Re-Temp: Relation-Aware Temporal Representation Learning for Temporal Knowledge Graph Completion

Kunze Wang[1] **Soyeon Caren Han**[1,2*] **Josiah Poon**[1]
[1]The University of Sydney
[2]The University of Western Australia
kwan4418@uni.sydney.edu.au {caren.han, josiah.poon}@sydney.edu.au

Abstract

Temporal Knowledge Graph Completion (TKGC) under the extrapolation setting aims to predict the missing entity from a fact in the future, posing a challenge that aligns more closely with real-world prediction problems. Existing research mostly encodes entities and relations using sequential graph neural networks applied to recent snapshots. However, these approaches tend to overlook the ability to skip irrelevant snapshots according to entity-related relations in the query and disregard the importance of explicit temporal information. To address this, we propose our model, Re-Temp (Relation-Aware Temporal Representation Learning), which leverages explicit temporal embedding as input and incorporates skip information flow after each timestamp to skip unnecessary information for prediction. Additionally, we introduce a two-phase forward propagation method to prevent information leakage. Through the evaluation on six TKGC (extrapolation) datasets, we demonstrate that our model outperforms all eight recent state-of-the-art models by a significant margin.

1 Introduction

A Knowledge Graph (KG) is a graph-structure database, composed of facts represented by triplets in the form of (*Subject Entity, Relation, Object Entity*) such as (*Alice, Is a Friend of, Bob*). In this graph, entities serve as nodes, and relations are depicted as direct edges connecting the nodes. However, facts in a KG are not static but undergo continuous updates over time. To incorporate temporal information into the KG, Temporal Knowledge Graphs (TKGs) are introduced. TKGs add the extra temporal information of each fact and extend each triple with a timestamp as a quadruplet (*Subject Entity, Relation, Object Entity, Timestamp*). A TKG can be represented as a sequence of snapshots, where each snapshot represents a static knowledge graph for one specific timestamp.

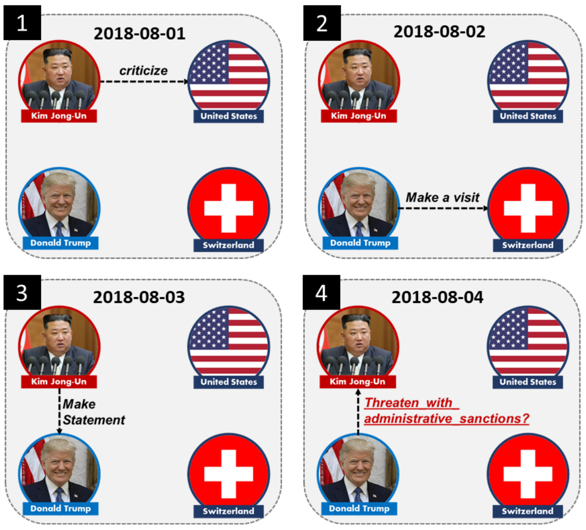

Figure 1: A case study of temporal knowledge graph completion under the extrapolation setting

Temporal Knowledge Graph Completion (TKGC) aims to predict the missing entity from a query (*Subject Entity, Relation, ?, Timestamp*) or (*?, Relation, Object Entity, Timestamp*). TKGC is difficult and even large-scale pre-trained language models such as ChatGPT(OpenAI, 2022) are prone to making factual errors(Borji, 2023). There are two main settings: interpolation and extrapolation setting. TKGC under the interpolation setting completes the facts in history, while TKGC under the extrapolation setting predicts facts at future timestamps. In this paper, we focus on TKGC in the extrapolation setting, which is more challenging and requires further improvement(Jin et al., 2020).

Enormous attention has been focused on static KGC problems, and numerous models have been employed to encode entities and relations. However, a key question remains: how can a static KGC model be extended to incorporate temporal infor-

[*]Corresponding author. caren.han@sydney.edu.au

mation for TKGC tasks? Recent works(Jin et al., 2020; Li et al., 2021, 2022a,b) have utilised sequential Graph Neural Networks (GNNs) to the previous snapshots for encoding the entities and relations. Then, they use a static score function as the decoder to assess the score of each candidate. Sequential GNNs are used because the facts shown in recent history can be helpful when making predictions in the future. An example is shown in Figure 1, the previous facts (*Kim Jong-Un, criticize, United States*) three days before and (*Kim Jong-Un, Make Statement, Donald Trump*) one day before may imply (*Donald Trump, Threaten with administrative sanction, Kim Jong-Un*) today. Since no explicit timestamp value is used, we can call it "implicit temporal information".

However, to effectively encode the timestamp, the temporal information, two additional considerations arise: First, explicit temporal information is crucial. For instance, the validity score of (Donald Trump, Threaten with administrative sanction, Kim Jong-un) may differ between 2018 and 2023 as Donald Trump was the president in 2018 but not in 2023, affecting his ability to threaten another nation with administrative sanctions in 2023. The nature of entities can change over time, necessitating the consideration of explicit temporal information to encode time-dependent factors. Second, not all the facts in the recent history are relevant. Given historical facts (Kim Jong-un, criticize, United States, 2018-08-01),(Donald Trump, Make a visit, Switzerland, 2018-08-02) and (Kim Jong-un, Make Statement, Donald Trump, 2018-08-03), when calculating the score of (Donald Trump, Threaten with administrative sanction, Kim Jong-un, 2018-08-01), the second quadruplet visiting Switzerland does not contribute to the prediction of the relation between Donald Trump and Kim Jong-un since Switzerland is neutral. In such case, the model should find a way to skip the irrelevant snapshots based on the entity-related relation in the query. Therefore, an optimal TKGC model should consider (1) explicit temporal information and (2) implicit temporal information with skipping irrelevant snapshots by considering the query.

In this paper, we propose Re-Temp, an innovative TKGC model designed for extrapolation settings that incorporates relation-aware temporal representation learning. The encoder of Re-Temp utilises explicit temporal embedding for each entity, combining static and dynamic embedding. Within the encoder, a sequential GNN is employed to capture the implicit temporal information with a skip information flow applied after each timestamp, taking into account the entity-related relation in the query. The main contributions of this paper can be summarised as follows:

- We introduce Re-Temp, a precise TKGC model that leverages both explicit and implicit temporal information, incorporates a relation-aware skip information flow to exclude irrelevant information and adopts a two-phase forward propagation method to prevent information leakage[1].
- We compare our Re-Temp against eight state-of-the-art baseline models from recent years using six publicly available TKGC datasets under the extrapolation setting. Our experimental results demonstrate that Re-Temp outperforms all of the baselines significantly.
- We conduct a detailed case study and statistical analysis to illustrate the distinct characteristics of each dataset and provide an explanation based on our experimental findings.

2 Related Work

KGC models normally adopt an encoder-decoder framework(Hamilton et al., 2017), where the encoder generates the embedding of entities and relations and the score function plays as a decoder. Most of the existing works extend the static KGC models into TKGC models by introducing temporal information.

2.1 TKGC(Interpolation)

To integrate the temporal information in the decoder, TTransE(Jiang et al., 2016) extends TransE(Bordes et al., 2013) with the summation of an extra timestamp embedding, and ConT(Ma et al., 2019) extends Tucker(Balažević et al., 2019) by replacing the learnable weight with the timestamp embedding. Some methods also focus on combining temporal information in the encoder: TA-DistMult(Garcia-Duran et al., 2018) encodes the temporal information into relation embedding by using LSTM, while DE-SimplE(Goel et al., 2020) encodes a diachronic entity embedding with temporal information. with decoders as DistMult and

[1]Code available at: https://github.com/adlnlp/re-temp

Table 1: Summary of TKGC(extrapolation) models and our proposed model. The column'Temporal' presents the trend of the approach to how the temporal information is used, and the column 'Query' shows the summary of the approach to how the model utilises query.

Method	**Core idea**	**Temporal**	**Query**
RE-NET(Jin et al., 2020)	estimate the future graph distribution	implicit	N/A
CyGNet(Zhu et al., 2021)	identify facts with repetition	explicit	repetitive queries
xERTE(Han et al., 2020)	sample subgraph according to query	implicit	query-related subgraph
REGCN(Li et al., 2021)	relation-GCN + GRU	implicit	N/A
TANGO(Han et al., 2021)	neural ODE on continuous-time reasoning	implicit	N/A
TITER(Haohai Sun, 2021)	path-based reinforcement learning	implicit	query-related path
CEN(Li et al., 2022a)	ensemble model with different history lengths	implicit	N/A
HiSMatch(Li et al., 2022b)	two separated encoders for entity and query information	implicit	repetitive queries
Re-Temp (Ours)	skip irrelevant information according to entity-related relations	both	query-related skip information flow

SimplE(Yang et al., 2015; Kazemi and Poole, 2018) accordingly. These models produced relatively lower performance on TKGC under the extrapolation setting tasks since they are unable to capture unseen temporal information.

2.2 TKGC(extrapolation)

For the last few years, more attention has been paid to TKGC tasks under the extrapolation setting. GNNs are typically used as the encoder: RE-NET(Jin et al., 2020) applies sequential neighbourhood aggregators such as R-GCN(Schlichtkrull et al., 2018) to get the distribution of the target timestamp snapshot, REGCN(Li et al., 2021) adopts CompGCN(Vashishth et al., 2020) at each timestamp and GRU for sequential information. CEN(Li et al., 2022a) uses an ensemble model of sequential GNNs with different history lengths, TANGO(Han et al., 2021) solves Neural Ordinary Equations and makes it as the input of a Multi-Relational GCN, and HiSMatch(Li et al., 2022b) builds two GNN encoders modelling the sequential candidate graph and query-related subgraphs separately and combines the representation from both sides into a matching function. Meanwhile, some methods do not follow the traditional encoder and decoder framework. xERTE(Han et al., 2020) extracts subgraph according to queries, CyGNet(Zhu et al., 2021) identifies the candidates with repetition, and TITer(Haohai Sun, 2021) uses reinforcement learning methods to search for the temporal evidence chain for prediction. To conclude, RE-NET, REGCN, and CEN adopt the entity evolvement information, while xERTE, CyGNet and TITer focus on the query. HiSMatch combines these two types of information with two separate encoders. However, none of the previous works encoded sequential and query-related information in one precise encoder. In addition to this, none of these methods considers explicit temporal information, except for CyGNet, which generates an independent timestamp vector but does not encode it into the entity or relation. Table 1 presents the summary of TKGC(extrapolation) models and emphasises the contribution of our proposed model.

3 Re-Temp

The overall architecture of Re-Temp can be found in Figure 2. Section 3.1 describes the notations of a TKGC task. The input of the model is represented by a combination of static and dynamic entity embedding, in Section 3.2, showing explicit temporal information. The encoder in Section 3.3 uses a sequential multi-relational GNN to learn implicit temporal information and after each timestamp, a relation-aware skip information flow mechanism is applied to retain the necessary information for prediction. The ConvTransE decoder together with the loss function is introduced in Section 3.4. To avoid information leaking, we apply a two-phase forward propagation method in Section 3.5.

3.1 Problem Formulation

To denote the set of entities, relations, timestamps and facts, $\mathcal{E}, \mathcal{R}, \mathcal{T}$ and $\mathcal{F}$ are selected. A temporal knowledge graph G can be treated as $|\mathcal{T}|$ sequential snapshots, $G = \{G_0, G_1, ..., G_T\}$, where $G_t = \{\mathcal{E}, \mathcal{R}, \mathcal{F}_t\}$ is a directed multi-relational graph at timestamp t. For each fact, a quadruplet is represented as (e_s, r, e_o, t), where $e_s, e_o \in \mathcal{E}$ are the subject and object entities, $r \in \mathcal{R}$ represents the relation and $t \in \mathcal{T}$ is the timestamp. The target of the temporal knowledge graph completion under the extrapolation setting is that for a query q, predicting $(e_s, r, ?, t_q)$ or $(?, r, e_o, t_q)$ given previous snapshots $\{G_0, G_1, ..., G_{t_q-1}\}$. Normally, the inverse of each quadruplet is added to the dataset, making all subject entity prediction prob-

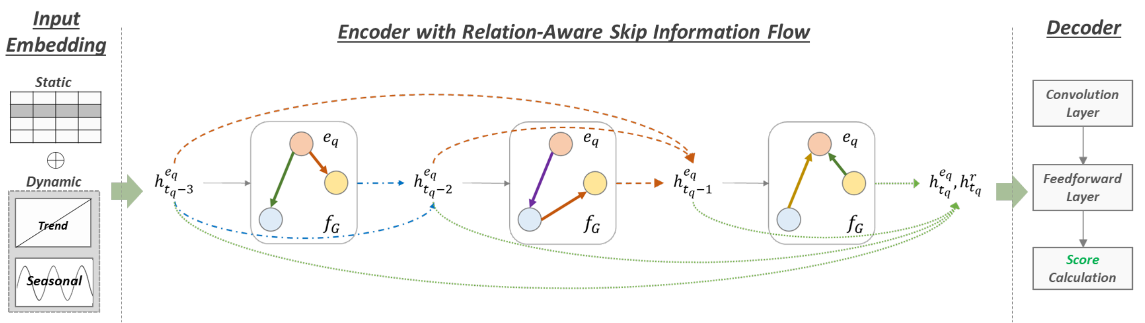

Figure 2: Illustration of Encoding and Decoding process in Re-Temp with history length as 3. For a query q, the input vector is $h^{e_q}_{t_q-3}$. The encoder with relation-aware skip information flow learns the entity and relation representation $h^{e_q}_{t_q}$ and h_{r_q}. Then the decoder measures the score of all the candidates.

lem $(?, r, e_o, t_q)$ into object entity prediction problem $(e_o, r^{-1}, ?, t_q)$.

3.2 Explicit Temporal Representation

For sequential snapshots with length k, let $h^{e_q}_{t_q-k} \in \mathbb{R}^{1\times d}$ denotes the input embedding of the subject entity e_q from query q, and d is the dimension of the input. In order to encode the explicit temporal information, we concatenated two kinds of input embedding; static and dynamic embedding. The static embedding reveals the nature of an entity that does not change through time, while the dynamic part reveals the time-dependent information.

Inspired by ATiSE(Xu et al., 2020), the dynamic embedding is decomposed into the trend component and seasonal component, and the trend component can be represented as a linear transformation on t while the seasonal component should be a periodical function of t. Thus, we model the dynamic temporal embedding at timestamp t by the summation of trend embedding $w_{e_q,0}t$ and seasonal embedding $sin(2\pi w_{e_q,1}t)$. After concatenation with the static embedding, a feed-forward layer is applied. Formally, the input of the encoder $h^{e_q}_{t_q-k}$ is derived by:

$$h^{e_q,S}_{t_q-k} = h^{e_q,S} \quad (1)$$

$$h^{e_q,D}_{t_q-k} = w_{e_q,0}(t_q-k)+sin(2\pi w_{e_q,1}(t_q-k)) \quad (2)$$

$$h^{e_q}_{t_q-k} = W_{tmp}(h^{e_q,S}_{t_q-k} \oplus h^{e_q,D}_{t_q-k}) \quad (3)$$

where $h^{e_q,S}_{t_q-k}$ in Equation 1 and $h^{e_q,D}_{t_q-k}$ in Equation 2 denote the static and dynamic embedding for subject entity e_q at timestamp $t_q - k$, $\oplus$ denotes the concatenation, and $h^{e_q,s}$, $w_{e_q,0}$, $w_{e_q,1}$, W_{tmp} are learnable parameters. The major difference between our explicit temporal representation and ATiSE lies in the fact that employing a learnable feed-forward layer to concatenate the dynamic embedding and static embedding, enables the model to determine the extent to which it should utilise information from each embedding rather than simply utilising both. Relation embedding h^r can simply be extracted from a static embedding lookup table since we do not expect the relation's nature to evolve through time.

3.3 Relation-Aware Skip Information Flow

In order to handle implicit temporal information, we use a sequential GNN-based encoder with a new relation-aware skip information flow mechanism. Following recent work(Li et al., 2021, 2022a,b), we adopt a variant of CompGCN(Vashishth et al., 2020) at each timestamp to model the multi-relational snapshot, outputting the entity embedding h_e and the relation embedding h_r. The details of CompGCN are shown in Appendix A.1.

Not all snapshots in the recent history are useful in predicting query q, hence, a relation-aware skip information flow is applied. Two things are considered: (1) Skip connection is used for filtering out the unnecessary information from each timestamp. (2) Relation-aware attention mechanism helps to determine whether some information should be filtered. Thus, after getting the output of CompGCN, they will be weighted-summed up with previous timestamps input to partially skip the irrelevant snapshots. The weights of the weighted sum are calculated by considering both the entity and the entity-related relation in the query.

Formally, for an entity e_q, the relation associated with e_q should be considered. To capture the entity-related relation information, mean pooling is applied on all relation embedding associated with e_q at timestamp t_q. The representation obtained

Table 2: Statistics Details of Benchmark Dataset

	ICEWS14	ICEWS18	ICEWS05-15	ICEWS14*	GDELT	WIKI
# Entities	7,128	23,033	10,094	7,128	7,691	12,554
# Relations	230	256	251	230	240	24
# Facts	89,730	468,558	461,329	90,730	2,277,405	669,934
# Snapshots	365	304	4,017	365	2,976	232
# Snapshots in Train/Val/Test set	304/30/31	240/30/34	3,243/404/370	262/52/51	2,304/288/384	211/11/10
# Facts per Snapshot	245.8	1541.3	32.2	248.6	765.3	2887.6
Time Interval	1 day	1 day	1 day	1 day	15 mins	1 year
Total Time Range	1 year	0.83 years	11 years	1 year	0.54 years	232 years

from mean pooling will serve as a reference vector to help the model determine the information to keep or skip. Then, this average relation embedding will be summed with all m previous timestamps one by one, followed by a feedforward layer. This calculation can also be treated as additive attention. After getting the attention weights $\beta_j^{e_q}$, the weighted sum using these attention weights is applied on the current CompGCN output $h_{t_i}^{e_q,L}$ and all m previous timestamp inputs. The detailed calculation shows as follows:

$$h_{r,t_q}^{e_q} = \frac{1}{|R_{t_q}^{e_q}|} \sum_{r \in R_{t_q}^{e_q}} h_r \tag{4}$$

$$attn_j^{e_q} = \begin{cases} \mathbf{0} & j = 0 \\ W_a(h_{t_i-j}^{e_q} + h_{r,t_q}^{e_q}) & j \in [1, m] \end{cases} \tag{5}$$

$$\beta_j^{e_q} = softmax(attn_j^{e_q}), j \in [0, m] \tag{6}$$

$$h_{t_i+1}^{e_q} = \beta_0^{e_q} h_{t_i}^{e_q,L} + \sum_{j=1}^{m} \beta_j^{e_q} h_{t_i-j}^{e_q} \tag{7}$$

Note that the output of each timestamp is also the input of the next timestamp. Equation 4 shows the entity-associated relation embedding and $R_{t_q}^{e_q}$ denotes the relation set which connects with entity e_q at timestamp t_q. Equation 5 and 6 denotes the attention score and weight calculation where W_a is learnable. By applying the relation-aware skip information flow, our model is capable of skipping irrelevant snapshots by considering the target query relations.

3.4 Decoder

ConvTransE(Shang et al., 2019) is widely used in both static KGC(Malaviya et al., 2020) and TKGC(Li et al., 2022b) as the score function, and ours is no exception. After getting the score of each candidate using ConvTransE, we train the model as a classification problem and the loss function for each query shows as follows:

$$L = -\sum_{e_c \in \mathcal{E}} z_c log(s(e_q, r_q, e_c, t_q)) \tag{8}$$

and z_c will be 1 if correctly classified, otherwise, it is 0. The training target is to minimise the total loss for all queries. Appendix A.2 introduces the details of ConvTransE.

3.5 Two-Phase Propagation

There is a potential information leakage problem by applying the relation-aware information flow mechanism. Suppose a query in the test set is (A, r, B, t), after adding the inverse of quadruplets, (B, r^{-1}, A, t) will be in the test set. When applying the encoder, with the relation-aware skip information flow, A and B will contain the information of r and r^{-1} accordingly. Therefore, when making predictions on $(A, r, ?, t)$ and calculating the score by dot product A and all candidates, there is a chance that the information of r in A can meet the information of r^{-1} in B. Since r and r^{-1} are paired, the model might find a shortcut to determine B is the right answer for $(A, r, ?, t)$. This information leakage will result in unreasonably high performance during evaluation.

To avoid such information leakage, we propose a two-phase forward propagation method. We divide the dataset into two subsets: the original set and the inverse set. The inverse set is the set of inverse quadruplets. The snapshot graph in the history will be built on the whole set, while during forward propagation, the original set and inverse set are used separately. The output of the original set and the inverse set will be collected for loss calculation or performance evaluation.

4 Experiments

4.1 Experiment Setup

Datasets We evaluated our model on six widely-used TKG datasets: *ICEWS14*(Li et al., 2021), *ICEWS18*(Jin et al., 2020), *ICEWS05-15*(Han et al., 2020), *ICEWS14**(Han et al., 2020), *GDELT*(Jin et al., 2020), and *WIKI*(Leblay and Chekol, 2018). The overall statistics of each dataset are presented in Table 2. All datasets are split into the Training, Validation and Test sets in chronological order. For example, the timestamps in **ICEWS14** are from 1st to 304th, from 305th to 334th and from 335th to 365th for training, validation and test set accordingly.

- **ICEWS14**, **ICEWS18**, **ICEWS05-15**, **ICEWS14*** are extracted from Integrated Crisis Early Warning System which is a database system recording political events. **14**, **18**, **05-15** represent the year of the dataset(2014, 2018, 2005-2015), and **ICEWS14*** uses a different split compared with **ICEWS14**. The time interval of **ICEWS** is 1 day. A sample from **ICEWS** datasets is (John_Kerry, Host_a_visit, Benjamin_Netanyahu, 2014-01-01)

- **GDELT** is also a political event temporal knowledge graph dataset from the **G**lobal **D**atabase of **E**vents, **L**anguage, and **T**one(Leetaru and Schrodt). Compared with **ICEWS** datasets, its time interval is only 15 minutes and **GDELT** is collected from a wider variety of sources. (Minist, Return, Nigeria, 0) is a sample in **GDELT**.

- **WIKI** is from Wikidata, an open knowledge base and not limited to political events. The temporal representation in the facts from Wikidata is not a single date/year but a range. For example, the fact (Wang Shu, educated at, Southeast University) is valid from 1981 to 1988. To represent a such range, **WIKI** generates eight quadruplets across eight snapshots during 1981-1988.

All the datasets are consistent with their intended use.

Baselines Our Re-Temp is compared with TKGC models under the extrapolation setting. Eight models from recent years are selected as baselines: **RE-NET**(Jin et al., 2020), *RE-GCN*(Li et al., 2021), *CyGNet*(Zhu et al., 2021), *xERTE*(Han et al., 2020), *TITer*(Haohai Sun, 2021), *TANGO*(Han et al., 2021), *CEN*(Li et al., 2022a), and *HiSMatch*(Li et al., 2022b). Models that are designed for static KG completion or TKGC under the interpolation setting tasks are not compared since they naturally perform badly in TKGC under the extrapolation setting tasks.

Hyperparameter Following the previous works(Li et al., 2022a,b), the dimension of the input is set to 200, which is also the hidden dimension of the graph model and decoder hidden dimension. The number of graph neural network layers is 2 and the dropout rate is set to 0.2. Adam(Kingma and Ba, 2015) with a learning rate of 1e-3 is used for optimisation. The model is trained on the training set with a maximum of 30 epochs and we stop training when the validation performance doesn't improve in 5 consecutive epochs. Then, the test set is evaluated using the trained model.

Evaluation Metrics Following the previous works(Han et al., 2020; Zhu et al., 2021; Li et al., 2022b), we employ widely used evaluation metrics, Mean Reciprocal Rank(MRR), hits@1, hits@3, and hits@10, which is explained in Appendix B.2. We adopt the way of filtering out the quadruplets occurring at the query time, followed by Haohai Sun (2021); Han et al. (2021), and we report the five-times running average result.

4.2 Performance Comparison

We use a history length of 3 for ICES14, ICEWS18, ICEWS05-15, ICEWS14* and GDELT, while 1 for WIKI. The influence of history length is discussed in Section 4.3. Table 3 presents the performance comparison of all baseline models. Our model, Re-Temp, outperforms significantly almost all the baseline models on all datasets, indicating the superiority of our Re-Temp model. In detail, three points can be observed:

Firstly, HiSMatch(Li et al., 2022b) achieved the second-highest performance on most of the datasets by considering both the query subgraph and entity subgraph. The concept considering both query and entity of HiSMatch is similar to our relation-aware attention mechanism in the skip information flow. However, HiSMatch only builds the query subgraph using the exact same relation of the query, which ignores the potential similarity between relations. For example, in ICEWS14, when making a prediction on (A, provide_aid, ?, t_q), relation 'provide_aid' and 'provide_military_aid' share similarities, but HisMatch only considers the entity with 'provide_military_aid' in the recent history while our method uses the embedding of relation to calculate the attention weights, making it gen-

Table 3: Performance(%) with Baseline models. The highest value is bold and the second highest is underlined.

Model	ICEWS14				ICEWS18				ICEWS05-15			
	MRR	**hits@1**	**hits@3**	**hits@10**	**MRR**	**hits@1**	**hits@3**	**hits@10**	**MRR**	**hits@1**	**hits@3**	**hits@10**
RE-NET(Jin et al., 2020)	37.01	27.02	39.66	54.85	29.02	20.03	33.14	48.60	44.03	34.43	49.03	64.03
CyGNet(Zhu et al., 2021)	35.02	25.72	39.06	53.50	25.03	16.03	29.28	43.42	37.03	27.01	42.23	56.98
xERTE(Han et al., 2020)	40.12	32.11	44.73	56.25	29.31	21.03	33.51	46.48	46.62	37.84	52.31	63.92
REGCN(Li et al., 2021)	41.50	30.86	46.60	62.47	30.55	20.00	34.73	51.46	46.41	35.17	52.76	67.64
TANGO(Han et al., 2021)	30.12	23.03	35.48	52.32	28.97	19.51	32.61	47.51	42.86	32.72	48.14	62.34
TITer(Haohai Sun, 2021)	41.73	32.74	46.46	58.44	29.96	22.06	33.41	44.92	47.78	38.05	53.11	65.93
CEN(Li et al., 2022a)	42.20	32.08	47.46	61.31	31.50	21.70	35.44	50.59	45.97	35.56	51.45	66.14
HiSMatch(Li et al., 2022b)	46.42	35.91	51.63	66.84	33.99	23.91	37.90	53.94	52.85	42.01	59.05	73.28
Re-Temp (Ours)	**48.04**	**37.32**	**53.60**	**68.90**	**35.82**	**25.02**	**40.36**	**57.30**	**56.30**	**45.49**	**62.80**	**77.17**

Model	ICEWS14*				GDELT				WIKI			
	MRR	**hits@1**	**hits@3**	**hits@10**	**MRR**	**hits@1**	**hits@3**	**hits@10**	**MRR**	**hits@1**	**hits@3**	**hits@10**
RE-NET(Jin et al., 2020)	38.28	28.68	41.43	54.52	19.63	12.39	21.03	34.02	49.66	46.98	51.23	53.49
CyGNet(Zhu et al., 2021)	33.13	24.16	37.02	51.23	18.98	12.32	20.56	33.89	43.78	39.02	46.12	51.92
xERTE(Han et al., 2020)	40.77	32.65	45.71	57.29	18.07	12.31	20.05	30.32	71.16	68.03	76.15	78.99
REGCN(Li et al., 2021)	41.79	31.55	46.67	61.53	19.31	11.99	20.61	33.59	77.58	73.72	80.39	83.69
TANGO(Han et al., 2021)	26.35	17.33	29.27	44.32	18.03	12.36	19.96	29.31	51.15	49.65	52.26	53.44
TITer(Haohai Sun, 2021)	41.76	32.69	46.35	58.46	17.02	11.23	19.81	26.92	75.51	72.98	77.51	79.32
CEN(Li et al., 2022a)	40.78	31.26	45.26	59.16	19.89	12.61	21.16	34.09	77.65	73.86	80.69	84.00
HiSMatch(Li et al., 2022b)	45.82	35.84	50.79	65.08	22.01	14.45	23.80	36.61	78.07	73.89	81.32	**84.65**
Re-Temp (Ours)	**46.40**	**35.86**	**51.69**	**67.12**	**25.05**	**15.70**	**27.14**	**44.16**	**78.51**	**74.80**	**81.33**	84.50

Table 4: Cases from WIKI Dataset about Lionel Messi from Year 2003 to Year 2005.

Subject Entity	**Relation**	**Object Entity**	**Year**
Lionel Messi	residence	Barcelona	2003
Lionel Messi	member of sports team	FC Barcelona C	2003
Lionel Messi	residence	Barcelona	2004
Lionel Messi	member of sports team	FC Barcelona C	2004
Lionel Messi	member of sports team	FC Barcelona Atlètic	2004
Lionel Messi	residence	Barcelona	2005
Lionel Messi	member of sports team	Argentina national football team	2005

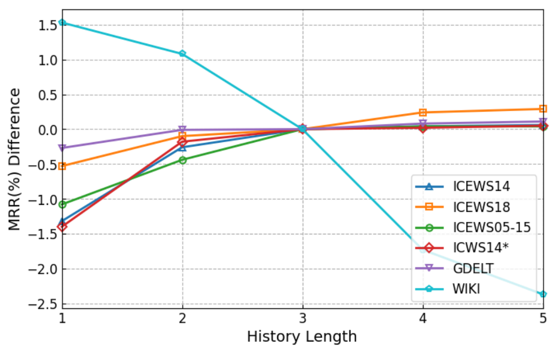

Figure 3: MRR(%) change of Re-Temp with the history lengths. The x-axis is the history length and the y-axis is the MRR(%) change compared with history length 3.

eral for different types of relations that are close in the embedding space and outperforming HiSMatch. Meanwhile, HiSMatch builds two separate encoders and fuses the output for the decoder while our model only applies one encoder for better information alignment.

Secondly, among four ICEWS datasets, our model achieves more improvement on ICEWS05-15. As shown in Table 2, the snapshots in ICEWS05-15 are sparser than others, showing the ability of our model to learn sequential information with less data.

Thirdly, our model only achieves a comparable performance with HiSMatch on WIKI, which might result from the nature of this dataset. Table 4 lists some cases of facts about Lionel Messi in WIKI. Suppose giving the quadruplets from 2003 and 2004, it is relatively easy to predict (Lionel Messi, residence, ?, 2005) based on his previous residence, however, it is almost impossible to have a correct prediction on (Lionel Messi, residence, ? , 2005) since the previous snapshots don't provide enough information on Argentina national football team. This is an issue in WIKI: the predictions are either too easy (using the previous facts), or too difficult (even humans can not make a correct prediction without any external knowledge). Thus, a relatively better model is not enough to generate an undoubtful better performance on WIKI, and our model and some previous baseline models (CEN, HiSMatch) share similar results on this dataset.

4.3 Impact of history length

To study the impact of history length on different datasets, experiments with different history lengths are conducted. The default value of history length is 3 and the MRR changes in percentage are shown in Figure 3 with history lengths from 1 to 5. Two

Table 5: The MRR(%) result of the ablation test of Re-Temp. The highest value is bold.

Model	ICEWS14	ICEWS18	ICEWS05-15	ICEWS14*	GDELT	WIKI
Re-Temp	**48.04**	**35.82**	**56.30**	**46.40**	**25.05**	78.51
- *dynamic*	47.52	35.33	55.12	45.89	24.85	76.04
- *relation_aware*	39.93	30.56	44.95	38.75	19.92	78.14
- *skip*	36.56	28.07	43.80	36.30	18.61	**79.60**

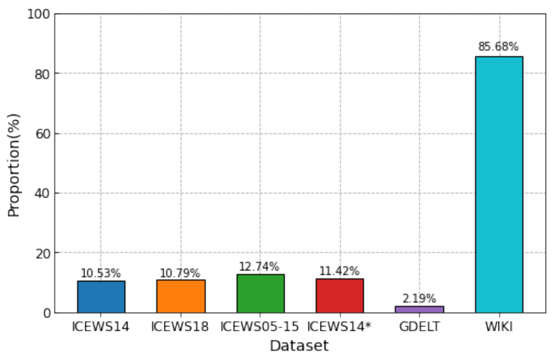

Figure 4: Proportion(%) of quadruplets shown in exact one timestamp before for each dataset. The x-axis is the name of the dataset and the y-axis is the proportion(%).

major points can be noticed:

(1) On most of the datasets (ICEWS14, ICEWS18, ICEWS05-15, ICEWS14*, and GDELT), a larger history length results in a higher MRR. Where the history length is small, enlarging the history length can substantially enhance performance. However, when the history length surpasses three, the degree of improvement becomes marginal. This aligns with the expectations that the recent several snapshots can help with inference, while in a long history, the irrelevant information does not contribute to the performance. By considering the model performance and calculation complexity, history length = 3 is selected as the final model for these datasets.

(2) An exception occurs on WIKI, where the model achieves the best performance when history length = 1. To investigate the factors, a detailed statistical analysis of the datasets is conducted. Table 4 in Section 4.2 shows some sample queries in WIKI, where some facts are the same as the facts at previous timestamps, the reason lies in that for a fact (s,r,o,t_1 - t_n), WIKI generates the same quadruplets across the time range from t_1 to t_n. Figure 4 shows the proportion of the quadruplets at t_q shown in the previous timestamp $t_q - 1$ for all timestamps in the test set on each dataset. 85.68% samples in the WIKI show in the one timestamp before, while fewer than 15% samples in ICEWS14, ICEWS18, ICEWS05-15, ICEWS14*, GDELT are from the previous timestamp. The same quadruplets shown across different timestamps in WIKI result in similar snapshots(graphs) at different timestamps. When a larger history length is applied, multiple graph neural network models applied on multiple similar graphs will be approximated to applying a multiple layers GNN model on one graph, which leads to the over-smoothing issue in a deep GNN(Li et al., 2018). Therefore, a large history length may decrease model performance on WIKI.

4.4 Ablation Study

Table 5 presents the ablation study of different components of our model.

Impact of explicit temporal embedding To evaluate the efficiency of the **explicit temporal representation**, we remove the dynamic embedding from the explicit temporal input, resulting in only the static embedding of each entity left. For all six benchmark datasets, removing dynamic embedding leads to worse performance. Compared with the performance drop in ICEWS14, ICEWS18, ICEWS14* and GDELT, it is clear that the MRR decreases more in WIKI and ICEWS05-15. The reason is that the total time range in these two datasets is large (232 years and 11 years), and the entity information can evolve over a long period, which can be captured by explicit temporal embedding.

Impact of relation-aware skip information flow To demonstrate how the relation-aware skip information flow contributes to the model performance, two ablation tests are conducted. (1)'-relation_aware' means that when calculating the attention score in skip information flow, the entity-related relation is omitted, formally, the attention score is Equation 5 is changed to:$attn_j^{e_q} = W_a(h_{t_i-j}^{e_q}), j \in [1, m]$. (2)'-skip' means removing the whole skip information flow, making the input of each timestamp the last timestamp the output: $h_{t_i+1}^{e_q} = h_{t_i}^{e_q,L}$.

The model performance drops heavily if no relation-aware attention mechanism is applied, showing the vital importance of the relation-aware

Table 6: MRR(%) of our model with different ensemble methods. The highest value is bold.

Ensemble Model	**ICEWS14**	**ICEWS18**	**ICEWS05-15**	**ICEWS14***	**GDELT**
Re-Temp	48.04	35.82	56.30	46.40	25.05
Ensemble (avg pooling)	48.58	36.16	**56.72**	46.56	25.04
Ensemble (max pooling)	**48.69**	**36.38**	56.69	**47.06**	**25.06**
Ensemble (min pooling)	47.55	35.72	55.58	46.23	25.03

attention mechanism. We can conclude that the entity-related relation information actually helps the model to select necessary information. In most cases, removing the skip connection worsens the model performance compared with only removing the relation-aware attention mechanism. Compared with '-relation_aware' setting, the models under the '-skip' setting learn from all the recent snapshots for prediction, leading to the involvement of irrelevant information during prediction.

However, WIKI shows better performance under this setting, even compared with our original Re-Temp model. The reason might be the same as that discussed in Section 4.3: More than 80% of facts in the WIKI show in the previous timestamp, and a graph model applied on the previous timestamp can easily capture that repetitive information for prediction.

4.5 Ensemble Modelling Evaluation

CEN(Li et al., 2022a) builds an ensemble model with different history lengths. Inspired by this, we test our model under an ensemble setting. For a model with a history length of k, suppose the score vector of all candidates for query q is s^q_k, a pooling method is applied on $\{s^q_1, s^q_2, ..., s^q_k\}$ to get the final score. Three different pooling methods are applied. Table 6 shows the MRR(%) results of our model under the ensemble setting. We applied the history lengths from one (1), and the maximum history length is set to three (3) as previously defined. We did not include the experiments on WIKI since the optimal history length is one (1), and no models with smaller history lengths can be used. First of all, our model can benefit under the ensemble setting on four of the datasets (ICEWS14, ICEWS18, ICEWS05-15, ICEWS14*), but only achieve similar performance on GDELT compared with the original Re-Temp model (25.05%). Considering the history length influence shown in Figure 3, the model achieves similar results with different history lengths. Therefore, models with different history lengths on GDELT might be similar making the ensemble models less effective. However, ICEWS datasets are history-length sensitive, and ensemble models can benefit from different models of different history lengths. In addition to this, max pooling usually achieves the best performance as the ensemble method while min pooling will worsen the performance.

5 Conclusion

We introduced Re-Temp, which integrates both explicit and implicit temporal information and applies a relation-aware skip information flow to adopt after each timestamp to remove unnecessary information for prediction by taking the entity-related relation in the query into consideration. The experimental results on six TKGC datasets present the superiority of our model, compared with eight baseline models. We also conduct a statistical analysis of the datasets to show the different nature between WIKI and other datasets. It is hoped that Re-temp presents insight into the importance of the relation in the query and both types of temporal information.

6 Limitations

Re-Temp still follows Knowledge Graph Completion encoder-decoder framework(Hamilton et al., 2017) while more frameworks can be explored. The graph model at each timestamp and the decoder score function follow the same methods widely used by other models.

Since we have shown that the explicit temporal embedding and the skip information flow contribute to model performance, more work can be done by combining these concepts into the graph model and score function, for example, combining the entity-related relation into the graph model at each timestamp to selectively propagate between nodes, or combining the explicit temporal embedding into the decoder score function. Also, like most TKGC models, Re-Temp can not handle new entities that do not show in the training data. More methods integrating the text description can be explored(Lv et al., 2022).

References

Ivana Balažević, Carl Allen, and Timothy Hospedales. 2019. Tucker: Tensor factorization for knowledge graph completion. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5185–5194.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. *Advances in neural information processing systems*, 26.

Ali Borji. 2023. A categorical archive of chatgpt failures. *arXiv preprint arXiv:2302.03494*.

Alberto Garcia-Duran, Sebastijan Dumančić, and Mathias Niepert. 2018. Learning sequence encoders for temporal knowledge graph completion. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4816–4821.

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

William L. Hamilton, Rex Ying, and Jure Leskovec. 2017. Representation learning on graphs: Methods and applications. *IEEE Data Eng. Bull.*, 40(3):52–74.

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.

Yunpu Ma Zhen Han Kun He. Haohai Sun, Jialun Zhong. 2021. Timetraveler: Reinforcement learning for temporal knowledge graph forecasting. In *EMNLP*.

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

Woojeong Jin, Meng Qu, Xisen Jin, and Xiang Ren. 2020. Recurrent event network: Autoregressive structure inference over temporal knowledge graphs. In *EMNLP*.

Seyed Mehran Kazemi and David Poole. 2018. Simple embedding for link prediction in knowledge graphs. *Advances in neural information processing systems*, 31.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Thomas N Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*.

Julien Leblay and Melisachew Wudage Chekol. 2018. Deriving validity time in knowledge graph. In *Companion Proceedings of the The Web Conference 2018*, pages 1771–1776.

Kalev Leetaru and Philip A Schrodt. Gdelt: Global data on events, location, and tone, 1979–2012. Citeseer.

Qimai Li, Zhichao Han, and Xiao-Ming Wu. 2018. Deeper insights into graph convolutional networks for semi-supervised learning. In *Thirty-Second AAAI conference on artificial intelligence*.

Zixuan Li, Saiping Guan, Xiaolong Jin, Weihua Peng, Yajuan Lyu, Yong Zhu, Long Bai, Wei Li, Jiafeng Guo, and Xueqi Cheng. 2022a. 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, Dublin, Ireland. Association for Computational Linguistics.

Zixuan Li, Zhongni Hou, Saiping Guan, Xiaolong Jin, Weihua Peng, Long Bai, Yajuan Lyu, Wei Li, Jiafeng Guo, and Xueqi Cheng. 2022b. Hismatch: Historical structure matching based temporal knowledge graph reasoning. *arXiv preprint arXiv:2210.09708*.

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.

Xin Lv, Yankai Lin, Yixin Cao, Lei Hou, Juanzi Li, Zhiyuan Liu, Peng Li, and Jie Zhou. 2022. Do pre-trained models benefit knowledge graph completion? a reliable evaluation and a reasonable approach. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 3570–3581.

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

Chaitanya Malaviya, Chandra Bhagavatula, Antoine Bosselut, and Yejin Choi. 2020. Commonsense knowledge base completion with structural and semantic context. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 2925–2933.

OpenAI. 2022. Introducing chatgpt.

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

Chao Shang, Yun Tang, Jing Huang, Jinbo Bi, Xiaodong He, and Bowen Zhou. 2019. End-to-end structure-aware convolutional networks for knowledge base completion. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 3060–3067.

Shikhar Vashishth, Soumya Sanyal, Vikram Nitin, and Partha Talukdar. 2020. Composition-based multi-relational graph convolutional networks. In *International Conference on Learning Representations*.

Bing Xu, Naiyan Wang, Tianqi Chen, and Mu Li. 2015. Empirical evaluation of rectified activations in convolutional network. *arXiv preprint arXiv:1505.00853*.

Chenjin Xu, Mojtaba Nayyeri, Fouad Alkhoury, Hamed Yazdi, and Jens Lehmann. 2020. Temporal knowledge graph completion based on time series gaussian embedding. In *International Semantic Web Conference*, pages 654–671. Springer.

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 *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

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 *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 4732–4740.

A Model Component Details

A.1 CompGCN

In CompGCN, at each layer, edges(relations) are conducted as the transformation on the connected node(entity), and then a weighted sum calculation from GCN(Kipf and Welling, 2017) is applied to the transformed entity. Self-loop is also calculated before the activation function. Formally, for a entity node e_q at timestamp t_i at lth layer, the propagation shows as follows:

$$h_{t_i}^{e_q,l+1} = \sigma(\frac{1}{|N_{t_i}^{e_q}|}\sum_{e_n \in N_{t_i}^{e_q}} W_{g,0}^{l} f(h_{t_i}^{e_n,l}, h_r) + W_{g,1}^{l} h_{t_i}^{e_q,l}) \tag{9}$$

where $N_{t_i}^{e_q}$ is the set of the neighbour entities of e_q at timestamp t_i, σ is the activation function and RReLU(Xu et al., 2015) is chosen. $W_{g,0}^{l}$ and $W_{g,1}^{l}$ are learnable parameters at layer l, and f is the composition function for neighbour entity embedding $h_{t_i}^{e_n,l}$ and relation embedding h_r, such as summation, subtraction, element-wise product, or circular-correlation(Xu et al., 2015).Summation is selected for better alignment of relation-aware skip information flow.

A.2 ConvTransE

By applying ConvTransE, the query subject entity embedding $h_{t_q}^{e_q}$ and query relation embedding h_{r_q} are concatenated first, and then a convolutional layer and a feed-forward layer are applied. The score of each candidate is the dot-product of the candidate entity embedding with the representation after the ConvTransE. To denote the process of calculating the score of the candidate entity e_c:

$$s(e_q, r_q, e_c, t_q) = h_{t_q}^{e_c}\text{FC}(\text{Conv1d}([h_{t_q}^{e_q} \oplus h_{r_q}])) \tag{10}$$

where e_c is the candidate entity.

B Experiment Setup Details

B.1 Running Details

All the models are trained by using 16 Intel(R) Core(TM) i9-9900X CPU @ 3.50GHz and NVIDIA Tesla P100 PCIe 16 GB.

The number of parameters of Re-Temp can be decomposed into three parts:

- **Input** Entity embedding: $3d|\mathcal{E}| + 2d^2$, Relation embedding: $2d|\mathcal{R}|$
- **Encoder** CompGCN: $2d^2$, Relation-aware information flow: d^2
- **Decoder** ConvTransE: $ch(2ke+d+2)$, where ch is the number of channels and ke is the kernal size.

The running time and number of parameters of Re-Temp on different datasets under the default hyperparameters can be found in Table 7.

B.2 Evaluation Metrics

For each query, the model produces a ranked list of all possible candidates and the reciprocal rank is the inverse of the rank position of the correct answer. MRR is calculated by $\frac{1}{Q}\sum_{q=1}^{Q}\frac{1}{rank_q}$, which is the average reciprocal rank of all queries. Hits@N measures the proportion of results, where the correct answer is in the top N ranked results. $N = 1, 3, 10$ are chosen, as all previous works

Table 7: Re-Temp running time and number of parameters

		ICEWS14	ICEWS18	ICEWS05-15	ICEWS14*	GDELT	WIKI
Running Time (min)	Training	11.6	22.1	99.5	8.8	112.6	8.1
	Inference	0.05	0.1	0.6	0.1	1.1	0.2
Number of Parameters	Input	4.4M	14.0M	6.2M	4.4M	4.8M	7.6M
	Encoder	0.1M	0.1M	0.1M	0.1M	0.1M	0.1M
	Decoder	2M	2M	2M	2M	2M	2M
	Total	6.6M	16.1M	8.4M	6.6M	6.9M	9.7M

Table 8: MRR(%) result of the Encoder and Decoder Variants test. The highest value is bold.

Model Variants	ICEWS14	ICEWS18	ICEWS05-15	ICEWS14*	GDELT	WIKI
Default	**48.04**	**35.82**	**56.30**	**46.40**	**25.05**	**78.51**
CompGCN (Element-Wise)	47.57	35.24	55.81	45.54	24.98	70.99
CompGCN (Circle-Correlation)	46.69	35.09	56.00	44.65	24.90	74.32
Tucker	46.36	35.14	56.84	44.48	24.65	78.28
DistMult	34.48	22.85	39.80	36.58	18.18	59.35

adopted. The higher value of MRR and hits@N indicates the better performance of a model.

C Model Varirants Experiments

We adopted CompGCN as a graph model in the encoder to model the multi-relational snapshot, and the transformation function is the sum: $f(h_{t_i}^{en,l}, h_r) = h_{t_i}^{en,l} + h_r$. Followed by Vashishth et al. (2020), we tested the default setting with the element-wise product or circle-correlation as the transformation function, as shown in Table 8. Even though good performance can be achieved by replacing the summation with other transformation functions, the summation is the best transformation function. The reason would be that during the skip information flow, additive attention is applied, which can benefit from the alignment of the entity embedding and relation embedding. Moreover, various decoders aside from ConvTransE are also experimented followed by TANGO(Han et al., 2021). As a decoder, Tucker(Balažević et al., 2019) achieves much better performance than DistMult(Yang et al., 2015). This is because DistMult lacks learnable parameters, while the learnable parameters in ConvTransE and Tucker give the model more complexity to have more possibility to find an optimal solution.

D Responsible Research - Risk

Most temporal knowledge graph datasets focus on political news, which might raise concerns when predicting future political events where people have different political leanings.