

T3DM: Test-Time Training-Guided Distribution Shift Modelling for Temporal Knowledge Graph Reasoning

Yuehang Si¹, Zefan Zeng¹, Jincai Huang¹, Qing Cheng¹,

¹National University of Defense Technology, Changsha, Hunan, China

Correspondence: siyuehang@nudt.edu.cn

Abstract

Temporal Knowledge Graph (TKG) is an efficient method for describing the dynamic development of facts along a timeline. Most research on TKG reasoning (TKGR) focuses on modelling the repetition of global facts and designing patterns of local historical facts. However, they face two significant challenges: inadequate modeling of the event distribution shift between training and test samples, and reliance on random entity substitution for generating negative samples, which often results in low-quality sampling. To this end, we propose a novel distributional feature modeling approach for training TKGR models, **Test-Time Training-guided Distribution shift Modelling (T3DM)**, to adjust the model based on distribution shift and ensure the global consistency of model reasoning. In addition, we design a negative-sampling strategy to generate higher-quality negative quadruples based on adversarial training. Extensive experiments show that T3DM provides better and more robust results than the state-of-the-art baselines in most cases.

1 Introduction

TKGs, as a knowledge representation framework, possess the distinct capability to store and process entity-relationship information enriched with a temporal dimension. TKGs are increasingly critical in various sub-domains such as information retrieval, and recommender systems (Chen et al., 2023). In addition, TKG’s application scope extends to important areas such as policy making, dialogue systems, and stock market forecasting (Ji et al., 2024).

The data structure of the TKG encapsulates event knowledge in the form of a quadruple (*subject, predicate, object, time*). TKG consists of a series of snapshots of the knowledge graph, containing all events occurring simultaneously. For example, in Figure 1, “Elon Musk” receives an appeal request from Tesla Motors at moment T_1 and

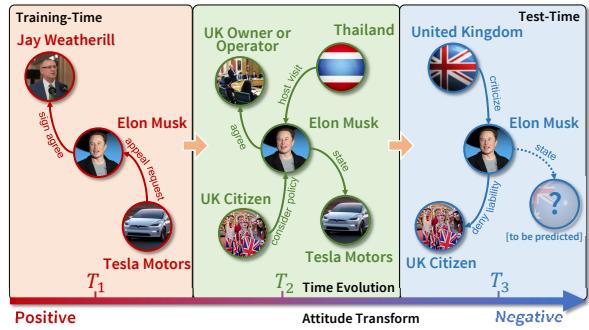


Figure 1: Illustration of the change of events in the evolution of the TKG over time in ICEWS18.

responds to Tesla Motors at moment T_2 . In addition, events of “Elon Musk” related to UK Citizens at moments T_2 and T_3 are “Consider policy” and “Deny liability”, respectively, where the roles of subject and object switch. Our study focuses on predicting future unknown events in TKG.

Recently, most existing TKGR methods encode the temporal evolution of factual relationships through time-embedded event triad data, offering a versatile and effective approach for predicting future facts on TKGs based on historical information (Zhu et al., 2021). Despite progress, TKGR field still faces two major challenges: event distribution shift and low negative sampling quality, as is shown in Figure 2.

(1) Event Distribution Shift. In TKG, the dynamic evolution of events is as a core research focus. Over time, the distribution of event types in TKGs shows a significant trend of change, a phenomenon known as event distribution shift. Event distribution shift refers to the change in the frequency of various types of events across distinct time periods, which can reflect the tendency of the social environment or policy tone at a specific moment. In most TKG training and test sets, event distribution shift is particularly significant. The Figure 2 demonstrate the distribution shift of event

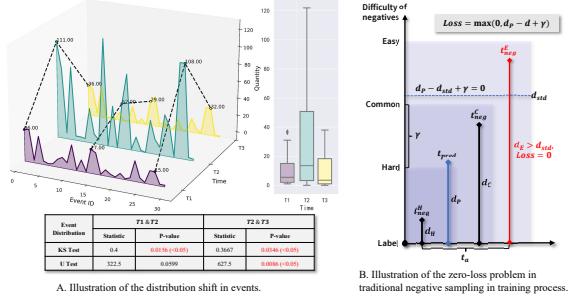


Figure 2: Illustration of event distribution shift and low negative sampling quality in ICEWS18.

relationships and results of non-parametric statistical tests (KS test and U test) for evaluating the divergence of event distributions at different time points in ICEWS18. This instance show that the event distributions exhibit shift in the time intervals from T_1 to T_2 and from T_2 to T_3 . This shifting phenomenon poses a challenge to the trained TKGR model in adapting to the new event distributions under the latest time. Therefore, it is particularly crucial to empower reasoning models with the ability to adapt to shifting event distributions. Some of existing methods model the evolution of event distributions to some extent, but they do not directly consider the problem of event distribution shift (Zhu et al., 2021).

(2) Low Negative Sampling Quality. Most of classical TKGR models are primarily trained by maximizing the distance between negative samples and minimizing the distance between positive samples. These methods generates negative samples by replacing one of the entities with another entity in the knowledge graph while keeping the relationship and timestamp unchanged (Xu et al., 2023). However, the vast majority of current TKGR models rely on the random negative sampling method, i.e., by randomly selecting other entities to replace the entities in the original fact quadruple, thus obtaining negative samples. This method ignores the logical relationships and structural features among entities, thus limiting the further improvement of the model performance due to the issue of “zero-loss” shown in Figure 2.

In this study, we propose a plug-and-play training method for distribution modelling in a test-time training (TTT) framework, named T3DM. We introduce LSTM (Long Short-Term Memory) as the distributional inference model to design an auxiliary training task in the testing phase. In addition, we propose an adversarial negative-sampling strat-

egy for TKGR, called **Temporal Knowledge Graph Generative Adversarial Networks** (TKGAN). We introduce a reinforcement learning algorithm to guide the sample generation process of TKGAN. We integrate T3DM into multiple TKGR baselines and conduct experiments on five publicly available datasets. The main contributions are as follows:

1) We highlight the challenges of event distribution shift and low negative sampling quality faced by existing TKGR models, which hinder their reasoning accuracy.

2) We propose a TTT-guided distribution modelling training method, T3DM, to address the event distribution shift issue in TKG. To the best of our knowledge, **we are the first to introduce the concept of TTT into knowledge graph domain to model the event distribution shift.**

3) We design TKGAN, an adversarial negative sampling strategy for TKGR, to improve the quality of negative sampling through adversarial training and incorporate a reinforcement learning strategy to guide training.

4) We conduct extensive experiments on five TKG datasets for reasoning tasks and verify the effectiveness of T3DM.

2 Related Work

2.1 Temporal Knowledge Graph Reasoning

Temporal attributes play a crucial role in TKGR and have garnered significant attention. Know-Evolve (Trivedi et al., 2017), as the first model to learn the nonlinear evolution of entities, lays the foundation for subsequent research. xERTE (Han et al., 2021a) and TLogic (Liu et al., 2022), while providing interpretable predictive evidence, are limited in their application. TANGO (Han et al., 2021b) utilises a neural frequent formula to model TKG, while CyGNet (Zhu et al., 2021) identifies high-frequency repetitive events through the replication construction mechanism. RE-GCN (Li et al., 2021) employs a reinforcement learning design. Some models attempt to incorporate neural network architectures to capture spatio-temporal patterns, such as RE-NET (Jin et al., 2020), HIP (He et al., 2024), TITer (Sun et al., 2021) and EvoKG (Park et al., 2022). In addition, CENET (Xu et al., 2023) integrates contrastive learning, while Coherence-Mode (Si et al., 2025) designs new synergistic relationship assessment units to mine the deeper correlations. However, when the event type distribution in the test set differs significantly from that in the

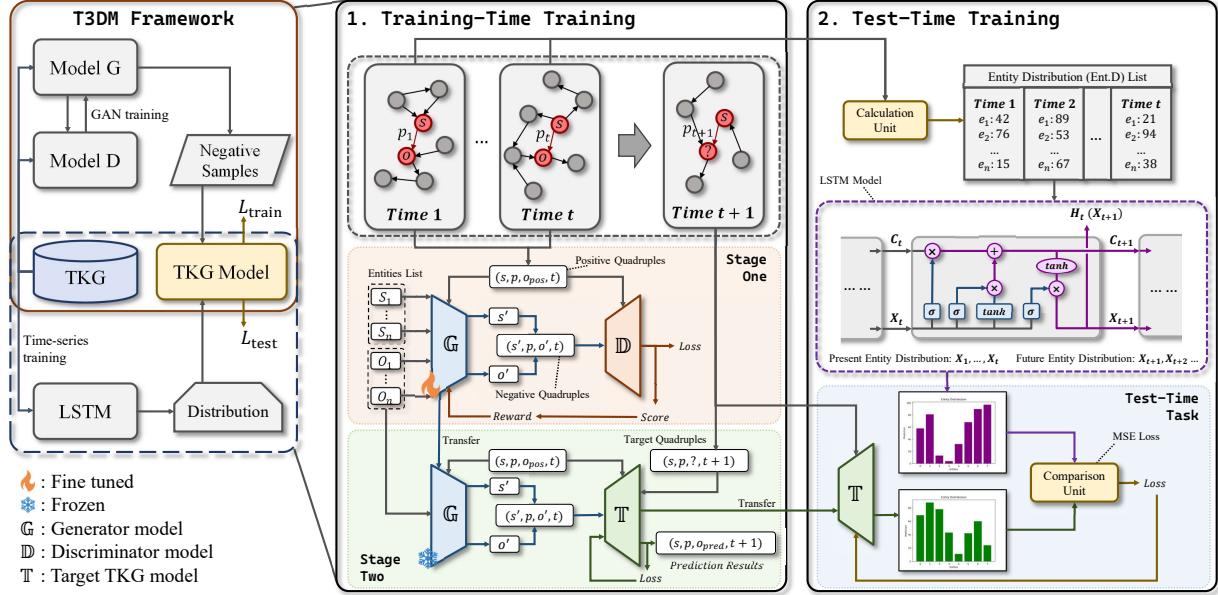


Figure 3: The training framework of T3DM. The first part represents Training Time Training, and the second part represents Test Time Training. The TKGAN part of Training Time Training is further divided into two stages.

training set, the inference performance of these models is severely constrained.

2.2 Test Time Training

TTT (Sun et al., 2020) works for solving the problem of distribution shift between training and test samples, which is based on partially tuning the test samples to optimise the model for changes in distributions between the training and test sets (Sun et al., 2019a). Meanwhile, TTT++ (Liu et al., 2021) used a regularised adaptive approach of offline feature extraction and online feature alignment. TTT-MAE (Gandelsman et al., 2022) uses a masked auto-encoder to solve the single-sample learning problem. TTT is also widely used in many domains. For example, LMTTT (Zhang et al., 2024) innovatively uses the large language model as an annotator to augment the TTT. DT3OR (Yang et al., 2024) addresses the problem of significant degradation in recommender systems due to shifts in user and item features. Therefore, it is of significance to address distributional shifts in TKGR using TTT techniques.

2.3 GAN for Negative Sampling

Generative Adversarial Network is a deep learning technique proposed by Ian Goodfellow (Goodfellow et al., 2014) in 2014. It consists of two parts: the generator and the discriminator, which improve each other's performance through adversarial training. GANs are effective in enhancing

the quality of negative samples in several domains. ANDA (Ruiz et al., 2019) designs data-efficient GANs to improve their negative sampling ability in image classification tasks. CDE-GAN (Chen et al., 2021) uses adversarial techniques to generate high-quality negative samples in novelty detection. NDAGAN (Shayesteh and Inkpen, 2022) successfully applies negative data enhancement techniques to text categorisation. GAN is gradually becoming a popular negative sampling method in knowledge graph. KBGAN (Cai and Wang, 2018) uses a knowledge graph embedding model as a negative sample generator to assist in training a reasoning model. FedEAN (Meng et al., 2024) is an entity-aware negative sampling strategy through joint training of the generator and the discriminator. However, none of them takes time into account, and applies GANs to negative sampling in TKGR.

3 The Proposed Framework

3.1 Overview

The training framework of T3DM consists of two parts: training time training and test time training (see Figure 3).

Training time training is the training process of traditional TKGR based on existing quadruples to reason about future quadruples. The whole phase of the training process is structured in two stages. The first stage is composed of generator and discriminator. The generator receives the positive

samples and the set of entities and generates the corresponding negative samples. The generated negative samples are used as inputs along with the positive samples for discriminator training. The discriminator’s training process is the same as that of the traditional TKGR model, which maximises the weights of the positive and minimises negative samples’ weights. The second stage is based on the trained generator and the target model to be trained. The target model is co-trained using the negative samples generated by the generator and the positive samples provided by the dataset with same training objective as the discriminator.

In the phase of test time training. We obtain a list of the distribution of the number of entities at different moments through the computational unit. Subsequently, LSTM is trained to predict entity number distributions at future moments. When the target model predicts the test set, the distribution of entities predicted is similarly calculated. The two distribution predictions are compared to calculate the loss, and the optimisation of the performance of the target model is continued instead.

3.2 Preliminaries

In TKGs, each fact contains a relation (or predicate) $p \in \mathcal{R}$ between the subject $s \in \mathcal{E}$ and the object $o \in \mathcal{E}$, which is located at time step $t \in \mathcal{T}$. Here, \mathcal{E} and \mathcal{R} denote the vocabularies of entities and relations, respectively, while \mathcal{T} denotes the collection of timestamps. \mathcal{G}_T denotes a snapshot of the TKG at time T , and $g = (s, p, o, t)$ denotes a quadruple (fact) in \mathcal{G}_T . The TKG is constructed around a set of events. These quadruple events are ordered chronologically, i.e., $\mathcal{G} = \{\mathcal{G}_1, \mathcal{G}_2, \dots, \mathcal{G}_T\}$.

The purpose of making predictions about omitted time facts is to infer that the omitted object entity $(s, p, ?, t)$ (or the supplied subject entity $(?, p, o, t)$, or the predictive relationship $(s, ?, o, t)$). Our model is described as predicting missing entities in time facts.

3.3 Adversarial Negative Sampling

Inspired by GAN, we propose a TKG adversarial training framework, TKGAN, for adversarial generation of negative quadratic samples. Corresponding to the terminology used in the classical GAN literature, in the rest of this paper, we refer to the two models used for adversarial training simply as the generator and discriminator, respectively. Our work has a similar training goal as the classical GAN framework: the ultimate goal of our

framework is to train a good generator to generate high-quality negative samples. The discriminator should assign a relatively high score to high-quality negative samples during training. Therefore, the goal of the generator should be set to maximise the score given by the discriminator for the quadruples it generates. Just like the traditional training process for TKGR models, the discriminator aims to minimise the marginal loss between the positive quadruple and the generated negative quadruple. In an adversarial training environment, the generator and discriminator are alternately trained on their respective goals.

Suppose that the generator generates a probability distribution $p_{\mathbb{G}}(s', p, o', t \mid s, p, o, t)$ over the negative quadruple (s', p, o', t) given the positive quadruple (s, p, o, t) , and by sampling from this distribution to generate the negative quadruple (s', p, o', t) . Let $f_{\mathbb{D}}(s, p, o, t)$ be the score function of the discriminator. The objective of discriminator can be stated as minimising the following marginal loss function:

$$L_{\mathbb{D}} = \sum_{g \in \mathcal{G}} [f_{\mathbb{D}}(g) - f_{\mathbb{D}}(s', p, o', t) + \gamma]_+, \quad (1)$$

where $(s', p, o', t) \sim p_{\mathbb{G}}(s', p, o', t \mid s, p, o, t)$ is the negative quadruple generated by generator, $g = (s, p, o, t)$, γ is the constant offset coefficient, and $[.]_+$ denotes the take positive operation.

The computation of the reward of the generator can be formulated as maximising the expectation of the negative distance of the discriminator:

$$R_{\mathbb{G}} = \sum_{(s, p, o, t) \in \mathcal{G}} E(-f_{\mathbb{D}}(s', p, o', t)), \quad (2)$$

where $(s', p, o', t) \sim p_{\mathbb{G}}(s', p, o', t \mid g)$, $E(\cdot)$ denotes the expectation operation.

The $R_{\mathbb{G}}$ involves a discrete sampling step, and the gradient cannot be computed by simple differentiation. We use the strategic gradient theorem to obtain the gradient of $R_{\mathbb{G}}$ with respect to the generator parameters:

$$\begin{aligned} \nabla_{\mathbb{G}} R_{\mathbb{G}} &= \sum_{(s, p, o, t) \in \mathcal{G}} \mathbb{E}_{(s', p, o', t) \sim p_{\mathbb{G}}(s', p, o', t \mid g)} \\ &\quad (-f_{\mathbb{D}}(s', p, o', t) \nabla_{\mathbb{G}} \log p_{\mathbb{G}}(s', p, o', t \mid g)) \\ &\simeq \sum_{g \in \mathcal{G}} \frac{1}{N} \sum_{(s'_i, p, o'_i, t) \sim p_{\mathbb{G}}(s', p, o', t \mid g), i=1 \dots N} \\ &\quad (-f_{\mathbb{D}}(s', p, o', t) \nabla_{\mathbb{G}} \log p_{\mathbb{G}}(s', p, o', t \mid g)). \end{aligned} \quad (3)$$

The approximation equation shows that we can approximate the expectation with sampled mean. This approximation implements the gradient computation of $R_{\mathbb{G}}$ and enables its optimisation using gradient-based algorithms.

The strategy gradient theorem is derived from reinforcement learning (RL). Thus, the generator can be viewed as an agent that interacts with environment by performing actions and improves itself by maximising the reward for its actions. Correspondingly, the discriminator serves as the environment, and its output corresponds to the feedback made by the environment. Using RL terminology, $g = (s, p, o, t)$ is the state (determining what actions the agent can take), $p_{\mathbb{G}}(s', p, o', t | g)$ is the strategy (how the agent chooses to take action), and (s', p, o', t) is the action and $-f_{\mathbb{D}}(s', p, o', t)$ is the reward. Unlike typical RL, where agent performs a series of actions, the agent in our model acts only once and does not affect the state.

To reduce the variance of the gradient algorithm, a baseline value, which depends solely on the state of the agent, is typically subtracted from its reward. In our model, we introduce a baseline term in $-f_{\mathbb{D}}(s', p, o', t)$. We replace it with $-f_{\mathbb{D}}(s', p, o', t) - b(g)$. b is a constant, the reward over the entire training set, approximated by the average of the rewards of the most recently generated negative quadruples:

$$b = \sum_{g \in \mathcal{G}} \mathbb{E}_{(s', p, o', t) \sim p_{\mathbb{G}}(s', p, o', t | g)} (-f_{\mathbb{D}}(s', p, o', t)). \quad (4)$$

Given a set of candidate negative quadruples $\text{Neg}(g) \subseteq \{(s', p, o, t) \mid s' \in \mathcal{E}\} \cup \{(s, p, o', t) \mid o' \in \mathcal{E}\}$, then the probability distribution $p_{\mathbb{G}}$ is modelled as:

$$p_{\mathbb{G}}(s', p, o', t | g) = \frac{\exp f_{\mathbb{G}}(s', p, o', t)}{\sum_{s^*, o^*} \exp f_{\mathbb{G}}(s^*, p, o^*, t)}. \quad (5)$$

where $(s^*, p, o^*, t) \in \text{Neg}(g)$, and $f_{\mathbb{G}}(g)$ is the generator score function.

Ideally, $\text{Neg}(g)$ should contain all possible negative quadruples. However, TKGs are usually highly incomplete, so the ‘‘hardest to distinguish’’ negative quadruples are likely to be false negatives (true facts). To address this problem, we generate $\text{Neg}(g)$ by uniformly sampling a certain number of entities (a minimal number compared to the number of all possible negatives) from the entity set \mathcal{E} .

Algorithm 1 The TKGAN algorithm

Input: Training set of positive fact quadruples $\mathcal{G} = \{g : (s, p, o, t)\}$; Pre-trained generator \mathbb{G} and \mathbb{D} with parameters $\theta_{\mathbb{G}}, \theta_{\mathbb{D}}$ and score function $f_{\mathbb{G}}(g), f_{\mathbb{D}}(g)$
Output: Adversarially trained generator \mathbb{G}

- 1: $b \leftarrow 0$; %baseline constant for policy gradient
- 2: **repeat**
- 3: Sample a small batch of positive quadruples $\mathcal{G}_{\text{batch}}$ from \mathcal{G}
- 4: Initial gradients of parameters: $G_{\mathbb{G}} \leftarrow 0, G_{\mathbb{D}} \leftarrow 0$
- 5: Total reward: $r_{\text{total}} \leftarrow 0$
- 6: **for** each $g \in \mathcal{G}_{\text{batch}}$ **do**
- 7: Uniformly randomly sample K negative quadruples $\text{Neg}(g) = \{(s'_i, p, o'_i, t)\}_{i=1 \dots K}$
- 8: Obtain probability of being generated: $p_i = \frac{\exp f_{\mathbb{G}}(s'_i, p, o'_i, t)}{\sum_{j=1}^K \exp f_{\mathbb{G}}(s'_j, p, o'_j, t)}$
- 9: Sample negative quadruple (s'_h, p, o'_h, t) from $\text{Neg}(g)$ with the highest probability p_h according to $\{p_i\}_{i=1 \dots K}$
- 10: $G_{\mathbb{D}} \leftarrow G_{\mathbb{D}} + \nabla_{\theta_{\mathbb{D}}} [f_{\mathbb{D}}(g) - f_{\mathbb{D}}(s'_h, p, o'_h, t) + \gamma]_+$
- 11: Calculate reward for \mathbb{G} : $r \leftarrow -f_{\mathbb{D}}(s'_h, p, o'_h, t)$, $r_{\text{total}} \leftarrow r_{\text{total}} + r$
- 12: $G_{\mathbb{G}} \leftarrow G_{\mathbb{G}} + (r - b) \nabla_{\theta_{\mathbb{G}}} \log p_h$
- 13: **end for**
- 14: $\theta_{\mathbb{G}} \leftarrow \theta_{\mathbb{G}} + \eta_{\mathbb{G}} G_{\mathbb{G}}, \theta_{\mathbb{D}} \leftarrow \theta_{\mathbb{D}} - \eta_{\mathbb{D}} G_{\mathbb{D}}$
- 15: $b \leftarrow r_{\text{total}} / |\mathcal{G}_{\text{batch}}|$
- 16: **until** convergence

to replace either s or t . As in real-world TKGs, the quantity of true negative quadruples significantly exceeds that of false negative quadruples. Consequently, it is improbable that a given set includes any false negative samples. Furthermore, the negative quadruples selected by the generator are highly likely to be classified as true negative quadruples.

In addition, we employ the ‘‘bern’’ sampling technique (Wang et al., 2014) to reduce false negatives further by replacing ‘‘1’’ of relationships with higher probability in ‘‘1-to-N’’ and ‘‘N-to-1’’. Algorithm 1 details the steps of TKGAN.

In the training phase, T3DM uses the negative quadruple samples generated based on the TKGAN method as a new part of generating negative samples in the traditional TKGR. Note that the choice of generator and discriminator for TKGAN is very flexible, and the target model to be trained can either be directly used as a discriminator or co-trained with the generator model. In our approach T3DM, we separate the training process of the target model to be trained from the generator model, i.e., the negative sample generator model is trained first (Stage 1) and then added to the training of the target model (Stage 2). This design allows T3DM to adapt more effectively by directly generating negative samples for TKGR models, eliminating the need to adjust the corresponding relationships within TKGAN.

3.4 Test-Time Training

To address the issue of changing event type distributions over time in TKGs, we integrate TTT with TKGR models. In the field of TKGR, we explore an auxiliary training strategy conducted at test-time, i.e., to improve the inference ability of the model by training its ability to predict the event type distribution during the Test-time phase. Specifically, since individual events refer to the associations generated between entities, the distribution of event types can be determined by the distribution of the number of various types of entities at a particular point in time, and the distribution of the number of entities can adequately reflect the changes in event types. This correspondence is crucial for understanding and predicting dynamic changes in TKG.

To achieve this goal, we employ LSTM to predict the distribution of the entities as the "pseudo labels" for TKGR in the Test-time phase. LSTM consists of forget gate, input gate, cell state, and output gate, as is shown in Figure 3.

$$S_{t+1} = \text{LSTM}(S_{t-l+1}, \dots, S_{t-1}, S_t) \quad (6)$$

where S_t denotes the event distribution at time t . l is the input sequence length of LSTM.

When testing, the TKGR model is used to predict the distribution of the number of entities. The prediction distribution is compared with the distribution predicted by LSTM model, and the loss between the two is calculated as follows:

$$\begin{aligned} L_{cmp} &= \sum_{i=1}^{T_{Pred}} CE_Loss(X_i^{True}, X_i^{Pred}) \\ &= - \sum_{i=1}^{T_{Pred}} \sum_{j=1}^N x_{i,j}^{True} \log(\text{softmax}(P_{i,j})), \end{aligned} \quad (7)$$

where $CE_Loss(\cdot)$ denotes the cross-entropy loss function, N denotes the number of object entities in the snapshot \mathcal{G}_t , and T_{Pred} denotes the predicting time period. X^{True} denotes the distribution predicted by the LSTM, and X^{Pred} denotes the distribution predicted by TKGR model. $P_{i,j}$ denotes the probability of $x_{i,j}^{Pred}$ on each entity.

By auxiliary training, the TKGR model can continuously adjust and optimise its parameters during the testing phase to improve prediction accuracy. This approach enhances the model's ability to adapt to dynamic changes in the knowledge graph and its robustness in complex temporal reasoning tasks.

3.5 Inference

We describe the reasoning process as predicting missing objects in time facts without loss of generality. In predicting the query $(s, p, ?, t)$, the TKGR model provides the object entity with the highest probability in the candidate space. The prediction result of the original baseline model is defined as follows:

$$o_t = \operatorname{argmax}_{o_{pred} \in \mathcal{E}} P(o_{pred} | s, p, o_{rand}, t) \quad (8)$$

where o_{pred} denotes the final prediction of the model and o_{rand} denotes the random entity input to the model.

4 Experiment

This section shows the effectiveness of the proposed T3DM¹ through a comprehensive experiment by solving the following questions: RQ1: What is the gain of T3DM's performance on TKGR tasks? RQ2: What are the advantages of the adversarial negative sampling method used in T3DM? RQ3: Can the unique training framework based on TTT in T3DM be applied to a more extensive range of baseline models? RQ4: How can different choices of generator and discriminators in TKGAN affect model performance? RQ5: How does the prediction of LSTM models with different input sequence length affect model performance?

4.1 Experimental Setup

4.1.1 Datasets and Metrics

We select five publicly available benchmark datasets for TKGR: GDELT (Leetaru and Schrot, 2013), ICEWS14 (Boschee et al., 2015), ICEWS18, YAGO11K (Mahdisoltani et al., 2015), and Wikidata12K (Leblay and Chekol, 2018).

To evaluate the performance of the T3DM model, we use standard metrics such as Mean Reciprocal Rank (MRR) and Hits@K (K=1,3,10). MRR denotes the average inverse ranking of all samples, while Hits@K measures the proportion of test samples ranked in the top K positions.

4.1.2 Baselines

To validate the effectiveness of T3DM, we choose a series of baseline models to start experiments, including TTransE (Jiang et al., 2016), TATransE (García-Durán et al., 2018), HyTE (Dasgupta et al., 2018), TADistmult (García-Durán et al., 2018) and

¹The released source code and documentation are available at <https://anonymous.4open.science/r/T3DM-6514>

Table 1: Experimental results of T3DM for TKG link prediction task on five datasets. The best results are presented in boldface, and the previous SOTA are underlined (if needed). H@1, H@3 and H@10 represent Hits@1, Hits@3 and Hits@10, respectively.

Baseline	Model	ICEWS18			ICEWS14			GDELT			WIKI			YAGO			
		MRR	H@1	H@3	H@10												
	TTransE	8.36	1.94	8.71	21.93	6.35	1.23	5.80	16.65	5.52	0.47	5.01	15.27	31.74	32.61	36.25	43.45
	TTransE&TKGAN	11.92	3.40	13.82	29.80	10.24	2.76	11.28	26.12	6.78	0.53	6.62	20.11	38.41	37.60	42.85	47.57
	TTransE&T3DM	11.98	3.50	13.91	29.87	10.29	2.83	11.30	26.13	6.84	0.58	6.71	20.30	38.44	37.65	42.86	47.59
TransE	TATransE	9.28	3.89	9.34	18.20	8.39	4.22	9.06	20.81	11.37	7.44	11.83	21.32	41.89	37.86	45.20	49.74
	TATransE&TKGAN	10.46	5.92	10.99	19.42	13.54	7.81	14.52	24.66	14.77	9.32	15.39	24.89	45.28	40.92	48.87	51.92
	TATransE&T3DM	10.50	5.94	11.06	19.48	13.59	7.82	14.56	24.69	14.83	9.40	15.43	24.92	45.33	40.99	48.96	51.99
	HyTE	7.31	4.03	7.50	14.95	11.48	4.30	13.04	22.51	6.37	4.78	6.72	18.63	43.02	27.99	45.12	49.49
	HyTE&TKGAN	8.31	5.00	9.14	14.96	10.98	4.48	13.17	23.06	10.11	6.66	10.88	18.93	44.56	28.84	45.87	50.45
	HyTE&T3DM	8.33	5.04	9.19	15.00	11.03	4.50	13.21	23.14	10.16	6.70	10.91	18.94	44.60	28.87	45.89	50.46
Distmult	TADistmult	28.53	20.30	31.57	44.96	20.78	13.43	22.80	35.26	29.35	22.11	31.56	41.39	48.09	45.84	49.51	51.70
	TADistmult&TKGAN	27.79	20.17	31.81	45.52	22.52	15.98	23.50	35.75	29.42	22.61	32.36	43.17	51.13	49.15	52.23	57.09
	TADistmult&T3DM	27.93	20.36	31.86	45.59	22.60	16.04	23.55	35.81	29.46	22.71	32.48	43.21	51.19	49.22	52.36	57.20
SimplE	DE-SimplE	12.56	9.84	19.87	27.07	16.08	9.71	13.90	16.55	20.49	6.33	17.00	23.97	18.17	9.92	18.79	32.44
	DE-SimplE&TKGAN	12.93	10.15	20.32	27.22	16.34	10.03	14.17	16.70	20.75	6.68	17.14	24.14	18.21	10.08	18.98	32.56
	DE-SimplE&T3DM	13.06	10.24	20.46	27.32	16.51	10.22	14.29	16.83	20.97	6.85	17.38	24.40	18.49	10.33	19.17	32.74

Table 2: Ablation experimental results of TTT part in T3DM for TKG link prediction task on five datasets.

Model	ICEWS18			ICEWS14			GDELT			WIKI			YAGO			
	MRR	H@1	H@3	H@10												
TransE	17.56	2.48	26.95	43.87	18.65	1.12	31.34	47.07	16.05	0.00	26.10	42.29	46.68	36.19	49.71	51.71
Distmult	22.16	12.13	26.00	42.18	19.06	10.09	22.00	36.41	18.71	11.59	20.05	32.55	46.12	37.24	49.81	51.38
ComplEx	30.09	21.88	34.15	45.96	24.47	16.13	27.49	41.09	22.77	15.77	24.05	36.33	47.84	38.15	50.08	51.39
RotatE	23.10	14.33	27.61	38.72	29.56	22.14	32.92	42.68	22.33	16.68	23.89	32.29	50.67	39.73	50.71	50.88
RE-NET	42.93	36.19	45.47	55.80	45.71	38.42	49.06	59.12	40.12	32.43	43.40	53.80	51.97	48.01	52.07	53.91
TANGO-TuckER	44.56	37.87	47.46	57.06	46.42	38.94	50.25	59.80	38.00	28.02	43.91	53.70	53.28	52.21	53.61	62.72
TANGO-Distmult	44.00	38.64	45.78	54.27	46.68	41.20	48.64	57.05	41.16	35.11	43.02	52.58	54.05	51.52	53.84	62.95
CyGNet	46.69	40.58	49.82	57.14	48.63	41.77	52.50	60.29	50.29	44.53	54.69	60.99	45.50	50.48	57.99	52.80
EvoKG	29.67	12.92	33.08	58.32	18.30	6.30	19.43	39.37	11.29	2.93	10.84	25.44	50.66	12.21	63.84	67.29
CENET	51.06	47.10	51.92	58.82	53.35	49.61	54.07	60.62	58.48	55.99	58.63	62.96	68.39	68.33	68.36	69.05
HIP network	48.37	43.51	51.32	58.49	50.57	45.73	54.28	61.65	52.76	46.35	55.31	61.87	54.71	53.82	54.73	56.46
CEC-BD	28.53	18.85	32.31	47.75	47.53	39.77	53.25	59.54	34.74	27.35	39.37	52.21	33.93	12.41	36.92	54.33
Co-CyGNet	47.93	43.21	51.38	58.99	49.57	43.04	53.64	60.03	50.77	45.05	54.93	61.42	46.31	52.19	52.34	54.11
Co-CENET	51.47	47.32	52.08	<u>59.11</u>	<u>53.37</u>	49.58	<u>54.81</u>	61.30	<u>58.63</u>	56.31	58.99	62.88	68.77	68.41	68.58	69.34
CyGNet&TTT	47.85	43.36	51.16	58.04	49.30	43.28	53.33	60.31	50.91	45.12	54.99	60.96	46.73	52.16	52.53	54.16
CENET&TTT	51.49	47.31	52.10	59.01	53.35	49.55	<u>54.81</u>	61.33	58.61	<u>56.35</u>	59.00	62.79	<u>68.78</u>	68.44	68.60	69.39
Co-CENET&TTT	51.51	47.34	52.12	59.15	53.41	49.64	54.83	<u>61.38</u>	58.65	56.37	59.02	<u>62.91</u>	68.80	68.49	68.64	69.40

DE-SimplE (Goel et al., 2020). We also compare with a range of static KG and TKGR models, including TransE (Bordes et al., 2013), DistMult (Yang et al., 2015), ComplEx (Trouillon et al., 2016), RotatE (Sun et al., 2019b), RE-NET (Jin et al., 2020), TANGO (Han et al., 2021b), CyGNet (Zhu et al., 2021), EvoKG (Park et al., 2022), CENET (Xu et al., 2023), HIP Network (He et al., 2024), CEC-BD (Yue et al., 2024), and Coherence-Mode (Si et al., 2025).

4.2 Experimental Performance (RQ1)

To address RQ1, we analyse the performance of the proposed models. Specifically, we integrate the proposed T3DM into three types of baselines totalling five models. We compare the performance of these baselines on five TKG datasets with the integration effect. Table 1 shows the results of using TTransE as the generator. It can be seen that the integration

of T3DM obtains entire performance gains on both TATransE and DE-SimplE (achieve performance improvements of 3.04 and 0.37 on average respectively), and only one metric is slightly inferior to the baseline on TTransE and HyTE. These results suggest that the TKG model can significantly improve its ability to reason about future events by incorporating the antagonistic negative sampling approach and adding training on factual distributions at test time. In contrast, a small number of metrics are down on TADistmult (with a lag on YAGO, from 62.80 to 59.74 on H@1), which is mainly attributed to the fact that the architecture of TADistmult and its handling of negative samples is very different from TTransE, and the return rewards have limited enhancement on negative sampling. Nevertheless, in terms of the experiment as a whole, baselines integrated with T3DM still show a significant performance improvement.

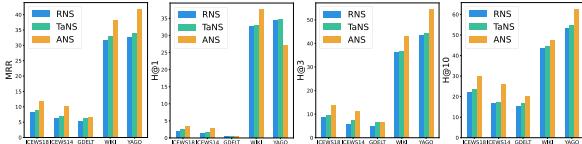


Figure 4: Comparison results of different negative sampling methods on five datasets. RNS, TaNS and ANS represents random, time-aware and adversarial negative sampling, respectively.

4.3 Ablation Study (RQ2 & RQ3)

For RQ2, we perform ablation analysis for the adversarial negative sampling method TKGAN. As shown in Table 1, we individually integrate the proposed TKGAN into five models from three types of baselines without the architectural design of TTT. The gain of TKGAN to the models is significant in almost all experimental settings compared to the traditional random negative sampling method. At the same time, only the individual metrics of individual models decreased, which is related to the choice of model framework and generator. Figure 4 compares TKGAN with random and time-aware negative sampling. The results suggest that TKGAN can bring better quality negative samples to the model and improve the model inference.

For the consideration of the TTT training structure in RQ3, in addition to the comparison of T3DM and TKGAN in Table 1, we apply it to more TKG models, as shown in Table 2. Due to their unique design, these models are not combined with negative samples, so we just use them to validate the TTT framework. From the results, it can be seen that for both CyGNet and Co-CENET, TTT obtains performance improvements in all metrics for all five datasets, and there are only two datasets with a decrease in H@10 and one H@1 on CENET. This phenomenon is mainly attributed to the limitations of CENET’s unique contrastive learning mechanism, which results in the loss of distributional differences for model updating. The results demonstrate that our TTT training model and the designed auxiliary training tasks can improve the model’s ability to cope with event distribution shift.

4.4 Sensitivity Analysis (RQ4 & RQ5)

For RQ4, we compare different generators in TKGAN. As shown in Table 3, we choose TTransE, HyTE and DE-SimplE as generators, and discriminators are the same as in main experiments. From the results, we find that choosing TTransE as the

Table 3: Experimental results of different TKGAN group settings for TKG link prediction task on YAGO.

Model G	TTransE		HyTE		DE-SimplE	
	MRR H@3	H@10	MRR H@3	H@10	MRR H@3	H@10
TTransE	41.84	27.05	36.87	19.67	40.21	25.17
	54.47	62.65	51.90	60.57	52.69	61.38
TATransE	54.37	47.92	54.98	49.03	54.25	47.86
	59.31	65.27	59.65	65.12	59.07	64.77
HyTE	42.86	38.51	56.32	50.65	53.85	47.91
	46.47	52.39	60.49	65.97	57.60	64.15
TADistmult	62.82	59.37	51.79	44.50	49.66	41.22
	63.67	71.46	57.55	63.81	56.42	63.38
DE-SimplE	30.80	34.39	22.65	14.90	14.62	8.99
	38.69	47.15	24.74	38.63	15.19	25.38

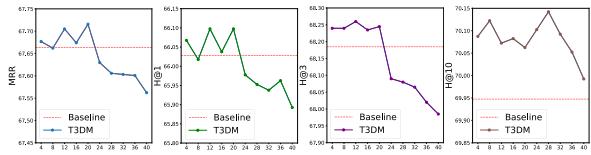


Figure 5: Sensitivity analysis results of input sequence length hyperparameter in LSTM.

generator gives the optimal performance with the most experimental setups (which is why choosing TTransE as the generator in main experiments). HyTE as the generator brings a better negative sampling effect for TATransE and HyTE, which is attributed to similar structural design and negative sampling process. The experiments show that for different TKG models, a model with similar structure can be chosen as the generator, and a suitable TKGAN design can improve their performance.

In addition, we analyse the hyperparameter of input sequence length in LSTM. As shown in Figure 5, the model effect tends to be optimal when the length does not exceed 20 and decreases with increasing length. Nonetheless, the length variation can get a stable effect enhancement on H@10, which is further evidence of the significant gain of the TTT framework to the TKG models.

5 Conclusions

In this paper, we propose T3DM, a plug-and-play training method for distribution modelling in a TTT framework. Specifically, we are the first to introduce the TTT framework and the GAN model to TKGR. We use TTT framework to enhance the model’s ability to model event distribution shift, and design an adversarial negative-sampling strategy to generate higher-quality negative quadruples. Extensive experiments on five datasets show that most baselines integrated with T3DM achieve better performance in link prediction.

Limitations

This paper primarily focuses on TKGR, which aims to predict future unknown events based on existing knowledge. In designing the training tasks for the test phase, we only obtain labels through LSTM predictions of future distributions, which leaves room for improvement. In future research, we will attempt to design new self-supervised training tasks, such as integrating large language models to improve the encoding of entities and relations. Furthermore, we only evaluate the robustness of TKGAN with five baseline models. To more convincingly demonstrate the high-quality negative sampling of TKGAN, we will incorporate additional tasks and baseline models in future work.

References

- Antoine Bordes, Nicolas Usunier, Alberto García-Durán, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *NIPS 2013*, pages 2787–2795.
- Elizabeth Boschee, Jennifer Lautenschlager, Sean O’Brien, Steve Shellman, James Starz, and Michael Ward. 2015. Icews coded event data. *Harvard Dataverse*, 12.
- Liwei Cai and William Yang Wang. 2018. KBGAN: adversarial learning for knowledge graph embeddings. In *NAACL-HLT 2018*, pages 1470–1480. Association for Computational Linguistics.
- Shiming Chen, Wenjie Wang, Beihao Xia, Xinge You, Qinmu Peng, Zehong Cao, and Weiping Ding. 2021. CDE-GAN: cooperative dual evolution-based generative adversarial network. *IEEE Trans. Evol. Comput.*, 25(5):986–1000.
- Zhongwu Chen, Chengjin Xu, Fenglong Su, Zhen Huang, and Yong Dou. 2023. Temporal extrapolation and knowledge transfer for lifelong temporal knowledge graph reasoning. In *Findings of the Association for Computational Linguistics: EMNLP 2023, Singapore, December 6-10, 2023*, pages 6736–6746. Association for Computational Linguistics.
- Shib Sankar Dasgupta, Swayambhu Nath Ray, and Partha P. Talukdar. 2018. Hyte: Hyperplane-based temporally aware knowledge graph embedding. In *EMNLP 2018*, pages 2001–2011. Association for Computational Linguistics.
- Yossi Gandelsman, Yu Sun, Xinlei Chen, and Alexei A. Efros. 2022. Test-time training with masked autoencoders. In *NeurIPS 2022*.
- Alberto García-Durán, Sebastijan Dumancic, and Mathias Niepert. 2018. Learning sequence encoders for temporal knowledge graph completion. In *EMNLP 2018*, pages 4816–4821. Association for Computational Linguistics.
- Rishab Goel, Seyed Mehran Kazemi, Marcus A. Brubaker, and Pascal Poupart. 2020. Diachronic embedding for temporal knowledge graph completion. In *AAAI 2020*, pages 3988–3995. AAAI Press.
- Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron C. Courville, and Yoshua Bengio. 2014. Generative adversarial networks. *CoRR*, abs/1406.2661.
- Zhen Han, Peng Chen, Yunpu Ma, and Volker Tresp. 2021a. Explainable subgraph reasoning for forecasting on temporal knowledge graphs. In *ICLR 2021*. OpenReview.net.
- Zhen Han, Zifeng Ding, Yunpu Ma, Yujia Gu, and Volker Tresp. 2021b. Learning neural ordinary equations for forecasting future links on temporal knowledge graphs. In *EMNLP 2021*, pages 8352–8364. Association for Computational Linguistics.
- Yongquan He, Peng Zhang, Luchen Liu, Qi Liang, Wenyuan Zhang, and Chuang Zhang. 2024. HIP network: Historical information passing network for extrapolation reasoning on temporal knowledge graph. *CoRR*, abs/2402.12074.
- Yixin Ji, Kaixin Wu, Juntao Li, Wei Chen, Mingjie Zhong, Xu Jia, and Min Zhang. 2024. Retrieval and reasoning on kgs: Integrate knowledge graphs into large language models for complex question answering. In *Findings of the Association for Computational Linguistics: EMNLP 2024, Miami, Florida, USA, November 12-16, 2024*, pages 7598–7610. Association for Computational Linguistics.
- Tingsong Jiang, Tianyu Liu, Tao Ge, Lei Sha, Baobao Chang, Sujian Li, and Zhifang Sui. 2016. Towards time-aware knowledge graph completion. In *COLING 2016*, pages 1715–1724. ACL.
- Woojeong Jin, Meng Qu, Xisen Jin, and Xiang Ren. 2020. Recurrent event network: Autoregressive structure inference over temporal knowledge graphs. In *EMNLP 2020*, pages 6669–6683. Association for Computational Linguistics.
- Julien Leblay and Melisachew Wudage Chekol. 2018. Deriving validity time in knowledge graph. In *WWW 2018*, pages 1771–1776. ACM.
- 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.
- Zixuan Li, Xiaolong Jin, Wei Li, Saiping Guan, Jiafeng Guo, Huawei Shen, Yuanzhuo Wang, and Xueqi Cheng. 2021. Temporal knowledge graph reasoning based on evolutionary representation learning. In *SIGIR ’21*, pages 408–417. ACM.

- Yuejiang Liu, Parth Kothari, Bastien van Delft, Baptiste Bellot-Gurlet, Taylor Mordan, and Alexandre Alahi. 2021. TTT++: when does self-supervised test-time training fail or thrive? In *NeurIPS 2021*, pages 21808–21820.
- Yushan Liu, Yunpu Ma, Marcel Hildebrandt, Mitchell Joblin, and Volker Tresp. 2022. Tlogic: Temporal logical rules for explainable link forecasting on temporal knowledge graphs. In *AAAI 2022*, pages 4120–4127. AAAI Press.
- Farzaneh Mahdisoltani, Joanna Biega, and Fabian M. Suchanek. 2015. YAGO3: A knowledge base from multilingual wikipedias. In *Seventh Biennial Conference on Innovative Data Systems Research, CIDR 2015*. www.cidrdb.org.
- Lingyuan Meng, Ke Liang, Hao Yu, Yue Liu, Sihang Zhou, Meng Liu, and Xinwang Liu. 2024. Fedean: Entity-aware adversarial negative sampling for federated knowledge graph reasoning. *IEEE Trans. Knowl. Data Eng.*, 36(12):8206–8219.
- 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 *WSDM '22*, pages 794–803. ACM.
- Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Z. Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, and 2 others. 2019. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8–14, 2019, Vancouver, BC, Canada*, pages 8024–8035.
- Daniel Vitor Ruiz, Bruno A. Krinski, and Eduardo Todt. 2019. ANDA: A novel data augmentation technique applied to salient object detection. In *ICAR 2019*, pages 487–492. IEEE.
- Shahriar Shayesteh and Diana Inkpen. 2022. Generative adversarial learning with negative data augmentation for semi-supervised text classification. In *FLAIRS 2022*.
- Yuehang Si, Xingchen Hu, Qing Cheng, Xinwang Liu, Shixuan Liu, and Jincai Huang. 2025. Coherence mode: Characterizing local graph structural information for temporal knowledge graph. *Inf. Sci.*, 686:121357.
- Haohai Sun, Jialun Zhong, Yunpu Ma, Zhen Han, and Kun He. 2021. Timetraveler: Reinforcement learning for temporal knowledge graph forecasting. In *EMNLP 2021*, pages 8306–8319. Association for Computational Linguistics.
- Yu Sun, Xiaolong Wang, Zhuang Liu, John Miller, Alexei A. Efros, and Moritz Hardt. 2019a. Test-time training for out-of-distribution generalization. *CoRR*, abs/1909.13231.
- Yu Sun, Xiaolong Wang, Zhuang Liu, John Miller, Alexei A. Efros, and Moritz Hardt. 2020. Test-time training with self-supervision for generalization under distribution shifts. In *ICML 2020*, volume 119, pages 9229–9248. PMLR.
- Zhiqing Sun, Zhi-Hong Deng, Jian-Yun Nie, and Jian Tang. 2019b. Rotate: Knowledge graph embedding by relational rotation in complex space. In *ICLR 2019*. OpenReview.net.
- Rakshit Trivedi, Hanjun Dai, Yichen Wang, and Le Song. 2017. Know-evolve: Deep temporal reasoning for dynamic knowledge graphs. In *ICML 2017*, volume 70, pages 3462–3471. PMLR.
- Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. 2016. Complex embeddings for simple link prediction. In *ICML 2016*, volume 48, pages 2071–2080. JMLR.org.
- Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014. Knowledge graph embedding by translating on hyperplanes. In *Proceedings of the AAAI conference on artificial intelligence*, volume 28.
- Yi Xu, Junjie Ou, Hui Xu, and Luoyi Fu. 2023. Temporal knowledge graph reasoning with historical contrastive learning. In *AAAI 2023*, pages 4765–4773. AAAI Press.
- Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. 2015. Embedding entities and relations for learning and inference in knowledge bases. In *ICLR 2015*.
- Xihong Yang, Yiqi Wang, Jin Chen, Wenqi Fan, Xiangyu Zhao, En Zhu, Xinwang Liu, and Defu Lian. 2024. Dual test-time training for out-of-distribution recommender system. *CoRR*, abs/2407.15620.
- Lupeng Yue, Yongjian Ren, Yan Zeng, Jilin Zhang, Kaisheng Zeng, Jian Wan, and Mingyao Zhou. 2024. Complex expressional characterizations learning based on block decomposition for temporal knowledge graph completion. *Knowl. Based Syst.*, 290:111591.
- Jiaxin Zhang, Yiqi Wang, Xihong Yang, Siwei Wang, Yu Feng, Yu Shi, Ruichao Ren, En Zhu, and Xinwang Liu. 2024. Test-time training on graphs with large language models (llms). In *ACM MM 2024*, pages 2089–2098. ACM.
- Cunchao Zhu, Muham Chen, Changjun Fan, Guangquan Cheng, and Yan Zhang. 2021. Learning from history: Modeling temporal knowledge graphs with sequential copy-generation networks. In *AAAI 2021*, pages 4732–4740. AAAI Press.

A Appendix

A.1 Implementation Details

The entire experimental implementation of T3DM is executed on a computational setup comprising an Intel (R) Core i9-10900K CPU and an NVIDIA GeForce RTX 3090 Ti GPU, based on the KGE open-source framework of PyTorch (Paszke et al., 2019). We focus on TTransE as the generator model and add the consideration of HyTE and DE-SimplE in the comparison. During training, the batch size of T3DM is set to 512, and the maximum training epoch limit is 1000. We choose Adam as the optimiser, and the learning rate is compared within a set of predefined values {0.0001, 0.0005, 0.001, 0.01} to select the optimal one. The input sequence length of LSTM is set to 20.

A.2 Evaluation Metrics

To evaluate the performance of TKGR models, we employ standard evaluation metrics like Mean Reciprocal Rank (MRR) and Hits@K. MRR represents the average of the inverse rankings of all samples:

$$\text{MRR} = \frac{1}{2 \cdot N(\mathcal{G})} \sum_{o,s \in \mathcal{G}} \left(\frac{1}{\text{RK}(o_t|o_p)} + \frac{1}{\text{RK}(s_t|s_p)} \right), \quad (9)$$

where, \mathcal{G} denotes the quadruple set, $N(\cdot)$ denotes the number of elements, o_t and s_t denote the true entities, while o_p and s_p denote the predicted entities. $\text{RK}(\cdot)$ denotes the recommendation ranking of the correct answer. Hits@K (where K=1,3,10) represents the proportion of test samples which are ranked in top K positions:

$$\begin{aligned} \text{Hits@K} = & \frac{1}{2 \cdot N(\mathcal{G})} \sum_{o,s \in \mathcal{G}} (\mathbf{1}\{\text{RK}(o_t|o_p) \leq K\} \\ & + \mathbf{1}\{\text{RK}(s_t|s_p) \leq K\}). \end{aligned} \quad (10)$$

A.3 Statistics of Datasets

Five public datasets differ in their factual representations: facts in GDELT and ICEWS are based on a specific time point, while facts in YAGO and Wikidata are based on time intervals. Statistics details of five datasets are shown in Table 4.

A.4 “Bern” sampling implementation details.

We set different probabilities for replacing the head or tail when corrupting the quadruples, which depends on the mapping property of relation. We

Table 4: Statistics details of five publicly available datasets.

Dataset	Entities	Relation	Time	Training	Validation	Test	Interval
ICEWS14	12,498	260	365	323,895	-	341,409	24 hours
ICEWS18	23,033	256	304	373,018	45,995	49,545	24 hours
GDELT	7,691	240	2,751	1,734,399	238,765	305,241	15 mins
WIKI	12,554	24	232	539,286	67,538	63,110	1 year
YAGO	10,623	10	189	161,540	19,523	20,026	1 year

tend to give more chance to replacing the head if the relation is 1-to-N and replace the tail if N-to-1. In this way, the chance of generating false negative labels is reduced. Specifically, the average number of tail per head is denoted as N_t , and the average number of head per tail is denoted as N_h . We corrupt the quadruple by replacing head with probability $\frac{N_t}{N_t+N_h}$, and replacing tail with probability $\frac{N_h}{N_t+N_h}$.