org/10.7910/DVN/28117`.

Julien Leblay and Melisachew Wudage Chekol. Deriving validity time in knowledge graph. In *Companion of the The Web Conference 2018 on The Web Conference 2018*, pp. 1771–1776. International World Wide Web Conferences Steering Committee, 2018.

Kalev Leetaru and Philip A Schrodt. Gdelt: Global data on events, location, and tone, 1979–2012. In *ISA annual convention*, volume 2, pp. 1–49. Citeseer, 2013.

Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo N Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick Van Kleef, Sören Auer, et al. Dbpedia–a large-scale, multilingual knowledge base extracted from wikipedia. *Semantic web*, 6(2):167–195, 2015.

Zhiyuan Liu, Yixin Cao, Liangming Pan, Juanzi Li, Zhiyuan Liu, and Tat-Seng Chua. Exploring and evaluating attributes, values, and structures for entity alignment. In *EMNLP*, 2020.

Farzaneh Mahdisoltani, Joanna Biega, and Fabian M Suchanek. Yago3: A knowledge base from multilingual wikipedias. In *CIDR*, 2013.

Franco Manessi, Alessandro Rozza, and Mario Manzo. Dynamic graph convolutional networks. *Pattern Recognition*, 97:107000, 2020.

Xin Mao, Wenting Wang, Huimin Xu, Man Lan, and Yuanbin Wu. Mraea: an efficient and robust entity alignment approach for cross-lingual knowledge graph. In *Proceedings of the 13th International Conference on Web Search and Data Mining*, pp. 420–428, 2020a.

Xin Mao, Wenting Wang, Huimin Xu, Yuanbin Wu, and Man Lan. Relational reflection entity alignment. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, pp. 1095–1104, 2020b.

Mojtaba Nayyeri, Chengjin Xu, Sahar Vahdati, Nadezhda Vassilyeva, Emanuel Sallinger, Hamed Shariat Yazdi, and Jens Lehmann. Fantastic knowledge graph embeddings and how to find the right space for them. In *International Semantic Web Conference*, pp. 438–455. Springer, 2020.

Aldo Pareja, Giacomo Domeniconi, Jie Chen, Tengfei Ma, Toyotaro Suzumura, Hiroki Kanezashi, Tim Kaler, Tao Schardl, and Charles Leiserson. Evolvegcn: Evolving graph convolutional networks for dynamic graphs. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 5363–5370, 2020.

Ali Sadeghian, Mohammadreza Armandpour, Anthony Colas, and Daisy Zhe Wang. Chronor: Rotation based temporal knowledge graph embedding. *arXiv preprint arXiv:2103.10379*, 2021.

Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne Van Den Berg, Ivan Titov, and Max Welling. Modeling relational data with graph convolutional networks. In *European semantic web conference*, pp. 593–607. Springer, 2018.

Zequn Sun, Wei Hu, and Chengkai Li. Cross-lingual entity alignment via joint attribute-preserving embedding. In *International Semantic Web Conference*, pp. 628–644. Springer, 2017.

Zequn Sun, Wei Hu, Qingheng Zhang, and Yuzhong Qu. Bootstrapping entity alignment with knowledge graph embedding. In *IJCAI*, volume 18, pp. 4396–4402, 2018.

Zequn Sun, Qingheng Zhang, Wei Hu, Chengming Wang, Muhao Chen, Farahnaz Akrami, and Chengkai Li. A benchmarking study of embedding-based entity alignment for knowledge graphs. *Proceedings of the VLDB Endowment*, 13(11):2326–2340, 2020. URL `http://www.vldb.org/pvldb/vol13/p2326-sun.pdf`.

Bayu Distiawan Trisedya, Jianzhong Qi, and Rui Zhang. Entity alignment between knowledge graphs using attribute embeddings. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pp. 297–304, 2019.

Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. Complex embeddings for simple link prediction. *arXiv preprint arXiv:1606.06357*, 2016.

Shikhar Vashishth, Soumya Sanyal, Vikram Nitin, and Partha Talukdar. Composition-based multi-relational graph convolutional networks. In *International Conference on Learning Representations*, 2020. URL `https://openreview.net/forum?id=BylA_C4tPr`.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

Zhichun Wang, Qingsong Lv, Xiaohan Lan, and Yu Zhang. Cross-lingual knowledge graph alignment via graph convolutional networks. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 349–357, 2018.

Jiapeng Wu, Meng Cao, Jackie Chi Kit Cheung, and William L Hamilton. Temp: Temporal message passing for temporal knowledge graph completion. *arXiv preprint arXiv:2010.03526*, 2020.

Yuting Wu, Xiao Liu, Yansong Feng, Zheng Wang, and Dongyan Zhao. Jointly learning entity and relation representations for entity alignment. *arXiv preprint arXiv:1909.09317*, 2019a.

Yuting Wu, Xiao Liu, Yansong Feng, Zheng Wang, and Dongyan Zhao. Jointly learning entity and relation representations for entity alignment. *arXiv preprint arXiv:1909.09317*, 2019b.

Chengjin Xu, Mojtaba Nayyeri, Fouad Alkhoury, Jens Lehmann, and Hamed Shariat Yazdi. Temporal knowledge graph embedding model based on additive time series decomposition. *arXiv preprint arXiv:1911.07893*, 2019.

Chengjin Xu, Mojtaba Nayyeri, Fouad Alkhoury, Hamed Shariat Yazdi, and Jens Lehmann. Tero: A time-aware knowledge graph embedding via temporal rotation. *arXiv preprint arXiv:2010.01029*, 2020a.

Chengjin Xu, Mojtaba Nayyeri, Yung-Yu Chen, and Jens Lehmann. Knowledge graph embeddings in geometric algebras. *arXiv preprint arXiv:2010.00989*, 2020b.

Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. Embedding entities and relations for learning and inference in knowledge bases. *arXiv preprint arXiv:1412.6575*, 2014.

Hao Zhu, Ruobing Xie, Zhiyuan Liu, and Maosong Sun. Iterative entity alignment via joint knowledge embeddings. In *IJCAI*, volume 17, pp. 4258–4264, 2017.

A Appendix

A.1 Time Complexity of TR-GAT Attention Head

In the Section 2.2 of the original GAT paper Veličković et al. (2017), the authors proposed "The time complexity of a single GAT attention head computing F' features may be expressed as $\mathcal{O}(|V|FF' + |E|F')$, where F is the number of input features, and $|V|$ and $|E|$ are the numbers of nodes and edges in the graph, respectively". Noteworthily, a weight matrix is used for each single attention head and the time complexity of the multiplications of and the feature vectors of $|V|$ nodes is $\mathcal{O}(|V|FF')$. Since we use a weight vector instead of the weight matrix used in the vanilla GAT, the time complexity of the multiplications of and the feature vectors of $|V|$ entities is $\mathcal{O}(|V|F)$. In our work, the number of nodes (entities), edges (quadruples) and input features are denoted as $\mathcal{E}$, $\mathcal{Q}$ and d. Thus, the time complexity of a single attention head in TR-GAT is denoted as $\mathcal{O} = d|\mathcal{E}| + d|\mathcal{Q}|$, and the complexity of a single attention head in vanilla GAT would be $\mathcal{O} = d^2|\mathcal{E}| + d|\mathcal{Q}|$ in our case. It can be seen that the computation of a TR-GAT attention head is more efficient than a vanilla GAT attention head.

A.2 Details of TKG Completion Datasets

ICEWS14 and ICEWS05-15 are the two most common TKG benchmarks extracted from a publicly available large-scale event-based database, Integrated Crisis Early Warning System (ICEWS) [1]. ICEWS is a repository that contains political events with specific time annotations, e.g. (*Barack Obama*, *Make a visit*, *Ukraine*, *2014-07-08*). It is noteworthy that time annotations in ICEWS are all time points. ICEWS14 contains events in 2014, and ICEWS05-15 contains events occurring between 2005-2015. These two datasets are filtered by only selecting the most frequently occurring entities in the graph.

To create YAGO15K, Garcia-Duran et.al. García-Durán et al. (2018) aligned the entities in FB15K with those from YAGO, which contains temporal information. The final dataset is the result of all facts with successful alignment. It is worth noting that since YAGO does not have temporal information for all facts, this dataset is also temporally incomplete and more challenging. Each temporal facts in YAGO15K have a temporal modifier *"occursSince"* or *"occursUntil"* to express if the timestamp is a start time or an end time. Some examples from YAGO15K include (*David Beckham*, *isAffiliatedTo*, *Manchester United F.C.*) and (*David Beckham*, *isMarriedTo*, *Victoria Beckham*, *occursSince*, *"1999-##-##"*). Follow the previous work, we drop the month and day information in timestamps from YAGO15K since most of them are unavailable. The statics of the three TKG completion datasets are listed in Table 6.

Dataset	#Entities	#Relations	Period(year)	#Train	#Valid	#Test
ICEWS14	6,869	230	201	72,826	8,941	8,963
ICEWS05-15	10,094	251	2005-2015	368,962	46,275	46,092
YAGO15K	15,403	34	1513-2017	110,441	13,815	13,800

Table 6: Statistics of TKG completion datasets.

A.3 Details of Time-aware Entity Alignment Datasets

We build two datasets **DICEWS-1K** and **DICEWS-200** from ICEWS05-15 in the similar way to the construction of DFB datasets Zhu et al. (2017). We first randomly divide ICEWS05-15 quadruples into two subsets $\mathcal{Q}_1$ and $\mathcal{Q}_2$ of similar size, and make the overlap ratio of the amount of shared quadruples between $\mathcal{Q}_1$ and $\mathcal{Q}_2$ to all quadruples equal to 50%. The only difference between DICEWS-1K and DICEWS-200 is the proportion of alignment seed $\mathcal{S}$. In DICEWS-1K and DICEWS-200, about 12% and 2% of entity pairs between TKGs are pre-known. Moreover, We set the time unit of ICEWS datasets as 1 day, which means that each day is an individual time step.

YAGO3 and Wikidata are two common large-scale knowledge bases containing time information of various forms including time points, beginning or end time, and time intervals. Lacroix et al. Lacroix

[1]https://dataverse.harvard.edu/dataverse/icews

et al. (2020) extract a subset[2] from Wikidata in which 90% of facts are non-temporal while others have time annotations attached. We select top 50,000 entities according to their frequencies in Wikidata and link them to their equivalent YAGO entities[3] according to their QIDs and the mappings of YAGO entities to Wikidata QIDs. We generate two TKGs only involving the selected entities from the original Wikidata dataset and all YAGO facts, and then attach complementary time information[4] to meta YAGO facts. We build two time-aware datasets **YAGO-WIKI50K-5K** and **YAGO-WIKI50K-1K** by removing non-temporal facts in the generated TKGs and using different ratios of alignment seeds $\mathcal{S}$. In addition, we build a hybrid dataset **YAGO-WIKI20K** containing both temporal and non-temporal facts with 400 pairs of alignment seeds by reducing sizes of entity sets of two TKGs to around 20,000. To generate the shared time set $\mathcal{T}^*$ for a YAGO-WIKI dataset, we drop month and date information and use the first time step τ_0 to represent unobtainable time information.

All time-aware EA datasets are submitted as supplementary materials and their statics are listed in Table 2, where $\mathcal{P}$ denotes the set of reference entity pairs. The set of reference entity pairs other than pre-aligned entity pairs, i.e., $\mathcal{P} - \mathcal{S}$ are used for testing.

A.4 Implementation Details

We implement TR-GAT and TU-GAT models using TensorFlow and test them on a single GeForce GTX Titan X (Pascal) GPU. For TKG completion, we follow the settings of TComplEx, TNTComplEx Lacroix et al. (2020) and ChronoR Sadeghian et al. (2021) to use an Adagrad optimizer with a learning rate of $lr \in \{0.1, 0.2\}$ and set the maximum embedding dimension k no more than 2000. The regularization weights λ_b and $\lambda_{\mathcal{T}}$ are tuned in a range of {0, 0.001, 0.005, 0.01, 0.05,. . . , 10}. We fix the batch size $b = 1000$, the number of multi-head attention mechanisms $M = 2$, the number of attentional layers $L = 1$ and the dropout rate $dr = 0.3$. We adopt the early-stop setting, and fix the max number of epochs $ep = 150$. The optimal configurations of other hyper-parameters used for our proposed models are listed in the Table 7.

Datasets	k	lr	λ_b	$\lambda_{\mathcal{T}}$
ICEWS14	2000	0.1	10	0.005
ICEWS05-15	1500	0.1	1	0.05
YAGO15K	2000	0.2	1	0

Table 7: Optimal hyperparameters of TR-GAT model for TKG completion.

For time-aware EA, we follow the the setting of MRAEA Mao et al. (2020a) to use an Adam optimizer with a learning rate of $lr = 0.001$ and tune k and γ in the ranges of $\{25, 50, 75, 100\}$ and $\{1, 2, 3, 5\}$, respectively. Moreover, we fix $M = 2$, $L = 2$, $dr = 0.3$, $b = |\mathcal{E}_1| + |\mathcal{E}_2|$ for the time-aware EA models. As mentioned in Section 5.2, we use the source codes [5][6][7][8][9] respective to baseline models for **time-aware EA** evaluation, except that we evaluate MTransE based on the implementation of OpenEA framework [10].

For all baseline EA models, we mostly follow their default optimal configurations regarding learning rates lr, batch sizes b, negative sampling rates η, dropout rates dr, numbers of attentional layers L, number of attention heads per layer M and mainly focus on the grid research of embedding dimensions k and margins γ (negative weights α for JAPE). We also follow the respective original paper to set the balance weight $\beta = 0.9$ for GCN-Align. For all baseline models and our proposed models,

[2]https://github.com/facebookresearch/tkbc
[3]http://resources.mpi-inf.mpg.de/yago-naga/yago3.1
[4]http://resources.mpi-inf.mpg.de/yago-naga/yago3.1/yagoMetaFacts.ttl.7z
[5]https://github.com/nju-websoft/JAPE
[6]https://github.com/nju-websoft/BootEA
[7]https://github.com/1049451037/GCN-Align/
[8]https://github.com/MaoXinn/MRAEA
[9]https://github.com/MaoXinn/RREA
[10]https://github.com/nju-websoft/OpenEA/

we tune k in the range of $(25, 50, 75, 100)$, and γ or α in the range of $(0, 0.5, 1, 2, 3, 5, 7, 10, 15, 20)$. Specially, we use the same margin hyperparameters as the original paper for AlignE and MuGNN. To make a fair comparison, we use the same setup for our proposed model as MRAEA and RREA to fix $M = 2$, $L = 2$, $dr = 0.3$, $ep = 6000$, $b = |\mathcal{E}_1| + |\mathcal{E}_2|$ and $\eta = b//|\mathcal{S}| + 1$ where $//$ denotes the round-down after division, and also conduct the same grid research of embedding dimensions k and margins γ for TR-GAT and TU-GAT as what we do for baseline models.

Hyperparameters used for getting the results reported in Table 4 and 5 are listed in Table 8, 9, 11, 12 and 10. In our experiments, for MRAEA, RREA and our proposed models, the usage of the higher-dimensional embeddings would lead to out-of-memory problems since we only use a single mid-range GPU device for training and the learning of large-scale graph neural networks with numerous nodes and high-dimensional embeddings needs excessive memory footprint during the training process. It is predictable that we can possibly further boost the performances of TR-GAT and TU-GAT on YAGOWIKI50K datasets by increasing their embedding dimensions with more high-performance GPU devices.

The source codes and datasets used in this work are submitted as the supplementary materials for reproducibility and will be released on Github after the anonymity period.

Models	k	lr	b	η	γ (or α)	dr
MTransE	100	0.01	20,000	10	10	-
JAPE	100	0.01	10,000	1	2	-
AlignE	100	0.01	20,000	10	0.01, 2, 0.7	-
GCN-Align	100	20	-	5	3	0
MuGNN	100	0.001	64	25	1.0, 1.0, 0.12	0.2
MRAEA	100	0.001	19,054	20	1	0.3
RREA	100	0.005	19,054	20	3	0.3
TU-GAT	100	0.001	19,054	20	1	0.3
TR-GAT	100	0.001	19,054	20	1	0.3

Table 8: Optimal hyperparameters of target models for DICEWS-1K.

Models	k	lr	b	η	γ (or α)	dr
MTransE	100	0.01	20,000	10	7	-
JAPE	100	0.01	10,000	1	3	-
AlignE	100	0.01	20,000	10	0.01, 2, 0.7	-
GCN-Align	100	20	-	5	7	0
MuGNN	100	0.001	64	25	1.0, 1.0, 0.12	0.2
MRAEA	100	0.001	19,054	96	2	0.3
RREA	100	0.005	19,054	96	2	0.3
TU-GAT	100	0.001	19,054	96	2	0.3
TR-GAT	100	0.001	19,054	96	1	0.3

Table 9: Optimal hyperparameters of target models for DICEWS-200.

Models	k	lr	b	η	γ (or α)	dr
TU-GAT	100	0.001	39,422	99	2	0.3
TR-GAT	100	0.001	39,422	99	2	0.3

Table 10: Optimal hyperparameters of target models for YAGO-WIKI20K.

Models	k	lr	b	η	γ (or α)	dr
MTransE	100	0.01	20,000	10	10	-
JAPE	100	0.01	10,000	1	3	-
AlignE	100	0.01	20,000	10	0.01, 2, 0.7	-
GCN-Align	100	20	-	5	3	0
MuGNN	100	0.001	64	25	1.0, 1.0, 0.12	0.2
MRAEA	75	0.001	98,851	20	1	0.3
RREA	50	0.005	98,851	20	1	0.3
TU-GAT	50	0.001	98,851	20	1	0.3
TR-GAT	50	0.001	98,851	20	1	0.3

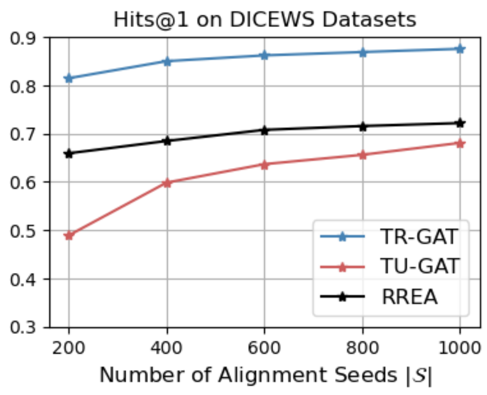
Table 11: Optimal hyperparameters of target models for YAGO-WIKI50K-5K.

Models	k	lr	b	η	γ (or α)	dr
MTransE	100	0.01	20,000	10	1	-
JAPE	100	0.01	10,000	1	2	-
AlignE	100	0.01	20,000	10	0.01, 2, 0.7	-
GCN-Align	100	20	-	5	10	0
MuGNN	100	0.001	64	25	1.0, 1.0, 0.12	0.2
MRAEA	75	0.001	98,851	99	2	0.3
RREA	50	0.005	98,851	99	1	0.3
TU-GAT	50	0.001	98,851	99	1	0.3
TR-GAT	50	0.001	98,851	99	1	0.3

Table 12: Optimal hyperparameters of target models for YAGO-WIKI50K-1K.

A.5 Robustness Study

As shown in Table 4, TR-GAT seems more robust to the size of alignment seed $|\mathcal{S}|$ compared against the previous state-of-the-art methods and its time-unaware variant. To verify this observation, we test TR-GAT, TU-GAT and RREA which obtain the best performance among all baselines, with $|\mathcal{S}|$ varying from 200 to 1000 with step size of 200. As shown in Figure 4, TR-GAT is not only significantly superior to TU-GAT and RREA in all seed sizes, but also has a more gradual slope curve. In practical applications, alignment seeds are difficult to obtain. Since our proposed time-aware EA model performs well with a small amount of pre-aligned entity pairs, it can more easily be applied in large-scale KGs compared to time-unaware EA methods.



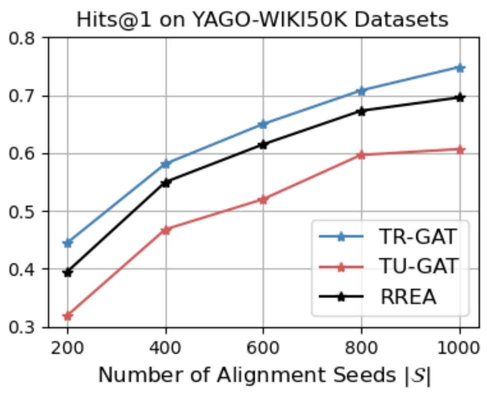


Figure 4: Hits@1 of TEA-GNN, TU-GNN and RREA on entity alignment, w.r.t. number of alignment seeds $|\mathcal{S}|$.