

Temporal Smoothness Regularisers for Neural Link Predictors

Manuel Dileo¹, Pasquale Minervini², Matteo Zignani¹, Sabrina Gaito¹

¹ Department of Computer Science, University of Milan, Milan, Italy

² School of Informatics, University of Edinburgh, Edinburgh, United Kingdom
manuel.dileo@unimi.it

Abstract

Most algorithms for representation learning and link prediction on relational data are designed for static data. However, the data to which they are applied typically evolves over time, including online social networks or interactions between users and items in recommender systems. This is also the case for graph-structured knowledge bases – knowledge graphs – which contain facts that are valid only for specific points in time. In such contexts, it becomes crucial to correctly identify missing links at a precise time point, i.e. the temporal prediction link task. Recently, Lacroix et al. [1] and Sadeghian et al. [2] proposed a solution to the problem of link prediction for knowledge graphs under temporal constraints inspired by the canonical decomposition of 4-order tensors, where they regularise the representations of time steps by enforcing temporal smoothing, i.e. by learning similar transformation for adjacent timestamps. However, the impact of the choice of temporal regularisation terms is still poorly understood. In this work, we systematically analyse several choices of temporal smoothing regularisers using linear functions and recurrent architectures. In our experiments, we show that by carefully selecting the temporal smoothing regulariser and regularisation weight, a simple method like TNTComplEx [1] can produce significantly more accurate results than state-of-the-art methods on three widely used temporal link prediction datasets. Furthermore, we evaluate the impact of a wide range of temporal smoothing regularisers on two state-of-the-art temporal link prediction models. We observe that linear regularisers for temporal smoothing based on specific nuclear norms can significantly improve the predictive accuracy of the base temporal link prediction methods. Our work shows that simple tensor factorisation models can produce new state-of-the-art results using newly proposed temporal regularisers, highlighting a promising avenue for future research.

1 Introduction

Knowledge Graphs (KGs) [3] are graph-structured knowledge bases where knowledge about the world is encoded in the form of relationships of various kinds between entities. KGs are an extremely flexible and versatile knowledge representation formalism, and are currently being used in a variety of domains, including bioinformatics [4–8], recommendation [9], linguistics [10], and industry applications [11]. Moreover, relational data from such domains is often temporal; for example, the action of buying an item or watching a movie is associated with a timestamp, and some medicines might interact differently depending on when they are administered to a patient. We refer to KGs augmented with temporal information as Temporal Knowledge Graphs (TKGs).

When reasoning with TKGs, one of the most crucial tasks is finding or completing missing links in the temporal knowledge graph at a precise time point, often referred to as the temporal link prediction task. One class of models for tackling the problem of identifying missing links in large

static or temporal KGs is neural link predictors [12], i.e. differentiable models which map entities and relationships into a d -dimensional embedding space, and use entity and relation embeddings for scoring missing links in the graph.

Recently, Lacroix et al. [1] and Sadeghian et al. [2] proposed two state-of-the-art approaches to temporal link prediction for temporal knowledge graphs. The key idea is to extend factorisation-based neural link prediction models with dense representations for each timestamp and to regularise such representations by enforcing them to change slowly over time. However, the impact of the choice of temporal regularisation terms is still not well understood.

Hence, in this work, we systematically analyse a comprehensive array of temporal regularisers for neural link predictors based on tensor factorisation. Starting from the temporal smoothing regulariser proposed in [1], we also consider a wide class of norms in the L_p and N_p families which allow to control the strength of the smoothing. We further extend the set of temporal regulariser by taking into account the regulariser proposed in [13], defining it as an explicit modelling of the temporal dynamic between adjacent timestamps. Lastly, we propose to adopt a recurrent architecture as an implicit temporal regulariser that can generate timestamp embeddings sequentially and learn the temporal dynamic by updating its parameters during the training phase.

We conducted the experimental evaluation over three well-known benchmark datasets: ICEWS14, ICEWS05-15, and YAGO15K. ICEWS14 and ICEWS05-15 are subsets of the widely used Integrated Crisis Early Warning System knowledge graph [ICEWS, 14], while YAGO15K [15] is a dataset derived from YAGO [16] covering the entities appearing in FB15k [17], which adds *occursSince* and *occursUntil* timestamps to each triple.

Our results show that using our proposed temporal regularisers, neural link predictors based on tensor factorisation models can significantly improve their predictive accuracy in temporal link prediction tasks. By carefully selecting a temporal regulariser and regularisation weight, our version of TNTComplEx [1] produces more accurate results than all of the baselines on ICEWS14, ICEWS05-15, and YAGO15K. Overall, linear regularizers for temporal smoothing that introduce smaller loss penalties for closer timestamp representations achieve the best performance. In contrast, recurrent architecture struggles to generate a long sequence of timestamps.

2 Background

A Knowledge Graph $\mathcal{G} \subseteq \mathcal{E} \times \mathcal{R} \times \mathcal{E}$ contains a set of subject-predicate-object $\langle s, p, o \rangle$ triples, where each triple represents a relationship of type $p \in \mathcal{R}$ between the subject $s \in \mathcal{E}$ and the object $o \in \mathcal{E}$ of the triple. Here, \mathcal{E} and \mathcal{R} denote the set of all entities and relation types, respectively. However, many real-world knowledge graphs are largely incomplete [18, 19, 12, 20] – link prediction focuses on the problem of identifying missing links in (possibly very large) knowledge graphs.

2.1 Static Neural Link Predictors

Neural link predictors, also referred to as KG embedding models, are neural models that yield state-of-the-art accuracy on a wide array of link prediction benchmarks [21–23]. A neural link predictor is a neural model where entities in \mathcal{E} and relation types in \mathcal{R} are represented in a continuous embedding space, and the likelihood of a link between two entities is a function of their representations. More formally, neural link predictors are defined by a parametric scoring function $\phi_\theta : \mathcal{E} \times \mathcal{R} \times \mathcal{E} \mapsto \mathbb{R}$, with parameters θ that, given a triple $\langle s, p, o \rangle$, produces the likelihood that entities s and o are related by the relationship p . See Appendix for a detailed description of static neural link predictors.

2.2 Temporal Knowledge Graph Completion

A Temporal Knowledge Graph (TKG) is referred to a set of quadruples $\mathcal{K} \subseteq \{(s, p, o, \tau) \mid s, o \in \mathcal{E}, p \in \mathcal{R}, \tau \in \mathcal{T}\}$. Each quadruple represents a temporal fact that is true in a world. \mathcal{E} is the set of all entities and \mathcal{R} is the set of all relations in the ontology. The fourth element in each quadruple represents time, which is often discretised. \mathcal{T} represents the set of all possible timestamps. Temporal Knowledge Graph Completion, also known as temporal link prediction, refers to the problem of completing a TKG by inferring facts from a given subset of its facts.

The subject of temporal link prediction has been studied in a wide range of approaches [34]. For a general overview of temporal link prediction models see the Appendix. In this work, we focus on methods that learn the temporal behaviour by using a representation for time. For instance, Lacroix et al. [1] perform tensor decomposition based on the time representation, while Sadeghian et al. [2] learns a d -dimensional rotation transformation parametrized by relation and time, such that after each fact’s head entity representation is transformed using the rotation, it falls near its corresponding tail entity. These methods achieve state-of-the-art results on most of the benchmark datasets [2, 13]. Hence, we systematically analyse a wide array of temporal regularisers to understand their impact on both performance and learning temporal behaviour.

3 Temporal KG Representation Learning

This section presents a framework for temporal knowledge graph representation learning [1]. Given a TKG, we want to learn representations for entities, relations, and timestamps (e.g., $\mathbf{s}, \mathbf{p}, \mathbf{o}, \mathbf{t}_\tau \in \mathbb{R}^d$) and a scoring function $\phi_\theta(s, p, o, \tau) \in \mathbb{R}$, such that true quadruples receive high scores. Thus, given ϕ_θ , the embeddings can be learned by optimising an appropriate cost function. Following Lacroix et al. [30], we minimise, for each of the train tuples (s, p, o, τ) , the instantaneous multi-class loss:

$$\ell(\phi_\theta; (s, p, o, \tau)) = -\phi_\theta(s, p, o, \tau) + \log \left(\sum_{o' \in \mathcal{E}} \exp(\phi_\theta(s, p, o', \tau)) \right), \quad (1)$$

Note that this loss is only suited to queries of the type (subject, predicate, ?, time), which are the queries that were considered in related work. For a training set S (augmented with reciprocal relations [30, 57]), and parametric tensor estimate ϕ_θ , we minimize the following objective, with a weighted embedding regulariser Ω :

$$\mathcal{L}(\phi_\theta) = \frac{1}{|S|} \sum_{(s, p, o, \tau) \in S} [\ell(\phi_\theta; (s, p, o, \tau)) + \lambda \Omega(\theta; (s, p, o, \tau))]. \quad (2)$$

In our experiments, embedding regularisation is performed using a nuclear tensor 3-norm [30]. In the following subsections, we introduce the models, scoring functions, and temporal regularisers considered for our analysis.

3.1 TNTComplEx

TNTComplEx [1] extends the ComplEx [30] decomposition to the TKG completion setting by adding a new factor T resulting in the following scoring function ϕ_θ :

$$\phi_\theta^{\text{TComplEx}}(s, p, o, \tau) = \text{Re}(\langle \mathbf{s}, \mathbf{p}, \bar{\mathbf{o}}, \mathbf{t}_\tau \rangle) \quad (3)$$

where \mathbf{s} and \mathbf{o} are the embeddings of entities $s, o \in \mathcal{E}$, \mathbf{p} is the embedding for the relation $p \in \mathcal{R}$, and \mathbf{t}_τ is the embedding for the timestamp $\tau \in \mathcal{T}$. Intuitively, they added timestamps embedding that modulate the multi-linear dot product.

In heterogeneous knowledge bases, where only part of the relation types are temporal, they introduce an embedding representation – \mathbf{p}^t – whether the relation p is temporal and an embedding \mathbf{p} otherwise. Thus, the scoring function $\phi_\theta^{\text{TComplEx}}$ is extended into the TNTComplEx scoring function $\phi_\theta^{\text{TNTComplEx}}$:

$$\phi_\theta^{\text{TNTComplEx}}(s, p, o, \tau) = \text{Re}(\langle \mathbf{s}, \mathbf{p}^t \odot \mathbf{t}_\tau + \mathbf{p}, \bar{\mathbf{o}} \rangle) \quad (4)$$

3.2 ChronoR

ChronoR [2] is inspired by rotation-based models in static KG completion [58]. For a training set S containing train tuples (s, p, o, τ) , with $|S| = n$, they expect that for true facts it holds:

$$\mathbf{Q}_{\mathbf{p}, \mathbf{t}_\tau} \odot \mathbf{s} = \mathbf{o} \quad (5)$$

where $\mathbf{s}, \mathbf{o} \in \mathbb{R}^{n \times d}$ denote the entity embeddings obtained by vertically concatenating the embedding of the entity s or o in each training tuple, and $\mathbf{Q}_{\mathbf{p}, \mathbf{t}_\tau}$ represents the (row-wise) linear operator parameterised by the relation and timestamp embeddings \mathbf{p} and \mathbf{t}_τ . Specifically, \mathbf{Q} is parameterized

by $\mathbf{p}, \mathbf{t}_\tau$ by concatenating the embeddings to get $\mathbf{Q}_{\mathbf{p}, \mathbf{t}_\tau} = [\mathbf{p} \parallel \mathbf{t}_\tau]$, where $\mathbf{p} \in \mathbb{R}^{n_p \times d}$ and $\mathbf{t}_\tau \in \mathbb{R}^{n_\tau \times d}$ are the representations of the relation type and time elements, and $n_p + n_\tau = n$. Thus, the scoring function of ChronoR is defined as:

$$\phi_\theta^{\text{ChronoR}}(s, p, o, \tau) = \langle \mathbf{s} \odot [\mathbf{p} \parallel \mathbf{t}_\tau]_{r,:}, \mathbf{o} \rangle, \quad (6)$$

where r indicates the row in $\mathbf{Q}_{\mathbf{p}, \mathbf{t}_\tau}$ corresponding to the train tuple (s, p, o, τ) . It is worth noting that, unlike RotatE [58], which uses a scoring function based on the Euclidean distance of $\mathbf{Q}_{\mathbf{p}} \odot \mathbf{s}$ and \mathbf{o} , they propose to use the angle between the two vectors, or the Frobenius inner product for the matrix formulation.

As well as for TNTComplEx, to treat TKGs that store a combination of static and dynamic facts, they allow an extra rotation operator $\mathbf{p}' \in \mathbb{R}^{n \times d}$, leading to the following scoring function:

$$\phi_\theta^{\text{ChronoR}}(s, p, o, \tau) = \langle \mathbf{s} \odot [\mathbf{p} \parallel \mathbf{t}_\tau]_{r,:} \odot \mathbf{p}'_{r,:}, \mathbf{o} \rangle. \quad (7)$$

3.3 Temporal regularisers

Temporal regularisers encourage tensor factorisation models to learn transformation for timestamp embeddings that capture specific temporal properties of real datasets. For instance, one would like the model to take advantage of the fact that most entities behave smoothly over time, i.e. learn similar transformations for closer timestamps. Alternatively, one would like to push away the representation of distant timestamps or allow the timestamp embeddings to be generated sequentially.

Formally, a temporal regulariser is a weighted penalty term $\Lambda(T)$ in the loss function, leading to the minimisation of the following objective:

$$\mathcal{L}(\phi_\theta) = \frac{1}{|S|} \sum_{(s,p,o,\tau) \in S} [\ell(\phi_\theta; (s, p, o, \tau)) + \lambda_1 \Omega(\theta; (s, p, o, \tau))] + \lambda_2 \Lambda(T). \quad (8)$$

Below we define the temporal regularisers considered in our analysis.

Temporal smoothing Most of the work in the literature [1, 2, 56] adds a temporal smoothness objective to the loss function to encourage neighbouring timestamps to have close representations. The temporal smoothing regulariser can be defined as:

$$\Lambda_p(T) = \frac{1}{|T| - 1} \sum_{\tau=1}^{|T|-1} \|\mathbf{t}_{\tau+1} - \mathbf{t}_\tau\|_p^p. \quad (9)$$

Despite being a very simple temporal regulariser, smoothing leads to an increase in performance on several benchmark datasets [1, 2]. However, it has been tested using only the N_3 norm. Hence, we decide to further investigate its impact by considering a wide class of norms in the L_p and N_p families, leading to the following temporal regularisation terms:

$$\Lambda_{L_p}(T) = \frac{1}{|T| - 1} \left(\sum_{\tau=1}^{|T|-1} |\mathbf{t}_{\tau+1} - \mathbf{t}_\tau|^p \right)^{\frac{1}{p}}, \quad (10)$$

$$\Lambda_{N_p}(T) = \frac{1}{|T| - 1} \sum_{\tau=1}^{|T|-1} |\mathbf{t}_{\tau+1} - \mathbf{t}_\tau|^p. \quad (11)$$

In our intuition, the choice of the norm function controls the strength of the smoothing. Thus, it provides several ways to penalize a change in the behaviour of the entities on neighbouring timestamps. We plot some of the considered norms in fig. 1. Using an L_p norm always penalises no equal timestamp representations, and the value of p determines the magnitude of the penalty. Using an N_p norm allows not penalising close representation for neighbouring timestamps (in the figure, N_5 starts to grow only for $|\mathbf{t}_{\tau+1} - \mathbf{t}_\tau| \geq 0.4$). In this case, the value of p controls the magnitude of the penalty and the range of distances to not penalise.

Linear3 Regulariser In [13], they propose a new temporal regulariser, namely Linear3, that can be defined as follows:

$$\Lambda_p(T) = \frac{1}{|T| - 1} \sum_{\tau=1}^{|T|-1} \|\mathbf{t}_{\tau+1} - \mathbf{t}_\tau - \mathbf{W}_b\|_p^p, \quad (12)$$

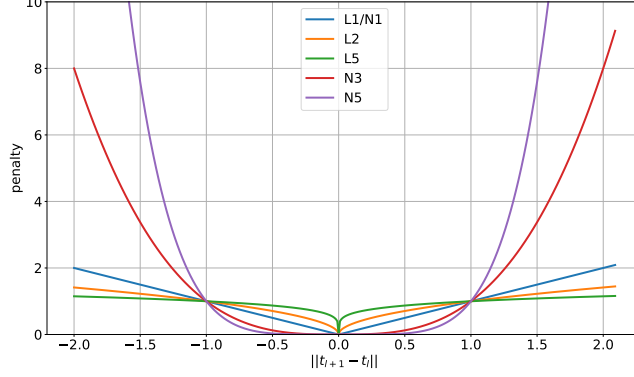


Figure 1: Plot of norm functions on the interval $[-2; 2]$.

where $\mathbf{W}_b \in \mathbb{R}^d$ denotes the embedding of a bias component between the neighbouring temporal embeddings, d the embedding size. The bias embedding is randomly initialised and then learned from the training process.

This linear regulariser promotes that the difference between embeddings of two adjacent timestamps is smaller than the difference between embeddings of two distant timestamps. In the context of this study, it is interesting to notice that Linear3 can be interpreted as the average score of triples $(\tau + 1, \textit{follows}, \tau)$, where the embedding of the relation *follows* is given by the bias component. From this point of view, Linear3 encourages similar embeddings for neighbouring timestamps by explicitly modelling their temporal dynamic through the predicate *follows*.

Modelling Temporal Dynamics via Recurrent Architectures Timestamp embeddings can be generated sequentially using a recurrent neural architecture that, starting from a random initialised hidden state, can learn implicitly the temporal dynamic through the learning process. Given a specific Recurrent Neural Network (RNN) architecture, the equation that describes its forward procedure acts as a temporal regulariser that maps the embedding of timestamp τ as:

$$\mathbf{t}_{\tau} = MLP(RNN(\mathbf{h}_{\tau-1}, \mathbf{0})); \tau \in \{1, \dots, |\mathcal{T}|\} \quad (13)$$

where $h_0 \in \mathbb{R}^m$ is the learnable initial hidden state, $\mathbf{0}$ is the zero vector, RNN is the function that describes the recurrent architecture, MLP is a function that describes one multi-layer perceptron layer that has (m, d) channels, d is the embedding dimension and $m < d$.

In our experiments, we consider RNN, LSTM, and GRU as recurrent architecture and their linear counterpart, i.e. by removing non-linear operations from the model.

4 Experiments

We evaluate the impact of temporal regularisers on TNTComplEx and ChronoR models for temporal link prediction on temporal knowledge graphs. We tune all the hyper-parameters on the validation set provided with each dataset using a grid search. We tune λ_1 and λ_2 from $\{10^i \mid -4 \leq i \leq 1\}$, the p value for N_p and L_p norms from $\{i \mid 1 \leq i \leq 5\}$, the embedding dimensions for tensors and recurrent architecture from $\{5, 25, 50, 100, 500, 2000\}$.

The training was done using mini-batch stochastic gradient descent with Adam and a learning rate of 0.1 with a batch size of 1000 quadruples. We implemented all our models in PyTorch. The source code to reproduce the full experimental results is made public on GitHub¹.

4.1 Datasets

We assess the performance of our model on three well-known benchmarks for Temporal Knowledge graph completion: ICEWS14, ICEWS05-15, and YAGO15K. All these datasets exclusively consist

¹<https://anonymous.4open.science/r/tkbc-reg-B77C/>

	ICEWS14	ICEWS05-15	YAGO15K
# Entities	7,128	10,488	15,403
# Relations	230	251	34
# Timestamps	365	4,017	198
# Facts	90,730	479,329	138,056
Time Spans	2014	2005 - 2015	1513 - 2017

Table 1: Dataset statistics for ICEWS14, ICEWS05-15, and YAGO15K.

Model	ICEWS14				ICEWS05-15				YAGO15K			
	MRR	Hit@1	Hits@3	Hit@10	MRR	Hit@1	Hit@3	Hit@10	MRR	Hit@1	Hit@3	Hit@10
TransE (2013)	28.0	9.4	-	63.70	29.4	8.4	-	66.30	29.6	22.8	-	46.8
DistMult (2014)	43.9	32.3	-	67.2	45.6	33.7	-	69.1	27.5	21.5	-	43.8
Simple (2018)	45.8	34.1	51.6	68.7	47.8	35.9	53.9	70.8	-	-	-	-
Complex (2016)	47.0	35.0	54.0	71.0	49.0	37.0	55.0	73.0	36.0	29.0	36.0	54.0
ConT (2018)	18.5	11.7	20.5	31.50	16.4	10.5	18.9	27.20	-	-	-	-
TTransE (2016)	25.5	7.4	-	60.1	27.1	8.4	-	61.6	32.1	23.0	-	51.0
TA-TransE (2018)	27.5	9.5	-	62.5	29.9	9.6	-	66.8	32.1	23.1	-	51.2
HyTE (2018)	29.7	10.8	41.6	65.5	31.6	11.6	44.5	68.1	-	-	-	-
TA-DistMult (2018)	47.7	-	36.3	68.6	47.4	34.6	-	72.8	29.1	21.6	-	47.6
DE-Simple (2020)	52.6	41.8	59.2	72.5	51.3	39.2	57.8	74.8	-	-	-	-
TIMEPLEX (2020)	60.40	51.50	-	77.11	63.99	54.51	-	81.81	-	-	-	-
TeRo (2020)	56.2	46.8	62.1	73.2	58.6	46.9	66.8	79.5	-	-	-	-
BoxTE (2022)	61.5	53.2	66.7	76.7	66.7	58.2	71.9	82.0	-	-	-	-
TeLM (2021) ²	61.41	53.39	66.0	76.12	66.70	58.99	71.28	80.85	-	-	-	-
ChronoR (2021) ²	56.97	46.50	63.66	76.06	60.64	49.39	67.97	81.13	32.89	25.88	33.51	50.71
ChronoR+Np/Lp (ours)	57.45	47.26	64.00	76.06	62.10	51.62	68.96	81.06	33.89	27.14	34.33	51.40
TNTComplex (2020) ²	60.72	51.91	65.92	77.17	66.64	58.34	71.82	81.67	35.94	28.49	36.84	53.75
TNTComplex+Np/Lp (ours)	61.80	53.60	66.55	76.97	67.70	59.90	72.35	82.30	37.05	29.00	39.62	54.02

Table 2: Evaluation on the YAGO15K, ICEWS14, and ICEWS05-15 datasets. Results reported for previous related works are the best numbers reported in their respective paper.² Results for our experiments on ChronoR and TNTComplex (ours) are reported for the best configuration of hyperparameters based on the validation MRR.

of positive triples. ICEWS14 and ICEWS05-15 are subsets of the widely used Integrated Crisis Early Warning System (ICEWS) knowledge graph. ICEWS14 covers the time span from 01/01/2014 to 12/31/2014, while ICEWS15-05 represents the subset between 01/01/2005 and 12/31/2015. Both datasets include timestamp information for each fact, with a temporal resolution of 24 hours. Importantly, these datasets are deliberately selected to encompass only the most frequently occurring entities in both the head and tail positions [1, 2]. To create YAGO15K, García-Durán et al. [15] aligned the entities in FB15K [17] with those from YAGO [16], which contains temporal information. The final dataset is the result of all facts with successful alignment. It is worth noting that since YAGO does not have temporal information for all facts, this dataset is also temporally incomplete and more challenging. To adapt YAGO15K to our models, following [1], for each fact, we group the relations *occursSince* and *occursUntil* together, in turn doubling our relation size. Note that this does not affect the evaluation protocol. table 1, summarises the statistics of used temporal KG benchmarks.

4.2 Evaluation setup

We follow the experimental set-up described in [15] and [49]. For each quadruple (s, p, o, τ) in the test set, we fill $(s, p, ?, \tau)$ and $(?, p, o, \tau)$ by scoring and sorting all possible entities in \mathcal{E} . We report Hits@k for $k = 1, 3, 10$ and filtered Mean Reciprocal Rank (MRR) for all datasets. We refer to Nickel et al. [12] for more details about the evaluation metrics.

We use baselines from both static and temporal KG embedding models. We use TransE, DistMult, Simple, and Complex from the static KG embedding models. These models ignore the timing information. An important consideration when evaluating these models on temporal KGs in the filtered setting is that it is necessary for each test quadruple to filter out previously encountered entities based on the corresponding fact and its associated timestamp. This step ensures a fair comparison between different models. To analyse the impact of temporal regularisers on tensor factorisation models, we compare our TNTComplex and ChronoR models against their original counterparts. To

Model	ICEWS14				ICEWS05-15				YAGO15K			
	MRR	Hit@1	Hits@3	Hit@10	MRR	Hit@1	Hit@3	Hit@10	MRR	Hit@1	Hit@3	Hit@10
ChronoR + N3 (2021)	56.97	46.50	63.66	76.06	60.64	49.39	67.97	81.13	32.89	25.88	33.51	50.71
ChronoR + L1	56.91	46.45	63.60	75.76	53.89	41.96	60.65	76.82	33.86	27.04	34.27	51.33
ChronoR + L2	57.42	47.12	63.97	76.35	59.75	48.99	66.77	79.01	33.89	27.14	34.33	51.40
ChronoR + L3	57.34	46.94	63.83	76.06	62.10	51.62	68.96	81.06	32.98	25.88	33.75	50.98
ChronoR + L4	57.45	47.26	64.00	76.06	61.39	50.53	68.49	81.00	33.31	26.27	33.97	51.20
ChronoR + L5	57.26	46.96	63.67	76.12	62.02	51.49	68.87	81.07	33.33	26.27	34.00	51.34
ChronoR + N2	56.89	46.32	63.71	76.14	61.68	50.87	68.83	81.14	33.08	26.08	33.57	51.12
ChronoR + N4	56.99	46.39	63.80	76.23	60.95	49.82	68.26	81.35	32.31	24.25	33.98	51.62
ChronoR + N5	57.01	46.37	63.82	76.41	60.26	49.00	67.58	81.01	33.03	25.18	34.43	51.96
TNTComplEx + N3 (2020)	60.72	51.91	65.92	77.17	66.64	58.34	71.82	81.67	35.94	28.49	36.84	53.75
TNTComplEx + L1	61.05	52.92	65.69	76.34	65.38	57.05	70.58	80.45	36.71	28.75	39.07	53.64
TNTComplEx + L2	61.40	53.20	66.27	76.65	66.69	58.63	71.56	81.40	36.88	28.84	39.42	54.00
TNTComplEx + L3	60.85	52.24	66.18	76.45	65.58	56.48	71.27	82.41	36.72	28.71	39.34	53.45
TNTComplEx + L4	61.34	53.07	66.16	76.51	66.81	58.49	72.05	81.92	36.87	28.64	39.66	54.25
TNTComplEx + L5	61.36	53.17	66.11	76.56	65.35	56.42	70.69	82.09	36.84	28.58	39.46	54.49
TNTComplEx + N2	61.18	52.87	66.27	76.26	66.96	59.01	71.79	81.56	36.52	29.00	37.65	54.39
TNTComplEx + N4	61.80	53.60	66.55	76.97	67.59	59.73	72.26	82.20	36.81	29.24	38.02	54.50
TNTComplEx + N5	61.60	53.41	66.42	76.73	67.70	59.90	72.35	82.30	37.05	29.00	39.62	54.02
TNTComplEx + RNN	57.87	47.95	63.85	76.33	54.35	42.49	61.38	76.94	36.15	29.55	36.42	53.53
TNTComplEx + Linear3	61.53	53.31	66.30	76.77	67.54	59.50	72.53	82.19	36.60	28.38	39.17	54.14

Table 3: Evaluation on the YAGO15K, ICEWS14, and ICEWS05-15 datasets for TNTComplEx and ChronoR models using different temporal regularisers. We highlight the best results for both families of models.

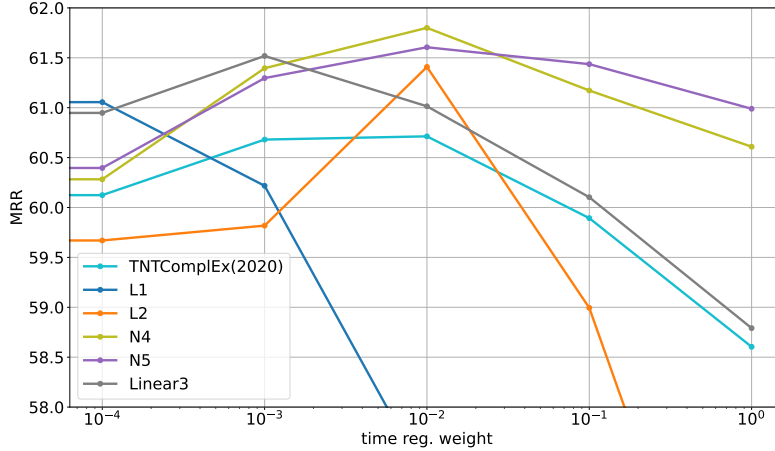


Figure 2: Comparison of various temporal regularisers with different regularisation weights on TNTComplEx trained on ICEWS14.

the best of our knowledge, we also compare our results with all temporal KG embedding models from the literature that has been evaluated on these datasets, which we discussed the details of in section 2, except for TeMP [53], as it uses negative sampling for the evaluation, leading to an unfair comparison with our models.

4.3 Results

In this section, we analyse and perform a quantitative comparison of our models and previous state-of-the-art ones. We also experimentally verify the impact of several temporal regularisers on tensor factorisation models.

table 2 demonstrates link prediction performance comparison on all datasets. Our TNTComplEx and ChronoR models achieve better performance than their original counterparts. Most importantly, TNTComplEx, with one of our proposed temporal regularisers, outperforms all the competitors in terms of link prediction MRR and Hits@1 metric on the three datasets.

table 3 shows link prediction performance comparison between all the considered temporal regularisers. The results show that the most accurate models on the three datasets are trained by setting the value of the hyperparameter p to four or five, i.e. the greatest choices considered. As described in section 3, the p value determines the magnitude of the penalty and contributes to controlling the strength of the temporal smoothness. In L_p norms, the penalty values decrease as p increases, while in N_p norms, larger values for p widen the range of differences not to penalise. Hence, we achieve state-of-the-art performances on temporal knowledge graph completion on the three datasets by weakening the strength of temporal smoothness. It is important to recall that temporal smoothing improves the performance of tensor factorisation models [1, 2, 13] so reduce its strength to zero (i.e. by removing its penalty term from the loss function) would not be beneficial. In ChronoR, the best performances are achieved using L_p norms, while an opposite trend characterizes TNTComplEx. Therefore, it seems the scoring function adopted by the tensor factorisation models influences the choice of the temporal regulariser. In our intuition, the assumption made by ChronoR in Eq. 5, which leads to its scoring function, imposes an additional constraint on the embedding representations, compared to TNTComplEx, that leads to benefit L_p norms.

Linear3 achieves the second-best results on all three datasets. Indeed, the embedding representation \mathbf{W}_b (see eq. (12)) is adjusted through the training process so the models can learn how to weaken the temporal smoothing. On the contrary, using a recurrent architecture for temporal regularisation leads to the worst results. In our intuition, RNNs struggle to generate very long sequences of embeddings, as in our three datasets (see Timestamps in table 1). Indeed, in our experiments, as the number of timestamps to generate increases, the output of the RNN at step τ converges to be equal to the output of step $\tau - 1$.

In fig. 2, we plot a detailed comparison of some of our proposed regularisers to the one proposed by Lacroix et al. [1] for TNTComplEx, on ICEWS14 dataset. N_4 increases MRR by 1.08 points, and carefully selecting regularisation weight can increase MRR up to 3.2 points.

5 Conclusion

In this work, we analyse the impact of several temporal regularisers on tensor factorisation models for link prediction in temporal knowledge graphs. Specifically, we compare several choices of temporal smoothing regularisers using N_p and L_p norms, linear functions, and recurrent architectures. Our experiments show that we can significantly improve the downstream link prediction accuracy in TNTComplEx and ChronoR by carefully selecting a temporal regulariser and corresponding weight hyperparameter. By doing so, our version of TNTComplEx outperforms all baselines in terms of MRR on three benchmark datasets. Overall, temporal regularisers that weaken the temporal smoothing penalty on shorter time embedding differences, like N_4 , N_5 , and Linear3, produce the best results in terms of temporal link prediction accuracy across all considered datasets.

Future Works We plan to extend our analysis to inductive tasks and show how temporal regularisers can be generalised to work with unseen timestamps, entities, and relation types – for example, by leveraging recent work connecting factorisation-based models and GNNs [59].

References

- [1] Timothée Lacroix, Guillaume Obozinski, and Nicolas Usunier. Tensor decompositions for temporal knowledge base completion. In *ICLR*. OpenReview.net, 2020.
- [2] Ali Sadeghian, Mohammadreza Armandpour, Anthony Colas, and Daisy Zhe Wang. Chronor: Rotation based temporal knowledge graph embedding. In *AAAI*, pages 6471–6479. AAAI Press, 2021.
- [3] Aidan Hogan, Eva Blomqvist, Michael Cochez, Claudia d’Amato, Gerard de Melo, Claudio Gutierrez, Sabrina Kirrane, José Emilio Labra Gayo, Roberto Navigli, Sebastian Neumaier,

²The original results for TNTComplEx were reported on the validation set; we use the code and hyperparameters from the official repository to re-run the model and report test set values. The original implementation of ChronoR used by Sadeghian et al. [2] is unavailable; we re-implement the solution in our framework, reporting the result according to the original configuration of hyperparameters. TeLM uses approximately the same number of parameters as TNTComplEx and ChronoR; more information is available in the appendix.

- Axel-Cyrille Ngonga Ngomo, Axel Polleres, Sabbir M. Rashid, Anisa Rula, Lukas Schmelzeisen, Juan F. Sequeda, Steffen Staab, and Antoine Zimmermann. Knowledge graphs. *ACM Comput. Surv.*, 54(4):71:1–71:37, 2022.
- [4] Marinka Zitnik, Monica Agrawal, and Jure Leskovec. Modeling polypharmacy side effects with graph convolutional networks. *Bioinform.*, 34(13):i457–i466, 2018.
 - [5] Aryo Pradipta Gema, Dominik Grabarczyk, Wolf De Wulf, Piyush Borole, Javier Antonio Alfaro, Pasquale Minervini, Antonio Vergari, and Ajitha Rajan. Knowledge graph embeddings in the biomedical domain: Are they useful? A look at link prediction, rule learning, and downstream polypharmacy tasks. *CoRR*, abs/2305.19979, 2023.
 - [6] Sameh K. Mohamed, Aayah Nounu, and Vít Nováček. Biological applications of knowledge graph embedding models. *Briefings Bioinform.*, 22(2):1679–1693, 2021.
 - [7] Brian Walsh, Sameh K. Mohamed, and Vít Nováček. Biokg: A knowledge graph for relational learning on biological data. In *CIKM*, pages 3173–3180. ACM, 2020.
 - [8] Daniel S. Himmelstein, Antoine Lizee, Christine Hessler, Leo Brueggeman, Sabrina L. Chen, Dexter Hadley, Ari Green, Pouya Khankhanian, and Sergio E. Baranzini. Systematic integration of biomedical knowledge prioritizes drugs for repurposing. *bioRxiv*, 2017. doi: 10.1101/087619. URL <https://www.biorxiv.org/content/early/2017/08/31/087619>.
 - [9] Qingyu Guo, Fuzhen Zhuang, Chuan Qin, Hengshu Zhu, Xing Xie, Hui Xiong, and Qing He. A survey on knowledge graph-based recommender systems. *IEEE Trans. Knowl. Data Eng.*, 34(8):3549–3568, 2022.
 - [10] George A. Miller. WORDNET: a lexical database for english. In *HLT*. Morgan Kaufmann, 1992.
 - [11] Natalya Fridman Noy, Yuqing Gao, Anshu Jain, Anant Narayanan, Alan Patterson, and Jamie Taylor. Industry-scale knowledge graphs: lessons and challenges. *Commun. ACM*, 62(8): 36–43, 2019.
 - [12] Maximilian Nickel, Kevin Murphy, Volker Tresp, and Evgeniy Gabrilovich. A review of relational machine learning for knowledge graphs. *Proc. IEEE*, 104(1):11–33, 2016.
 - [13] Chengjin Xu, Yung-Yu Chen, Mojtaba Nayyeri, and Jens Lehmann. Temporal knowledge graph completion using a linear temporal regularizer and multivector embeddings. In *NAACL-HLT*, pages 2569–2578. Association for Computational Linguistics, 2021.
 - [14] Elizabeth Boschee, Jennifer Lautenschlager, Sean O’Brien, Steve Shellman, James Starz, and Michael Ward. Icews coded event data. *Harvard Dataverse*, 12, 2015.
 - [15] Alberto García-Durán, Sebastijan Dumancic, and Mathias Niepert. Learning sequence encoders for temporal knowledge graph completion. In *EMNLP*, pages 4816–4821. Association for Computational Linguistics, 2018.
 - [16] Thomas Rebele, Fabian M. Suchanek, Johannes Hoffart, Joanna Biega, Erdal Kuzey, and Gerhard Weikum. YAGO: A multilingual knowledge base from wikipedia, wordnet, and geonames. In *ISWC (2)*, volume 9982 of *Lecture Notes in Computer Science*, pages 177–185, 2016.
 - [17] Antoine Bordes, Nicolas Usunier, Alberto García-Durán, Jason Weston, and Oksana Yakhnenko. Translating embeddings for modeling multi-relational data. In *NIPS*, pages 2787–2795, 2013.
 - [18] Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. Convolutional 2d knowledge graph embeddings. In *AAAI*, pages 1811–1818. AAAI Press, 2018.
 - [19] Michael Färber, Frederic Bartscherer, Carsten Menne, and Achim Rettinger. Linked data quality of dbpedia, freebase, opencyc, wikidata, and YAGO. *Semantic Web*, 9(1):77–129, 2018.
 - [20] Marie Destandau and Jean-Daniel Fekete. The missing path: Analysing incompleteness in knowledge graphs. *Inf. Vis.*, 20(1):66–82, 2021.

- [21] Daniel Ruffinelli, Samuel Broscheit, and Rainer Gemulla. You CAN teach an old dog new tricks! on training knowledge graph embeddings. In ICLR. OpenReview.net, 2020.
- [22] Rudolf Kadlec, Ondrej Bajgar, and Jan Kleindienst. Knowledge base completion: Baselines strike back. In Rep4NLP@ACL, pages 69–74. Association for Computational Linguistics, 2017.
- [23] Prachi Jain, Sushant Rathi, Mausam, and Soumen Chakrabarti. Knowledge base completion: Baseline strikes back (again). CoRR, abs/2005.00804, 2020.
- [24] Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. Embedding entities and relations for learning and inference in knowledge bases. In ICLR (Poster), 2015.
- [25] F. L. Hitchcock. The expression of a tensor or a polyadic as a sum of products. J. Math. Phys., 6 (1):164–189, 1927.
- [26] Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. A three-way model for collective learning on multi-relational data. In ICML, pages 809–816. Omnipress, 2011.
- [27] Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. Complex embeddings for simple link prediction. In ICML, volume 48 of JMLR Workshop and Conference Proceedings, pages 2071–2080. JMLR.org, 2016.
- [28] Nico Vervliet, Otto Debals, and Lieven De Lathauwer. Tensorlab 3.0 - numerical optimization strategies for large-scale constrained and coupled matrix/tensor factorization. In ACSSC, pages 1733–1738. IEEE, 2016.
- [29] Jean Kossaifi, Yannis Panagakis, Anima Anandkumar, and Maja Pantic. Tensorly: Tensor learning in python. J. Mach. Learn. Res., 20:26:1–26:6, 2019.
- [30] Timothée Lacroix, Nicolas Usunier, and Guillaume Obozinski. Canonical tensor decomposition for knowledge base completion. In ICML, volume 80 of Proceedings of Machine Learning Research, pages 2869–2878. PMLR, 2018.
- [31] Antoine Bordes, Jason Weston, Ronan Collobert, and Yoshua Bengio. Learning structured embeddings of knowledge bases. In AAAI. AAAI Press, 2011.
- [32] Xavier Glorot, Antoine Bordes, Jason Weston, and Yoshua Bengio. A semantic matching energy function for learning with multi-relational data. In ICLR (Workshop Poster), 2013.
- [33] Rodolphe Jenatton, Nicolas Le Roux, Antoine Bordes, and Guillaume Obozinski. A latent factor model for highly multi-relational data. In NIPS, pages 3176–3184, 2012.
- [34] Seyed Mehran Kazemi, Rishab Goel, Kshitij Jain, Ivan Kobyzev, Akshay Sethi, Peter Forsyth, and Pascal Poupard. Representation learning for dynamic graphs: A survey. J. Mach. Learn. Res., 21:70:1–70:73, 2020.
- [35] David Liben-Nowell and Jon M. Kleinberg. The link-prediction problem for social networks. J. Assoc. Inf. Sci. Technol., 58(7):1019–1031, 2007.
- [36] Umang Sharan and Jennifer Neville. Temporal-relational classifiers for prediction in evolving domains. In ICDM, pages 540–549. IEEE Computer Society, 2008.
- [37] Nahla Mohamed Ahmed Ibrahim and Ling Chen. Link prediction in dynamic social networks by integrating different types of information. Appl. Intell., 42(4):738–750, 2015.
- [38] Nahla Mohamed Ahmed Ibrahim and Ling Chen. An efficient algorithm for link prediction in temporal uncertain social networks. Inf. Sci., 331:120–136, 2016.
- [39] Nahla Mohamed Ahmed Ibrahim, Ling Chen, Yulong Wang, Bin Li, Yun Li, and Wei Liu. Sampling-based algorithm for link prediction in temporal networks. Inf. Sci., 374:1–14, 2016.
- [40] Lin Yao, Luning Wang, Lv Pan, and Kai Yao. Link prediction based on common-neighbors for dynamic social network. In ANT/SEIT, volume 83 of Procedia Computer Science, pages 82–89. Elsevier, 2016.

- [41] Ismail Günes, Sule Gündüz Ögüdücü, and Zehra Çataltepe. Link prediction using time series of neighborhood-based node similarity scores. *Data Min. Knowl. Discov.*, 30(1):147–180, 2016.
- [42] Behnaz Moradabadi and Mohammad Reza Meybodi. A novel time series link prediction method: Learning automata approach. *Physica A: Statistical Mechanics and its Applications*, 482:422–432, 2017. ISSN 0378-4371. doi: <https://doi.org/10.1016/j.physa.2017.04.019>. URL <https://www.sciencedirect.com/science/article/pii/S0378437117303199>.
- [43] Purnamrita Sarkar, Sajid M. Siddiqi, and Geoffrey J. Gordon. A latent space approach to dynamic embedding of co-occurrence data. In *AISTATS*, volume 2 of *JMLR Proceedings*, pages 420–427. JMLR.org, 2007.
- [44] Zhen Han, Yuyi Wang, Yunpu Ma, Stephan Günnemann, and Volker Tresp. The graph hawkes network for reasoning on temporal knowledge graphs. *arXiv preprint arXiv:2003.13432*, 2020.
- [45] Zixuan Li, Xiaolong Jin, Wei Li, Saiping Guan, Jiafeng Guo, Huawei Shen, Yuanzhuo Wang, and Xueqi Cheng. Temporal knowledge graph reasoning based on evolutionary representation learning. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR ’21*, page 408–417, New York, NY, USA, 2021. Association for Computing Machinery. ISBN 9781450380379. doi: 10.1145/3404835.3462963. URL <https://doi.org/10.1145/3404835.3462963>.
- [46] Tingsong Jiang, Tianyu Liu, Tao Ge, Lei Sha, Sujian Li, Baobao Chang, and Zhifang Sui. Encoding temporal information for time-aware link prediction. In *EMNLP*, pages 2350–2354. The Association for Computational Linguistics, 2016.
- [47] Cristóbal Esteban, Volker Tresp, Yinchong Yang, Stephan Baier, and Denis Krompass. Predicting the co-evolution of event and knowledge graphs. In *FUSION*, pages 98–105. IEEE, 2016.
- [48] Shib Sankar Dasgupta, Swayambhu Nath Ray, and Partha P. Talukdar. Hyte: Hyperplane-based temporally aware knowledge graph embedding. In *EMNLP*, pages 2001–2011. Association for Computational Linguistics, 2018.
- [49] Rishab Goel, Seyed Mehran Kazemi, Marcus A. Brubaker, and Pascal Poupart. Diachronic embedding for temporal knowledge graph completion. In *AAAI*, pages 3988–3995. AAAI Press, 2020.
- [50] Chengjin Xu, Mojtaba Nayyeri, Fouad Alkhoury, Jens Lehmann, and Hamed Shariat Yazdi. Temporal knowledge graph embedding model based on additive time series decomposition. *CoRR*, abs/1911.07893, 2019.
- [51] Michael Sejr Schlichtkrull, Thomas N. Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. Modeling relational data with graph convolutional networks. In *ESWC*, volume 10843 of *Lecture Notes in Computer Science*, pages 593–607. Springer, 2018.
- [52] Xiao Wang, Houye Ji, Chuan Shi, Bai Wang, Yanfang Ye, Peng Cui, and Philip S Yu. Heterogeneous graph attention network. In *The World Wide Web Conference, WWW ’19*, page 2022–2032, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450366748. doi: 10.1145/3308558.3313562. URL <https://doi.org/10.1145/3308558.3313562>.
- [53] Jiapeng Wu, Meng Cao, Jackie Chi Kit Cheung, and William L. Hamilton. Temp: Temporal message passing for temporal knowledge graph completion. In *EMNLP (1)*, pages 5730–5746. Association for Computational Linguistics, 2020.
- [54] Yunpu Ma, Volker Tresp, and Erik A. Daxberger. Embedding models for episodic knowledge graphs. *J. Web Semant.*, 59, 2019.
- [55] Chengjin Xu, Mojtaba Nayyeri, Fouad Alkhoury, Hamed Shariat Yazdi, and Jens Lehmann. Tero: A time-aware knowledge graph embedding via temporal rotation. In *COLING*, pages 1583–1593. International Committee on Computational Linguistics, 2020.

- [56] Johannes Messner, Ralph Abboud, and İsmail İlkan Ceylan. Temporal knowledge graph completion using box embeddings. In AAAI, pages 7779–7787. AAAI Press, 2022.
- [57] Seyed Mehran Kazemi and David Poole. Simple embedding for link prediction in knowledge graphs. In NeurIPS, pages 4289–4300, 2018.
- [58] Zhiqing Sun, Zhi-Hong Deng, Jian-Yun Nie, and Jian Tang. Rotate: Knowledge graph embedding by relational rotation in complex space. In ICLR (Poster). OpenReview.net, 2019.
- [59] Yihong Chen, Pushkar Mishra, Luca Franceschi, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. Refactor gnns: Revisiting factorisation-based models from a message-passing perspective. In NeurIPS, 2022.

A Appendix

A.1 Description of Static Neural Link Predictors

Scoring Functions Neural link prediction models can be characterised by their scoring function ϕ_θ . For example, in TransE [17], the score of a triple $\langle s, p, o \rangle$ is given by $\phi_\theta^{\text{TransE}}(s, p, o) = -\|s + p - o\|_2$, where $s, p, o \in \mathbb{R}^d$ denote the embedding representations of s, p , and o , respectively. In DistMult [24], the scoring function is defined as $\phi_\theta^{\text{DistMult}}(s, p, o) = \langle s, p, o \rangle = \sum_{z=1}^d s_z p_z o_z$, where $\langle \cdot, \cdot, \cdot \rangle$ denotes the tri-linear dot product. Canonical Tensor Decomposition [CP, 25] is similar to DistMult, with the difference that each entity $e \in \mathcal{E}$ has two representations, $e_{\text{head}} \in \mathbb{R}^d$ and $e_{\text{tail}} \in \mathbb{R}^d$, depending on whether it is being used as a head (subject) or tail (object): $\phi_\theta^{\text{CP}}(s, p, o) = \langle s_{\text{head}}, p, o_{\text{tail}} \rangle$. In RESCAL [26], the scoring function is a bilinear model given by $\phi_\theta^{\text{RESCAL}}(s, p, o) = s^\top P o$, where $s, o \in \mathbb{R}^d$ is the embedding representation of s and o , and $P \in \mathbb{R}^{d \times d}$ is the representation of p . DistMult is equivalent to RESCAL if P is constrained to be diagonal. Another variation of this model is ComplEx [27], where the embedding representations of s, p , and o are complex vectors – i.e. $s, p, o \in \mathbb{C}^d$ – and the scoring function is given by $\phi_\theta^{\text{ComplEx}}(s, p, o) = \Re(\langle s, p, \bar{o} \rangle)$, where $\Re(\mathbf{x})$ represents the real part of \mathbf{x} , and $\bar{\mathbf{x}}$ denotes the complex conjugate of \mathbf{x} .

Training Objectives Another dimension for characterising neural link predictors is their training objective. Early neural link prediction models such as RESCAL and CP were trained to minimise the reconstruction error of the whole adjacency tensor [26, 28, 29]. To scale to larger Knowledge Graphs, subsequent approaches such as Bordes et al. [17] and Yang et al. [24] simplified the training objective by using negative sampling: for each training triple, a corruption process generates a batch of negative examples by corrupting the subject and object of the triple, and the model is trained by increasing the score of the training triple while decreasing the score of its corruptions. This approach was later extended by Dettmers et al. [18] where, given a subject s and a predicate p , the task of predicting the correct objects is cast as a $|\mathcal{E}|$ -dimensional multi-label classification task, where each label corresponds to a distinct object and multiple labels can be assigned to the (s, p) pair; this training objective is referred to as KvsAll by Ruffinelli et al. [21]. Another extension was proposed by Lacroix et al. [30] where, given a subject s and a predicate p , the task of predicting the correct object o in the training triple is cast as a $|\mathcal{E}|$ -dimensional multi-class classification task, where each class corresponds to a distinct object and only one class can be assigned to the (s, p) pair; this is referred to as 1vsAll by Ruffinelli et al. [21].

Regularisers As noted by Bordes et al. [17], imposing regularisation terms on the learned entity and relation representations prevents the training process from trivially optimising the training objective by increasing the embedding norms. Early works such as Bordes et al. [17, 31], Glorot et al. [32], Jenatton et al. [33] proposed constraining the embedding norms. More recently, Yang et al. [24], Trouillon et al. [27] proposed adding a L_2 regularisation term on entity and relation representations to the training objective. Lastly, Lacroix et al. [30] observed systematic improvements by replacing the L_2 norm with a nuclear tensor 3-norm.

A.2 Description of temporal knowledge graph completion models

The subject of temporal link prediction has been studied in a wide range of approaches [34]. One approach is to create a static KG by combining links from different time points without considering timestamps [35]. Afterwards, static embeddings are learned for individual entities. Some attempts have been made to enhance this method by assigning higher importance to more recent links [36–39]. In contrast to these methods, [40] involves learning embeddings for each KG snapshot and then combining them using weighted averaging. Various techniques have been proposed for determining the weights of the embedding aggregation, including approaches based on ARIMA [41] or reinforcement learning [42].

Some additional studies employ sequence models for TKGs. Sarkar et al. [43] apply a Kalman filter to learn dynamic node embeddings. García-Durán et al. [15] extends DistMult and TransE to TKG using recurrent neural nets (RNN) for temporal data. For each relation, a temporal embedding has been learned by feeding a string version for timestamp (e.g. a date in US format) and the static relation embedding to an LSTM. This method learns dynamic embedding for relations but not for entities. Furthermore, [44] employ a temporal point process parameterised by a deep neural architecture.

Model	Number of Parameters
DE-Simple	$2d((3\gamma + (1 - \gamma)) E + R)$
TComplEx	$2d(E + T + 2 R)$
BoxTE	$d(2 E + k T + 2 R) + k R $
TeLM	$4d(E + T + 4 R)$
ChronoR	$2d(E + T + 4 R)$
TNTComplEx	$2d(E + T + 4 R)$

Table 4: Model parameter count for TNTComplEx and other competing models. For DE-Simple, γ denotes the share of temporal embedding features. For BoxTE, k denotes the dimension for the scalar vector.

One of the earliest works that used representation learning techniques for reasoning over TKGs [45] introduced an approach involving both embedding and rule mining methods for TKG reasoning. Another related model, t-TransE [46], indirectly learns time-based embeddings by capturing the order of time-sensitive relations, such as *wasBornIn* followed by *diedIn*. In a similar vein, Esteban et al. [47] enforce temporal order constraints on their data by augmenting their quadruple $(s, p, o, \tau : \text{Bool})$, where Bool indicates if the fact vanishes or continues after time t . However, the model’s evaluation is limited to medical and sensory data.

Some approaches have attempted to apply diachronic word embeddings to the TKG problem, motivated by their success. [15, 48]. Diachronic methods map every (node, timestamp) or (relation, timestamp) pair to a hidden representation. Goel et al. [49] propose learning dynamic embeddings by masking a fraction of the embedding weights with an activation function of frequencies and Xu et al. [50] embed the vectors as a direct function of time.

Other methods like Schlichtkrull et al. [51], Wang et al. [52], Wu et al. [53] leverage message passing graph neural networks (MPNNs) to learn structure-based entity representations. For instance, [53] learns entity representation at every timestamp and then aggregates representations across all timestamps using a gated recurrent unit or self-attention encoder. Similarly to other models, the final entity representations can subsequently be used with a static KGE model such as ComplEx.

Finally, most of the proposed methods do not evolve the embedding of entities over time [54, 1, 55, 2, 13, 56]. Instead, they learn the temporal behaviour by using a representation for time. For instance, Lacroix et al. [1] perform tensor decomposition based on the time representation, while Sadeghian et al. [2] learns a d -dimensional rotation transformation parametrized by relation and time, such that after each fact’s head entity representation is transformed using the rotation, it falls near its corresponding tail entity.

In this work, we focus on the latter methods as they achieve state-of-the-art results on most of the benchmark datasets [2, 13]. Specifically, we systematically analyse a wide array of temporal regularisers to understand their impact on both performance and learning temporal behaviour.

A.3 Details related to experimental evaluation and results

Model parameter counts table 4 reports the model parameter counts for TNTComplEx, ChronoR and other competing models. For all the models, d is the embedding size, $|E|$, $|T|$, and $|R|$ are the number of entities, timestamps and relations in the dataset, respectively. Note that ChronoR and TNTComplEx have the same number of parameters, while TeLM, which uses multi-vector embeddings, has exactly double.

Results of previous related works. The results of previous related works are the best numbers reported in their respective paper except for TNTComplEx, ChronoR and TeLM. The original results for TNTComplEx were reported on the validation set, we use the code and hyper-parameters from the official repository to re-run the model and report test set values. The same approach to report TNTComplEx results was adopted by [2]. The original implementation of ChronoR (2021) is not available, we re-implement the solution in our framework, reporting the result according to the original configuration of hyperparameters stated in their paper. TeLM is trained using almost the same number of parameters as TNTComplEx and ChronoR. As shown in table 4, fixed an embedding

Dataset	Embedding Dimension	λ_1	λ_2	Temporal Regulariser
ICEWS14	2000	0.001	0.01	N4
ICEWS05-15	2000	0.001	1	N5
YAGO15K	2000	0.0001	0.0001	N5

Table 5: Best hyper-parameter configuration of TNTComplEx on ICEWS14, ICEWS5-15, and YAGO15K – λ_1 and λ_2 denote the N3 and the temporal regularisation weights, respectively.

dimension d , TeLM has double the number of parameters of TNTComplEx, hence we report the results of TeLM setting $d = 1000$. In its original paper [13], the results refer to $d = 2000$.

Best configuration of hyperparameters We tune all the hyper-parameters using a grid search using the validation set provided with each of the datasets. We tune λ_1 and λ_2 from $\{10^i \mid -4 \leq i \leq 1\}$, the p value for Np and Lp norms from $\{i \mid 1 \leq i \leq 5\}$, the embedding dimensions for tensors and recurrent architecture from $\{5, 25, 50, 100, 500, 2000\}$.

The training was done using mini-batch stochastic gradient descent with Adam and a learning rate of 0.1 with a batch size of 1000 quadruples.

We report the best configuration of hyperparameters for TNTComplEx in Table 5.