Learning to Walk across Time for Interpretable Temporal Knowledge Graph Completion

Jaehun Jung
Seoul National University,
Kakao Enterprise
Seoul, Korea
sharkmir1@snu.ac.kr

Jinhong Jung
Jeonbuk National University
Jeonju, Korea
jinhongjung@jbnu.ac.kr

U Kang
Seoul National University
Seoul, Korea
ukang@snu.ac.kr

ABSTRACT

Static knowledge graphs (KGs), despite their wide usage in relational reasoning and downstream tasks, fall short of realistic modeling of knowledge and facts that are only temporarily valid. Compared to static knowledge graphs, temporal knowledge graphs (TKGs) inherently reflect the transient nature of real-world knowledge. Naturally, automatic TKG completion has drawn much research interests for a more realistic modeling of relational reasoning. However, most of the existing models for TKG completion extend static KG embeddings that do not fully exploit TKG structure, thus lacking in 1) accounting for temporally relevant events already residing in the local neighborhood of a query, and 2) path-based inference that facilitates multi-hop reasoning and better interpretability. In this paper, we propose T-GAP, a novel model for TKG completion that maximally utilizes both temporal information and graph structure in its encoder and decoder. T-GAP encodes query-specific substructure of TKG by focusing on the temporal displacement between each event and the query timestamp, and performs path-based inference by propagating attention through the graph. Our empirical experiments demonstrate that T-GAP not only achieves superior performance against state-of-the-art baselines, but also competently generalizes to queries with unseen timestamps. Through extensive qualitative analyses, we also show that T-GAP enjoys transparent interpretability, and follows human intuition in its reasoning process.

CCS CONCEPTS

• **Computing methodologies** → **Knowledge representation and reasoning**; **Temporal reasoning**; **Neural networks**.

KEYWORDS

Knowledge Graph Completion; Graph Neural Networks; Relational Reasoning

ACM Reference Format:
Jaehun Jung, Jinhong Jung, and U Kang. 2021. Learning to Walk across Time for Interpretable Temporal Knowledge Graph Completion. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '21), August 14–18, 2021, Virtual Event, Singapore.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3447548.3467292

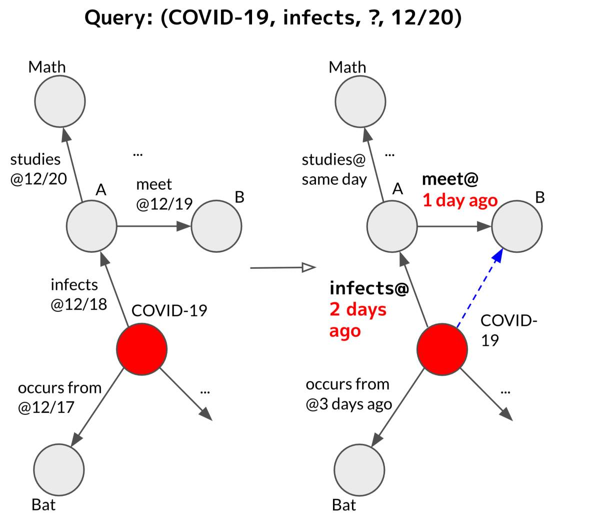

Figure 1: Example of temporal displacement. Edges in bold are important events relevant to the input query.

1 INTRODUCTION

Knowledge graph (KG), due to its expressiveness over structured knowledge, has been widely used in various applications including recommender system [21], information retrieval [15], concept discovery [10], and question answering [26]. Moreover, the inherent sparseness of KGs gives rise to research interests on automatic knowledge graph completion which predicts missing entity for incomplete queries in form of (subject, predicate, ?).

Recent advancements in KG completion tasks have extended to a more challenging domain of temporal knowledge graphs (TKGs), as they model realistic events that are only temporarily valid. Triples in temporal graphs are annotated with the corresponding time token, taking form of (subject, predicate, object, timestamp). Naturally, TKG completion task can be formulated as predicting missing tail entity for queries in the form of (subject, predicate, ?, timestamp).

Majority of existing approaches to TKG completion propose a straightforward extension of conventional KG embeddings onto the temporal graphs [4, 12]. There are two major rooms of improvement from the existing models, each from the encoding phase, and the decoding phase. In the encoding phase, a model could benefit from the rich neighborhood information residing in the structure of TKGs. Extracting, and encoding query-relevant information from the neighborhood nodes and their associated edges would help in fine-grained modeling of entity representation. The importance of neighborhood encoding has already been appreciated in static KGs [2, 17], but extension of these models to TKG is non-trivial, due to the additional time dimension in each triple.

Next, in the decoding phase, relational reasoning on TKG could leverage path-based inference. Several works adopted path-traversal model in static KGs [3, 23], showing preferable performance in relational reasoning compared to embedding-based models. Although path-based inference helps in capturing long-term dependency between nodes and gives better interpretability over model's reasoning process, these approaches are yet to be examined in TKG completion tasks.

To this end, we propose T-GAP (**T**emporal **G**NN with **A**ttention **P**ropagation), a novel model for TKG completion, that tackles both challenges stated above. In the encoder, we introduce a new type of temporal graph neural network (GNN), which attentively aggregates query-relevant information from each entity's local neighborhood. Specifically, we focus on encoding the temporal displacement between the timestamps of the input query and each edge being encoded. An intuitive example is presented in Figure 1. Evidently, the two most important facts to answer the given query (COVID-19, infects, ?, 12/20), are that A has been infected to COVID-19 at 12/18, and A met B at 12/19. Here, one should note that the valuable information lies in the fact that A got infected *2 days before* the time of interest, not that he was infected at a *specific day* of 12/18. What matters most when accounting for temporal events, is the relative displacement between the event and the time of interest, rather than the absolute time of the event. To effectively capture the temporal displacement, our proposed encoder separately encodes both the sign of the displacement (i.e. whether the time of the event belongs to past, present, or future), and the magnitude of the displacement (i.e. how far is the event from the time of our interest).

Also, T-GAP performs a generalized path-based inference over TKG, based on the notion of Attention Flow [24]. In each decoding step, our model explores KG by propagating attention value at each node to its reachable neighbor nodes, rather than sampling one node to walk from the neighborhood. The soft approximation of path traversal with attention propagation not only allows our model to be easily trained with end-to-end supervised learning, but also provides better interpretation over its reasoning process, compared to embedding-based models. [1]

In summary, our contributions are as follows:

- We propose a new GNN encoder that effectively captures query-relevant information from temporal KGs.
- Based on the encoder, we present T-GAP, a novel path-based TKG reasoning model. We examine T-GAP in 3 benchmark datasets in TKG completion task, and the quantitative metrics show clear improvement in all benchmarks compared to the state-of-the-art baselines.
- By analyzing the inferred attention distribution, we show that T-GAP possesses clear *interpretability* over its reasoning process.

The symbols used in the paper are provided in Table 1.

[1]Source code available at https://github.com/sharkmir1/T-GAP.

Table 1: Notations of symbols used in the paper.

Notation	Description
G_{KG}	A graph $\subseteq V_{KG} \times R_{KG} \times V_{KG} \times T_{KG}$
V_{KG}	Set of nodes in G_{KG}
R_{KG}	Set of relations in G_{KG}
T_{KG}	Set of timestamps in G_{KG}
v_i	The i-th node
e_{ij}	The edge from v_i to v_j
$\mathbf{q}$	Query context vector
$\mathbf{h}_i$	Preliminary node feature of v_i in G_{KG}
$\boldsymbol{\rho}_{ij}$	Edge feature of e_{ij} in G_{KG}
$G_{sub}^{(t)}$	Query-dependent subgraph of G_{KG} at t-th decoding step
$\mathbf{g}_i^{(t)}$	Query-dependent node feature of v_i in $G_{sub}^{(t)}$
$a_i^{(t)}$	Node attention value assigned to v_i at the t-th decoding step
$\widetilde{a}_{ij}^{(t)}$	Edge attention value propagated through e_{ij} at the t-th decoding step
$\mathbf{W}_{\cdot}$	Various weight matrices
$\|\|$	Concatenation

2 RELATED WORK

Various approaches have been made toward automatic completion of static KGs. Majority of conventional approaches propose embedding-based models, including translative [1, 22], and factorization based models [19, 25]. To complement for the weak representation power of KG embeddings, several recent works incorporate neural network either to the scoring function, or as an additional encoding layer. ConvE [5] adopts convolutional layer to model sophisticated interaction between entities in the scoring function. KBGAT [17], and RGHAT [27] adopt a variant of graph attention network to contextualize entity embedding with the corresponding neighborhood structure. DPMPN [23] employs two GNNs to encode both the original graph and an induced subgraph, for a scalable learning of KG structure. We extend the neighborhood encoding scheme of these prior works to temporal graphs, especially focusing on query dependent encoding of KG structure.

Existing works on TKG completion primarily focus on extending static KG embedding to dynamic graphs. Different models mainly differ in how to represent independent timestamps, and incorporate it to their scoring functions. HyTE [4] extends TransH [22], projecting entity and relation embedding to time-specific hyperplane. García-Durán et al. (2018) propose to represent temporal relation as a sequence of relation type and characters in the timestamp, and encode the sequence using RNN. TComplEx [12] considers the score of each triple as canonical decomposition of order 4 tensors in complex domain, adding time embedding to the order 3 decomposition of ComplEx. Goel et al. [7] suggest to learn entity representation that changes over time, transforming part of the embedding with sinusoidal activation of learned frequencies. While successfully extending to temporal graphs, these models limit their inference to the interaction of embeddings in the latent space, hence inherently lack interpretation over model's reasoning process.

Meanwhile, path-based reasoning has been actively employed for node prediction on knowledge graphs. Lin et al. [14] infuse

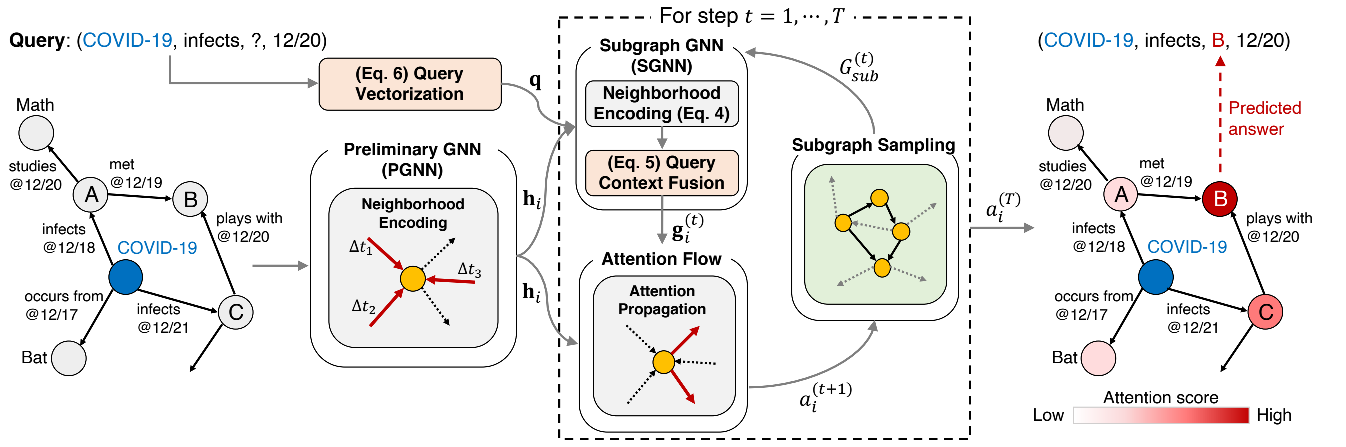

Figure 2: Overview of T-GAP. Starting from the query head, T-GAP explores relevant nodes and edges by iteratively propagating attention, and reaches at the target entity after the final propagation step.

multi-hop path information into entity representation, using additive composition of relation embeddings. Das et al. [3] and Lin et al. [13] consider KG reasoning problem as partially observed Markov Decision Process, training a policy network that starts traversing from the query head and reaches at the predicted tail entity. Xu et al. [24] propose a soft approximation of path traversal with attention distribution. T-GAP aligns with these line of works by exploring informative paths relevant to the input query with attention propagation, to maximally utilize the KG structure during the decoding process.

Lastly, we find connections to our work from recently suggested GNNs for dynamic graphs. Pareja et al. [18] employs RNN to evolve GCN parameters across time. Han et al. [8] present Graph Hawkes Neural Network, modeling temporal dependency between events with Hawkes process. While these works focus on modeling the evolution of the graph as whole, we discuss a new methodology of encoding temporal displacement between each event and the input query, which better suits our goal to explore query-relevant paths and reach the answer node.

3 PROPOSED METHOD

3.1 Overview

First, we denote TKG as $G_{KG} = \{(v, r, u, t)\} \subseteq V_{KG} \times R_{KG} \times V_{KG} \times T_{KG}$, where V_{KG} is a set of entities, R_{KG} is a set of relations, and T_{KG} is a set of timestamps associated with the relations. Given the graph G_{KG} and query $\mathbf{q} = (v_{query}, r_{query}, ?, t_{query})$, TKG completion is formulated as predicting $u \in V_{KG}$ the most probable to fill in the query. Note that head entity prediction $(?, r_{query}, u_{query}, t_{query})$ can be evaluated in a similar way after adding inverse edge of each triple to G_{KG}. We denote $\overleftarrow{N_i}$ as a set of incoming neighbor nodes of v_i, i.e. nodes posessing edges toward v_i, and $\overrightarrow{N_i}$ as a set of outgoing neighbor nodes of v_i. With T-GAP, we aim to resolve the two aforementioned issues of concurrent TKG completion models - (1) *Encoder-side*: suboptimal utilization of temporal and structural information residing in the graph, and (2) *Decoder-side*: lack of multi-hop reasoning and interpretability in finding the optimal answer.

Regarding the first issue, Hu et al. [9] has shown preferable performance in node classification by introducing HGT, a GNN with relative temporal encoding between node-associated timestamps. However, one should note that TKG completion is inherently different from other graph-related tasks in that it is *query-dependent*. Depending on the type of query, the usefulness of temporal information in the neighborhood of each entity dramatically shifts. Hence, unlike HGT, T-GAP fuses neighborhood information of each entity while focusing on the temporal displacement between the *query timestamp*, and each edge to be encoded. Also, we decompose the encoding stage into two GNNs operating on different topology of the input graph: Preliminary GNN (PGNN) and Subgraph GNN (SGNN). The decomposition not only helps in extracting only the query-relevant information from the original graph, but also scales up the encoder by query-dependent pruning of irrelevant edges.

T-GAP handles the second issue by attention propagation-based decoder, namely Attention Flow. Unlike scoring function and recurrent decoder, Attention Flow naturally allows multi-hop reasoning by propagating attention through existing edges in the graph. Also, the inferred attention distribution provides us with better interpretability, which will be thoroughly analyzed in the experiments.

Figure 2 illustrates an overview of T-GAP's reasoning process. Given G_{KG} and the query, in the encoding phase, T-GAP first uses PGNN to create preliminary node feature $\mathbf{h}_i$ for all entities in the G_{KG}. Next, at each decoding step $t = 1, \cdots, T$, T-GAP iteratively samples a subgraph $G_{sub}^{(t)}$ from G_{KG}, that consists only of query-relevant nodes and edges. For each entity i included in $G_{sub}^{(t)}$, SGNN creates query-dependent node feature $\mathbf{g}_i^{(t)}$, incorporating the query vector $\mathbf{q}$ and the preliminary feature $\mathbf{h}_i$. Using both $\mathbf{h}_i$ and $\mathbf{g}_i^{(t)}$, Attention Flow computes transition probability to propagate the attention value of each node to its reachable neighbor nodes, creating the next step's node attention distribution $a_i^{(t+1)}$. After the final propagation step T[2], the answer to the input query is inferred as the node with the highest attention value $a_i^{(T)}$.

3.2 Preliminary GNN

Given G_{KG}, T-GAP first randomly initializes node feature $\mathbf{h}_i$ for all $v_i \in V_{KG}$. Then, to contextualize the representation of entities in G_{KG} with the graph structure, each layer in PGNN updates node feature $\mathbf{h}_i$ of entity v_i by attentively aggregating v_i's neighborhood information. The important intuition underlying PGNN is that the temporal displacement between timestamps of the query and each event is integral to capture the time-related dynamics of each entity. Therefore, for each timestamp t_{ij} of edge e_{ij} in G_{KG}, we resolve to separately encode the sign and magnitude of the temporal displacement $\triangle t_{ij} = t_{ij} - t_{query}$. Concretely, PGNN computes message $\mathbf{m}_{ij}$ from entity v_i to v_j as follows:

$$\mathbf{m}_{ij} = \mathbf{W}_{\lambda(\triangle t_{ij})}(\mathbf{h}_i + \boldsymbol{\rho}_{ij} + \boldsymbol{\tau}_{|\triangle t_{ij}|})$$
$$\text{where } \mathbf{W}_{\lambda(\triangle t_{ij})} = \begin{cases} \mathbf{W}_{past} & \text{if } \triangle t_{ij} < 0 \\ \mathbf{W}_{present} & \text{if } \triangle t_{ij} = 0 \\ \mathbf{W}_{future} & \text{if } \triangle t_{ij} > 0 \end{cases} \tag{1}$$

$\boldsymbol{\rho}_{ij}$ is a relation-specific parameter associated with r_{ij} which denotes the relation that connects v_i to v_j. In addition to the entity and relation, we learn the discretized embedding of size of temporal displacement, i.e. $\boldsymbol{\tau}_{|\triangle t_{ij}|}$. We take account of the sign of displacement by applying sign-specific weight for each event.

Next, the new node feature $\mathbf{h}'_j$ is computed by attention weighted sum of all incoming messages to v_j:

$$\begin{aligned} \mathbf{h}'_j &= \sum_{i \in \overleftarrow{N}_j} a_{ij}\mathbf{m}_{ij}, \\ a_{ij} &= \mathtt{softmax}_i(\alpha_{ij}), \\ \alpha_{ij} &= \mathtt{LeakyReLU}\left((\mathbf{W}_Q\mathbf{h}_j)^\top(\mathbf{W}_K\mathbf{m}_{ij})\right) \end{aligned} \tag{2}$$

The attention values are computed by applying softmax over all incoming edges of v_j, with $\mathbf{h}_j$ as query and $\mathbf{m}_{ij}$ as key.

In addition, we extend this attentive aggregation scheme to multi-headed attention, which helps to stabilize the learning process and jointly attend to different representation subspaces [20]. Hence our message aggregation scheme is modified to:

$$\mathbf{h}'_j = \Big\|_{k=1}^{K} \sum_{i \in \overleftarrow{N}_j} a_{ij}^k \mathbf{m}_{ij}^k \tag{3}$$

concatenating independently aggregated neighborhood features from each attention heads, where K is a hyperparameter indicating the number of attention heads.

3.3 Subgraph GNN

At each decoding step t, SGNN updates node feature $\mathbf{g}_i$ for all entities that are included in the induced subgraph of current step, $G_{sub}^{(t)}$. We present the detailed procedure of subgraph sampling in upcoming section. Essentially, SGNN not only contextualizes $\mathbf{g}_i$ with respective incoming edges, but also infuses the query context vector with the entity representation. First, the subgraph features, for entities newly added to the subgraph, are initialized to their corresponding preliminary features $\mathbf{h}_j$. Next, SGNN performs message propagation, using the same message computation and aggregation scheme as PGNN (Eq. 1-3), but with separate parameters:

$$\widetilde{\mathbf{g}}'_j = \Big\|_{k=1}^{K} \sum_{i \in \overleftarrow{N}_j} a_{ij}^k \mathbf{m}_{ij}^k \tag{4}$$

This creates an intermediate node feature $\widetilde{\mathbf{g}}'_j$. The intermediate features are then concatenated with query context vector $\mathbf{q}$, and linear-transformed back to the node embedding dimension, creating new feature $\mathbf{g}'_j$:

$$\mathbf{g}'_j = \mathbf{W}_g[\widetilde{\mathbf{g}}'_j \,\|\, \mathbf{q}] \tag{5}$$

$$\mathbf{q} = \mathbf{W}_c \times \mathtt{LeakyReLU}\Big(\mathbf{W}_{present}(\mathbf{h}_{query} + \boldsymbol{\rho}_{query})\Big) \tag{6}$$

where $\mathbf{h}_{query}$ is the preliminary feature of v_{query}, and $\boldsymbol{\rho}_{query}$ is the relation parameter for r_{query}.

3.4 Attention Flow

T-GAP models path traversal with the soft approximation of attention flow, iteratively propagating the attention value of each node to its outgoing neighbor nodes. Initially, the node attention is initialized to 1 for v_{query}, and 0 for all other entities. Hereafter, at each step t, Attention Flow propagates edge attention $\widetilde{a}_{ij}^{(t)}$ and aggregates them to node attention $a_j^{(t)}$:

$$\widetilde{a}_{ij}^{(t+1)} = \mathcal{T}_{ij}^{(t+1)} a_i^{(t)}, \quad a_j^{(t+1)} = \sum_{i \in \overleftarrow{N}_j} \widetilde{a}_{ij}^{(t+1)}$$
$$s.t. \ \sum_i a_i^{(t+1)} = 1, \ \sum_{ij} \widetilde{a}_{ij}^{(t+1)} = 1$$

The key here is the transition probability $\mathcal{T}_{ij}$. In this work, we define $\mathcal{T}_{ij}$ as applying softmax over the sum of two scoring terms, regarding both the preliminary feature $\mathbf{h}$, and the subgraph feature $\mathbf{g}$:

$$\begin{aligned} \mathcal{T}_{ij}^{(t+1)} = \mathtt{softmax}_j(&score(\mathbf{g}_i^{(t)}, \mathbf{g}_j^{(t)}, \boldsymbol{\rho}_{ij}, \boldsymbol{\tau}_{|\triangle t_{ij}|}) + \\ &score(\mathbf{g}_i^{(t)}, \mathbf{h}_j, \boldsymbol{\rho}_{ij}, \boldsymbol{\tau}_{|\triangle t_{ij}|})), \end{aligned}$$
$$score(\mathbf{i}, \mathbf{j}, \mathbf{r}, \boldsymbol{\tau}) = \sigma\left((\mathbf{W}_Q\mathbf{i})^\top(\mathbf{W}_K(\mathbf{j} + \mathbf{r} + \boldsymbol{\tau}))\right)$$

The first scoring term $score(\mathbf{g}_i^{(t)}, \mathbf{g}_j^{(t)}, \boldsymbol{\rho}_{ij}, \boldsymbol{\tau}_{|\triangle t_{ij}|})$ accounts only for subgraph feature $\mathbf{g}_i$ and $\mathbf{g}_j$, giving additional point to entities that are already included in the subgraph (note that $\mathbf{g}_i$ is initialized to zero for entities not yet included in the subgraph). Meanwhile, the second scoring term $score(\mathbf{g}_i^{(t)}, \mathbf{h}_j, \boldsymbol{\rho}_{ij}, \boldsymbol{\tau}_{|\triangle t_{ij}|})$ could be regarded as *exploring term*, as it relatively prefers entities not included in the subgraph, by modeling the interaction between $\mathbf{g}_i$ and $\mathbf{h}_j$.

3.5 Subgraph Sampling

The decoding process of T-GAP depends on the iterative sampling of query-relevant subgraph $G_{sub}^{(t)}$. The initial subgraph $G_{sub}^{(0)}$ before the first propagation step contains only one node, v_{query}. As the propagation step proceeds, edges with high relevance to the input query, measured by the size of attention value assigned to the edges, are added to the previous step's subgraph. Specifically, the subgraph sampling at step t proceeds as follows:

- Find x number of *core nodes* with highest (nonzero) node attention value $a_i^{(t-1)}$ at the previous step.

[2] T is a hyperparameter specific to the input graph.

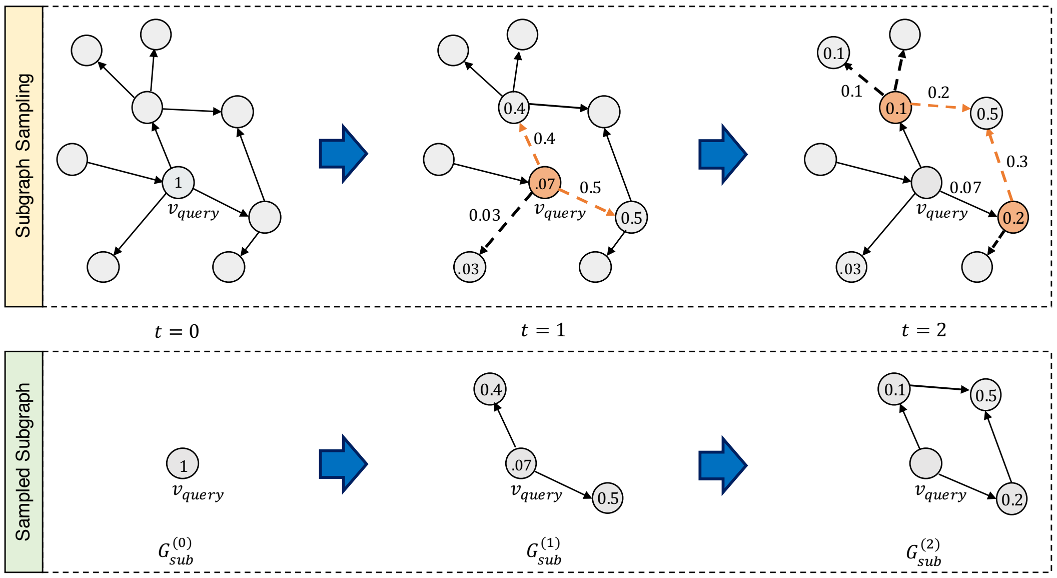

Figure 3: Example of subgraph sampling in T-GAP (Section 3.5). The graphs above represent the respective node / edge attention distribution at the initial state ($t = 0$), after the first propagation step ($t = 1$), and after the second propagation step ($t = 2$). The graphs below show the sampled subgraph at each step t. x ($= 2$) orange nodes are the core nodes retrieved at a step, and y ($= 3$) dashed edges from each core node are candidate edges for the sampled subgraph. Among the candidate edges, z ($= 2$) orange edges are newly added to the previous subgraph.

- For each of the core node, sample y number of edges that originate from the node.
- Among $x \cdot y$ sampled edges, find z number of edges with highest edge attention value $\widetilde{a}_{ij}^{(t)}$ at the current step.
- Add the z edges to $G_{sub}^{(t-1)}$.

In this module, x, y, z are hyperparameters. Intuitively, we only collect 'important' events that originate from 'important' entities (core nodes) with respect to the query, while keeping the subgraph size under control (edge sampling).

Figure 3 is an illustrative example for the subgraph sampling procedure in T-GAP. The hyperparameters for the example are as follows: $x = 2$ (the maximum number of core nodes), $y = 3$ (the maximum number of candidate edges, considered by each core node), and $z = 2$ (the number of sampled edges added to the subgraph).

In the initial state, the only node associated with nonzero attention $a_i^{(0)}$ is the query head v_{query}. Also, the initial subgraph $G_{sub}^{(0)}$ consists only of the node v_{query}. At the first decoding step $t = 1$, T-GAP first finds top-x core nodes (where $x = 2$) w.r.t. nonzero node attention scores $a_i^{(0)}$ of the previous step $t = 0$. Since the only node with nonzero attention value is v_{query}, it is retrieved as the core node. Next, T-GAP randomly samples at most $y = 3$ edges that originate from the core node (e.g. dashed edges with weights). Among the sampled edges, it selects top-z (where $z = 2$) edges in the order of edge attention values $\widetilde{a}_{ij}^{(1)}$ at the current step; then, they are added to $G_{sub}^{(0)}$, resulting in the new subgraph $G_{sub}^{(1)}$.

At the second decoding step $t = 2$, T-GAP again finds x core nodes that correspond to highest attention values $a_i^{(1)}$ (e.g. nodes annotated with 0.1 and 0.2, respectively). Then, y outgoing edges for each core node are sampled; among $x \cdot y$ sampled edges, z edges with highest edge attention values $\widetilde{a}_{ij}^{(2)}$ are added to $G_{sub}^{(1)}$, creating the new subgraph $G_{sub}^{(2)}$. As seen in Figure 3, the incremental subgraph sampling scheme allows our model to iteratively expand the range of nodes and edges to attend, while guaranteeing that the critical nodes and edges in the previous steps are kept included in the latter subgraphs.

By flexibly adjusting the subgraph related hyperparameters x, y, and z, T-GAP is readily calibrated between reducing computational complexity and optimizing the predictive performance. Intuitively, with more core nodes, more sampled edges, and more edges added to the subgraph, T-GAP can better attend to the substructure of TKG that otherwise might have been discarded. Meanwhile, with small x, y, and z, T-GAP can easily scale-up to large graphs by reducing the number of message-passing operations in SGNN.

3.6 Model Training

We partition the triples in G_{KG} into train, valid, and test sets. In the training phase, from each of the triple $(v_{train}, r_{train}, u_{train}, t_{train})$ in the train set, we generate a query $(v_{train}, r_{train}, ?, t_{train})$ by masking u_{train}. As T-GAP consists only of differentiable operations, the model can be trained end-to-end by given each query $(v_{train}, r_{train}, ?, t_{train})$, maximizing on the node attention assigned to u_{train} after T propagation steps. Hence, the loss function is simply defined as:

$$\mathcal{L} = -\sum_{i \in train} \log a_{u_i}^{(T)}$$

Note that although edge sampling brings in stochasticity to T-GAP's inference, this does not hinder the end-to-end training of

Table 2: T-GAP outperforms baselines in all three benchmarks, over all metrics. We use official implementation of DE-SimplE, and T(NT)ComplEx for Wikidata11k. Other results with [▼] are from García-Durán et al. [6], results with [⋄] are from Goel et al. [7], and results on T(NT)ComplEx are from Lacroix et al. [12].

	ICEWS14				ICEWS05-15				Wikidata11k			
Model	*MRR*	*Hits@1*	*Hits@3*	*Hits@10*	*MRR*	*Hits@1*	*Hits@3*	*Hits@10*	*MRR*	*Hits@1*	*Hits@3*	*Hits@10*
TransE [▼]	0.280	9.4	-	63.7	0.294	9.0	-	66.3	0.316	18.1	-	65.9
DistMult [▼]	0.439	32.3	-	67.2	0.456	33.7	-	69.1	0.316	18.1	-	66.1
ConT [⋄]	0.185	11.7	20.5	31.5	0.163	10.5	18.9	27.2	-	-	-	-
TTransE [▼]	0.255	7.4	-	60.1	0.271	8.4	-	61.6	0.488	33.9	-	80.6
HyTE [⋄]	0.297	10.8	41.6	65.5	0.316	11.6	44.5	68.1	-	-	-	-
TA-TransE [▼]	0.275	9.5	-	62.5	0.299	9.6	-	66.8	0.484	32.9	-	80.7
TA-DistMult [▼]	0.477	36.3	-	68.6	0.474	34.6	-	72.8	0.700	65.2	-	78.5
DE-SimplE [⋄]	0.526	41.8	59.2	72.5	0.513	39.2	57.8	74.8	0.310	18.4	31.8	62.5
TComplEx	0.560	47.0	61.0	73.0	0.580	49.0	64.0	76.0	0.731	67.3	76.2	84.5
TNTComplEx	0.560	46.0	61.0	74.0	0.600	50.0	65.0	78.0	0.718	65.4	74.9	85.6
T-GAP	**0.610**	**50.9**	**67.7**	**79.0**	**0.670**	**56.8**	**74.3**	**84.5**	**0.778**	**69.7**	**84.4**	**90.3**

the model. Since the sampling is not parameterized and we only use node feature g from the sampled subgraph, gradients back-propagate through g, not through the sampling operation.

4 EXPERIMENT

We run experiments to answer the following questions.

- **Performance (Section 4.2)**. How well does T-GAP complete incomplete queries on different benchmarks?
- **Temporal generalization (Section 4.3)**. How well does T-GAP generalize to unseen timestamps?
- **Interpretability (Section 4.4)**. Does T-GAP provide interpretability, and align with human intuition?

4.1 Experimental Setting

4.1.1 Datasets. We evaluate our proposed method on three standard benchmark datasets shown in Table 7 for TKG completion: ICEWS14, ICEWS05-15, and Wikidata11k, all suggested by García-Durán et al. [6]. ICEWS14 and ICEWS05-15 are subsets of ICEWS[3], each containing socio-political events in 2014, and from 2005 to 2015 respectively. Wikidata11k is a subset of Wikidata[4], posessing facts of various timestamps that span from A.D. 25 to 2020. All facts in Wikidata11k are annotated with additional temporal modifier, *occurSince* or *occurUntil*. For the sake of consistency and simplicity, we follow García-Durán et al. [6] to merge the modifiers into predicates rather than modeling them in separate dimension (e.g. (A, loves, B, since, 2020) transforms to (A, loves-since, B, 2020)).

4.1.2 Baselines. We compare T-GAP with representative static KG embeddings, TransE and DistMult, and state-of-the-art embedding -based baselines on temporal KGs, including ConT [16], TTransE [11], HyTE [4], TA [6], DE-SimplE [7], and T(NT)ComplEx [12].

4.1.3 Implementation Details. For each dataset, we create G_{KG} with only the triples in the train set. We add inverse edges to G_{KG} for proper path-based inference on reciprocal relations. Also, we follow Xu et al. [23] by adding self-loops to all entities in the graph, allowing the model to stay at the 'answer node' if it reaches an optimal entity in $t < T$ steps. Following prior works [6, 7, 12], we evaluate T-GAP in both head and tail entity prediction and average the metrics. To measure T-GAP's performance in head entity prediction, we add reciprocal triples to valid and test sets too. For all datasets, we find through empirical evaluation that setting the maximal path length $T = 3$ results in the best performance. Following previous works, we fix the dimension of entity / relation / displacement embedding to 100. Except for the embedding size, we search for the best set of hyperparameters using grid-based search, choosing the value with the best *Hits@1* while all other hyperparameters are fixed. We provide further implementation details including hyperparameter search bounds and the best configuration in Supplement A.

[3]https://dataverse.harvard.edu/dataverse/icews
[4]https://www.wikidata.org/wiki/Wikidata:Main_Page

Table 3: Ablation study results on ICEWS14, compared to the best configuration. Results annotated 'T-GAP' are the best performance of T-GAP with temporal displacement encoding, 1-layer PGNN, and subgraph sampling.

Model	*MRR*	*Hits@1*	*Hits@3*	*Hits@10*
No Displacement	0.477	35.7	53.9	71.5
No Subgraph	0.598	49.3	66.8	78.5
No PGNN	0.590	49.9	66.5	78.4
2-layer PGNN	0.605	50.2	67.3	78.9
T-GAP	**0.610**	**50.9**	**67.7**	**79.0**

4.2 Benchmark Performance

Table 2 shows the overall evaluation result of T-GAP against baseline methods. Along with Hits@1, 3, 10, we report MRR of the ground-truth entity, compared to the baseline methods. As seen in the table, T-GAP outperforms baseline models in all the benchmarks, improving up to 10% relative performance consistently over all metrics. We find through ablation study that the improvements are mainly contributed from resolving the two shortcomings of pre-existing TKG embeddings, which we indicate in earlier sections

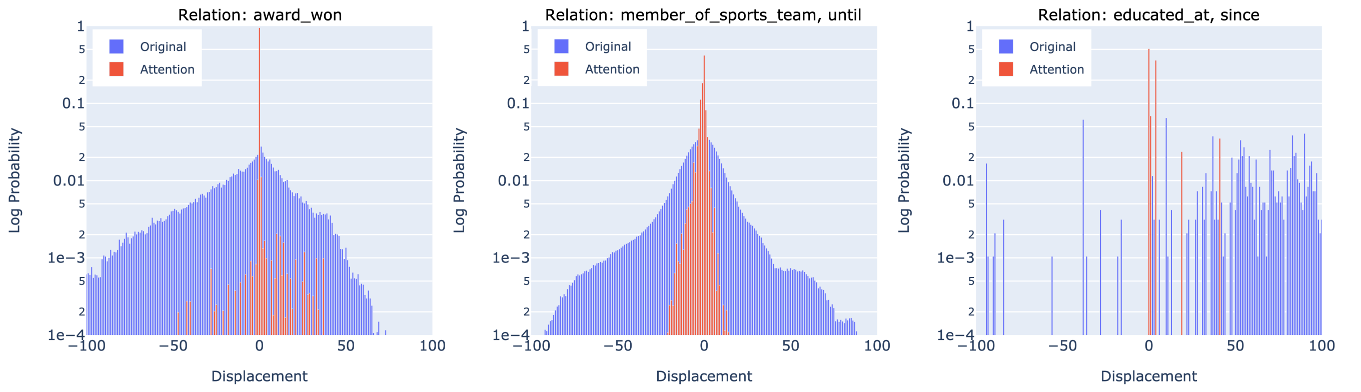

Figure 4: Edge attention distribution over different temporal displacements. Note that y-axis is in log scale. For better visualization, we cut the range of temporal displacement to $[-100, 100]$ in all charts. T-GAP learns to effectively attend on events with specific temporal distance from the query time, depending on the relation type of the query.

- absence of 1) neighborhood encoding scheme, and 2) path-based inference with scalable subgraph sampling.

4.2.1 Model Variants and Ablation. To further examine the effect of our proposed method in solving the aforementioned two problems, we conduct an ablation study as shown in Table 3. First, we consider T-GAP without temporal displacement encoding. In both PGNN and SGNN, we do not consider the sign and magnitude of temporal displacement, and simply learn the embedding of each timestamp as is. While computing the message $\mathbf{m}_{ij}$, the two GNNs simply add the time embedding $\boldsymbol{\tau}_{t_{ij}}$ to $\mathbf{h}_i + \boldsymbol{\rho}_{ij}$. No sign-specific weight is multiplied, and all edges are linear-transformed with the same weight matrix. In this setting, T-GAP's performance on ICEWS14 degrades about 30% in Hits@1, and performs similar to TA-DistMult in Table 2. This result attests to the importance of temporal displacement for the neighborhood encoding in temporal KGs.

Next, to analyze the effect of subgraph sampling on overall performance, we resort to a new configuration of T-GAP where no subgraph sampling is applied, and SGNN creates node feature $\mathbf{g}_i$ for all entities in G_{KG}. Here, T-GAP's performance slightly degrades about 1 percent in all metrics. This implies the importance of subgraph sampling to prune query-irrelevant edges, helping T-GAP concentrate on the plausible substructure of the input graph.

Finally, we analyze the effect of PGNN by training T-GAP with different numbers of PGNN layers. We find that T-GAP, trained with 1-layer PGNN, performs superior to the model without PGNN by absolute gain of 1% in MRR. However, adding up more layers in PGNN gives only a minor gain, or even aggravates the test set accuracy, mainly owing to early overfitting on the train set.

4.3 Temporal Generalization

We conduct an additional study that measures the performance of T-GAP in generalizing to queries with unseen timestamps. Following Goel et al. [7], we modify ICEWS14 by including all triples except those on 5th, 15th, 25th day of each month in the train set, and creating valid and test sets using only the excluded triples. The performance of T-GAP against the strongest baselines in this dataset are presented in Table 4. In this setting, DE-SimplE and T(NT)ComplEx perform much more similar to each other than in Table 2, while T-GAP performs superior to all baselines. DE-SimplE shows strength in generalizing over time, as it represents each entity as a continuous function over temporal dimension. However, the model is weak when the range of timestamps is large and sparse, as shown for Wikidata in Table 2. Meanwhile, TComplEx and TNTComplEx show fair performance in Wikidata, but poorly infer for unseen timestamps, as they only learn independent embeddings of discrete timestamps. On the contrary to these models, T-GAP not only shows superior performance in all benchmarks but also is robust to unseen timestamps, by accounting for the temporal displacement, not the independent time tokens.

Table 4: Generalization performance over unseen timestamps in ICEWS14. Accounting for relative displacement rather than independent timestamps, T-GAP is the most robust to queries with unseen timestamps.

Model	*MRR*	*Hits@1*	*Hits@3*	*Hits@10*
DE-SimplE	0.434	33.3	49.2	62.4
TComplEx	0.443	34.8	49.2	62.5
TNTComplEx	0.444	34.6	49.4	63.5
T-GAP	**0.483**	**37.2**	**54.6**	**69.0**

4.4 Interpretability

We provide an in-depth analysis on the model's relational reasoning process to give a clear presentation of T-GAP's interpretability. The analysis is twofold: first we discuss the relation between temporal displacements and the type of the query, using the attention distribution over different timestamps. Next, we give a case study in T-GAP's reasoning process for a given query, specifically listing the salient edges that are given high attention values during the path exploration.

4.4.1 Relation Type and Temporal Displacement. Intuitively, the query relation type, and the temporal displacement between relevant events and the query are closely correlated. For a query such as

Table 5: List of predominant edges for a case study. Numbers in the right are the corresponding edge attention assigned to each edge. Predicates in red color carry negative meaning, while predicates in blue color hold positive meaning. We find through the case study that the attention propagation allows T-GAP to fix its misleading focus on sub-optimal entities.

Query	*(North_Korea, threaten, ?, 2014/04/29)*	
1st Step	*(North_Korea, threaten, Japan, 2014/05/12)*	0.057
	(North_Korea, threaten, Japan, 2014/12/18)	0.054
	(North_Korea, make_statement, South_Korea, 2014/04/29)	0.044
	(North_Korea, make_an_appeal, Japan, 2014/10/01)	0.041
	(North_Korea, threaten, South_Korea, 2014/08/01)	0.040
2nd Step	*(South_Korea, threaten, North_Korea, 2014/04/22)*	0.150
	(Japan, release_person, North_Korea, 2014/07/28)	0.078
	(South_Korea, criticize_or_denounce, North_Korea, 2014/04/28)	0.051
	(Japan, express_intent_to_cooperate, North_Korea, 2014/02/28)	0.039
	(South_Korea, make_statement, North_Korea, 2014/04/28)	0.037
3rd Step	*(North_Korea, make_statement, South_Korea, 2014/04/29)*	0.189
	(North_Korea, accuse, South_Korea, 2014/04/15)	0.121
	(North_Korea, criticized_or_denounced_by, South_Korea, 2014/04/28)	0.052
	(North_Korea, deny_responsibility, South_Korea, 2014/04/25)	0.032
	(South_Korea, release_person, North_Korea, 2014/04/14)	0.021
Answer	*South_Korea*	

(PersonX, member_of_sports_team, ?, t_1), events that happened 100 years before t_1 or 100 years after t_1 will highly likely be irrelevant. On the contrary, for a query given as *(NationX, wage_war_against, ?, t_2)*, one might have to consider those events far-off the time of interest. To verify whether T-GAP understands this implicit correlation, we analyze the attention distribution over edges with different temporal displacements, when T-GAP is given input queries with a specific relation type.

The visualization of the distributions for three relation types are presented in Figure 4. For all queries in the test set of WikiData11k with a specific relation type, we visualize the average attention value assigned to edges for each temporal displacement (red bars). We compare this with the original distribution of temporal displacement between query and each edge, counted for all edges reachable in T steps from the head entity v_{query} (blue bars). Remarkably, on the contrary with the original distribution of high variance over wide range of displacement, T-GAP tends to focus most of the attention to edges with specific temporal displacement, depending on the relation type.

For queries with relation *award_won*, the attention distribution is extremely skewed, focusing over 90% of the attention to events with displacement = 0 (i.e. events in the same year with the query). The skewed distribution mainly results from the fact that the majority of the 'award' entities in Wikidata11k are annual awards, such as *Latin Grammy Award*, or *Emmy Award*. The annual property of the candidate entities naturally makes T-GAP to focus on clues such as the nomination of awardees, or significant achievement of awardees in the year of interest.

Next, we test T-GAP for queries with relation *member_of_sports-_team-occurUntil*. In this case, the attention is more evenly distributed than the former case, but slightly biased toward past events. We find that this bias is mainly due to the existence of temporally reciprocal edge (i.e. *member_of_sports_team-occurSince*) in G_{KG}, which is a crucial key in solving the given query. Here, T-GAP sends more than half of the attention value (on average) to an edge that starts from the query head with relation *member_of_sports_team-occurSince*, that happened few years before the time of interest. The inference follows our intuition to look for the last sports club where the player became member of, before the timestamp of the query.

The third case with relation *educated_at-occurSince* is the opposite of the second case. Majority of the attention have been concentrated on events in 1-5 years after the query time, searching for the first event with relation *educated_at-occurUntil*, that happened after the time of interest.

As the analysis suggests, T-GAP discovers important clues for each relation type, adequately accounting for the temporal displacement between the query and related events. The results show that T-GAP not only finds correlation between the displacement and query type in an interpretable way, but also aligns with human intuition when looking for meaningful temporal events.

4.4.2 *Reasoning Process.*

We resort to a case study to provide a detailed view on T-GAP's attention-based decoding. In this study, our model is given an input query *(North_Korea, threaten, ?, 2014/04/29)* in ICEWS14 where the correct answer is *South_Korea*. For each propagation step, we list top-5 edges that received the highest attention value in the step.

The predominant edges and their associated attention values are shown in Table 5. In the first step, T-GAP attends to various events pertinent to *North_Korea*, that mostly include negative predicates against other nations. As seen in the table, the two plausible query-filling candidates are *Japan*, and *South_Korea*. *Japan* receives slightly more attention than *South_Korea*, as it is associated with more relevant facts such as "*North_Korea* threatened *Japan* at May 12th".

In the second step, however, T-GAP discovers additional relevant facts, that could be crucial in answering the given query. As these facts have either *Japan* or *South _Korea* as head entity, they could not be discovered in the first propagation step, which only propagates the attention from the query head *North_Korea*. T-GAP attends to the events *(South_Korea, threaten / criticize_or_denounce,* *North_Korea)* that happened only a few days before our time of interest. These facts imply the strained relationship between the two nations around the query time. Also, T-GAP finds that most of the edges that span from *Japan* to *North_Korea* before/after few months the time of interest, tend to be positive events. As a result, in the last step, T-GAP propagates most of the node attention in *North_Korea* to the events associated with *South_Korea*. Highest attention have been assigned to the relation *make_statement*. Although the relation itself does not hold a negative meaning, in ICEWS14, *make_statement* is typically accompanied by *threaten*, as entities do formally threaten other entities by making statements.

Through the case study, we find that T-GAP leverages the propagation based decoding as a tool to fix its traversal over wrongly-selected entities. Although *Japan* seemed like an optimal answer in the first step, it understands through the second step that the candidate was sub-optimal with respect to the query, propagating the attention assigned to *Japan* back to *North_Korea*. T-GAP fixes its attention propagation in the last step, resulting in a completely different set of attended events compared to the first step. Such an amendment would not have been possible with conventional approaches to path-based inference, which greedily select an optimal entity to traverse at each decoding step.

5 CONCLUSION

In this paper, we propose a novel approach to TKG completion named T-GAP, which explores a query-relevant substructure of TKG with attention propagation. Unlike other embedding-based models, the proposed method effectively gathers useful information from the existing KG, by accounting for the temporal displacement between the query and respective edges. Quantitative results show that T-GAP not only achieves the state-of-the-art performance consistently over all three benchmarks, but also competently generalizes to queries with unseen timestamps. Through extensive analysis, we also show that the propagated attention distribution well serves as an interpretable proxy of T-GAP's reasoning process and aligns well with human intuition.

ACKNOWLEDGMENTS

This work was supported by the National Research Foundation of Korea(NRF) funded by MSIT(2019R1A2C2004990). The Institute of Engineering Research and ICT at Seoul National University provided research facilities for this work. U Kang is the corresponding author. This work is done as a research project inside Seoul National University.

REFERENCES

[1] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Advances in neural information processing systems*. 2787–2795.

[2] Jianlong Chang, Jie Gu, Lingfeng Wang, Gaofeng Meng, Shiming Xiang, and Chunhong Pan. 2018. Structure-aware convolutional neural networks. In *Advances in neural information processing systems*. 11–20.

[3] Rajarshi Das, Shehzaad Dhuliawala, Manzil Zaheer, Luke Vilnis, Ishan Durugkar, Akshay Krishnamurthy, Alex Smola, and Andrew McCallum. 2018. Go for a Walk and Arrive at the Answer: Reasoning Over Paths in Knowledge Bases using Reinforcement Learning. In *International Conference on Learning Representations*.

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

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

[6] Alberto García-Durán, Sebastijan Dumančić, and Mathias Niepert. 2018. Learning Sequence Encoders for Temporal Knowledge Graph Completion. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Brussels, Belgium, 4816–4821. https://www.aclweb.org/anthology/D18-1516

[7] Rishab Goel, Seyed Mehran Kazemi, Marcus Brubaker, and Pascal Poupart. 2019. Diachronic embedding for temporal knowledge graph completion. *arXiv preprint arXiv:1907.03143* (2019).

[8] Zhen Han, Yuyi Wang, Yunpu Ma, Stephan Günnemann, and Volker Tresp. 2020. Graph Hawkes Neural Network for Future Prediction on Temporal Knowledge Graphs. In *Automated Knowledge Base Construction*.

[9] Ziniu Hu, Yuxiao Dong, Kuansan Wang, and Yizhou Sun. 2020. *Heterogeneous Graph Transformer*. Association for Computing Machinery, New York, NY, USA, 2704–2710. https://doi.org/10.1145/3366423.3380027

[10] Inah Jeon, Evangelos E. Papalexakis, Christos Faloutsos, Lee Sael, and U. Kang. 2016. Mining billion-scale tensors: algorithms and discoveries. *VLDB J.* 25, 4 (2016), 519–544.

[11] Tingsong Jiang, Tianyu Liu, Tao Ge, Lei Sha, Baobao Chang, Sujian Li, and Zhifang Sui. 2016. Towards time-aware knowledge graph completion. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*. 1715–1724.

[12] Timothée Lacroix, Guillaume Obozinski, and Nicolas Usunier. 2019. Tensor Decompositions for Temporal Knowledge Base Completion. In *International Conference on Learning Representations*.

[13] Xi Victoria Lin, Richard Socher, and Caiming Xiong. 2018. Multi-Hop Knowledge Graph Reasoning with Reward Shaping. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Brussels, Belgium, 3243–3253.

[14] Yankai Lin, Zhiyua Liu, Huanbo Luan, Maosong Sun, Siwei Rao, and Song Liu. 2015. Modeling Relation Paths for Representation Learning of Knowledge Bases. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Lisbon, Portugal, 705–714.

[15] Zhenghao Liu, Chenyan Xiong, Maosong Sun, and Zhiyuan Liu. 2018. Entity-Duet Neural Ranking: Understanding the Role of Knowledge Graph Semantics in Neural Information Retrieval. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Melbourne, Australia, 2395–2405.

[16] Yunpu Ma, Tresp, Volker, and Erik A Daxberger. 2019. Embedding models for episodic knowledge graphs. *Journal of Web Semantics* 59 (2019), 100490.

[17] Deepak Nathani, Jatin Chauhan, Charu Sharma, and Manohar Kaul. 2019. Learning Attention-based Embeddings for Relation Prediction in Knowledge Graphs. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Florence, Italy. https://www.aclweb.org/anthology/P19-1466

[18] Aldo Pareja, Giacomo Domeniconi, Jie Chen, Tengfei Ma, Toyotaro Suzumura, Hiroki Kanezashi, Tim Kaler, Tao B. Schardl, and Charles E. Leiserson. 2020. EvolveGCN: Evolving Graph Convolutional Networks for Dynamic Graphs. In *Proceedings of the Thirty-Fourth AAAI Conference on Artificial Intelligence*.

[19] Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. 2016. Complex embeddings for simple link prediction. In *International Conference on Machine Learning*. International Conference on Machine Learning (ICML).

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

[21] Xiang Wang, Dingxian Wang, Canran Xu, Xiangnan He, Yixin Cao, and Tat-Seng Chua. 2019. Explainable reasoning over knowledge graphs for recommendation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 5329–5336.

[22] Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014. Knowledge Graph Embedding by Translating on Hyperplanes. In *AAAI*.

[23] Xiaoran Xu, Wei Feng, Yunsheng Jiang, Xiaohui Xie, Zhiqing Sun, and Zhi-Hong Deng. 2019. Dynamically Pruned Message Passing Networks for Large-scale Knowledge Graph Reasoning. In *International Conference on Learning Representations*.

[24] Xiaoran Xu, Songpeng Zu, Chengliang Gao, Yuan Zhang, and Wei Feng. 2018. Modeling Attention Flow on Graphs. *arXiv preprint arXiv:1811.00497* (2018).

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

[26] Yuyu Zhang, Hanjun Dai, Zornitsa Kozareva, Alexander J Smola, and Le Song. 2018. Variational reasoning for question answering with knowledge graph. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

[27] Zhao Zhang, Fuzhen Zhuang, Hengshu Zhu, Zhi-Ping Shi, Hui Xiong, and Qing He. 2020. Relational Graph Neural Network with Hierarchical Attention for Knowledge Graph Completion.. In *AAAI*. 9612–9619.

A ADDITIONAL IMPLEMENTATION DETAIL

Table 6: Additional implementation detail of T-GAP. We report the hyperparameter search bounds and the used configurations, along with the experimental environments.

Computing Infrastructure	Tesla V100 GPU
Search Strategy	Manual Tuning
Best Validation *Hits@1*	52.1 (ICEWS14), 56.8 (ICEWS05-15), 70.7 (Wikidata11k)

Hyperparameter	Search Bound	Experimental Setting
max path length T	*choice*[2, 3, 4]	3
number x of core nodes	*choice*[5, 10, 50, 100]	100 (ICEWS14), 10 (ICEWS05-15), 50 (Wikidata11k)
number y of edges to sample	*choice*[10, 50, 100, 200, 500]	500 (ICEWS14), 100 (ICEWS05-15), 200 (Wikidata11k)
number z of edges to add	*choice*[10, 50, 100, 200, 500]	500 (ICEWS14), 100 (ICEWS05-15), 200 (Wikidata11k)
number K of attention heads	*choice*[3, 4, 5, 6]	5
embedding dimension	100	100
number of epochs	10	10
batch size	*choice*[8, 16, 32]	16 (ICEWS14), 8 (ICEWS05-15, Wikidata11k)
optimizer	*Adam*	*Adam*
learning rate	*loguniform-float*[5e-2, 5e-5]	5e-4
lr scheduler	*reduce_on_plateau*	*reduce_on_plateau*
lr reduction factor	0.1	0.1
gradient clip norm	*uniform-integer[1, 5]*	3 (ICEWS14, ICEWS05-15), 5 (Wikidata11k)

B DATASET STATISTICS

Table 7: Dataset Statistics for ICEWS14, ICEWS05-15, and Wikidata11k. The unit of the time span is year.

Dataset	ICEWS14	ICEWS05-15	Wikidata11k
$\lvert V_{KG} \rvert$	7,128	10,488	11,134
$\lvert R_{KG} \rvert$	230	251	95
$\lvert T_{KG} \rvert$	365	4017	328
$\lvert G_{KG} \rvert$	90,730	479,329	150,079
Time Span	2014	2005-2015	25-2020