Temporal Knowledge Graph Completion using a Linear Temporal Regularizer and Multivector Embeddings

Chengjin Xu
University of Bonn / Germany
xuc@iai.uni-bonn.de

Yung-Yu Chen
University of Bonn / Germany
s6ynchen@uni-bonn.de

Mojtaba Nayyeri
University of Bonn / Germany
nayyeri@iai.uni-bonn.de

Jens Lehmann
University of Bonn / Germany
Fraunhofer IAIS/ Germany
jens.lehmann@iais.fraunhofer.de

Abstract

Representation learning approaches for knowledge graphs have been mostly designed for static data. However, many knowledge graphs involve evolving data, e.g., the fact (The President of the United States is Barack Obama) is valid only from 2009 to 2017. This introduces important challenges for knowledge representation learning since the knowledge graphs change over time. In this paper, we present a novel time-aware knowledge graph embebdding approach, **TeLM**, which performs 4th-order tensor factorization of a **Te**mporal knowledge graph using a **L**inear temporal regularizer and **M**ultivector embeddings. Moreover, we investigate the effect of the temporal dataset's time granularity on temporal knowledge graph completion. Experimental results demonstrate that our proposed models trained with the linear temporal regularizer achieve the state-of-the-art performances on link prediction over four well-established temporal knowledge graph completion benchmarks.

1 Introduction

Numerous large-scale knowledge graphs (KGs) including DBpedia (Auer et al., 2007), FreeBase (Bollacker et al., 2008) and WordNet (Miller, 1995) have been established in recent years. Such KGs abstract knowledge from the real world into a complex network graph consisting of billions of triples. Each triple is denoted as (s, r, o), where s is the subject entity, o is the object entity, and r is the relation between the entities.

Knowledge graph completion is one of the main challenges in the KG field since most KGs are incomplete. To tackle this problem, knowledge graph embedding (KGE) approaches embed entities and relations into a low-dimensional embedding space and measure the plausibility of triples by inputting embeddings of the entities and their relation to a score function (Wang et al., 2017). For instance, ComplEx (Trouillon et al., 2016) has been proven to be a highly effective KGE model, where entities and relations are represented as complex embeddings, and the score of a triple (s, r, o) is computed with the asymmetric Hermitian dot product.

Some KGs involve temporal facts, e.g., the triple (*Barack Obama*, *presidentOf*, *USA*) is only valid in a specific time period [*2009*, *2017*]. Temporal KGs like Wikidata (Erxleben et al., 2014), YAGO3 (Mahdisoltani et al., 2013) and ICEWS (Lautenschlager et al., 2015) incorporate time information into triples. Triples attached with time information are represented as quadruples, shaped like (s, r, o, T), where T denotes the timestamp. Traditional KGE models disregard time information, leading to an ineffectiveness of performing link prediction on TKGs involving temporary relations, e.g., (*?*, *presidentOf*, *USA*, *2010*). Recent researches show that the temporal knowledge graph embedding (TKGE) models, which encode time information in their embeddings, have better performances on link prediction over TKGs than traditional KGE models (Dasgupta et al., 2018; García-Durán et al., 2018; Xu et al., 2019; Goel et al., 2020; Lacroix et al., 2020).

In this paper, we present a novel temporal KG embedding approach TeLM. We move beyond the complex-valued representations and introduce more expressive multivector embeddings from 2-grade geometric algebras to model entities, relations, and timestamps for TKGE. At a high level, our approach performs 4th-order tensor factorization of a temporal KG, using the asymmetric geometric product. The geometric product provides a

Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 2569–2578
June 6–11, 2021.

greater extent of expressiveness compared to the complex Hermitian operator.

Specially, each relation is represented as a pair of dual multivector embeddings used to handle the beginning and the end of the relation. In this way, TeLM can adapt well to datasets where time annotations are represented in the various forms: time points, begin or end time, time intervals.

Moreover, we develop a new linear temporal regularization function for time representation learning which introduces a bias component in the temporal smoothing function and empirically study the effect of the time granularity for a TKG dataset on the performance of our models.

Experimental results on four well-established TKG datasets show that our approach outperforms the state-of-the-art TKGE models, and the linear temporal regularization function improves the performance of our model compared to three common temporal regularization functions.

2 Related Work

Tensor decomposition-based KGE approaches have led to good results in static KG completion. Such approaches (Yang et al., 2014; Trouillon et al., 2016; Kazemi and Poole, 2018; Zhang et al., 2019; Xu et al., 2020b) model a static KG as a low-dimensional 3rd-order tensor and consider knowledge graph completion as a tensor decomposition problem. A typical tensor decomposition model ComplEx (Trouillon et al., 2016) has been proven to be fully expressive with complex embeddings. Apart from tensor decomposition approaches, distance-based KGE models are also commonly used for KG completion. However, distance-based KGE models like TransE (Bordes et al., 2013) and its variants (Wang et al., 2014; Lin et al., 2015; Nayyeri et al., 2019, 2020) have been proven to have limitations in modeling various relation patterns which does not lead to the state-of-the-art results on the current benchmarks.

The above KGE approaches achieve satisfactory results on link prediction over static KGs. Recent research on TKG completion shows that the inclusion of time information can improve the performances of KGE models on TKGs. TTransE (Leblay and Chekol, 2018), HyTE (Dasgupta et al., 2018), ATiSE and TeRo (Xu et al., 2019, 2020a) propose scoring functions which incorporate time representations into a distance-based score function in different ways. Furthermore, RTGE (Xu et al., 2020c) introduces the concept of temporal smoothness to optimize and learn the hyperplanes of adjacent time intervals jointly on the basis of HyTE. García-Durán et al. (2018) utilize recurrent neural networks to learn time-aware representations of relations and use standard scoring functions from the existing KG embedding model, e.g. TransE and DistMult. DE-SimplE (Goel et al., 2020) uses diachronic entity embeddings to represent entities at different time steps and exploit the same score function as SimplE to score the plausibility of a quadruple. TIMEPLEX (Jain et al., 2020) and TComplEx (Lacroix et al., 2020) extend the time-agnostic ComplEx model in different ways. Among them, TComplEx performs a 4th-order tensor decomposition of a TKG using the quadranomial Hermitian product which involves the embedding of timestamp T. Similarly to RTGE, TComplEx also uses the temporal smoothness to improve its performance. Thanks to the strong expressiveness provide by the complex embeddings and the 4th-order tensor decomposition, TComplEx achieves state-of-the-art results on TKG completion.

3 Geometric Algebras

In this section, we provides a brief introduction to the 2-grade Geometric Algebra $\mathbb{G}^2$. The contents are sufficient to understand the rest of the work.

Members of $\mathbb{G}^2$ are called 2-grade multivectors. The multivector space $\mathbb{G}^2$ is build with vectors from the vector space $\mathbb{R}^2$. Let $\{e_1, e_2\}$ be an orthonormal basis of $\mathbb{R}^2$. The multivector space $\mathbb{G}^2$ is based on two rules: $e_1e_1 = e_2e_2 = 1$ and $e_1e_2 = -e_2e_1$. The multivector space $\mathbb{G}^2$ is 4-dimensional with basis:

$$\begin{gathered} 1 \text{ spans 0-vectors, scalars,} \\ \{e_1, e_2\} \text{ spans 1-vectors, vectors, and} \\ \{e_1e_2\} \text{ spans 2-vectors, bivectors.} \end{gathered}$$

A 2-grade multivector $M \in \mathbb{G}^2$ can be written as $M = a_0 + a_1e_1 + a_2e_2 + a_{12}e_1e_2$. Noteworthly, the unit bivectors from $\mathbb{G}^2$ has similar algebraic properties as the imaginary unit $\mathbf{i}$, i.e.,

$$(e_1e_2)^2 = -e_1e_1e_2e_2 = -1 = \mathbf{i}^2.$$

Thus, the complex numbers $C \in \mathbb{C}$ can be embedded into a subalgebra of $\mathbb{G}^2$ which are formed with scalars and bivectors. In other words, a 2-grade multivector $M = a_0 + a_{12}e_1e_2$ consisting of a

scalar plus a bivector is isomorphic to a complex number $C = a_0 + a_{12}\mathbf{i}$.

The norm of a multivector $M \in \mathbb{G}^2$ is equal to the root of the square sum of real values of its all elements. Taking a 2-grade multivector as an example, its norm is defined as: $||M|| = \sqrt{a_0^2 + a_1^2 + a_2^2 + a_{12}^2}$.

Geometric algebra also introduces a new product **geometric product** denoted as $\times_n$ where n is the grade of multivectors, as well as three multivector involutions, **space inversion**, **reversion** and **Clifford conjugation**.

The **geometric product** of two 2-grade multivectors comprises of multiplications between scalars, vectors and bivectors. The product of two 2-grade multivectors $M_a = a_0 + a_1e_1 + a_2e_2 + a_{12}e_1e_2$ and $M_b = b_0 + b_1e_1 + b_2e_2 + b_{12}e_1e_2$ from $\mathbb{G}^2$ is equal to

$$\begin{aligned} M_a \times_2 M_b &= a_0b_0 + a_1b_1 + a_2b_2 - a_{12}b_{12} \\ &+ (a_0b_1 + a_1b_0 - a_2b_{12} + a_{12}b_2)e_1 + \\ &(a_0b_2 + a_1b_{12} + a_2b_0 - a_{12}b_1)e_2 \\ &+ (a_0b_{12} + a_1b_2 - a_2b_1 + a_{12}b_0)e_1e_2, \end{aligned} \tag{1}$$

The **Clifford conjugation** of a 2-grade multivector M is a subsequent composition of **space inversion** M^* and **reversion** $M^\dagger$ as $\overline{M} = M^{\dagger *}$, where **space inversion** M^* is obtained by changing e_i to $-e_i$ and **reversion** is obtained by reversing the order of all products i.e. changing e_1e_2 to e_2e_1. Thus, the **Clifford conjugation** of an 2-grade multivector $M_a = a_0 + a_1e_1 + a_2e_2 + a_{12}e_1e_2$ is computed as $\overline{M_a} = a_0 - a_1e_1 - a_2e_2 - a_{12}e_1e_2$. Note that the product of a multivector M_a and its conjugation $\overline{M}_2$ is a scalar, i.e., given a 2-grade multivector $M_a = a_0 + a_1e_1 + a_2e_2 + a_{12}e_1e_2$, we have

$$M_a \times_2 \overline{M_a} = a_0^2 - a_1^2 - a_2^2 + a_{12}^2, \tag{2}$$

producing a real number.

4 Our Method

4.1 TeLM Model

Let $\mathcal{E}$ denote the set of entities, $\mathcal{R}$ denote the set of relations. A TKG denoted as Ω is a collection of numerous quadruples shaped like (s, r, o, T) where $s, o \in \mathcal{E}$, $r \in \mathcal{R}$ and T denotes the timestamp. The timestamp T can be represented as various forms, e.g., a time interval $[t_b, t_e]$, a begin time $[t_s, -]$ or an end time $[-, t_e]$ and a time point t. A time point t can be denoted as a special time interval $[t_b, t_e]$ where $t = t_b = t_e$. We extend the relation set $\mathcal{R}$ of a TKG to a pair of dual relation sets, $\mathcal{R}_b$ and $\mathcal{R}_e$. A relation $r_b \in \mathcal{R}_b$ is used to handle the begin of relation r, meanwhile a relation $r_e \in \mathcal{R}_e$ is used to handle the end of relation r. By doing this, we score a fact (s, r, o, $[t_b, t_e]$) as the mean value of scores of two quadruples, (s, r_b, o, t_b) and (s, r_e, o, t_e) which represent the begin and the end of this fact respectively, i.e., $f(s, r, o, [t_b, t_e]) = \frac{1}{2}(f(s, r_b, o, t_b) + f(s, r_e, o, t_e))$. For a fact missing the begin time or the end time, e.g., (s, r, o, $[t_b, -]$) or (s, r, o, $[-, t_e]$), the score of this fact is equal to the score of the quadruple involving the known time, i.e., $f(s, r, o, [t_b, -]) = f(s, r_b, o, t_b)$, $f(s, r, o, [-, t_e]) = f(s, r_e, o, t_e)$. We construct a set of time steps $\mathcal{T}$ for a TKG. For any time t appearing in the TKG, we can find a time step $\tau \in \mathcal{T}$ to represent t. The time set $\mathcal{T}$ changes with time granularity of the TKG.

Our approach TeLM embeds a TKG in a multiple-dimensional 2-grade multivector space $\mathbb{G}^{2\times k}$ where k is the dimension of embeddings, and score a quadruple with an element-wise geometric product. TeLM embeds each entity, relation and time step as a k-dimensional 2-grade multivector embedding $\mathbf{M}$ where each component is a multivector, i.e., $\mathbf{M} = [M_1, \ldots, M_k]$, $i = 1, \ldots, k, M_i \in \mathbb{G}^2$. We can define the score function of TeLM as

$$f(s, r, o, t) = \langle \mathbf{Sc}(\mathbf{M}_s \otimes_2 \mathbf{M}_{r\tau} \otimes_2 \overline{\mathbf{M}_o}), \mathbf{1} \rangle, \tag{3}$$

where τ is the time step corresponding to time t, $\mathbf{M}_{r\tau} = \mathbf{M}_r \otimes_2 \mathbf{M}_\tau$, $\mathbf{M}_s$, $\mathbf{M}_r$, $\mathbf{M}_o$ and $\mathbf{M}_\tau$ denote the k-dimensional multivector embeddings of s, r, o and τ respectively. $\otimes_2$ denotes the element-wise geometric product between 2-grade multivector embeddings, e.g., $\mathbf{M}_r \otimes_2 \mathbf{M}_\tau = [M_{r_1} \times_2 M_{\tau_1}, \cdots, M_{r_k} \times_2 M_{\tau_k}]$. $\mathbf{Sc}(\cdot)$ denotes the real-valued vector of the scalar component of a multivector embedding, $\mathbf{1}$ denotes a $k \times 1$ vector having all k elements equal to one, $\overline{\mathbf{M}}$ denotes the element-wise conjugation of multivectors i.e. $\overline{\mathbf{M}} = [\overline{M}_1, \ldots, \overline{M}_k]$. and $\langle a, b \rangle := \sum_k a_k b_k$ is the dot product.

In our approach, the total number of parameters increases linearly with embedding dimension k, i.e., the space complexity of a TeLM model is $\mathcal{O}(k)$. Since the score is computed with an asymmetric quadranomial geometric product between k-dimensional multivector embeddings, the time complexity is also equal to $\mathcal{O}(k)$, which are the same as some common KGE models, e.g., TransE and DistMult.

4.2 Loss Function

Using full multiclass log-softmax loss function and N3 regularization has been proven to be helpful in boosting the performances of tensor decomposition-based (T)KGE models (Lacroix et al., 2018; Xu et al., 2020b; Lacroix et al., 2020; Jain et al., 2020). In this work, we follow such setting for TeLM and utilize the reciprocal learning for simplifying the training process.

For each relation r, we create an inverse relation r^{-1} and create a quadruple (o, r^{-1}, s, t) for each training quadruple (s, r, o, t). At the evaluation phase, queries of the form $(?, r, o, t)$ are answered as $(o, r^{-1}, ?, t)$. By doing this, the multiclass log-loss of a training quadruple $\omega = (s, r, o, t)$ can be defined as follows,

$$\begin{aligned}\mathcal{L}_\omega = &-\log\left(\frac{\exp(f(s,r,o,t))}{\sum_{s'\in\mathcal{E}}\exp(f(s',r,o,t))}\right)\\ &-\log\left(\frac{\exp(f(o,r^{-1},s,t))}{\sum_{o'\in\mathcal{E}}\exp(f(o',r^{-1},s,t))}\right)\\ &+\lambda_\omega\sum_{i=1}^{k}(||M_{s_i}||_3^3+||M_{r\tau_i}||_3^3+||M_{o_i}||_3^3),\end{aligned} \tag{4}$$

where λ_ω denotes the N3 regularization weight.

4.3 Temporal Regularization

A common approach to leverage the temporal aspect of temporal graphs is to use time as a regularizer to impose a smoothness constraint on time embeddings. RTGE (Xu et al., 2020c) and TComplEx (Lacroix et al., 2020) introduce the temporal smoothness between hyperplanes and embeddings of adjacent time steps, respectively, based on the assumption that the neighboring time steps should have close representations. The smoothing temporal regularizer is defined as,

$$\mathcal{L}_\mathcal{T} = \sum_{i=1}^{n_\tau-1}||\mathbf{M}_{\tau_{i+1}}-\mathbf{M}_{\tau_i}||_p^p, \tag{5}$$

where n_τ is the number of time steps and $p = 3$ in this work since we use N3 regularization.

Apart from the basic temporal smoothness, various temporal regularization methods are used for learning temporal embeddings. Singer et al. (2019) add a rotation projection to align the neighboring temporal embeddings. The loss of such projective temporal regularization can be defined as,

$$\mathcal{L}_\mathcal{T} = \sum_{i=1}^{n_\tau-1}||\mathbf{M}_{\tau_{i+1}}-\mathbf{M}_w\otimes_2\mathbf{M}_{\tau_i}||_p^p, \tag{6}$$

where $\mathbf{M}_w$ is the rotation embedding. Yu et al. (2016) propose an autoregressive temporal regularizer based on the assumption that the change of temporal embeddings fits an AR model. This autoregressive temporal regularizer is defined as,

$$\mathcal{L}_\mathcal{T} = \sum_{i=1}^{n_\tau-m}||\mathbf{M}_{\tau_{i+m}}-\sum_{j=0}^{m-1}\mathbf{M}_j\otimes_2\mathbf{M}_{\tau_{i+j}}||_p^p, \tag{7}$$

where $m = 3$ is the order of the AR model used in our work, and $\mathbf{M}_j$ denote the weight of the embeddings of previous time steps which are learned during the training process.

In this work, we develop a novel linear temporal regularizer by adding a bias component between the neighboring temporal embeddings, which can be defined as,

$$\mathcal{L}_\mathcal{T} = \sum_{i=1}^{n_\tau-1}||\mathbf{M}_{\tau_{i+1}}-\mathbf{M}_{\tau_i}-\mathbf{M}_b||_p^p. \tag{8}$$

where $\mathbf{M}_b$ denotes the bias embedding which are randomly initialized and then learned from the training process. This linear regularizer promotes that the difference between embeddings of two adjacent time steps is smaller than the difference between embeddings of two distant time steps, i.e., $||\mathbf{M}_{\tau_{i+m}}-\mathbf{M}_{\tau_i}|| > ||\mathbf{M}_{\tau_{i+1}}-\mathbf{M}_{\tau_i}||$ when $m \gg 1$. This formulation can be helpful for effectively clustering and ordering time embeddings $\mathbf{M}_{\tau_i}$.

The total loss $\mathcal{L}_b$ of a training batch Ω_b is the sum of the quadruple loss and the temporal regularization term, i.e.,

$$\mathcal{L}_b = \frac{1}{b}\sum_{\omega\in\Omega_b}\mathcal{L}_\omega + \lambda_\mathcal{T}\mathcal{L}_\mathcal{T}. \tag{9}$$

where $\lambda_\mathcal{T}$ denotes the coefficient of the temporal regularizer. In this work, we use the linear temporal regularizer for TeLMand compare its performance with other three temporal regularizers.

5 Experiments

5.1 Datasets

To compare our model with baselines, we used the following three datasets, namely ICEWS14, ICEWS05-15, and YAGO11k, released by (Dasgupta et al., 2018) and (García-Durán et al., 2018).

ICEWS14 and ICEWS05-15 are the two most common TKG benchmarks extracted from the large-scale event-based database, Integrated Crisis Early Warning System (ICEWS) (Lautenschlager

Dataset	#Entities	#Relations	Period(year)	#Train	#Valid	#Test
ICEWS14	6,869	230	201	72,826	8,941	8,963
ICEWS05-15	10,094	251	2005-2015	368,962	46,275	46,092
YAGO11k	10,623	10	-431-2844	16,406	2,050	2,051
Wikidata12k	12,554	24	19-2020	32,497	4,062	4,062

Table 1: Statistics of datasets.

et al., 2015). ICEWS is a repository that contains political events with specific time annotations, e.g. (*Barack Obama*, *Make a visit*, *Ukraine*, *2014-07-08*). It is noteworthy that time annotations in ICEWS are all time points. ICEWS14 contains events in 2014, and ICEWS05-15 contains events occurring between 2005-2015. These two datasets are filtered by only selecting the most frequently occurring entities in the graph.

YAGO3 (Mahdisoltani et al., 2013) and Wikidata (Erxleben et al., 2014) are two temporal KGs where time annotations are represented in various forms, i.e., time points like [*2003-01-01*, *2003-01-01*], beginning or end time like [*2003*, *##*], and time intervals like [*2003*, *2005*]. YAGO15k, Wikidata11k (García-Durán et al., 2018), YAGO11k and Wikidata12k (Dasgupta et al., 2018) are subsets of YAGO3 and Wikidata. In YAGO15k and Wikidata11k, time information is represented as either begin time or end time of each fact and some facts do not include time annotations. In this paper, we focus on performing link prediction on time-aware facts where time annotations are represented as various forms. Based on this consideration, we use YAGO11k and Wikidata12k as datasets, where all of facts involve time annotations.

The statics of datasets are listed in Table 1. All datasets can be downloaded from `https://github.com/soledad921/ATISE`.

5.2 Time granularity

In the previous work (García-Durán et al., 2018; Goel et al., 2020; Lacroix et al., 2020), the time granularity of ICEWS14 and ICEWS05-15 was set as 1 day. For YAGO11k and Wikidata12k, Dasgupta et.al (2018) and Xu et.al (2019) dropped the month and day information. They took care of the unbalance that might occur in terms of number of facts in a particular interval by clubbing neighboring years which are less frequently mentioned into the same time step and applying a minimum threshold of 300 facts per interval during construction. To illustrate, in Wikidata12k, there were time steps like [*1596-1777*], [*1791-1815*] with a large span as the facts occurring on those years were relatively less in KG. The years like 2013 being highly frequent were self-contained. This setting was used to alleviate the effect of the long-tail property of time data in YAGO11k and Wikidata12k. As shown in Figure 1, the time distribution of facts in ICEWS14 is relatively uniform, while the frequency distribution of time data in YAGO11k has a long tail.

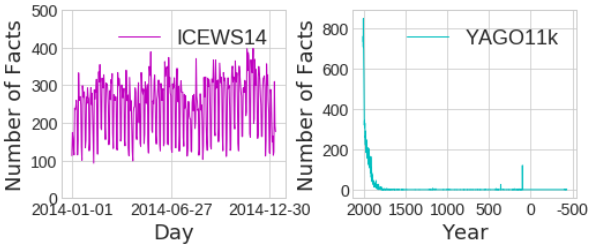

Figure 1: Time distribution of facts in TKGs.

In this work, we study the effect of time granularity on TKG completion. For ICEWS datasets, we test our model with different time units, denoted as u, in a range of {1, 2, 3, 7, 14, 30, 90 and 365} days. Dasgupta et al. (2018) and Xu et al. (2019) applied a minimum threshold of 300 triples per interval during construction for YAGO11k and Wikidata12k. We follow their time-division approaches for these two datasets and test different minimum thresholds, denoted as tr, amongst {1, 10, 100, 1000, 10000} for grouping years into different time steps. The change of time granularity will reconstruct the set of time steps $\mathcal{T}$. To illustrate, the total number of time steps in ICEWS14 is 365 with $u = 1$. When the time unit u changes from 1 to 2, the set of time steps $\mathcal{T}$ will be reconstructed and include 188 different time steps. In YAGO11k, there are totally 388 different time steps when $tr = 1$. Years like *-453*, *100* and *2008* are taken as independent time steps. When tr for YAGO11k rises to 100, the number of time steps drops to 118 and years between *-431* and *100* are clubbed into a same time step.

5.3 Evaluation Metrics

We evaluate our models on link prediction over the above-mentioned TKG benchmarks. To perform

	ICEWS14				ICEWS05-15			
Metrics	MRR	Hits@1	Hits@3	Hits@10	MRR	Hits@1	Hits@3	Hits@10
ComplEx-N3$^\diamond$	.47	.35	.54	.71	.49	.37	.55	.73
TTransE*	.255	.047	-	.601	.271	.084	-	.616
HyTE	.297	.108	.416	.655	.316	.116	.445	.681
TA-TransE	.275	.095	-	.625	.299	.096	-	.668
TA-DistMult	.477	.363	-	.686	.474	.346	-	.728
DE-SimplE	.526	.418	.592	.725	.513	.392	.578	.748
ATiSE	.550	.436	.629	.750	.519	.378	.606	.794
TeRo	.562	.468	.621	.732	.586	.469	.668	.795
TIME-PLEX(base)	.589	.499	-	.761	.632	.542	-	.802
TComplEx	.61	.53	.66	.77	.66	.59	.71	.80
TeLM	**.625**	**.545**	**.673**	**.774**	**.678**	**.599**	**.728**	**.823**

Table 2: Link prediction results on ICEWS14 and ICEWS05-15. *: results are taken from (García-Durán et al., 2018). $^\diamond$: results are taken from (Lacroix et al., 2020). Dashes: results are not reported in the responding literature. Other results are taken from the original papers. The best results among all models are written bold.

a time-aware link prediction query $(s, r, ?, T)$, we first generate the candidate list $\mathcal{C} = \{(s, r, o', T) : o' \in \mathcal{E}\}$. Following the time-wise filtered setting used in most previous TKGE-related work, e.g., TComplEx (Lacroix et al., 2020), we then remove the candidate quadruples appearing in the train Ω_{train}, valid Ω_{train} and test set Ω_{train} from the candidate list. The filtered candidate list is denoted as $\overline{\mathcal{C}} = \{\omega : \omega \in \mathcal{C}, \omega \notin \Omega_{\text{train}} \cup \Omega_{\text{valid}} \cup \Omega_{\text{test}}\}$. We get the rank of test quadruple (s, r, o, T) among the candidate quadruples $\overline{\mathcal{C}}$ by sorting their scores. We use Mean Reciprocal Rank (MRR) and Hits@N as evaluation metrics. The Mean Reciprocal Rank (MRR) is the average of the reciprocal values of all computed ranks. The percentage of testing quadruples which are ranked lower than N is considered as Hits@N.

5.4 Baselines

We compare our models with the state-of-the-art KGE model, ComplEx-N3 (Lacroix et al., 2018) and several existing TKGE approaches including TTransE (Leblay and Chekol, 2018), HyTE (Dasgupta et al., 2018), TA-TransE, TA-DistMult (García-Durán et al., 2018), ATiSE (Xu et al., 2019), TeRo (Xu et al., 2020a), DE-SimplE (Goel et al., 2020), TIME-PLEX(base) (Jain et al., 2020) and TComplEx (Lacroix et al., 2020). We do not use the complete TIME-PLEX model and the TNTComplEx model as baselines since the former incorporates additional temporal constraints for some specific relations and the latter is designed for modelling a KG where some facts involve time information and others do not. Among the existing TKGE approaches, TComplEx achieves state-of-the-art results on TKG completion.

5.5 Experimental Setup

We implement our proposed model TeLM in PyTorch. We use the Adagrad optimizer with a learning rate of 0.1 to train both models. The batch size b is fixed as 1000. The regularization weights λ_ω and $\lambda_\mathcal{T}$ are tuned in a range of {0, 0.001, 0.0025, 0.005, 0.0075, 0.01,. . . , 0.1}. To avoid too much memory consumption, we follow the setting in (Lacroix et al., 2020) to make the maximum embedding no more than 2000. The above experimental setup is also used for evaluating TComplEx on YAGO11k and Wikidata12k. Notably, the time granularity parameters u and tr are also regraded as hyperparameters for TeLM as mentioned in the previous section. The optimal hyperparameters for TeLM are as follows: $\lambda_\omega = 0.0075$, $\lambda_\mathcal{T} = 0.01$, $u = 1$ on ICEWS14; $\lambda_\omega = 0.0025$, $\lambda_\mathcal{T} = 0.1$, $u = 1$ on ICEWS05-15; $\lambda_\omega = 0.025$, $\lambda_\mathcal{T} = 0.001$, $tr = 100$ on YAGO11k; $\lambda_\omega = 0.025$, $\lambda_\mathcal{T} = 0.0025$, $tr = 1$ on Wikidata12k. The optimal embedding dimension is $k = 2000$ in all cases. The training processes of a TeLM model with $k = 2000$ on ICEWS14, YAGO11K and Wikidata12k all cost less than half an hour with a GeForce RTX 2080 GPU. On ICEWS05-15, It takes about 2 hours to train a 2000-dimensional TeLM model.

6 Results and Analysis

6.1 Link Prediction

Table 2 and 3 list the link prediction results of our models and all baseline models on four TKG

	YAGO11k				Wikidata12k			
Metrics	MRR	Hits@1	Hits@3	Hits@10	MRR	Hits@1	Hits@3	Hits@10
ComplEx-N3†	.181	.115	-	.311	.248	.143	-	.489
TTransE°	.108	.020	.150	.251	.172	.096	.184	.329
HyTE†	.136	.033	-	.298	.253	.147	-	.483
TA-DistMult†	.155	.098	-	.267	.230	.130	-	.461
TeRo	.187	.121	**.197**	.319	.299	.198	.329	.507
ATiSE	.185	.126	.189	.301	.252	.148	.288	.462
TIME-PLEX(base)	.184	.110	-	.319	.324	.220	-	.528
TComplEx△	.185	.127	.183	.307	.331	**.233**	.357	.539
TeLM	**.191**	**.129**	.194	**.321**	**.332**	.231	**.360**	**.542**

Table 3: Link prediction results on YAGO11k and Wikidata12k. †: results are taken from (Jain et al., 2020). °: results are taken from (Xu et al., 2019). △: results are obtained from our experiments. Dashes: results are not reported in the responding literature. Other results are taken from the original papers. The best results among all models are written bold.

datasets. As shown in Table 2, TeLM surpasses all baseline models on ICEWS datasets regarding all metrics. Compared to TComplEx, TeLM obtains the improvements of 1.5 MRR points on ICEWS14 and 1.8 MRR points on ICEWS05-15.

TA-TransE is not included in Table 3 since there is no literature reporting the results of TA-TransE on YAGO11k and Wikidata12k and the performances of TA-TransE are worse than most baseline models on other TKG datasets. The results of DE-SimplE on YAGO11k and Wikidata12k can not be obtained since DE-SimplE mainly focuses on event-based datasets and cannot model time intervals or time annotations missing moth and day information which are common in YAGO and Wikidata. On YAGO11k, TeLM outperforms all baseline models other than TA-TransE and DE-SimplE regarding MRR, Hits@1 and Hits@10, though performs slightly worse than TeRo on Hits@3. Additionally, TeLM also achieves the state-of-the-art results except the the Hits@1 of TComplEx is 0.1 point higher than TeLM.

6.2 Effect of Linear Temporal Regularizer

We compare the performances of the TeLM model trained with various temporal regularizers mentioned before, e.g., the smoothing temporal regularizer, the projective temporal regularizer, the 3-order autoregressive temporal regularizer, and our proposed linear temporal regularizer. As shown in Figure 2, the TeLM model trained with the linear temporal regularizer outperforms the TeLM model trained with other temporal regularizer on ICEWS14. Compared to the smoothing temporal regularizer, the linear temporal regularizer improves MRR by 0.2 point and Hits@1 by 0.3 point.

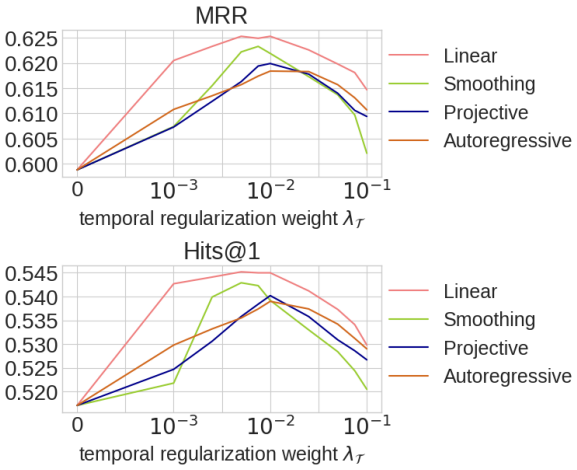

Figure 2: Results of TeLM trained with different temporal regularizers on ICEWS14.

And the linear temporal regularizer is also less sensitive to the temporal regularization weight λ_T amongst the range of $\{0.001, ..., 0.1\}$ since its bias component is learned during the training process and thus can be partly adaptive to different λ_T.

In Figure 3, In we show 2-d PCA projections of the 2000-dimensional time embeddings of TeLM models trained with/without a linear temporal regularizer. Adjacent time embeddings of TeLM trained without the temporal regularization naturally come together. However, the time embeddings representing time points in different months are not well divided. By contrast, time embeddings of TeLM trained with the linear temporal regularizer are forming good clusters in chronological order. Overall, the linear temporal regularizer provides good geometric meanings of time embeddings by effectively retaining the time sequence information in temporal KGs and thus improves the performances

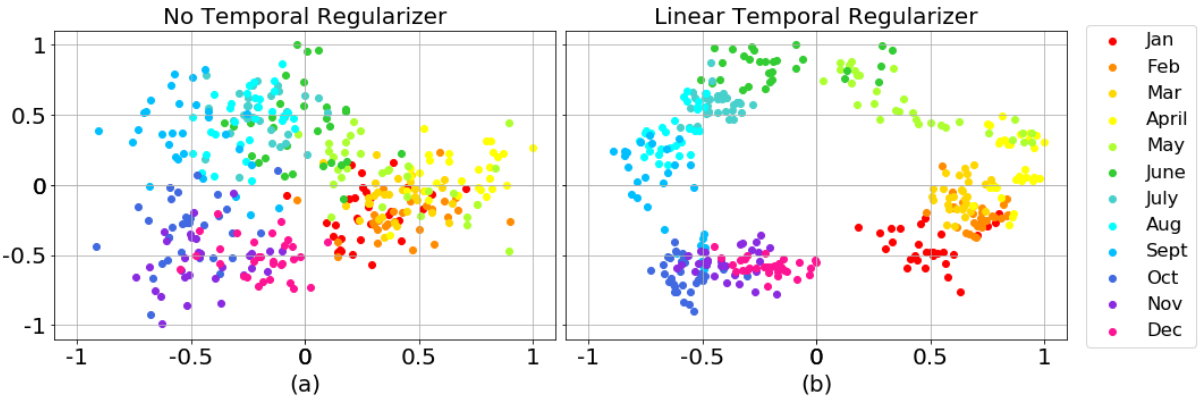

Figure 3: The figure illustrates 2-d PCA projection of the 2000 dimensional time embeddings which are obtained after training TeLM on ICEWS14 with a smoothing temporal regularizer and a linear temporal regularizer. Time points in different months are represented with different colors

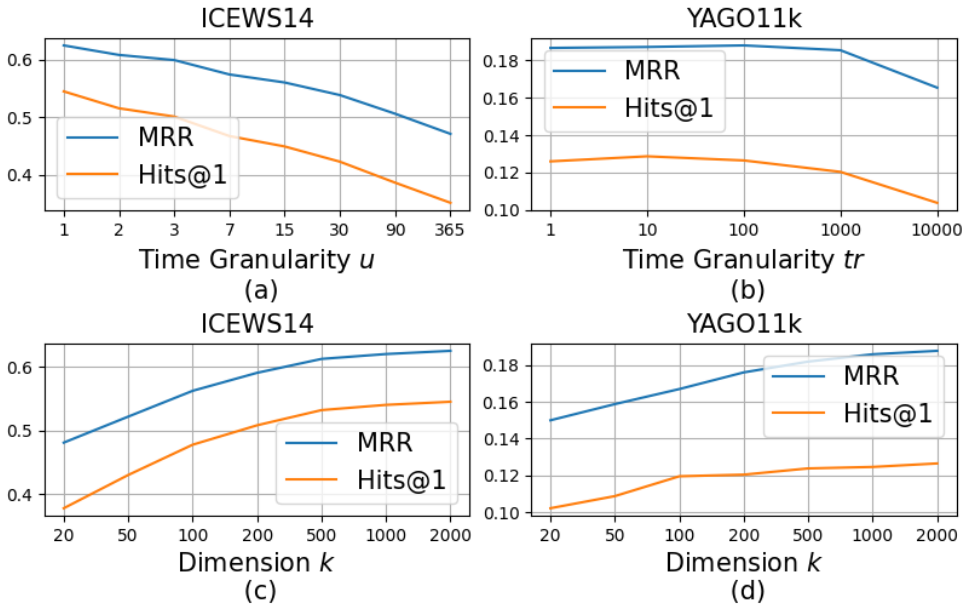

Figure 4: Results of TeLM with different time granularities and embedding dimensions on TKGs.

of TeLM.

6.3 Effect of Time Granularity and Embedding Dimension

In this work, we analyze the effect of the change of the time granularity on the performance of our model. As mentioned in the previous section, we adopt two different time-division approaches for event-based datasets, i.e., ICEWS datasets, and time-wise KGs involving time intervals, i.e., YAGO11k as well as Wikidata12k. As shown in Figure 4(a), on ICEWS14 where time distribution of facts is relatively uniform, the performance of TeLM decreases with the time unit u increasing, since representing time with a small time granularity can provide more abundant time information. On the other hand, Figure 4(b) illustrates that using the smallest time granularity is non-optimal for YAGO11k due to the long-tail property of time data. An appropriate minimum threshold used for generating time steps, e.g., $tr = 100$, can improve the link prediction results of TeLM by alleviating the effect of the long-tail property of time data and decrease the memory usage with fewer time steps. Meanwhile, using overly coarse-grained time units always leads to low performances since the time information is not fully expressed in these cases.

Figure 4(c) and (d) show that the performances on ICEWS14 and YAGO11k of TGeomE2 improve with the increasing of the embedding dimension in a range of $k = \{20, 50, 100, 200, 500, 1000, 2000\}$. TeLM with

$k = 500$ has fewer adjustable parameters than TComplEx with $k = 1740$ used in (Lacroix et al., 2020) but performs closely (0.612 vs 0.61 on MRR). It will still be interesting to explore the performances of TeLM models with higher-dimensional embeddings, e.g., Ebisu et al. (2018) use 10000-dimensional embeddings for TorusE, although it would bring more memory pressure.

7 Conclusion

We propose a new time-aware approach for TKG completion, TeLM, which performs 4th-order tensor factorization of a temporal knowledge graph using multivector embeddings for knowledge graph representation and a linear temporal regularizer for learning time embeddings. Compared to real-valued and complex-valued embeddings, multivector embeddings provides better generalization capacity and richer expressiveness with higher degree of freedom for TKGE. Moreover, the linear temporal regularizer provides better geometric meanings for time embeddings and improves the performances of TeLM compared to the temporal smoothness. Additionally, two time division methods are used for different types of TKG datasets to study the effect of the time granularity on TKG completion. Our proposed models trained with the linear temporal regularizer achieve the state-of-the-art results on time-wise link prediction over four well-known datasets involving various forms of time information, e.g., time points, begin or end time, and time intervals. Experimental results also show that choosing a reasonable time division method with an appropriate time granularity is helpful for TKG completion.

Acknowledgements

This work is supported by the EC Horizon 2020 grant LAMBDA (GA no. 809965), the CLEOPATRA project (GA no. 812997) and the China Scholarship Council (CSC).

References

Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. 2007. Dbpedia: A nucleus for a web of open data. *The semantic web*, pages 722–735.

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 1247–1250. AcM.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Advances in Neural Information Processing Systems*, pages 2787–2795.

Shib Sankar Dasgupta, Swayambhu Nath Ray, and Partha Talukdar. 2018. Hyte: Hyperplane-based temporally aware knowledge graph embedding. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2001–2011.

Takuma Ebisu and Ryutaro Ichise. 2018. Toruse: Knowledge graph embedding on a lie group. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Fredo Erxleben, Michael Günther, Markus Krötzsch, Julian Mendez, and Denny Vrandečić. 2014. Introducing wikidata to the linked data web. In *International Semantic Web Conference*, pages 50–65. Springer.

Alberto García-Durán, Sebastijan Dumančić, and Mathias Niepert. 2018. Learning sequence encoders for temporal knowledge graph completion. *arXiv preprint arXiv:1809.03202*.

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

Prachi Jain, Sushant Rathi, Soumen Chakrabarti, et al. 2020. Temporal knowledge base completion: New algorithms and evaluation protocols. *arXiv preprint arXiv:2005.05035*.

Seyed Mehran Kazemi and David Poole. 2018. Simple embedding for link prediction in knowledge graphs. In *Advances in neural information processing systems*, pages 4284–4295.

Timothée Lacroix, Guillaume Obozinski, and Nicolas Usunier. 2020. Tensor decompositions for temporal knowledge base completion. *arXiv preprint arXiv:2004.04926*.

Timothée Lacroix, Nicolas Usunier, and Guillaume Obozinski. 2018. Canonical tensor decomposition for knowledge base completion. International Conference on Machine Learning (ICML).

Jennifer Lautenschlager, Steve Shellman, and Michael Ward. 2015. Icews event aggregations.

Julien Leblay and Melisachew Wudage Chekol. 2018. Deriving validity time in knowledge graph. In *Companion of the The Web Conference 2018 on The Web Conference 2018*, pages 1771–1776. International World Wide Web Conferences Steering Committee.

Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. 2015. Learning entity and relation embeddings for knowledge graph completion. In *Twenty-ninth AAAI conference on artificial intelligence*.

Farzaneh Mahdisoltani, Joanna Biega, and Fabian M Suchanek. 2013. Yago3: A knowledge base from multilingual wikipedias. In *CIDR*.

George A Miller. 1995. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41.

Mojtaba Nayyeri, Chengjin Xu, Sahar Vahdati, Nadezhda Vassilyeva, Emanuel Sallinger, Hamed Shariat Yazdi, and Jens Lehmann. 2020. Fantastic knowledge graph embeddings and how to find the right space for them. In *International Semantic Web Conference*, pages 438–455. Springer.

Mojtaba Nayyeri, Chengjin Xu, Yadollah Yaghoobzadeh, Hamed Shariat Yazdi, and Jens Lehmann. 2019. Toward understanding the effect of loss function on the performance of knowledge graph embedding.

Uriel Singer, Ido Guy, and Kira Radinsky. 2019. Node embedding over temporal graphs. *arXiv preprint arXiv:1903.08889*.

Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. 2016. Complex embeddings for simple link prediction. *arXiv preprint arXiv:1606.06357*.

Quan Wang, Zhendong Mao, Bin Wang, and Li Guo. 2017. Knowledge graph embedding: A survey of approaches and applications. *IEEE Transactions on Knowledge and Data Engineering*, 29(12):2724–2743.

Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014. Knowledge graph embedding by translating on hyperplanes. In *AAAI*, pages 1112–1119. Citeseer.

Chengjin Xu, Mojtaba Nayyeri, Fouad Alkhoury, Jens Lehmann, and Hamed Shariat Yazdi. 2019. Temporal knowledge graph embedding model based on additive time series decomposition. *arXiv preprint arXiv:1911.07893*.

Chengjin Xu, Mojtaba Nayyeri, Fouad Alkhoury, Hamed Shariat Yazdi, and Jens Lehmann. 2020a. Tero: A time-aware knowledge graph embedding via temporal rotation. *arXiv preprint arXiv:2010.01029*.

Chengjin Xu, Mojtaba Nayyeri, Yung-Yu Chen, and Jens Lehmann. 2020b. Knowledge graph embeddings in geometric algebras. *arXiv preprint arXiv:2010.00989*.

Yonghui Xu, Shengjie Sun, Yuan Miao, Dong Yang, Xiaonan Meng, Yi Hu, Ke Wang, Hengjie Song, and Chuanyan Miao. 2020c. Time-aware graph embedding: A temporal smoothness and task-oriented approach. *arXiv preprint arXiv:2007.11164*.

Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. 2014. Embedding entities and relations for learning and inference in knowledge bases. *arXiv preprint arXiv:1412.6575*.

Hsiang-Fu Yu, Nikhil Rao, and Inderjit S Dhillon. 2016. Temporal regularized matrix factorization for high-dimensional time series prediction. In *Advances in neural information processing systems*, pages 847–855.

Shuai Zhang, Yi Tay, Lina Yao, and Qi Liu. 2019. Quaternion knowledge graph embeddings. In *Advances in Neural Information Processing Systems*, pages 2731–2741.