Natural Evolution-based Dual-Level Aggregation for Temporal Knowledge Graph Reasoning

Bin Chen[1], **Chunjing Xiao**[2][*], **Fan Zhou**[1,3]
[1]University of Electronic Science and Technology of China, Chengdu, China
[2]Henan University, Kaifeng, China
[3]Kash Institute of Electronics and Information Industry, Kashgar, China
binchen4110@gmail.com, chunjingxiao@gmail.com, fan.zhou@uestc.edu.cn

Abstract

Temporal knowledge graph (TKG) reasoning aims to predict missing facts based on a given history. Most of the existing methods unifiedly model the evolution process of different events and ignore their inherent asynchronous characteristics, resulting in suboptimal performance. To tackle this challenge, we propose a Natural Evolution-based Dual-level Aggregation framework (NEDA) for TKG reasoning. Specifically, we design a natural division strategy to group TKGs into different patches according to the occurrence of a given target entity. Then, we present a dual-level aggregation scheme to extract local representations from information within patches and then aggregate these representations with adaptive weights as the final entity representations. By assigning varying weights to different patches, this aggregation scheme can incorporate the asynchronous characteristics of event evolution for representation computation, thus enhancing prediction performance. Extensive experiments demonstrate the significant improvement of our proposed model.

1 Introduction

Temporal Knowledge Graphs (TKGs) are essential for effectively capturing temporal facts and hold remarkable values in diverse real-world applications, such as information retrieval (Li et al., 2023), recommendation systems (Wang et al., 2018; Cheng et al., 2024), and question answering (Huang et al., 2019; Liu et al., 2022b). Each fact in TKGs is represented as a quadruple (*s, r, o, t*), such as (*Lebron James, join, LA Lakers, 2018*). Due to the significant practical value, reasoning over TKGs has garnered multidisciplinary research interests (Li et al., 2021; Xu et al., 2023; Han et al., 2020).

Reasoning over TKGs primarily has two settings: interpolation and extrapolation. The former intends to infer missing facts that happened in the past. While the extrapolation attempts to predict facts that are likely to occur in the future. In this paper, we focus on predicting future facts (i.e., extrapolation). Given a TKG from timestamp t_0 to t_n, we aim to predict future facts at timestamp t with $t > t_n$, i.e., answer the queries such as (*Lebron James, play for, ?, 2024*).

To answer such queries, existing solutions (Jin et al., 2020; Li et al., 2021; Han et al., 2020) have attempted to learn temporal evolutionary information by modeling structural dependencies and sequential patterns. However, they unifiedly model the evolution process of different events, ignoring evolution discrepancies (i.e., asynchronous characteristics) of events. Whereas, in the real world, event evolution is typically asynchronous. Different events often exhibit distinct life cycles (i.e., starting and ending time) and evolution types (i.e., continuous and discontinuous) (Jiang et al., 2023; Nolting et al., 2023; Yang et al., 2023). The unified processing schemes may incorporate irrelevant information into the reasoning process or overlook critical information, leading to inferior performance.

Considering asynchronous evolution for TKG reasoning is non-trivial and faces two main challenges: (1) For a given query on TKGs, identifying the correlation between the query and each snapshot is challenging. As an event typically evolves across multiple snapshots, each individual snapshot contains only limited topic information. This makes determining their relevance a complex task. However, to capture asynchronous evolution, it is vital to ascertain their correlation and further eliminate unrelated snapshots (e.g., those before the starting time) to accurately answer the query. (2) Current approaches cannot accurately model the evolution of events with long life-cycles, as they fail to efficiently capture long-term historical information. These methods primarily employ GCNs and GRUs to model event evolution, which limits

[*]Corresponding Author: chunjingxiao@gmail.com

them to capturing only a short history window to model structural dependencies and sequential patterns (Li et al., 2021, 2022a). However, short-range historical information is insufficient to encompass the entire evolution process of events with long life-cycles (Wang et al., 2023; Liang et al., 2022). Therefore, capturing long-range historical information is radical for effectively modeling the evolution patterns of events with varying life-cycles.

To address these issues, we propose a Natural Evolution-based Dual-level Aggregation framework (NEDA) for TKG reasoning, which can efficiently capture the asynchronous evolution characteristics of events to boost reasoning performance. This framework primarily involves the *patch construction*, which aims to group TKGs into different patches, as well as the *first-level aggregation* (i.e., intra-patch encoder) and the *second-level aggregation* (i.e., inter-patch attention), both of which intend to compute entity representations by combining the intra-patch and inter-patch information with adaptive weights.

To summarize, the major contributions can be listed as follows:

- We propose a Natural Evolution-based Dual-level Aggregation framework (NEDA) for TKG reasoning. To the best of our knowledge, we are the first to consider asynchronous characteristics of event evolution for TKG reasoning.
- We design a natural division strategy and a dual-level aggregation scheme to hierarchically aggregate event information for entity representation computation, which can efficiently capture the asynchronous evolution characteristics and long-range dependencies.
- We validate the effectiveness of NEDA on six TKG datasets for TKG reasoning and achieve state-of-the-art performance.

2 Preliminaries

In this section, we introduce the essential background of TKGs and formally formulate the task of TKG reasoning under the extrapolation setting.
Temporal Knowledge Graph. Let E and R represent the set of entities and relations. A quadruple (s, r, o, t) denotes a subject entity $s \in E$ has a relation $r \in R$ with an object entity $o \in E$ at timestamp t. For each quadruple (s, r, o, t), an inverse relation quadruple (o, r^{-1}, s, t) is often added to the dataset. All quadruples (a.k.a. facts) that occur within a period $(t - \Delta t, t]$, denoted as t for simplicity, constitute a knowledge graph G_t. A TKG, denoted as $G = \{G_0, G_1, \cdots, G_t\}$, consists of a series of knowledge graphs in chronological order.
Temporal Knowledge Graph Reasoning. TKG reasoning mainly falls into two settings: interpolation and extrapolation. The former aims to predict the missing historical facts, while the latter concentrates on predicting future events based on given historical facts. TKG reasoning under the extrapolation setting has two sub-tasks: entity prediction and relation prediction. In the context of extrapolation setting, entity prediction aims to infer the missing *object* entity in quadruple $(s, r, ?, t + 1)$ or *subject* entity in quadruple $(?, r, o, t + 1)$ based the given TKG i.e., $G = \{G_1, G_2, \ldots, G_t\}$. Similarly, relation prediction tries to predict the missing relation in quadruple $(s, ?, o, t + 1)$. Note that the target entity refers to the entity known in the given query. For example, the given *subject* entity in query $(s, r, ?, t + 1)$ or the given *object* entity in query $(?, r, o, t+1)$. In this work, we mainly focus on the task of entity prediction under the extrapolation setting.

3 Method

In this section, we elaborate our proposed NEDA for TKG reasoning. As illustrated in Figure 1, NEDA consists of three main modules: (1) *Patch Construction*, which divides the historical TKG for a given query into various explicit and implicit patches according to the occurrence of the target entity in this query, and further extracts connected subgraphs centered at the target entity; (2) *Intra-Patch Encoder* (i.e., the first-level aggregation), which extracts the local entity representations by exploring structural and temporal dependencies within patches. (3) *Inter-patch Attention* (i.e., the second-level aggregation), which aggregates local representations across patches to generate final entity representations. Finally, the obtained representations are adopted to compute the prediction scores for each query.

3.1 Patch Construction

Explicit and Implicit Patch Division. To capture asynchronous nature of event evolution, we divide the historical TKG for each query into different patches based on the occurrence of the target entity. Specifically, consecutive snapshots where the target entity appears are grouped into *explicit patches*, as

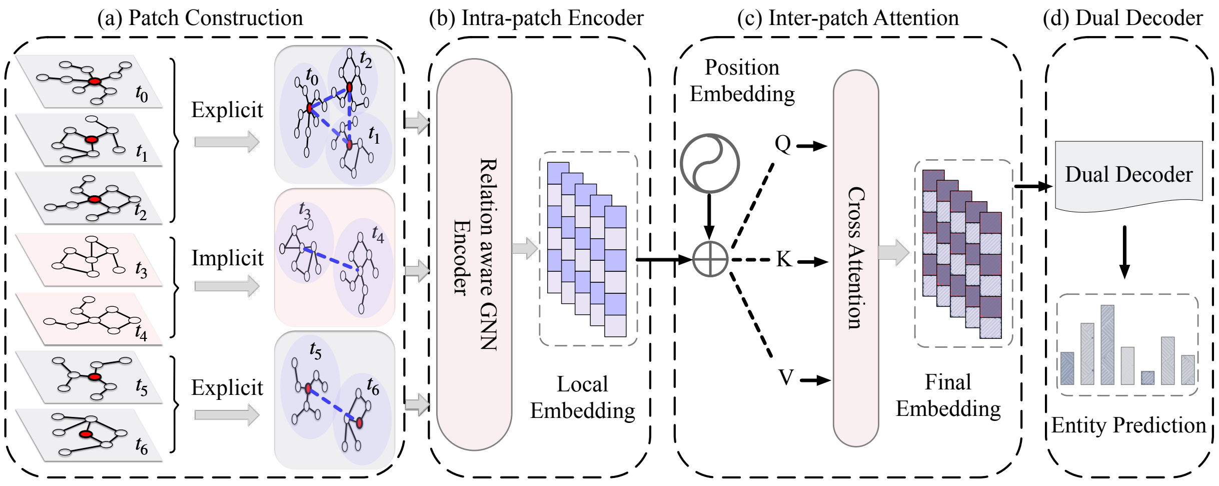

Figure 1: An illustration of NEDA.

the entity explicitly evolves over time in the patch. Conversely, consecutive snapshots where the entity does not appear form *implicit patches*. In implicit patches, even though the target entity is absent, its neighbors continue to evolve over time, impacting the target entity when it reappears in the future.

An example of patch division is illustrated in Figure 1 (a). As shown, given a query $q = (s, r, ?, t_7)$ where entity s appears at timestamps $\{t_0, t_1, t_2, t_5, t_6\}$, we group the snapshots at $\{t_0, t_1, t_2\}$ and $\{t_5, t_6\}$ into two explicit patches. While, the snapshots at $\{t_3, t_4\}$ are grouped into an implicit patch.

This natural division strategy can facilitate the consideration of asynchronous evolution characteristics to enhance reasoning performance. In each explicit patch generated by this strategy, since all its snapshots contain the target entity, they can be regarded as a cohesive segment that emphasizes specific topics related to the target entity. Compared to a single snapshot, the topics can be easily extracted from the explicit patch because such cohesive segment contains more comprehensive topic information. Consequently, the topic relevance between the explicit patch and the query can be effectively determined. This allows for the elimination of unrelated patches and the highlighting of relevant ones by adjusting the patch weights according to topic relevance, thus advocating learned representations and reasoning results.

Subgraph Extraction. Given that only nearby neighbors substantially impact the evolution of the target entity (Liang et al., 2022), we extract the k-hop neighbors of the target entity within each snapshot to form a k-hop subgraph for entity representation computation. For explicit patches, since the target entity consistently appears throughout the snapshots within the patch, this phase can be considered as a cohesive segment. Consequently, we transform the subgraphs within the patch into a *connected subgraph*, in which subgraphs at different timestamps are connected into a whole graph by linking the target entity among them. For implicit patches, where the target entity is absent, we utilize the time-aware exponentially weighted sampling strategy (Han et al., 2020) to sample n important neighbors of the target entity to form subgraphs. Subsequently, we link the sampled neighbors to generate *connected subgraphs*.

3.2 Intra-patch Encoder

Here, we extract local representations of entities by exploiting complex structural and temporal dependencies within patches. For explicit patches, we directly compute the entity representation. For implicit patches, we calculate the representations of the neighboring entities, which serve as the context for evolution, thereby enriching the representation of the target entity in the subsequent explicit patch. In other words, the learned neighbor representations in the current implicit patch are used to initialize themselves in the next explicit patch.

To effectively exploit continuous evolution patterns within patches, we design a relation-aware GNN encoder to extract structural and temporal dependencies in patches. This encoder first extracts structural embedding by capturing the complex structural dependencies among concurrent events within the k-hop subgraph at a timestamp, and then models the temporal correlations among target entities across various timestamps within the patch.

Structural Dependency. Given a connected subgraph G_p obtained by linking the k-hop subgraph sequence $\{G_1^s, G_2^s, \cdots, G_m^s\}$, we extract the structural dependencies in each G_i^s via the relational graph attention network (Chen et al., 2024). Supposing r_i is the representation of relation r in G_i^s, the structural-dependency aggregator is defined as follows:

$$\alpha_{s,o}^{i,l} = \frac{\exp\left(\mathbf{a}^{\mathrm{T}}\mathrm{g}\left(\mathrm{W}_1^l\left[\mathrm{h}_{s,i}^l, \mathrm{h}_{o,i}^l, \mathrm{r}_i\right]\right)\right)}{\sum\limits_{(o',r')\in N_{G_i^s}} \exp\left(\mathbf{a}^{\mathrm{T}}\mathrm{g}\left(\mathrm{W}_1^l\left[\mathrm{h}_{s,i}^l, \mathrm{h}_{o',i}^l, \mathrm{r}_i'\right]\right)\right)},$$

$$\mathrm{h}_{s,i}^{l+1} = \sum_{(o,r)\in N_{G_i^s}} \alpha_{s,o}^{i,l}(\mathrm{h}_{o,i}^l + \mathrm{r}_i) + \mathrm{W}_2^l\mathrm{h}_{s,i}^l, \quad (1)$$

where $\mathrm{h}_{s,i}^l$ and $\mathrm{h}_{o,i}^l$ denote the embeddings of entities s and o at the l-th layer, $\alpha_{s,o}^{i,l}$ represents the influence coefficient between entity s and o via relation r at the l-th layer, $N_{G_i^s}$ is a set consisting of the neighbors of s and their corresponding relationship combinations, $\mathrm{W}_1^l, \mathrm{W}_2^l$ and $\mathbf{a}$ denote learnable weights, and $\mathrm{g}(\cdot)$ is the Leaky ReLU function. $\mathrm{h}_{o,i}^l + \mathrm{r}_i$ implies the translational property between the subject entity and the corresponding object entity via the relation r at the l-th layer. After L-layer aggregation, we can obtain the structural embedding $\mathrm{h}_{s,i}^L$ of the target entity for G_i^s, which captures the structural dependencies within patches.

Temporal Correlation. To capture the temporal correlation across different k-top subgraphs within the patch, we link the structural embeddings (e.g., $\mathrm{h}_{s,i}^L$) in different subgraphs computed by Equation 1, and then adopt a GNN to aggregate them to produce the local representation of the target entity. Supposing $\mathrm{h}_{s,m}^L$ and $\mathrm{h}_{s,j}^L$ are the structural embeddings of target entity s in G_m^s and G_j^s, respectively, the local representation is computed by the GNN encoder :

$$\mathrm{h}_{s,m}^{L+1} = \sigma\left(\sum_{j\in[1,m-1]} \frac{1}{m-1}\mathrm{W}_3\mathrm{h}_{s,j}^L + \mathrm{W}_4\mathrm{h}_{s,m}^L\right), \quad (2)$$

where m is the number of subgraphs in the patch, W_3 and W_4 denote learnable weights, and $\sigma(\cdot)$ is the sigmoid activation function.

3.3 Inter-patch Attention

Having the local representations computed by considering the information within patches, we here further aggregate them by considering correlation among the patches to generate the final entity representation. Since different explicit patches are separated by implicit patches, each explicit patch tends to have relatively independent topics. Correspondingly, different explicit patches may have different correlations with the target query in terms of topics. Hence, we assign different weights to the local representations of the entity during the aggregation process. This can efficiently capture the asynchronous nature of event evolution, i.e., the impact of patches preceding the start time of the query event can be minimized by assigning them lower or even zero weights, while emphasizing the relevant patches by assigning them higher weights. To this end, we design an inter-patch attention module to assign adaptive weights to the local representations and combine them as the final representation.

In particular, assuming there are M explicit patches for the query $(s, r, ?, t_n)$, we stack the local representations $\mathrm{h}_{s,m}^{L+1} \in \mathbb{R}^d$ from all explicit patches computed by Equation 2 to form the input embedding matrix $\mathbf{P} \in \mathbb{R}^{M\times d}$. Then, we add a learnable position embedding $\mathbf{P}_s \in \mathbb{R}^{M\times d}$ for the input embedding matrix $\mathbf{P}$. This operation enables the model to preserve the sequential information of the explicit evolution patches (Vaswani et al., 2017). Therefore, we can obtain the input embedding as $\mathbf{X} = \mathbf{P} + \mathbf{P}_s \in \mathbb{R}^{M\times d}$.

We then apply a cross-attention operation on $\mathbf{X}$ to model distant and critical asynchronous temporal dependencies, and further adaptively aggregate entity representations across patches to obtain the final representation. Specifically, we perform trainable linear transformations on $\mathbf{X}$ to derive the key and value matrices for the attention mechanism, denoted as $\mathbf{K}$ and $\mathbf{V}$, respectively. Further, we employ a trainable query matrix $\mathbf{Q} \in \mathbb{R}^{1\times d}$ to integrate the comprehensive context of all the patches. Finally, we compute the cross-attention among $\mathbf{Q}, \mathbf{K}$, and $\mathbf{V}$ to model the global temporal dependencies:

$$\mathrm{h}_s = \mathrm{Softmax}\left(\frac{\mathbf{QK}^T}{\sqrt{d}}\right)\mathbf{V}, \quad (3)$$

where h_s is the final representation of the target entity s. This representation can involve asynchronous characteristics by assigning different weights to various patches and capture long-range history information by hierarchically aggregate the information of all the patches.

In summary, the intra-patch encoder extracts local representations of entities by aggregating com-

Dataset	#Entities	#Relation	#Training	#Validation	#Test	#Granularity	#Snapshots
ICEWS18	23,033	256	373,018	45,995	49,545	24 hours	304
ICEWS14	7,128	230	74,845	8,514	7,371	1 day	365
ICEWS05-15	10,094	251	368,868	46,302	46,159	24 hours	4017
GDELT	7,691	240	1,734,399	238,765	305,241	15 mins	2751
ICEWS22	23,530	259	435,049	54,222	54,499	1 day	365
ICEWS23	13,076	233	112,969	14,340	14,297	1 day	100

Table 1: Statistics of the datasets.

plex structural and temporal information within patches. Then the inter-patch attention aggregates the local representations with adaptive weights to generate the final representations, which will be adopted for predictions. This dual-level aggregation scheme can capture asynchronous characteristics of event evolution by utilizing our designed inter-patch attention to assign adaptive weights to different patches. The higher the correlation between the patch and the query, the greater the assigned weight will be. Also, this strategy can capture long-range history information by aggregating information from all the patches.

3.4 Event Prediction and Learning Objective

Based on the extracted final representations, we further compute the prediction scores for each query $(s, r, ?, t_n)$ and present the learning objective for model training.

Prediction Scores. We introduce the dual decoder (Li et al., 2022a), which performs predictions from the perspective of both the repeated and non-repeated patterns of events, to obtain the prediction scores for each query $(s, r, ?, t_n)$ based on the learnable representations. Formally, the probability of interaction between subject s and object o under the relation r at timestamp t_n can be calculated as follows:

$$\mathbf{p}\left(o|s, r, t_n\right) = \Phi\left(\mathrm{h}_s, \mathrm{r}, \mathrm{h}_o\right), \tag{4}$$

where $\Phi(\cdot)$ is the dual decoder, h_s and h_o are the final representations of subject s and object o, respectively, and r denotes the representation of r.

Learning Objective. The cross-entropy function is employed to calculate the loss during the training process. The objective function is formulated as:

$$\mathcal{L} = -\sum_{t_n \in T} \sum_{i \in G_{t_n}} \sum_{j \in E} o_i^{t_n} \ln \mathbf{p}(y_i^j \mid s, p, t_n), \tag{5}$$

where $o_i^{t_n}$ represents the i-th ground truth object entity for queries in snapshot G_{t_n}, and $\mathbf{p}(y_i^j \mid s, p, t_n)$ is the combined probability value of j-th object entity for the i-th query in G_{t_n}.

4 Experiments

In this section, we perform experiments on six TKG datasets to evaluate the performance of our NEDA. We aim to answer the following questions through experiments.

- *Q1: Superiority.* How does NEDA perform compared with the state-of-the-art baselines?
- *Q2: Effectiveness.* How do the different components affect the NEDA's performance?
- *Q3: Scalability.* How does NEDA perform when there is limited training data available?
- *Q4: Sensitivity.* How does the performance of NEDA fluctuate with different hyperparameters?
- *Q5: Asynchrony.* How does NEDA capture the inherent asynchronous evolution characteristics of events?

4.1 Basic Setting

Datasets & Evaluation Metrics. We conduct experiments on four commonly used TKG datasets including ICEWS14 (Garcia-Duran et al., 2018), ICEWS18 (Jin et al., 2020), ICEWS05-15 (Garcia-Duran et al., 2018), and GDLET (Leetaru and Schrodt, 2013), as well as two benchmark datasets we newly collected, ICEWS22 and ICEWS23. The two datasets are obtained from the ICEWS database[1] and sorted out using the typical processing method in the work (Jin et al., 2020). Table 1 presents the statistic results. In addition, we utilize the widely used metrics MRR and Hits@{1,3,10} to evaluate the performance. MRR represents the average reciprocal values of the ranks assigned to the true entity candidates across all queries, and Hits@{1,3,10} represents the proportion of times the true entity appears in the top 1,3 and 10 ranking candidates. Following the works (Han et al., 2020; Li et al., 2022a; Zhang et al., 2023), we use the time-aware filtered setting to report the experimental results.

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

Method	ICEWS14				ICEWS18				ICEWS05-15			
	MRR	H@1	H@3	H@10	MRR	H@1	H@3	H@10	MRR	H@1	H@3	H@10
RE-Net (2020)	39.86	30.11	44.02	58.21	29.78	19.73	32.55	48.46	43.67	33.55	48.83	62.72
xERTE (2020)	40.79	32.70	45.67	57.30	29.31	21.03	33.51	46.48	46.62	37.84	52.31	63.92
CyGNet (2021)	37.65	27.43	42.63	57.90	27.12	17.21	30.97	46.85	40.42	29.44	46.06	61.60
RE-GCN (2021)	42.00	31.63	47.20	61.65	32.62	22.39	36.79	52.68	48.03	37.33	53.90	68.51
TITer (2021)	41.73	32.74	46.46	58.44	29.98	22.05	33.46	44.83	47.60	38.29	52.74	64.86
TANGO (2021)	36.94	27.60	41.10	54.41	28.97	19.51	32.61	47.51	42.86	32.72	48.14	62.34
CEN (2022)	42.20	32.08	47.46	61.31	31.50	21.70	35.44	50.59	48.71	38.30	54.39	68.68
EvoKG (2022)	27.18	-	30.84	47.67	29.28	-	33.94	50.09	-	-	-	-
TiRGN (2022)	44.33	33.73	49.85	64.46	33.58	23.10	37.90	54.20	49.84	39.07	55.75	70.11
CENET (2023)	38.58	30.18	41.79	55.20	31.43	23.47	34.05	47.27	46.34	38.14	49.89	62.53
GenTKG (2023)	-	36.85	47.95	53.50	-	24.25	37.25	42.10	-	-	-	-
RETIA (2023)	43.00	32.47	48.01	63.64	32.10	21.96	36.18	51.95	44.37	34.02	49.64	64.42
L^{2}TKG (2023)	47.40	35.36	-	71.05	33.36	22.15	-	55.04	57.43	41.86	-	80.69
RPC (2023)	44.55	34.87	49.80	65.08	34.91	24.34	38.74	55.89	51.14	39.47	57.11	71.75
NEDA (Ours)	**50.51**	**39.42**	**56.54**	**73.46**	**36.84**	**25.55**	**41.70**	**59.16**	**60.19**	**46.23**	**63.75**	**82.76**

Table 2: Performance under time-aware metrics (in percentage) on ICEWS14, ICEWS18, and ICEWS05-15.

Method	GDELT				ICEWS22				ICEWS23			
	MRR	H@1	H@3	H@10	MRR	H@1	H@3	H@10	MRR	H@1	H@3	H@10
RE-Net (2020)	19.55	12.38	20.80	34.00	30.06	20.25	33.86	50.37	27.52	18.36	30.81	45.25
xERTE (2020)	19.45	11.92	20.84	34.18	-	-	-	-	-	-	-	-
CyGNet (2021)	20.22	12.35	21.66	35.82	26.07	16.25	29.90	45.52	24.79	15.53	28.42	42.81
RE-GCN (2021)	19.69	12.46	20.93	33.81	34.69	24.13	39.03	55.63	33.68	23.53	37.86	53.78
TITer (2021)	18.19	11.52	19.20	31.00	31.73	23.94	35.32	46.19	30.88	23.09	34.36	45.32
TANGO (2021)	19.66	12.50	20.93	33.55	30.13	20.48	33.89	49.10	28.02	19.23	31.23	45.20
CEN (2022)	21.17	13.44	22.71	36.40	34.06	23.79	38.38	54.19	32.37	22.76	36.48	51.01
EvoKG (2022)	19.28	-	20.55	34.44	-	-	-	-	-	-	-	-
TiRGN (2022)	21.67	13.63	23.27	37.60	35.86	24.96	40.66	57.36	34.90	24.52	39.27	55.06
CENET (2023)	-	-	-	-	33.52	25.26	36.26	49.61	30.07	22.20	32.75	45.46
GenTKG (2023)	-	13.90	22.55	30.45	-	-	-	-	-	-	-	-
RETIA (2023)	-	-	-	-	-	-	-	-	-	-	-	-
L^{2}TKG (2023)	20.53	12.89	-	35.83	-	-	-	-	-	-	-	-
RPC (2023)	22.41	14.42	24.36	38.33	-	-	-	-	-	-	-	-
NEDA (Ours)	**23.07**	**14.71**	**25.03**	**40.12**	**39.23**	**27.58**	**44.59**	**62.12**	**38.06**	**26.94**	**43.05**	**59.71**

Table 3: Performance under time-aware metrics (in percentage) on GDELT, ICEWS22, and ICEWS23.

Baseline Methods. We compare our method with the following fourteen baselines: RE-Net (Jin et al., 2020), xERTE (Han et al., 2020), CyGNet (Zhu et al., 2021), RE-GCN (Li et al., 2021), TITer (Sun et al., 2021), TANGO (Han et al., 2021), CEN (Li et al., 2022b), TiRGN (Li et al., 2022a), EvoKG (Park et al., 2022), CENET (Xu et al., 2023), RETIA (Liu et al., 2023), L^{2}TKG (Zhang et al., 2023), RPC (Liang et al., 2023), and GenTKG (Liao et al., 2023). For datasets ICEWS18, ICEWS14, ICEWS05-15, and GDELT, the baseline results are obtained from the previous papers. For the newly collected ICEWS22 and ICEWS23, we select the optimal hyper-parameters by performing a grid search on all models.

Implementation Details. NEDA is implemented based on *Pytorch* (Paszke et al., 2019) and *DGL* (Wang, 2019) with the Adam optimizer (Kingma and Ba, 2014) and learning rate of $1e^{-3}$. For all the datasets, the embedding dimension d is set to 200, and both L and k are set to 2. The optimal n is set to 4, 4, 11, 8, 4, and 4 for ICEWS14, ICEWS18, ICEWS05-15, GDELT, ICEWS22 and ICEWS23, respectively. The parameter settings of the dual decoder are the same as those in the work (Li et al., 2022a). To ensure statistical significance, we report the mean results of five times experiments. The experiments are run on a machine with an NVIDIA GeForce RTX 3090 GPU and an Intel i7-13700KF CPU.

4.2 Performance Comparison (RQ1)

Table 2 and Table 3 show the comparison results on the six TKG datasets, where the best performance is highlighted in boldface and the second-best is underlined. We have the following observations. First, NEDA significantly outperforms all the baselines in both MRR and Hits@{1,3,10}, which confirms the effectiveness of considering asynchronous characteristics for TKG reasoning. For example, compared to the second-best performances, for the ICEWS datasets, NEDA achieves on average about 5.6%, 8.9%, and 3.9% performance improvements on MRR, Hits@1, and Hits@10, respectively. Second, the performance improvement of our method on the GDELT dataset is smaller compared to the ICEWS datasets. This can be attributed to the fact that GDELT has more facts and corresponding topics per patch on average compared to other datasets. This makes it more difficult for our NEDA to summarize the most important and relevant topics in the patch.

4.3 Ablation Study (RQ2)

Here, we conduct ablation study to assess the effectiveness of different components in NEDA. Table 4 illustrates the distinct impact attributed to each component. As shown, incorporating implicit patches is essential for computing the target entity's representation. This emphasizes the importance of capturing the evolution of the surrounding environment, thereby enhancing the model's prediction accuracy. Constructing the connected subgraph within each patch facilitates the extraction of complex structural dependencies and temporal correlations across different timestamps. Besides, inter-patch attention with position embedding is particularly effective at capturing distant asynchronous temporal dependencies, and adaptively aggregating the local representation of each patch by distinguishing the global evolution characteristics and patterns. Also, the design of the relation-aware GNN encoder significantly enhances information propagation within the connected subgraph in a patch. It effectively exploits the structural and temporal dependencies of events throughout their continuous evolution by identifying the crucial neighboring entities and timestamps.

4.4 Performance under Limited Training Data (RQ3)

Figure 2 reports the performance of our proposed NEDA compared to the leading baseline, TiRGN, with varying amounts of training data. The results indicate that NEDA consistently outperforms TiRGN across various training data sizes. This advantage persists even when data is severely limited, with only 10% or 20% of the data available for training. By incorporating the asynchronous characteristics of event evolution, NEDA successfully identifies the unique evolution patterns and characteristics of each event. This capability holds even with limited training data, enabling NEDA to consistently deliver superior performance.

Model	ICE14	ICE18	ICE05-15	GDELT	ICE22	ICE23
w/o IP	48.21	34.83	58.25	21.12	37.64	36.39
w/o CG	49.82	35.74	59.11	22.32	38.56	37.05
w/o PE	50.18	36.09	59.72	22.79	38.90	37.49
w/o Inter	48.97	35.17	58.39	21.83	38.01	36.53
w/o RGNN	47.55	34.01	57.31	20.63	36.84	35.81
NEDA	**50.51**	**36.84**	**60.19**	**23.07**	**39.23**	**38.06**

Table 4: Results of ablation studies with time-aware MRR. w/o IP, w/o CG, w/o PE, w/o Inter, and w/o RGNN represent removing the implicit patches, connected subgraph, position embedding, inter-patch attention, and relation-aware GNN encoder, respectively.

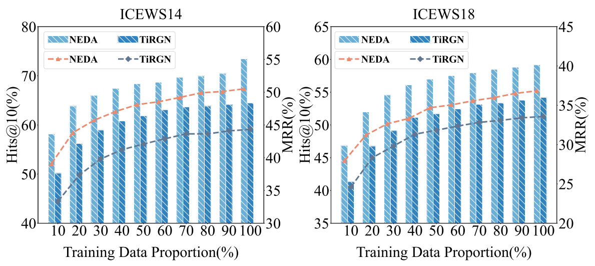

Figure 2: The impact of training data. The lines represent MRR, and the bars denote Hits@10.

4.5 Sensitivity Analysis (RQ4)

We here investigate the influence of two crucial hyperparameters: the layer number of the relational graph attention network L and the number of neighbors extracted in the implicit patch n on the ICEWS14, ICEWS18, and GDELT datasets.

The sensitivity analysis results are reported in Figure 3. Detailed findings are elucidated below: **Hyperparameter** L controls the depth of neighborhood information extraction for structural dependencies. The experimental results with varying L reveal that the performance of NEDA initially improves as the layer number L increases, but it experiences a sharp decline when L becomes excessively large. This phenomenon can be attributed to the fact that an initial increase in L adequately extracts structural dependencies within patches, but an overly large L results in information from more distant neighbors being disseminated to the target entity, thereby introducing more noise.
Hyperparameter n regulates the strength of im-

plicit evolution for prediction. As shown in Figure 3, a smaller n results in the loss of implicit evolution information, whereas a larger n introduces excessive implicit evolution information, overshadowing the more critical explicit evolution information. Therefore, selecting an appropriate value for n is crucial for the effectiveness of our model.

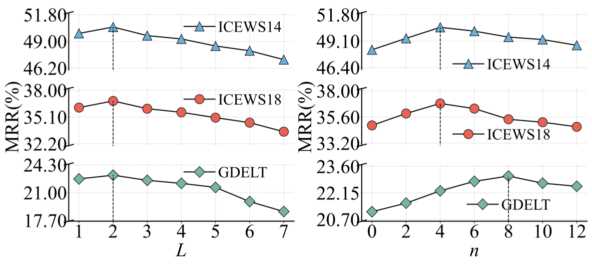

Figure 3: Sensitivity analysis of hyper-parameter L and n on ICEWS14, ICEWS18, and GDELT.

4.6 Case Study (RQ5)

Here, we inspect how our proposed dual-level aggregation scheme capture the asynchronous evolution characteristics of events. As shown in Table 5, we select an event query (*Rishi Sunak, hreaten, United Kingdom, 2022-12-31*) from the test set in ICEWS22 as the prediction case, and we extract the top 3 influential patches assigned the highest weights by NEDA. We can observe that the historical events in these patches predominantly involve interactions between Rishi Sunak and the United Kingdom. As expected, these influential patches and their associated events are highly relevant to the query and play a critical role in the prediction. This demonstrates that by identifying and assigning the highest weights to the most relevant historical event patches, the proposed dual-level aggregation scheme can effectively capture the asynchronous evolution characteristics of the event query.

5 Related Work

Depending on the timing of predicted events, TKG reasoning models can be divided into two settings: interpolation and extrapolation (Liang et al., 2022). **TKG reasoning under interpolation.** The task in the interpolation setting aims to infer missing facts from the past (Xiong et al., 2022; Bai et al., 2021; Wu et al., 2020). For example, TTransE (Jiang et al., 2016) integrates temporal information by treating both relations and time as translations of entities, whereas HyTE (Dasgupta et al., 2018) associates each timestamp with specific hyperplanes. ChronoR (Sadeghian et al., 2021) combines relation and temporal embeddings to create a comprehensive rotational embedding, which is then applied to the final entity representation. TILP (Xiong et al., 2022) introduces a constrained random walk mechanism and integrates an array of temporal operators into the learning framework.

TKG reasoning under extrapolation. The extrapolation methods try to predict future events (Li et al., 2021; Lee et al., 2023; Zhu et al., 2021). Following the work (Li et al., 2022c), these approaches can be divided into two main categories according to the historical structural information used for event prediction: candidate-based and query-based methods.

Candidate-based methods try to utilize the historical facts of all candidate entities without considering the query during the encoding process. The query information is incorporated only in the decoding stage (Li et al., 2021, 2022a; Zhang et al., 2023). For instance, RE-GCN (Li et al., 2021) learns the evolutionary representations for entities and relations by incorporating the gate recurrent unit to extract sequential patterns and taking externally injected static entity attributes into consideration. TiRGN (Li et al., 2022a) learns sequential, repetitive, and periodic historical facts from local and global perspectives, respectively. L^2TKG (Zhang et al., 2023) exploits the latent relation between entities and learns latent relational graphs based on similarities between entities.

Query-based methods focus on modeling the query-specific historical dependencies for event prediction (Zhu et al., 2021; Xu et al., 2023; Liu et al., 2022a). Among them, CyGNet (Zhu et al., 2021) calculates the occurrence frequencies of events to narrow the candidate entities when making predictions, while CENET (Xu et al., 2023) utilizes contrastive learning to identify whether an event has occurred before. DA-Net (Liu et al., 2022a) tries to capture distributed attention on repeated facts across various historical timestamps to better understand future events. In addition, some recent approaches attempt to utilize the powerful reasoning capabilities of LLMs for TKG reasoning (Liao et al., 2023; Luo et al., 2024). For example, GenTKG (Liao et al., 2023) formulates a generative forecasting framework with LLMs, and ICL (Lee et al., 2023) uses in-context learning to the potential of LLMs for TKG reasoning.

2022.12.31	Rishi Sunak	Threaten	United Kingdom
2022.12.24	Daily Mail Rishi Sunak	Criticize or denounce Make statement	Rishi Sunak Ministry (United Kingdom)
2022.12.05 – 2022.12.21	Citizen (United Kingdom) Citizen (United Kingdom) Daily Mail United Kingdom Rishi Sunak	Accuse Reject Criticize or denounce Consult Consult	Rishi Sunak Rishi Sunak Rishi Sunak Rishi Sunak United Kingdom
2022.11.27 – 2022.12.02	Rishi Sunak Presidential Family (United Kingdom) Rishi Sunak	Make statement Consult Make statement	United Kingdom Rishi Sunak United Kingdom

Table 5: Case study on the dual-level aggregation scheme on ICEWS22.

6 Conclusions

This paper proposes a Natural Evolution-based Dual-level Aggregation framework (NEDA) to consider the inherent asynchronous evolution characteristics for TKG reasoning. A natural division strategy is presented to split TKGs into different patches according to the occurrence of a given target entity. Besides, a dual-level aggregation scheme is designed to extract local representations from information within patches and further aggregate them with adaptive weights to generate the final entity representations. This hierarchical aggregation can efficiently capture the event-specific asynchronous characteristics and long-range history information to enhance reasoning performance. Experimental results on six benchmarks demonstrate the superiority of NEDA in TKG reasoning.

Limitations

While our proposed NEDA demonstrates superior performance compared to the state-of-the-art baselines, it still has some limitations. Firstly, unlike the baseline models that predict all queries in a snapshot at once, our NEDA predicts each query separately, leading to additional overhead. Secondly, our model only considers the most dominant topic within a patch, ignoring other minor topics. Performance could potentially be enhanced by considering multiple topics within a single patch.

Acknowledgements

This work was supported by National Natural Science Foundation of China (Grant No.62176043 and No.62072077), and Kashgar Science and Technology Bureau (Grant No.KS2023025).

References

Luyi Bai, Wenting Yu, Mingzhuo Chen, and Xiangnan Ma. 2021. Multi-hop reasoning over paths in temporal knowledge graphs using reinforcement learning. *Applied Soft Computing*, 103:107144.

Bin Chen, Kai Yang, Wenxin Tai, Zhangtao Cheng, Leyuan Liu, Ting Zhong, and Fan Zhou. 2024. Interpreting temporal knowledge graph reasoning (student abstract). In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 23451–23453.

Zhangtao Cheng, Jienan Zhang, Xovee Xu, Goce Trajcevski, Ting Zhong, and Fan Zhou. 2024. Retrieval-augmented hypergraph for multimodal social media popularity prediction. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 445–455.

Shib Sankar Dasgupta, Swayambhu Nath Ray, 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*, pages 2001–2011.

Alberto Garcia-Duran, Sebastijan Dumančić, and Mathias Niepert. 2018. Learning sequence encoders for temporal knowledge graph completion. In *EMNLP*, pages 4816–4821.

Zhen Han, Peng Chen, Yunpu Ma, and Volker Tresp. 2020. Explainable subgraph reasoning for forecasting on temporal knowledge graphs. In *International Conference on Learning Representations*.

Zhen Han, Zifeng Ding, Yunpu Ma, Yujia Gu, and Volker Tresp. 2021. Learning neural ordinary equations for forecasting future links on temporal knowledge graphs. In *Proceedings of the 2021 conference on empirical methods in natural language processing*, pages 8352–8364.

Xiao Huang, Jingyuan Zhang, Dingcheng Li, and Ping Li. 2019. Knowledge graph embedding based question answering. In *WSDM*, pages 105–113.

Tingsong Jiang, Tianyu Liu, Tao Ge, Lei Sha, Baobao Chang, Sujian Li, and Zhifang Sui. 2016. Towards time-aware knowledge graph completion. In *COLING*, pages 1715–1724.

Xuhui Jiang, Chengjin Xu, Yinghan Shen, Xun Sun, Lumingyuan Tang, Saizhuo Wang, Zhongwu Chen, Yuanzhuo Wang, and Jian Guo. 2023. On the evolution of knowledge graphs: A survey and perspective. *arXiv preprint arXiv:2310.04835*.

Woojeong Jin, Meng Qu, Xisen Jin, and Xiang Ren. 2020. Recurrent event network: Autoregressive structure inferenceover temporal knowledge graphs. In *EMNLP*, pages 6669–6683.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Dong-Ho Lee, Kian Ahrabian, Woojeong Jin, Fred Morstatter, and Jay Pujara. 2023. Temporal knowledge graph forecasting without knowledge using in-context learning. In *The 2023 Conference on Empirical Methods in Natural Language Processing*.

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

Qian Li, Shu Guo, Yangyifei Luo, Cheng Ji, Lihong Wang, Jiawei Sheng, and Jianxin Li. 2023. Attribute-consistent knowledge graph representation learning for multi-modal entity alignment. In *The Web Conference*, pages 2499–2508.

Yujia Li, Shiliang Sun, and Jing Zhao. 2022a. Tirgn: time-guided recurrent graph network with local-global historical patterns for temporal knowledge graph reasoning. In *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI 2022, Vienna, Austria, 23-29 July 2022*, pages 2152–2158. ijcai. org.

Zixuan Li, Saiping Guan, Xiaolong Jin, Weihua Peng, Yajuan Lyu, Yong Zhu, Long Bai, Wei Li, Jiafeng Guo, and Xueqi Cheng. 2022b. Complex evolutional pattern learning for temporal knowledge graph reasoning. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 290–296.

Zixuan Li, Zhongni Hou, Saiping Guan, Xiaolong Jin, Weihua Peng, Long Bai, Yajuan Lyu, Wei Li, Jiafeng Guo, and Xueqi Cheng. 2022c. Hismatch: Historical structure matching based temporal knowledge graph reasoning. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 7328–7338.

Zixuan Li, Xiaolong Jin, Wei Li, Saiping Guan, Jiafeng Guo, Huawei Shen, Yuanzhuo Wang, and Xueqi Cheng. 2021. Temporal knowledge graph reasoning based on evolutional representation learning. In *SIGIR*, pages 408–417.

Ke Liang, Lingyuan Meng, Meng Liu, Yue Liu, Wenxuan Tu, Siwei Wang, Sihang Zhou, and Xinwang Liu. 2023. Learn from relational correlations and periodic events for temporal knowledge graph reasoning. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1559–1568.

Ke Liang, Lingyuan Meng, Meng Liu, Yue Liu, Wenxuan Tu, Siwei Wang, Sihang Zhou, Xinwang Liu, and Fuchun Sun. 2022. A survey of knowledge graph reasoning on graph types: Static, dynamic, and multimodal. *arXiv preprint arXiv:2212.05767*.

Ruotong Liao, Xu Jia, Yunpu Ma, and Volker Tresp. 2023. Gentkg: Generative forecasting on temporal knowledge graph. In *NeurIPS 2023 Workshop: New Frontiers in Graph Learning*.

Kangzheng Liu, Feng Zhao, Hongxu Chen, Yicong Li, Guandong Xu, and Hai Jin. 2022a. Da-net: Distributed attention network for temporal knowledge graph reasoning. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, pages 1289–1298.

Kangzheng Liu, Feng Zhao, Guandong Xu, Xianzhi Wang, and Hai Jin. 2023. Retia: relation-entity twin-interact aggregation for temporal knowledge graph extrapolation. In *IEEE International Conference on Data Engineering*. IEEE.

Lihui Liu, Boxin Du, Jiejun Xu, Yinglong Xia, and Hanghang Tong. 2022b. Joint knowledge graph completion and question answering. In *SIGKDD*, pages 1098–1108.

Ruilin Luo, Tianle Gu, Haoling Li, Junzhe Li, Zicheng Lin, Jiayi Li, and Yujiu Yang. 2024. Chain of history: Learning and forecasting with llms for temporal knowledge graph completion. *arXiv preprint arXiv:2401.06072*.

Soeren Nolting, Zhen Han, and Volker Tresp. 2023. Modeling the evolution of temporal knowledge graphs with uncertainty. *arXiv preprint arXiv:2301.04977*.

Namyong Park, Fuchen Liu, Purvanshi Mehta, Dana Cristofor, Christos Faloutsos, and Yuxiao Dong. 2022. Evokg: Jointly modeling event time and network structure for reasoning over temporal knowledge graphs. In *Proceedings of the fifteenth ACM international conference on web search and data mining*, pages 794–803.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32.

Ali Sadeghian, Mohammadreza Armandpour, Anthony Colas, and Daisy Zhe Wang. 2021. Chronor: Rotation based temporal knowledge graph embedding. In *AAAI*, volume 35, pages 6471–6479.

Haohai Sun, Jialun Zhong, Yunpu Ma, Zhen Han, and Kun He. 2021. Timetraveler: Reinforcement learning for temporal knowledge graph forecasting. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 8306–8319.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Hongwei Wang, Fuzheng Zhang, Jialin Wang, Miao Zhao, Wenjie Li, Xing Xie, and Minyi Guo. 2018. RippleNet: Propagating user preferences on the knowledge graph for recommender systems. In *CIKM*, pages 417–426.

Jiapu Wang, Boyue Wang, Meikang Qiu, Shirui Pan, Bo Xiong, Heng Liu, Linhao Luo, Tengfei Liu, Yongli Hu, Baocai Yin, et al. 2023. A survey on temporal knowledge graph completion: Taxonomy, progress, and prospects. *arXiv preprint arXiv:2308.02457*.

Minjie Yu Wang. 2019. Deep graph library: Towards efficient and scalable deep learning on graphs. In *ICLR workshop on representation learning on graphs and manifolds*.

Jiapeng Wu, Meng Cao, Jackie Chi Kit Cheung, and William L Hamilton. 2020. Temp: Temporal message passing for temporal knowledge graph completion. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5730–5746.

Siheng Xiong, Yuan Yang, Faramarz Fekri, and James Clayton Kerce. 2022. Tilp: Differentiable learning of temporal logical rules on knowledge graphs. In *The Eleventh International Conference on Learning Representations*.

Yi Xu, Junjie Ou, Hui Xu, and Luoyi Fu. 2023. Temporal knowledge graph reasoning with historical contrastive learning. In *AAAI*, volume 37, pages 4765–4773.

Yu Yang, Hongzhi Yin, Jiannong Cao, Tong Chen, Quoc Viet Hung Nguyen, Xiaofang Zhou, and Lei Chen. 2023. Time-aware dynamic graph embedding for asynchronous structural evolution. *IEEE Transactions on Knowledge and Data Engineering*.

Mengqi Zhang, Yuwei Xia, Qiang Liu, Shu Wu, and Liang Wang. 2023. Learning latent relations for temporal knowledge graph reasoning. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12617–12631.

Cunchao Zhu, Muhao Chen, Changjun Fan, Guangquan Cheng, and Yan Zhang. 2021. Learning from history: Modeling temporal knowledge graphs with sequential copy-generation networks. In *AAAI*, volume 35, pages 4732–4740.