Path of Time: Explanations for Temporal Knowledge Graph Completion through Chronological Regulation

Lukas Gehring[1], Moritz Blum[1,*], Basil Ell[1,2] and Philipp Cimiano[1]

[1]Bielefeld University, CITEC, Inspiration 1, 33619, Bielefeld, Germany
[2]University of Oslo, Problemveien 11, 0313 Oslo, Norway

Abstract

Temporal Knowledge Graph Completion (TKGC) uses the facts available in a TKG to make it less incomplete. State-of-the-art Graph Neural Networks (GNNs) for TKGC are black boxes that provide results without explanations. Existing explanation methods for static KGC are difficult to transfer to TKGC as they do not capture the temporal properties and likely generate large explanation graphs.

As the chronological order of facts is relevant for TKGC, we infuse this characteristic into the explanation subgraphs. In this work, we (i) propose a regulation method that incentivizes a chronological order in the explanations and (ii) investigate the effect of the chronological regulation on the explanations of two state-of-the-art TKGC models.

Our results show that in most scenarios, the chronological regulation can improve explanations of TKGCs. For example, we observe an improvement of the fidelity characterization score by up to 2% and significant improvements for small explanations.

Keywords

XAI, Temporal Knowledge Graph Completion

1. Introduction

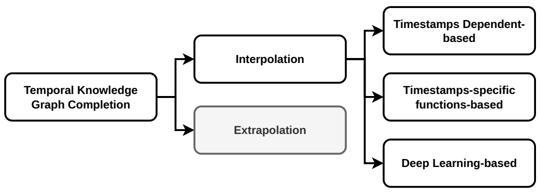
A Temporal Knowledge Graph (TKG) captures knowledge about facts and their temporal dynamics in a graph-structured form as a set of quadruples of *(subject, relation, object, timestamp)*, like *(Barack Obama, Consult, Angela Merkel, 2014-05-01)*. Fig. 1a shows a visualization of a TKG.

TKGs are inherently incomplete due to various reasons. Temporal Knowledge Graph Completion (TKGC) attempts to complete a TKG by deriving missing information based on existing facts. A typical TKGC query might be *(Barack Obama, Criticize or denounce, ?, 2014-04-30)*. A TKGC model aims to predict the missing subject or object by using the existing facts in the past, present, and future. The missing information is visualized as the unknown node *?* in Fig. 1a.

Various TKGC approaches have been developed recently [1, 2, 3] and Graph Neural Networks (GNNs) are considered state-of-the-art. These GNN-based approaches often remain difficult or impossible for humans to interpret. Explainable Artificial Intelligence (XAI) is a crucial aspect of ensuring the secure and trustworthy integration of neural models [4], especially in critical infrastructures. Additionally, explainability can help to develop more robust approaches by providing insights into model decision-making processes, enabling the identification and mitigation of potential biases, errors, or vulnerabilities.

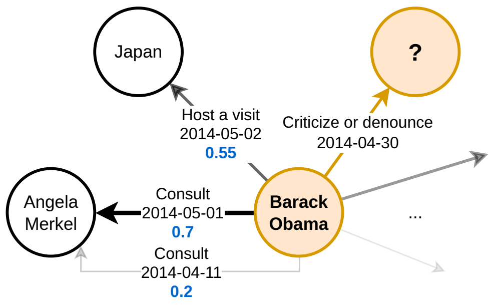
In this work, we focus on perturbation-based methods for explaining TKGC predictions, which construct a subgraph of the input TKG by perturbing the input of the target model to identify the most important input features for the model's prediction. The subgraph then serves as an explanation for a target model prediction. A possible explanation subgraph for our presented example about *Barack Obama* is shown in Fig. 1b.

ESWC 2025 Workshops and Tutorials
*Corresponding author.
lgehring@techfak.uni-bielefeld.de (L. Gehring); mblum@techfak.uni-bielefeld.de (M. Blum); bell@techfak.uni-bielefeld.de (B. Ell); cimiano@techfak.uni-bielefeld.de (P. Cimiano)
0009-0009-3335-1679 (L. Gehring); 0000-0003-4924-3903 (M. Blum); 0000-0002-8863-3157 (B. Ell); 0000-0002-4771-441X (P. Cimiano)



Temporal Knowledge Graph Completion
Interpolation
Extrapolation
Timestamps Dependent-based
Timestamps-specific functions-based
Deep Learning-based

(a) Temporal Knowledge Graph



(b) Explanation Graph

Figure 1: (a) TKGC query *(Barack Obama, Criticize or denounce, ?)* with graph context that might be used by a model, and (b) an explanation graph one could expect from an explainer model for the model's prediction on this query.

Most explanation methods only consider entities and relations and ignore the temporal information or do not consider time-aware models in their evaluation. Since the temporal information of TKGs can significantly influence the prediction of such a model, time should also be considered as part of the explanation.

The goal of this work is to: i) establish a baseline for GNN-based TKGC explanations using non-temporal GNN explainers; ii) propose a temporal regulation method that enables non-temporal explainers to incorporate additional temporal information into their explanations. Inspired by the concept of path-based explanations used for XAI of static KG approaches [5, 6], our regulation method encourages a chronological order of the facts in the explanations; iii) investigate the effect of chronological regulation on the explanations of two state-of-the-art TKGC models. We evaluate explanation quality using common metrics and introduce a new metric better suited for graphs in the temporal setting than existing metrics.

2. Foundations

2.1. Temporal Knowledge Graph Completion

A Knowledge Graph (KG) stores facts as triples (s, r, o), where $s \in \mathcal{E}$ is called the subject, $r \in \mathcal{R}$ the relation, and $o \in \mathcal{E}$ the object. $\mathcal{E}$ and $\mathcal{R}$ are finite sets of entity and relation identifiers, respectively.

A Temporal Knowledge Graph (TKG) is a KG extended by the temporal information about the facts. Facts in a TKG are represented as quadruples (s, r, o, t), with $t \in \mathcal{T}$ adding time information to the fact. t is a specific point in time (e.g. *05-11-2014*) from a finite set of timestamps $\mathcal{T}$.

Temporal Knowledge Graph Completion (TKGC) is about adding missing quadruples to a TKG. TKGC models predict a missing entity of a given query $q = (s_q, r_q, ?, t_q)$ or $q = (?, r_q, t_q, t_q)$, where $s_q, t_q, ? \in \mathcal{E}$, $r_q \in \mathcal{R}$, and $t_q \in \mathcal{T}$.

2.2. Graph Neural Networks for TKGC

Graph Neural Networks (GNNs) are a type of neural network designed to process graphs as input. The core concept of a GNN is message-passing, first introduced by Gilmer et al. [7], which enables the GNN to learn node embeddings that capture its features but also include information from the neighborhood.

Given a node i, with its features $\mathbf{x}_i$ and its incoming neighborhood $\mathcal{N}_i$, message passing computes updated node features $\hat{\mathbf{x}}_i$ as

$$\hat{\mathbf{x}}_i = \phi\left(\mathbf{x}_i, \bigoplus_{j \in \mathcal{N}_i} \psi(\mathbf{x}_i, \mathbf{x}_j)\right), \tag{1}$$

where message function ψ and update function ϕ are trainable functions and $\oplus$ denotes a nonparametric operation such as sum, mean, or maximum [8].

In a GNN, this message passing scheme is typically repeated layerwise and can include edge features $\mathbf{e}_{ji}$ for each edge connecting node j and i. Two commonly used GNNs are the Graph Attention Networks (GATs) and the Graph Convolution Networks (GCNs).

2.3. GNN Explainability

We focus on post-hoc explanations, which are generated for a target model $\mathcal{M}$. We utilize a perturbation-based method that modifies the model's input by masking to identify minimal subgraphs that explain the model's prediction.

One of the initial and well-known perturbation-based GNN explainer models for non-temporal KGs is the GNNExplainer [9]. This model is designed to identify the subgraph and node features most relevant to a GNN's prediction, by applying a learnable mask $M \in [0,1]^{|\mathcal{E}|\times|\mathcal{R}|\times|\mathcal{E}|}$ to the graph's adjacency matrix A_c, to minimize the following cross-entropy objective:

$$\min_M -\sum_{c=1}^{C} \mathbb{1}[y=c] \log P_{\mathcal{M}}(Y=y|G=A_c \odot \sigma(M)). \tag{2}$$

Here, $\mathbb{1}[y=c]$ is the indicator function for the target class y, $P_{\mathcal{M}}$ is the probability of target model $\mathcal{M}$ predicting y, and $\sigma(M)$ maps the mask to a continuous range $[0,1]$. The framework learns the mask M to minimize the conditional entropy of the predictions when restricted to the masked subgraph. Sparse explanations are encouraged through regularization terms, and additional thresholds can be applied to refine the resulting subgraph, retaining only the most important edges and nodes.

3. Related Work

XAI aims to help humans understand the predictions of neural networks and other AI models, which are normally considered black boxes. Given a target model $\mathcal{M}$, the goal of XAI is to provide a human-understandable textual or visual explanation of $\mathcal{M}$'s predictions.

Perturbation-based instance-level explanation methods investigate the behavior of the target model's predictions on varying inputs to identify a subgraph relevant to the prediction, which then serves as an explanation.

A perturbation-based instance-level explanation should reflect the model's prediction, i. e., the explanation graph should only contain information important for the prediction. Similarly, the result should change if crucial information is removed from the input. At the same time, an explanation should be sufficiently sparse to be interpretable by a human [10].

The GNNExplainer proposed by Ying et al. [9] is one of the most well-known and initial approaches in explainability for GNNs.

Recently, path-based explanations for KGC have gained attention [5, 6]. Instead of subgraphs, they generate a set of paths connecting the query entities, naturally capturing their connections. Such explanations are expected to be better interpretable and more user-friendly.

While TKGC and XAI are well-researched subjects, there is still little literature on using XAI in TKGC.

Some works combine TKGC and TKGC Explainability in a single model, also known as self-interpretable models. Examples of self-interpretable models are xERTE [11] and T-GAP [2], which both construct a subgraph using attention propagation during inference that can also serve as an explanation graph. However, in this work, we are interested in model-agnostic explainers, i. e., explainers that can be used for different target models $\mathcal{M}$ without large modifications.

The perturbation-based explainer Temporal Motifs Explainer (TempME) proposed by Chen et al. [12] identifies the most important recurring temporal patterns of connections in a TKG.

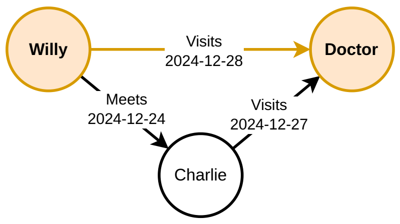


Figure 2: Example of a simple chronological path for the fact (query and model prediction) that Willy visits the doctor at $t =$ *2024-12-28*. Before this, Charlie met Willy, who visited the doctor one day before Charly. This could indicate that Willy was ill and infected Charly.

He et al. [13] extend an existing explainer to the temporal setting. First, the TKG is divided into several non-temporal KGs, i. e., a sequence of KGs. Second, a non-temporal explainer is then used to explain the instance on each static snapshot. Finally, a time-aware explanation is constructed by combining the most dominant static explanations.

The existing TKGC explainers fail to incorporate the temporal aspect sufficiently, address the unique challenges posed by the TKG graph characteristics, or tailor explanations to meet user requirements for time-based interpretations.

4. Method

GNN approaches for TKGC learn how information evolves over time to predict new facts. Since the temporal order of facts conveys information, models process the graph in chronological order rather than random order to leverage the causal relationships and temporal dependencies [14, 15]. In consequence, this should be captured in explanations, too. E. g., if we want to explain why a person visits a doctor, it can be interesting to know what happened the days before or after. This can build up a temporal chain of facts, see Fig. 2.

Inspired by how path-based explanations incorporate connections between query entities in the explanations [5, 6], we propose chronological paths to infuse temporal properties into the explanation of non-temporal explainers. We propose a chronological regulation that favours the temporal chain of facts. To reinforce the effect, relations not being on such a path might be penalized.

4.1. Chronological Path

A chronological path is a path in a graph with chronologically ascending or descending timestamps along its edges. For any two consecutive edges on a chronologically ascending path, the timestamp of the second edge is greater than or equal to the timestamp of the first edge. Chronological descending is defined analogously. Given a TKG G and the target models predicted entity o' for query $q = (s_q, r_q, ?, t_q)$ with entity s_q, relation r_q and timestamp t_q, we denote $p_{(s_q,o')}$ as a chronological path from s_q to o'. The set of all chronological paths between s_q and o' is defined as

$$P_{(s_q,o')} = \{p^{(k)}_{(s_q,o')} \mid k \leq K_{\text{max}}\}, \tag{3}$$

where K_{max} is the maximal length of the chronological paths.

We set $K_{\text{max}} = 3$ for all experiments as longer paths may be less relevant, and most TKG models only consider a maximum of 3 hops around a query.

4.2. Chronological Regulation

Chronological regulation rewards edges on chronological paths between the query's subject s_q and the target model's predicted object o' and might penalize edges that are not. We propose two methods to implement chronological regulation.

Loss Regulation Given all chronological paths $P_{(s_q,o')}$ from s_q to the target model's prediction o', chronological regulation can be applied to the edge mask $\sigma(M)$ by defining a regulation loss that measures the distance between $\sigma(M)$ and the optimal edge mask regarding the chronological paths $y_n^{(\text{loss\_reg})} \in [0,1]$. For each incoming edge i in s_q that connects the nodes s_q and $e_i \in \mathcal{N}_{s_q}$ with the relation r_i at time t_i, we define the optimal edge mask regarding the chronological paths as

$$y_i^{(\text{loss\_reg})} = \begin{cases} 1 - \beta \cdot \dfrac{\log_e(|p|_{\text{min}})}{\log_e(K_{\text{max}})}, & \text{if } (s_q, r_i, e_i, t_i) \in p \in P_{(s_q,o')}, \\ 0, & \text{otherwise} \end{cases} \tag{4}$$

where $\beta \in [0,1]$ is a hyperparameter that determines the logarithmic value decrease for edges on chronological paths that exceed a length of 1 and $|p|_{\text{min}}$ is the length of the shortest chronological path between s_q and o'. If $\beta = 0$, $y_i^{(\text{loss\_reg})}$ is equal to 1 regardless of the length of the chronological path. If $\beta = 1$, the decrease is maximum and $y_i^{(\text{loss\_reg})}$ is 0 for paths of length K_{max}. Note that we only reward edges in the direct neighborhood of s_q, if they lie on a chronological path. We expect that we can guide the explanation in the direction of the chronological paths, and not regulate them individually.
Now we can define a loss $\ell_{\text{reg}}(\sigma(M), y^{(\text{loss\_reg})})$ between $\sigma(M)$ and $y^{(\text{loss\_reg})}$ which we can add to the explainer loss. We choose the mean absolute error.

$$\ell_{\text{reg}}(\sigma(M), y^{(\text{loss\_reg})}) = \text{mean}(\{l_1, ..., l_N\}), \quad l_n = \gamma \cdot |\sigma(M)_n - y_n^{(\text{loss\_reg})}| \tag{5}$$

We use a hyperparameter γ, to scale the strength of the regulation.

Gradient Regulation The second regulation method directly applies the regulation to the gradients. The chronological paths are used to create a function $y_i^{(\text{grad\_reg})}$ that rewards or penalizes the gradients of the mask. This function is similar to the one used in Eq. 4, but with hyperparameter α scaling the maximum reward and penalty.

$$y_i^{(\text{grad\_reg})} = \begin{cases} \alpha \left(1 - \beta \cdot \dfrac{\log_e(|p|_{\text{min}})}{\log_e(K_{\text{max}})}\right), & \text{if } (s_q, r_i, e_i, t_i) \in p \in P_{(s_q,o')}, \\ -\alpha, & \text{otherwise} \end{cases} \tag{6}$$

Let Θ_e be the edge mask parameters and $\nabla\ell(\Theta_e)$ the computed gradients. To regulate the edge mask, before doing gradient descent, the computed gradient is subtracted by $y^{(\text{grad\_reg})}$. This regulation increases the gradient for edges on a chronological path and decreases it otherwise.

$$\nabla\ell(\Theta_e) = \nabla\ell(\Theta_e) - y^{(\text{grad\_reg})} \tag{7}$$

Note that we do not need a scaling parameter γ as in the previous method since we can scale $y^{(\text{grad\_reg})}$ directly with α.

5. Experiments

5.1. Datasets & Target Models

Commonly used real-world benchmark datasets for TKGC are subsets of ICEWS[1] and WIKIDATA [16]. We utilize ICEWS14 and WIKIDATA11K [17].

ICEWS14 contains socio-political events. The entities are, for example, countries, institutions, or persons; the relations are predicates like *Consult* or *Make statement*, and the timestamps are the dates on which the event occurred [17].

[1]https://www.lockheedmartin.com/en-us/capabilities/research-labs/advanced-technology-labs/icews.html (last visit september 30, 2024)

WIKIDATA11K contains entities such as historical figures, places, and artifacts, connected by relations like *Was born in* or *Founded*. The characteristics of both datasets can be found in App. A Tab. 2.

In this work, we use two state-of-the-art TKGC models for predictions. TARGCN [1] aggregates a subset of the temporal neighborhood with a single GCN layer to compute the time-dependent representation of an entity. T-GAP [2] utilizes multiple GNN layers and attention-based subgraph sampling to account for distant nodes, which increases the representativeness of predictions due to increased information flow. A detailed description of both target models can be found in appendix C. We then explain these target models using the GNNExplainer.

5.2. Metrics for Graph Neural Network Explainers

Following [18], we evaluate our explanations using *Fidelity*, *charact*, and *Sparsity*, as well as with our proposed *SparseFid*, which combines *fidelity* and *sparsity*.

Fidelity [19] measures the faithfulness of an explanation to the target model. This means the model's prediction should change if important entities or relations are removed from the explanation graph (*fid*$_+$). However, if unimportant entities or relations are removed, the prediction should remain the same (*fid*$_-$).

$$fid_+ = 1 - \frac{1}{|Q|}\sum_{q\in Q} \mathbb{1}(\hat{y}_q^{G_{C\setminus S}} = y_q), \qquad fid_- = 1 - \frac{1}{|Q|}\sum_{q\in Q} \mathbb{1}(\hat{y}_q^{G_S} = y_q) \tag{8}$$

If *fid_* is close to 0, the provided explanation is *sufficient*, and if *fid*$_+$ is close to 1, the explanation is *necessary*. An explanation should be *sufficient* and *necessary*. A metric to combine *fid*$_+$ and *fid*$_-$ is the *charact* score [19].

$$charact = \frac{w_+ + w_-}{\frac{w_+}{fid_+} + \frac{w_-}{1 - fid\_}} \quad \text{, with } w_+ + w_- = 1 \tag{9}$$

We give equal weight to *fid*$_+$ and *fid_*.

Sparsity is also an important property of explanation graphs to provide human-understandable explanations as TKGs often have a high avg. node degree and high information density due to the additional temporal information compared to static KGs. We define the *sparsity* of an explanation subgraph G_{S_i} as

$$sparsity = \frac{1}{N}\sum_{i=0}^{N}\left(1 - \frac{log(|G_{S_i}| + 1)}{log(|G_{C_i}| + 1)}\right), \tag{10}$$

where $|G_{S_i}|$ denotes the number of edges in the explanation subgraph and $|G_{C_i}|$ the number of edges in the computation graph. Note that we are taking the log of the number of edges because we want to focus on explanations that are as small as possible. Reducing an already small explanation has a greater effect on the *sparsity* than reducing a large explanation.

Sparse-Fidelity Finally, we propose a new combined metric based on the *charact* score and *sparsity*. As with the *charact* score, we calculate the harmonic mean between *charact* and *sparsity*.

$$SparseFid = \frac{w_c + w_s}{\frac{w_c}{charact} + \frac{w_s}{sparsity}} \quad \text{, with } w_c + w_s = 1 \tag{11}$$

With $w_c = w_s = 0.5$ we give equal weight to *charact* and *sparsity*.

5.3. Baseline

We use the GNNExplainer without temporal edge mask regularization as the baseline to compare the proposed temporal regulation methods. Although the GNNExplainer was originally developed for static KGs only, it can be adapted to the temporal setting by extending the edge mask to the temporal

Table 1
Results of the GNNExplainer without and with regulation.

		ICEWS14						WIKIDATA11K					
Model	Mask Type	*fid*$_\times$	*fid*$_-$	*charact*	*sparsity*	*SparseFid*	*Threshold*	*fid*$_\times$	*fid*$_-$	*charact*	*sparsity*	*SparseFid*	*Threshold*
threshold 100													
TARGCN	No Regulation	0.7987	0.0006	0.8878	0.6615	0.7582	100	0.8798	**0.0003**	0.9359	0.7335	0.8224	100
	Loss-Regulation	**0.8311**	**0.0004**	**0.9076**	0.6615	**0.7653**	100	**0.8974**	0.0004	**0.9458**	0.7335	**0.8262**	100
	Gradient-Regulation	0.8286	0.0010	0.9058	0.6615	0.7647	100	0.8808	0.0004	0.9364	0.7335	0.8226	100
T-GAP	No Regulation	0.7282	0.0393	0.8285	0.6122	0.7041	100	0.7285	0.0366	0.8297	0.6283	0.7151	100
	Loss-Regulation	0.7283	0.0391	0.8286	0.6122	0.7042	100	0.7285	0.0366	0.8296	0.6283	0.7150	100
	Gradient-Regulation	**0.7334**	**0.0369**	**0.8327**	0.6122	**0.7056**	100	**0.7311**	**0.0328**	**0.8327**	0.6283	**0.7162**	100
optimal threshold (according to *SparseFid*)													
TARGCN	No Regulation	0.7989	0.0235	0.8788	**0.7140**	**0.7879**	40	0.8832	**0.0261**	**0.9263**	0.7530	0.8307	30
	Loss-Regulation	0.8073	**0.0117**	**0.8887**	0.7004	0.7834	50	0.8966	0.0635	0.9161	0.7697	**0.8365**	20
	Gradient-Regulation	0.8091	0.0157	0.8881	0.7004	0.7832	50	0.8774	0.0570	0.9090	0.7697	0.8335	20
T-GAP	No Regulation	**0.7154**	0.0438	**0.8184**	0.6210	0.7062	90	**0.7077**	**0.0703**	**0.8037**	0.6566	0.7227	70
	Loss-Regulation	0.7152	**0.0437**	0.8183	0.6210	0.7061	90	0.7077	**0.0703**	0.8036	0.6566	0.7227	70
	Gradient-Regulation	0.7014	0.0456	0.8086	**0.6308**	**0.7087**	80	0.6865	0.1072	0.7762	**0.6831**	**0.7267**	50

adjacency matrix. The use of this inflated mask $M \in [0,1]^{|\mathcal{E}|\times|\mathcal{R}|\times|\mathcal{E}|\times|\mathcal{T}|}$ allows the GNNExplainer to indirectly model the temporal information with the edge mask since each relation between two entities can be considered independently at all possible times. This is indirect because the timestamp is not masked independently of the edge type, and the temporal information might also be utilized in other model components not affected by the edge mask. A description of how the edge mask can be applied to the target models TARGCN and T-GAP can be found in App. C.

6. Results

This chapter provides the evaluation results of the GNNExplainer w. and w/o. temporal regulation on TKGC. We apply the GNNExplainer to the two target models TARGCN and T-GAP. Details about the hyperparameter-tuning can be found in App. B. Tab. 1 shows an overview of the results using the edge mask and the proposed edge mask regulation methods loss and gradient regulation. We report all metrics with a threshold of 100 edges for each mask.[2]

We observe that the explanations for the predictions of the target model TARGCN achieve notably better scores compared to those for T-GAP. Using the target model TARGCN, the proposed regulation methods can outperform the edge mask, with loss regulation providing the best results. In contrast, the best results for the target model T-GAP are obtained with gradient regulation, while loss regulation cannot improve upon the baseline. Since the *sparsity* of the explanations remains constant at a fixed threshold for the edge mask, we observe the same values with and without regulation.

Case Study: Impact of Chronological Regulation on Edge Mask Evolution: We investigate the evolution of masks on one randomly selected ICEWS14 quadruple to see how the regulation methods influence edge mask learning.

The mask history using TARGCN as the target model can be seen in Fig. 3. We highlighted two edges for better visualization: one on a chronological path (black) and one that is not (red). When using loss regulation, an initial increase in the mask value can be observed for the edge not on a chronological path, followed by a steep decrease after about 60 epochs. The explainer seems to have found a minimum for the loss after a few epochs. An instant decrease of the edge mask for edges not lying on a chronological path can be observed using the gradient regulation. Since this method does not minimize a loss, the influence of the regulation is immediately apparent, which generally seems to lead to a more precise separation of important and unimportant edges. Please note that this behavior does not apply to all

[2]The source code and target model checkpoints for our experiments are publicly available at https://anonymous.4open.science/r/ExplainableTKGC-1908/

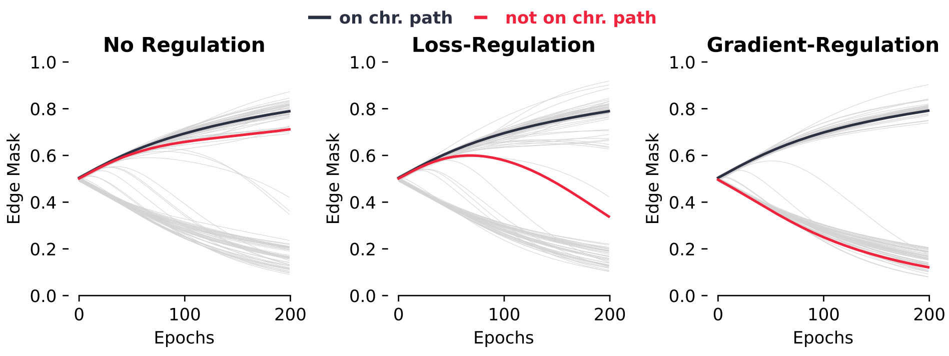


Figure 3: Comparison of the edge mask evolution with TARGCN as target model with and without regulation for a random sample from ICEWS14.

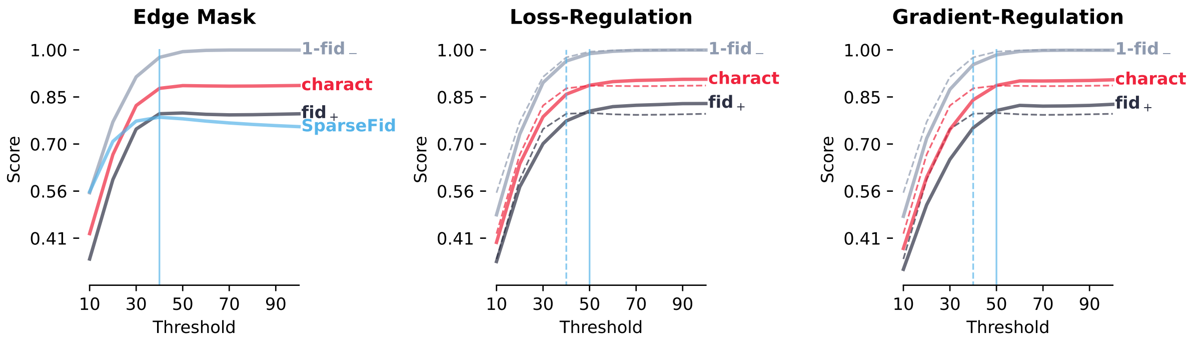


Figure 4: Performance of edge mask with and without regulation of TARGCN on ICEWS14 at different thresholds. The dotted lines show the edge mask performance without regulation. The vertical blue line shows the threshold that optimizes *SparseFid*.

edges on a non-chronological path. If we look at the same sample with the target model T-GAP, which can be found in App. D Fig. 5, we see that using the threshold has already removed all edges that the explainer considers unimportant. It can be seen that, compared to the target model TARGCN, the loss regulation seems not to influence the edge weights. For the gradient regulation, some edges are influenced.

Edge Masks Across Different Thresholds: The previous results show explanations with an edge mask threshold of 100. However, since we are not only interested in the fidelity of the explanation but also in achieving a high degree of sparsity, we have also evaluated low thresholds for the masks. The lower part of Tab. 1 reports the results of the GNNExplainer for both target models using the edge mask with and without the two regulations with the best threshold regarding the *SparseFid* score. We observe that the best threshold for all methods is below 100. For the target model TARGCN on the ICEWS14 dataset, the edge mask can achieve the highest *SparseFid* score. On the WIKIDATA11K dataset, loss regulation is still the best method. Gradient regulation also remains the best method for the target model T-GAP.

In the following, we look at the results for the target models i) TARGCN and ii) T-GAP in detail.

i) TARGCN The comparison of loss and gradient regulation to the baseline on the ICEWS14 dataset in Fig. 4 shows a similar trend of the scores depending on the threshold. However, the baseline can provide better results for lower thresholds. This is also indicated by the smaller optimal threshold of the baseline compared to the regulation methods. Loss and gradient regulation can only improve the baseline with thresholds of 50 or higher. Gradient regulation, in particular, struggles with high fidelity for very small thresholds.

The results on the WIKIDATA11K dataset show a significant improvement of the baseline for small thresholds using loss and gradient regulation, as can be observed in Fig. 6a in the appendix. This results in the optimal threshold being improved from 30 to 20 for both regulation methods. The *charact* score of the loss regulation is superior to the baseline for every threshold. Thus, the *charact* score of the loss regulation at a threshold of 30 is already above the baseline score with the maximum threshold of 100. Gradient regulation, on the other hand, can outperform the baseline for small thresholds. Above a threshold of 40, the improvements are minimal.

ii) T-GAP Since the results of the GNNExplainer for the target model T-GAP using the loss regulation for different thresholds show no difference to the baseline performance, we only report the results of the edge and node mask and the gradient regulation.

We report the results of T-GAP in the appendix in Fig. 6. In comparison to the baseline, the gradient regulation can only achieve very small improvements for a threshold of 100 on ICEWS14, as can be seen in Tab. 1. However, if we look at smaller explanations in Fig. 6b, we see an improvement in the *charact* score when using the gradient regulation. For a threshold of 30 to 60, a noticeable improvement can be observed compared to the baseline. The optimal threshold can be decreased to 80 using gradient regulation.

A very similar behavior can be found in the results on the WIKIDATA11K dataset in Fig. 6c. The optimal threshold for the edge mask without regulation is 70 but can be reduced to 50 using gradient regulation.

7. Discussion

Our results show improvements through temporal regulation for all models on all datasets in most scenarios. Often, we observe improvements through both regulation methods, or at least through one of them.

With an edge mask threshold of 100, the GNNExplainer can obtain the best *charact* score through loss regulation for TARGCN and gradient regulation for T-GAP. While the GNNExplainer for TARGCN can achieve an improvement with the gradient regulation compared to the edge mask without regulation, the loss regulation for T-GAP had no noticeable influence on the quality of the explanations according to the metrics used.

The explainer uses a significantly smaller edge mask for TARGCN than for T-GAP, which may be easier to optimize. This is because message passing is only performed for a sampled temporal 1-hop neighborhood of the subject node in this model. Since TARGCN limits this neighborhood to a maximum of 100 edges, the edge mask includes, at most, 100 parameters to optimize. In contrast, with T-GAP, message passing is performed for each edge in the graph, which means that the number of parameters in the edge mask is significantly larger than with TARGCN. This might cause the loss regulation to have only a small impact on explanations of T-GAP's predictions.

We can observe that the explanation quality seems to depend highly on the target model to be explained. Tab. 1 shows that TARGCN explanations are considerably better than explanations for T-GAP. This might be caused by i) the larger neighborhood context of T-GAP and the resulting complex inference of T-GAP compared to TARGCN ; ii) a difference in the TKGC prediction quality as TARGCN performs better than T-GAP on both datasets,[3] which makes explanations more difficult.

Furthermore, the optimal size of the edge mask seems to depend heavily on the underlying dataset and target model. The explainer consistently achieves a smaller optimal explanation threshold for TARGCN than for T-GAP. One reason could be that T-GAP considers the 3-hop neighborhood around the query node for its prediction, while TARGCN only considers the direct neighborhood. Therefore, T-GAP generally requires more edges to provide a reliable prediction than TARGCN. This is further

[3]With the original source code, we reproduced the original experiments and achieved TKGC scores close the ones publications with the models. TARGCN: 0.606 MRR on ICEWS14, 0.715 MRR on WIKIDATA11K; T-GAP: 0.56 MRR on ICEWS14, 0.663 MRR on WIKIDATA11K.

supported by the observation of the *charact* score curve in relation to the threshold shown in Fig. 4 and Fig. 6b in the appendix.

We evaluated the models following common methods and standards in XKGC and introduced a new metric to better reflect the explanations' size. A human evaluation to verify a model's capabilities in real-world scenarios is not common in XKGC tue to open challenges, especially with TKGs, as i) the standard KGC benchmark datasets require human experts in the respective domains, ii) no commonly accepted dataset for X(T)KGC exists, iii) existing state-of-the-art TKGC models require large subgraphs to make TKGC predictions. Even though our chronological regulation can reduce the size of explanations, a human evaluation still poses significant challenges and would be an interesting topic for future work.

8. Conclustion

In this work, we address the explainability of GNN-based TKGC models. We implement a baseline for GNN-based TKGC explanations using non-temporal GNN explainers and report the explanation quality according to established metrics. Furthermore, we proposed a regulation method that incentivizes a chronological order in the explanations to improve explanations over TKCs. We see this in improved explainability scores in most scenarios across models and datasets, e. g., with fidelity characterization scores increased by up to 2% compared to the baselines. We observe that the regulation methods can reduce the size of the explanation graph while maintaining the same explanation quality according to explainability metrics in most scenarios.

Acknowledgments

This work was supported by the Research Council of Norway through its Centres of Excellence scheme, Integreat – Norwegian Centre for Knowledge-driven Machine Learning, project number 332645.

Declaration on Generative AI

We used ChatGPT and Grammarly to check grammar and spelling and to make minor rephrasings for improved clarity. All changes were reviewed by us, and we take full responsibility for the content of this publication.

References

[1] Z. Ding, Y. Ma, B. He, J. Wu, Z. Han, V. Tresp, A Simple But Powerful Graph Encoder for Temporal Knowledge Graph Completion, in: Intelligent Systems and Applications, Springer Nature Switzerland, 2024, pp. 729–747.

[2] J. Jung, J. Jung, U. Kang, Learning to Walk across Time for Interpretable Temporal Knowledge Graph Completion, in: Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining, Association for Computing Machinery, 2021, p. 786–795.

[3] J. Wu, M. Cao, J. C. K. Cheung, W. L. Hamilton, TeMP: Temporal message passing for temporal knowledge graph completion, in: B. Webber, T. Cohn, Y. He, Y. Liu (Eds.), Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), Association for Computational Linguistics, Online, 2020, pp. 5730–5746. URL: https://aclanthology.org/2020.emnlp-main.462/. doi:10.18653/v1/2020.emnlp-main.462.

[4] H. Yuan, H. Yu, S. Gui, S. Ji, Explainability in Graph Neural Networks: A Taxonomic Survey, IEEE Transactions on Pattern Analysis and Machine Intelligence 45 (2023) 5782–5799.

[5] H. Chang, J. Ye, A. Lopez-Avila, J. Du, J. Li, Path-based explanation for knowledge graph completion, in: Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining,

KDD '24, Association for Computing Machinery, New York, NY, USA, 2024, p. 231–242. URL: https://doi.org/10.1145/3637528.3671683. doi:10.1145/3637528.3671683.

[6] S. Zhang, J. Zhang, X. Song, S. Adeshina, D. Zheng, C. Faloutsos, Y. Sun, Page-link: Path-based graph neural network explanation for heterogeneous link prediction, in: Proceedings of the ACM Web Conference 2023, WWW '23, Association for Computing Machinery, New York, NY, USA, 2023, p. 3784–3793. URL: https://doi.org/10.1145/3543507.3583511. doi:10.1145/3543507.3583511.

[7] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, G. E. Dahl, Neural Message Passing for Quantum Chemistry, in: Proceedings of the 34th International Conference on Machine Learning, PMLR, 2017, pp. 1263–1272.

[8] M. M. Bronstein, J. Bruna, T. Cohen, P. Veličković, Geometric Deep Learning: Grids, Groups, Graphs, Geodesics, and Gauges, 2021. URL: https://arxiv.org/abs/2104.13478. arXiv:2104.13478.

[9] Z. Ying, D. Bourgeois, J. You, M. Zitnik, J. Leskovec, GNNExplainer: Generating Explanations for Graph Neural Networks, in: Advances in Neural Information Processing Systems, volume 32, Curran Associates, Inc., 2019.

[10] J. Kakkad, J. Jannu, K. Sharma, C. Aggarwal, S. Medya, A Survey on Explainability of Graph Neural Networks, 2023. URL: https://arxiv.org/abs/2306.01958. arXiv:2306.01958.

[11] Z. Han, P. Chen, Y. Ma, V. Tresp, Explainable Subgraph Reasoning for Forecasting on Temporal Knowledge Graphs, in: International Conference on Learning Representations, 2021.

[12] J. Chen, R. Ying, TempME: Towards the Explainability of Temporal Graph Neural Networks via Motif Discovery, in: Advances in Neural Information Processing Systems, volume 36, Curran Associates, Inc., 2023, pp. 29005–29028.

[13] W. He, M. N. Vu, Z. Jiang, M. T. Thai, An Explainer for Temporal Graph Neural Networks, in: GLOBECOM 2022 - 2022 IEEE Global Communications Conference, 2022, pp. 6384–6389.

[14] W. Xu, B. Liu, M. Peng, X. Jia, M. Peng, Pre-trained language model with prompts for temporal knowledge graph completion, in: A. Rogers, J. Boyd-Graber, N. Okazaki (Eds.), Findings of the Association for Computational Linguistics: ACL 2023, Association for Computational Linguistics, Toronto, Canada, 2023, pp. 7790–7803. URL: https://aclanthology.org/2023.findings-acl.493/. doi:10.18653/v1/2023.findings-acl.493.

[15] M. Peng, B. Liu, W. Xu, Z. Jiang, J. Zhu, M. Peng, Deja vu: Contrastive historical modeling with prefix-tuning for temporal knowledge graph reasoning, in: K. Duh, H. Gomez, S. Bethard (Eds.), Findings of the Association for Computational Linguistics: NAACL 2024, Association for Computational Linguistics, Mexico City, Mexico, 2024, pp. 1178–1191. URL: https://aclanthology.org/2024.findings-naacl.75/. doi:10.18653/v1/2024.findings-naacl.75.

[16] D. Vrandečić, M. Krötzsch, Wikidata: a free collaborative knowledgebase, Commun. ACM 57 (2014) 78–85.

[17] A. García-Durán, S. Dumančić, M. Niepert, Learning Sequence Encoders for Temporal Knowledge Graph Completion, in: Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, 2018, pp. 4816–4821.

[18] K. Amara, R. Ying, Z. Zhang, Z. Han, Y. Shan, U. Brandes, S. Schemm, C. Zhang, Graphframex: Towards systematic evaluation of explainability methods for graph neural networks, 2024. URL: https://arxiv.org/abs/2206.09677. arXiv:2206.09677.

[19] K. Amara, Z. Ying, Z. Zhang, Z. Han, Y. Zhao, Y. Shan, U. Brandes, S. Schemm, C. Zhang, Graph-FramEx: Towards Systematic Evaluation of Explainability Methods for Graph Neural Networks, in: The First Learning on Graphs Conference, 2022.

[20] B. Yang, W. Yih, X. He, J. Gao, L. Deng, Embedding Entities and Relations for Learning and Inference in Knowledge Bases, in: Proceedings of the 3rd International Conference on Learning Representations, 2015.

A. Dataset Statistics

Tab. 2 shows the dataset characteristics of the datasets used in our experiments.

Table 2
Training set graph characteristics of ICEWS14 and WIKIDATA11K.

Stats	ICEWS14	WIKIDATA11K
Number of Nodes	7,128	11,153
Number of Edges	90,730	150,079
Number of Relations	230	181
Number of Timestamps	365	328
Graph Density [1]	1.79	1.21
Maximum Node Degree	6,083	586
Average Node Degree	25.46	26.91
Longest of All Shortest Paths	11	7
Shortest of All Shortest Paths	1	1
Average Shortest Path Length	4.11	1.20

[1] density multiplied by 1000

B. Hyper-parameter Search

The proposed chronological regulation methods add new hyperparameters to the GNNExplainer.

For all other parameters added for the chronological regulation, we use grid-search hyperparameter tuning with the parameters reported in Tab. 3 on 1000 samples for TARGCN and 500 for T-GAP. Note that it is also possible to optimize the hyperparameters for each sample individually since the GNNExplainer has to be trained separately for each sample by default. The best hyperparameters are determined by the *charact* score. However, as this can be artificially inflated with a very large explanation, we limit the explanation size to 100 edges. Except for the number of training epochs (200 for TARGCN and 100 for T-GAP), we do not change any default hyperparameters of the GNNExplainer.

C. TKGC Models

Time-aware Relational Graph Convolutional Network (TARGCN) [1] is based on a single GCN layer to aggregate graph neighborhood information. For every query $q = (s_q, r_q, ?, t_q)$, the model samples the temporal neighborhood $\bar{\mathcal{N}}(s_q, t_q) \subseteq \mathcal{N}(s_q, t_q)$ of the query node s_q at time t_q. Then, a GCN layer is used to aggregate information of $\bar{\mathcal{N}}(s_q, t_q)$ to encode the time-aware representation of entity s_q at time t_q, by combining time-invariant representations of relation r, entity e and implicit time difference information from the subset of all temporal neighbors.

$$\mathbf{h}_{(s_q,t_q)} = \frac{1}{|\bar{\mathcal{N}}_{(s_q,t_q)}|} \sum_{(e,t)\in\bar{\mathcal{N}}_{(s_q,t_q)}} \mathbf{W}(\mathbf{h}_{(e,t)}||\mathbf{h}_r), \tag{12}$$

where $\mathbf{h}_r$ denotes the time-invariant embedding of relation r and $\mathbf{h}_{(e,t)}$ the time-aware entity embedding $\mathbf{h}_{(e,t)}$ for $(e, t) \in \bar{\mathcal{N}}_{(s_q,t_q)}$.

For each possible candidate object o', a simplified time-aware representation is compared to s_q using DistMult decoding [20].

To apply the edge mask to TARGCN, we need to adjust Eq. 12 as follows:

$$\mathbf{h}_{(s_q,t_q)} = \frac{1}{|\bar{\mathcal{N}}_{(s_q,t_q)}|} \sum_{(e,t)\in\bar{\mathcal{N}}_{(s_q,t_q)}} \mathbf{W}((\mathbf{h}_{(e,t)}||\mathbf{h}_r) \odot \sigma(M)_{(e,t)}), \tag{13}$$

where $\sigma(M)_{(e,t)}$ is the sigmoid applied edge mask parameter for the edge connecting e with s_q at time t. M masks a feature that is based on the time-aware entity embedding and the time-invariable relation embedding r.

Temporal GNN with Attention Propagation (T-GAP) [2], another state-of-the-art TKGC model, considers distant nodes for encoding through multiple GNN layers. This allows the model to capture a richer context and potentially increase representativeness due to the increased information flow. T-GAP iteratively samples a subgraph based on node and edge attention values. Starting from a single node, each iteration adds nodes and edges based on their attention values to the subgraph. To complete the query, the node within the subgraph with the highest attention is predicted. T-GAP performs message-passing initially for each edge of the graph, as well as for all edges of the sampled subgraph in each iteration. While the weights vary across different layers and may also depend on the timestamp, the following message-passing scheme can always be found:

$$\mathbf{m}_{ij} = \mathbf{W}(\mathbf{h}_i + \mathbf{p}_{ij} + \tau_{|\Delta t_{ij}|}), \tag{14}$$

where $\mathbf{h}_i$ denotes the node features, $\mathbf{p}_{ij}$ the relation embedding, and $\tau_{|\Delta t_{ij}|}$ a temporal displacement embedding.

The implementation of the edge mask in T-GAP is similar to TARGCN. The message passing from Eq. 14 is modified as follows:

$$\mathbf{m}_{ij} = \mathbf{W}\left((\mathbf{h}_i + \mathbf{p}_{ij} + \tau_{|\Delta t_{ij}|}) \odot \sigma(M)_{ij}\right), \tag{15}$$

where the edge mask M is multiplied with each of the messages between node i and node j.

D. Further results

Fig. 6 shows the performance of the explainer with and without regulation at different thresholds.

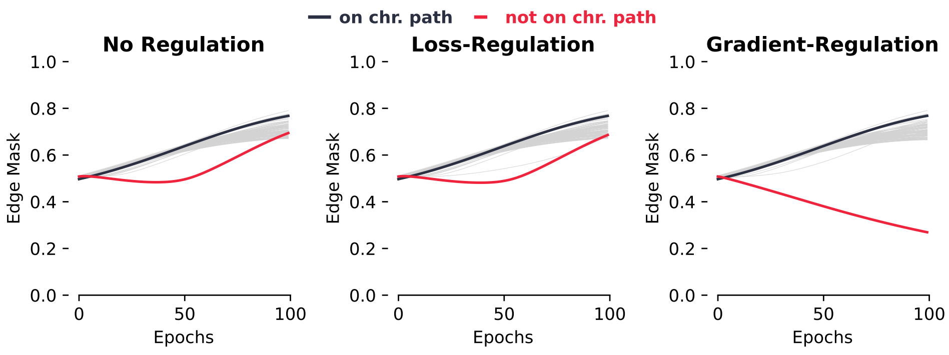
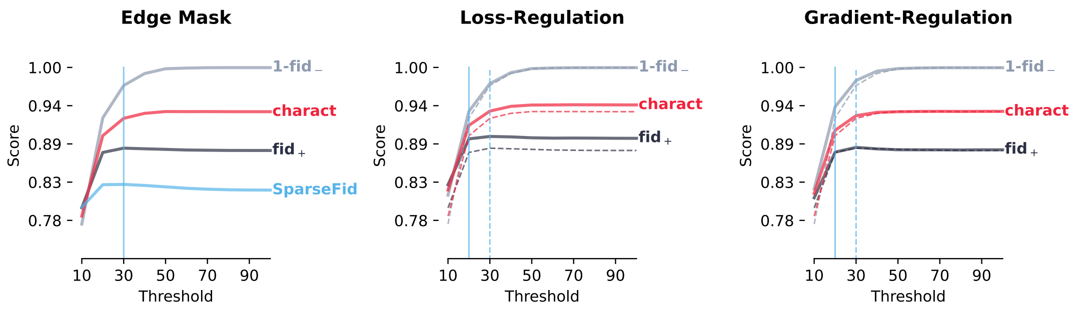


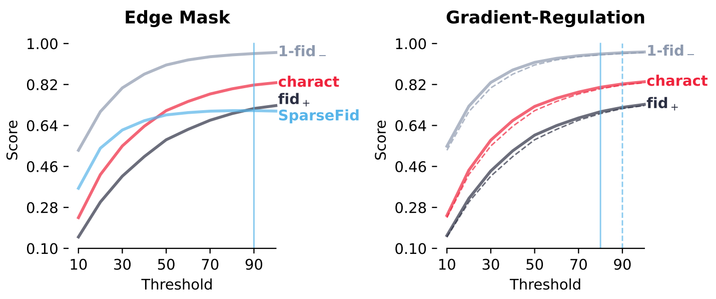
Figure 5: Comparison of the edge mask history of the GNNExplainer with T-GAP as target model with and without regulation for a random sample from ICEWS14.

E. Computing Resource

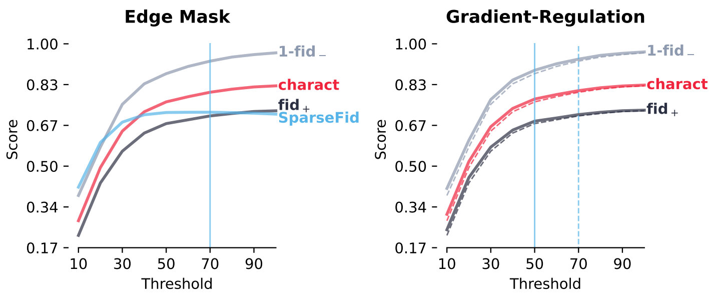
We ran the experiments on our GPU cluster with Nvidia A40 GPUs (older GPUs with less VRAM, e. g., Nvidia Tesla cards, are sufficient, too). For both target model training and the hyperparameter tuning, we used approx. 450h GPU hours. Note that our approach does not substantially increase the computation time of the existing GNNExplainer. We evaluated our approaches on existing TKGC datasets for comparability. These datasets were not developed for XAI and, therefore, contain large test sets that cause the runtime of our experiments. Furthermore, the large computation times are related to the target model T-GAP and are thus independent of our proposed approach.



(a) TARGCN on WIKIDATA11K



(b) T-GAP on ICEWS14



(c) T-GAP on WIKIDATA11K

Figure 6: Scores achieved with and without regulation at different thresholds. The dotted lines show the edge mask performance without regulation. The vertical blue line shows the threshold that optimizes *SparseFid*.

Table 3

GNNExplainer hyperparameter search space for target models TARGCN and T-GAP.

Hyperparameter	**Search Space**	**Best Result**			
		TARGCN		T-GAP	
		ICEWS14	*WIKIDATA11K*	*ICEWS14*	*WIKIDATA11K*
alpha[1](α)	{0.05, 0.1, 0.2, 0.4, 0.8}	0.8	0.4	0.05	0.8
beta[1](β)	{0.0, 0.33, 0.66, 1.0}	0.0	0.66	1.0	1.0
gamma[2](γ)	{0.05, 0.1, 0.2, 0.4, 0.8}	0.8	0.2	(0.8)	(0.8)
beta[2](β)	{0.0, 0.33, 0.66, 1.0}	0.33	0.0	(0.0)	(0.0)
epochs		200		100	
samples		1000		500	

[1] gradient-regulation only
[2] loss-regulation only