Knowledge-Based Systems 238 (2022) 107841

Contents lists available at ScienceDirect

Knowledge-Based Systems

journal homepage: www.elsevier.com/locate/knosys

Tucker decomposition-based temporal knowledge graph completion

Pengpeng Shao [a], Dawei Zhang [a], Guohua Yang [a], Jianhua Tao [a,b,c,*], Feihu Che [a], Tong Liu [a]

[a] National Laboratory of Pattern Recognition, Institute of Automation, Chinese Academy of Sciences, Beijing, China
[b] School of Artificial Intelligence, University of Chinese Academy of Sciences, Beijing, China
[c] CAS Center for Excellence in Brain Science and Intelligence Technology, Beijing, China

ARTICLE INFO

Article history:
Received 17 July 2021
Received in revised form 22 November 2021
Accepted 27 November 2021
Available online 17 December 2021

Keywords:
Temporal knowledge graphs
Tucker decomposition
Reconstruction
Contrastive learning

ABSTRACT

Knowledge graphs have been demonstrated to be an effective tool for numerous intelligent applications. However, a large amount of valuable knowledge still exists implicitly in the knowledge graphs. To enrich the existing knowledge graphs, recent years have witnessed that many algorithms for link prediction and knowledge graphs embedding have been designed to infer new facts. But most of these studies focus on the static knowledge graphs and ignore the temporal information which reflects the validity of knowledge. Developing the model for temporal knowledge graphs completion is an increasingly important task. In this paper, we build a new tensor decomposition model for temporal knowledge graphs completion inspired by the Tucker decomposition of order-4 tensor. Furthermore, to further improve the basic model performance, we provide three kinds of methods including cosine similarity, contrastive learning, and reconstruction-based to incorporate the prior knowledge into the proposed model. Because the core tensor contains a large number of parameters on the proposed model, thus we present two embedding regularization schemes to avoid the over-fitting problem. By combining these two kinds of regularization with the proposed model, our model outperforms baselines with an explicit margin on three temporal datasets (i.e. ICEWS2014, ICEWS05-15, GDELT).

1. Introduction

Knowledge graphs (KGs) which are represented as a collection of triples (subject, predicate, object), are graph-structured representations of knowledge and facts in the real-world and have been demonstrated to be available for various downstream tasks, such as recommendation [1], question answering [2], and information retrieval [3]. However, a great deal of knowledge remains hidden in the KGs, namely many links are missing between entities in KGs. In recent years, many algorithms for KGs completion are proposed to enrich KGs. As a representative method, KGs embedding is a current research hotspot and presents its efficiency and effectiveness on KGs completion. This type of method aims to model the nodes and relations in KGs and learn their low-dimensional hidden representation on the premise of preserving the graph structure and knowledge, then feeding the representation of each triple to score function for its validity.

As the increase of structured data, many facts involve temporal property, such as (Trump, PresidentOf, America) is true from 2017 to 2021, (Einstein, WonPrize, Nobel Prize) only hold in 1922, and knowledge presents dynamic and temporal gradually. However, static KGs completion algorithms without taking temporal information cannot achieve considerable link prediction [4] performance on these temporal facts. Therefore, temporal KGs completion becomes an increasingly important task. On a temporal KGs with timestamps varying from t_0 to t_T, temporal KGs completion has two settings — interpolation and extrapolation. In the interpolation setting, new facts are inferred out for time t ($t_0 \leq t \leq t_T$) by using historical information and future information. Corresponding extrapolation setting only employs historical knowledge to predict future facts for time t ($t > t_T$). In this work, we focus on the interpolation temporal KGs completion. To be specific, our task is to answer the queries (subject, predicate, ?, timestamps) and (?, predicate, object, timestamps) on temporal KGs with timestamps varying from t_0 to t_T.

Recently, tensor factorization methods have been successfully applied to KGs completion as Table 1. [5,6] apply Canonical Polyadic (CP) decomposition and Tucker decomposition to static KGs completion, respectively. And they frame KGs completion problem as an order-3 tensor completion problem. Inspired by the CP decomposition of order-4 tensor, TNTComplEx [7] expresses temporal KGs completion problem as an order-4 tensor completion problem and presents an extension of ComplEx for temporal KGs completion. However, it is an NP-hard problem [8] to calculate the rank of tensors for CP decomposition, thus TNTComplEx has to put great effort into selecting a proper rank of tensors manually for temporal KGs completion. In addition,

* Corresponding author at: National Laboratory of Pattern Recognition, Institute of Automation, Chinese Academy of Sciences, Beijing, China.
E-mail address: jhtao@nlpr.ia.ac.cn (J. Tao).

https://doi.org/10.1016/j.knosys.2021.107841
0950-7051/© 2021 Published by Elsevier B.V.

Table 1
Tensor decomposition-based KGs completion models.

Models	CP decomposition	Tucker decomposition
Static KGs	DistMult [10], ComplEx [11], SimplE [12]	TuckER [6]
Temporal KGs	DE-SimplE [13], TNTComplEx [7], TIMEPLEX [9]	Ours

TNTComplEx integrates the prior that adjacent timestamps have close representations into its model, and fails to consider the non-adjacent timestamps, thus it does not model more complete prior knowledge. TIMEPLEX [9] performs CP order-3 tensor decomposition on temporal KGs, which constructs a base score by augmenting the ComplEx score and employs three time-dependent terms to replace a single four-way product. Moreover, it designs three temporal constraints to incorporate the prior into its model, including relation recurrence, relations order, and the time gaps between relations. However, these three temporal constraints fail to help the basic model much in terms of performance. Therefore, designing a powerful temporal KGs completion model incorporating prior knowledge remains an important task.

To this end, we develop a new Tucker decomposition model for temporal KGs completion inspired by the Tucker decomposition of order-4 tensor, and it is capable of achieving considerable performance by virtue of its powerful expressive ability. Based on this, to further improve the basic model performance, we design three novel methods including cosine similarity, contrastive learning, and reconstruction-based to incorporate prior knowledge about timestamps into the proposed model. Note that, the core tensor on our model contains a large number of parameters, thus we present two embedding regularization schemes to avoid the overfitting problem. Overall, by combining these two kinds of regularization with the basic model, the proposed model not only achieves considerable prediction performance and promising generalization performance, but also provides a support for applying Tucker decomposition to temporal KGs as Table 1. Our contributions are as follows:

- Developing a new tensor decomposition model for temporal KGs completion inspired by the Tucker decomposition of order-4 tensor.
- Presenting three methods including cosine similarity, contrastive learning, and reconstruction-based to incorporate prior knowledge into the proposed model to help to improve the basic model.
- Introducing two embedding regularization schemes to avoid the overfitting problem caused by the core tensor in the proposed model containing a large number of parameters.
- Experimental studies on three temporal datasets demonstrate that our algorithm achieves state-of-the-art performance.

2. Related work

In this section, we introduce the previously proposed methods for static and temporal KGs completion. Please kindly note that most of the related works are based on tensor decomposition.

2.1. Static KGs completion

Static KGs completion methods based on embedding can be broadly classified into three paradigms: Translational distance-based models such as TransE [14] and TransD [15], Tensor factorization based methods, and Neural network-based models including ConvE [16] and R-GCN [17]. In particular, tensor decomposition has been favored in KGs completion for its high efficiency and powerful function. RESCAL [18] is the seminal tensor decomposition method for static KG completion, which imposes a score function of a bilinear product on the two entity vectors and predicate matrix. Although the model is fully expressive, it tends to arise over-fitting problems as the predicate matrix includes a large number of parameters. Later, DistMult [10] is aware of the above defects, then simplifying RESCAL by diagonalizing the predicate matrix. This also poses a problem that the diagonal predicate matrix only models symmetric relation but asymmetric relation. To address this problem, ComplEx [11] projects entity and predicate embeddings into complex space to better model asymmetric relation. From another perspective, HoLE [19] applies circular correlation operation to subject and object entity vectors to obtain a compositional vector, which then matches the predicate vectors to score the fact, thus the model absorbs the advantages of RESCAL and DistMult. The above methods based on CP tensor decomposition learn the subject and object entity representation independently. This is also the main reason for conducting link prediction poorly. In view of this, SimplE [12] presents a new CP method that takes advantage of the inverse of the predicate to address the obstacle. In addition to the methods based on CP decomposition, TuckER [6] based on Tucker decomposition also frames KGs completion as a order-3 binary tensor completion problem and factorizes the binary tensor of known facts into core tensor and three orthogonal matrixes.

2.2. Temporal KGs completion

Most static completion models fail to take temporal information while learning embeddings of the KG elements. The temporal KGs completion remains a valuable but rarely studied research issue, and recent years witness that only a handful of temporal KGs completion models are presented.

t-TransE [20]: To model the transformation between the time-aware predicate of two adjacent facts, t-TransE imposes temporal order constraints on the geometric structure of the embedding space to enforce the embeddings to be temporally consistent and more accurate. Then t-TransE optimizes the joint model consisting of temporal order constraints and the TransE model to make the embedding space compatible with the observed triple in the fact dimension.

HyTE [21]: Inspired by TransH [22], HyTE associates temporal information with entity and predicate by projecting them to the hyperplane modeled by temporal information. Then HyTE accomplishes the embedding learning of entity and predicate which incorporate temporal information by minimizing translation distance.

TA-DistMult [23]: To incorporate temporal information, TA-DistMult employs a recurrent neural network to learn the time-aware representation of predicate which then be utilized in DistMult and TransE.

ConT [24]: To model the cognitive function, the work generalized several static KGs approaches including Tucker and RESCAL to temporal/episodic KGs. Tree and ConT are two novel generalizations of RESCAL to episodic tensors, and ConT obtains outstanding performance overall through introducing the latent representation of time for sparse episodic tensors.

DE-SimplE [13]: Motivated by diachronic word embeddings, DE-SimplE, combining the diachronic entity embedding function with the static model SimplE, is capable of instructing the model to learn the temporal features of the entity at any point in time for temporal KG completion.

TNTComplEx [7]: Inspired by the canonical decomposition of order-4 tensor, TNTComplEx introduces an extension of ComplEx for temporal KG completion. Although TNTComplEx obtains considerable performance, it is hard to determine the rank of the tensor accurately.

ATiSE [25]: To capture the evolution process of KG representations, ATiSE decomposes a time series into three components, including a trend component, a seasonal component, and a random component. Meanwhile, ATiSE also considers the temporal uncertainty during the evolution of entity/relation representations over time and projects the representations of temporal KGs into the space of multi-dimensional Gaussian. distributions.

TIMEPLEX [9]: TIMEPLEX constructs a base score including three time-dependent terms by augmenting the ComplEx score and designs three temporal constraints to incorporate the prior knowledge, including relation recurrence, relations order, and the time gaps between relations.

TeLM [26]: TeLM performs 4th-order tensor factorization of a Temporal knowledge graph using a Linear temporal regularizer and Multivector embedding. In addition, TeLM develops a novel linear temporal regularizer by adding a bias component between the neighboring temporal embeddings to promote that the difference between embeddings of two adjacent time steps is smaller than the difference between embeddings of two distant time steps.

RTFE-TComplEx [27]: RTFE-TComplEx which uses TComplEx as a baseline model treats the KGs of each timestamp as a Markov chain and approximates the state transition as the gradient update process. In brief, RTFE tracks the state transition of TKG by passing updated parameters/features between timestamps recursively.

In this work, we propose a Tucker decomposition of an order-4 tensor for temporal KG completion and introduce three methods including contrastive learning and reconstruction-based to incorporate prior knowledge into the proposed model. Experimental results show that our algorithm achieves state-of-the-art performance.

3. Background

CP decomposition [28] factorizes a tensor $\mathcal{X}$ into sum of R rank-one tensors, which is described as follows,

$$\mathcal{X} = \sum_{r=1}^{R} a_r \circ b_r \circ c_r \tag{1}$$

Here, R is the rank of the tensor $\mathcal{X}$, $\circ$ represents the vector outer product. From the above formula, we can obtain that the first problem in calculating CP decomposition is how to determine the rank R of tensor. But [8] proves that it is an NP-hard problem to calculate the rank of tensors for CP decomposition, thus TNT-ComplEx [7] faces the problem of selecting an appropriate rank of tensor.

While Tucker decomposition [29] is a form of high-order PCA, which factorizes a tensor into a core tensor multiplied by a matrix along each mode. Specifically, given a tensor $\mathcal{X} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$, Tucker decomposition can factorize $\mathcal{X}$ along three mode into core tensor $\mathcal{G}$ and three matrix A, B, C,

$$\begin{aligned}\mathcal{X} &\approx \mathcal{G} \times_1 A \times_2 B \times_3 C \\ &= \sum_p \sum_k \sum_q g^{(pkq)} a^{(p)} \circ b^{(k)} \circ c^{(q)} = [\![\mathcal{G}; A, B, C]\!]\end{aligned} \tag{2}$$

where the core tensor $\mathcal{G} \in \mathbb{R}^{r_1 \times r_2 \times r_3}$ can capture the information of interaction between the different components. The matrix in each mode $A \in \mathbb{R}^{n_1 \times r_1}$, $B \in \mathbb{R}^{n_2 \times r_2}$, $C \in \mathbb{R}^{n_3 \times r_3}$ are orthogonal to each other. $\times_n$ suggests the tensor product along the nth mode. In addition, if the core tensor is super-diagonal and satisfies $r_1 = r_2 = r_3$, the Tucker decomposition is equivalent to CP decomposition.

Latter, TuckER [6] employs this type of decomposition for KGs completion, which views matrix A and C as entity embedding E, $E = A = C \in \mathbb{R}^{n_e \times d_e}$, with n_e indicates the number of entities, d_e suggests the dimensionality of entity embedding. Regarding B as predicate embedding R, $R = B \in \mathbb{R}^{n_r \times d_r}$, where n_r and d_r denote the number of predicates and the dimensionality of predicate embedding, respectively. The scoring function is represented as follows,

$$\phi(e_s, e_r, e_o) = \mathcal{W} \times_1 \mathrm{e}_s \times_2 \mathrm{e}_r \times_3 \mathrm{e}_o = [\![\mathcal{W}; \mathrm{e}_s, \mathrm{e}_r, \mathrm{e}_o]\!] \tag{3}$$

where $\mathcal{W} \in \mathbb{R}^{d_e \times d_r \times d_e}$ also is core tensor, and the number of parameter in $\mathcal{W}$ only relies on the embedding dimensionality of entity and predicate, not on the number of entities or predicates. Meanwhile, TuckER also justifies that ComplEx based on CP decomposition is a special case of TuckER.

4. The proposed model

The temporal fact (s, r, o, t) gives the triple (s, r, o) a temporal label t to ensure its accuracy. We argue that the temporal information is contained in the entity or predicate of the correct triple fact implicitly. For example, if the fact (Trump, PresidentOf, America) is universally true, then the temporal element [2017, 2021] of the 4-tuple is contained implicitly in the subject entity of Trump, or the predicate of PresidentOf, or the object entity of America. We can express the above three cases in the following forms,

- (Trump & [2017, 2021], PresidentOf, America): Trump of that period from 2017 to 2021 was President of America. That is, the entity of Trump in the triple refers to Trump of that period from 2017 to 2021.
- (Trump, PresidentOf & [2017, 2021], America): Trump was President of that period from 2017 to 2021 of America.
- (Trump, PresidentOf, America & [2017, 2021]): Trump was President of America of the period from 2017 to 2021.

Although the timestamps information emphasizes different objects to express their temporal nature, respectively, these three facts accurately express the same meaning. Therefore, we can extract the temporal information T from the triples to express the temporal facts in form of 4-tuple. Meanwhile, the embedding dimensionality of timestamps is the same as the embedding dimensionality of predicate or entity. Thus the Tucker decomposition can naturally be extended to the following general form by adding temporal information T [30],

$$\phi(E, R, T) = [\![\mathcal{M}; E, R, E, T]\!] \tag{4}$$

where $\mathcal{M} \in \mathbb{R}^{d_e \times d_r \times d_e \times d_t}$ is core tensor. We call this kind of decomposition TuckERT, E represents entity matrix, R denotes predicate matrix, and T indicates temporal matrix. Given the temporal fact $(\mathrm{e}_s, \mathrm{e}_r, \mathrm{e}_o, \mathrm{e}_t)$, the TuckERT decomposition is expressed in following inner product form,

$$\phi(E, R, T)_{s,r,o,t} = \langle \mathcal{M}; \mathrm{e}_s, \mathrm{e}_r, \mathrm{e}_o, \mathrm{e}_t \rangle \tag{5}$$

As described in the above example, the time-wise information can be employed to associate with the subject, predicate or object to obtain time-dependent embedding equivalently. In addition, obtaining the time-dependent embedding can be viewed as the inverse process of TuckERT decomposition,

$$\begin{aligned}\phi(E, R, T)_{s,r,o,t} = \langle \mathcal{M}; \mathrm{e}_s, \mathrm{e}_r, \mathrm{e}_o, \mathrm{e}_t \rangle &= \langle \mathcal{W}; \mathrm{e}_s \odot \mathrm{e}_t, \mathrm{e}_r, \mathrm{e}_o \rangle \\ &= \langle \mathcal{W}; \mathrm{e}_s, \mathrm{e}_r \odot \mathrm{e}_t, \mathrm{e}_o \rangle \\ &= \langle \mathcal{W}; \mathrm{e}_s, \mathrm{e}_r, \mathrm{e}_o \odot \mathrm{e}_t \rangle\end{aligned} \tag{6}$$

Here, $\odot$ denotes dot product, $\mathcal{W} \in \mathbb{R}^{d_e \times d_r \times d_e}$ is the folding version of $\mathcal{M}$ on dimension 2 and 4 when associating the time-wise information with predicate. In addition, it is worth noting that some facts may vary with time-wise dimension, while some facts are independent of time. As mentioned above, the fact (Trump, PresidentOf, America) hold from 2017 to 2021, (Einstein, WonPrize, Nobel Prize) is true only at 1922. While the correctness of the facts (Beijing, CityOf, China) does not change over time. To model the two kinds of knowledge, temporal facts and non-temporal facts, following [7,13], we propose a TuckERTNT model which is a variant of TuckERT,

$$\phi(E,R,T) = [\![\mathcal{M}; E, R, E, T]\!] + [\![\mathcal{M}; E, R, E, \mathbf{1}]\!] \tag{7}$$

Similarly, given the temporal fact $(\mathrm{e}_s, \mathrm{e}_r, \mathrm{e}_o, \mathrm{e}_t)$, the specific form of the TuckERTNT decomposition can be described as follows,

$$\begin{aligned}\phi(E,R,T)_{s,r,o,t} &= \left\langle \mathcal{W}; \mathrm{e}_s, \mathrm{e}_r^t, \mathrm{e}_o, \mathrm{e}_t \right\rangle + \langle \mathcal{W}; \mathrm{e}_s, \mathrm{e}_r, \mathrm{e}_o, 1\rangle \\ &= \left\langle \mathcal{W}; \mathrm{e}_s, \mathrm{e}_r^t \odot \mathrm{e}_t, \mathrm{e}_o \right\rangle + \langle \mathcal{W}; \mathrm{e}_s, \mathrm{e}_r \odot 1, \mathrm{e}_o\rangle \\ &= \left\langle \mathcal{W}; \mathrm{e}_s, \mathrm{e}_r^t \odot \mathrm{e}_t + \mathrm{e}_r, \mathrm{e}_o \right\rangle\end{aligned} \tag{8}$$

where NT (the suffix of TuckERTNT) is an abbreviation of non-temporal. e_r^t and e_r are two different representations of same relation, respectively, that is, temporal relation representation and static relation representation. Compared with TNTComplEx, not only does our model not need to select the rank of tensors manually, but it also has a powerful expressive ability. We associate the temporal information with the predicate to learn time-dependent embeddings based on the Tucker decomposition and use the parameters of the core tensor to increase the level of interaction in each dimension between the entity and time-dependent predicate, thus obtaining state-of-the-art performance. Furthermore, [6] proved that ComplEx is equivalent to TuckER on the premise of imposing certain constraints on the core tensor, or ComplEx is a special case of TuckER. We give a similar result that TComplEx can be viewed as equivalent to TuckERT on the premise of certain constraints.

4.1. Time regularization

We expect to integrate more prior knowledge into the proposed model to improve model performance. Here, we provide three kinds of models and the corresponding prior knowledge about timestamps that they are modeled.

(1) **Cosine Similarity** : It uses the cosine value of the angle between two vectors to measure the similarity. Here, we employ cosine similarity measure to model the prior that adjacent timestamps have close representations, that is

$$\mathcal{L}_s(T) = -\frac{e_{t_{i+1}}^T \cdot e_{t_i}}{|e_{t_{i+1}}||e_{t_i}|} \tag{9}$$

(2) **Contrastive learning** : Inspired by first order Markov chain (Given the state at the current moment, the state at the next moment is independent of the state at the past moment), thus, in addition to that the representations between the adjacent timestamps are similar, we add a constraint that the representations between the non-adjacent timestamps are dissimilar. This can be modeled in contrastive learning fashion as,

$$\mathcal{L}_c(T) = -\log \frac{exp(e_{t_i}^T e_{t_{i+1}})}{exp(e_{t_i}^T e_{t_{i+1}}) + \sum_{k=i+2}^N exp(e_{t_i}^T e_{t_k})} \tag{10}$$

where N denotes the number of the timestamps.

(3) **Reconstruction-based** : (1) only considers adjacent timestamps. (2) considers adjacent and non-adjacent timestamps simultaneously, and it argues that the representations between the adjacent timestamps are similar and the representations between the non-adjacent timestamps are dissimilar. This part proposes a "soft" smooth constraint that the similarity of the representation of the two timestamps will decrease as the time distance increases. To model this assumption, we first construct a template by the following function:

$$M_{t_i,t_j} = e^{-\frac{\|t_i - t_j\|^2}{2\sigma^2}} \tag{11}$$

where σ is the kernel width. The element m_{t_i,t_j} of M represents the inverse of the distance between t_i and t_j. The similar matrix S can be obtained by cosine similarity measure,

$$S = \frac{e_{t_i}^T \cdot e_{t_j}}{|e_{t_i}||e_{t_j}|} \tag{12}$$

Thus, to achieve the proposed assumption, we enforce the representation similarity of two timestamps to confirm to the inverse distance relations,

$$\mathcal{L}_r(T) = \|S - M\|_2^2 \tag{13}$$

Please kindly note that the larger the value of σ, the smoother the results, so it has a powerful impact on learning the representation of timestamps.

4.2. Embedding regularization

To prevent the model from over-fitting, we impose embedding regularization constraints on the elements of the proposed model. Note that, because the core tensor contains a large number of parameters, we argue that exerting a constraint on the core tensor is important and may have an appreciable impact on our model. Therefore, based on the regularization scheme, we study the impact of imposing constraints on the core tensor and without constraints, respectively.

$$\begin{aligned}\mathcal{L}_p(E) &= \frac{1}{4}(2\|e_s\|_p^q + \|e_r^t \odot e_t\|_p^q + 2\|e_o\|_p^q + \|e_r\|_p^q) \\ \mathcal{L}_p(E,\mathcal{W}) &= \frac{1}{5}(2\|e_s\|_p^q + \|e_r^t \odot e_t\|_p^q + 2\|e_o\|_p^q + \|e_r\|_p^q + \|\mathcal{W}\|_p^q)\end{aligned} \tag{14}$$

where $\|\cdot\|_p$ is the l_p norm of the matrix, and $\|\cdot\|^q$ denotes the q power of tensor norm.

4.3. Learning

We employ data augmentations to add reciprocal predicates (object, predicate^{-1}, subject, timestamps) into training sets. The model parameters are learned utilizing stochastic gradient descent with mini-batches. Then we expect to minimize the instantaneous multi-class loss [5] to train our model :

$$\mathcal{L}(\theta) = -\phi(\theta; s, r, o, t) + \log\left(\sum_{o'} \exp\left(\phi(\theta; s, r, o', t)\right)\right) \tag{15}$$

Here, (s, r, o, t) is a positive sample, (s, r, o', t) represents the negative sample obtained by replacing the true object with a false. Note that, the loss function can only be used to train the model to answer the queries of this form (subject, predicate, ?, timestamps). Due to the existence of inverse samples (object, predicate^{-1}, subject, timestamps), answering (object, predicate^{-1}, ?, timestamps) is equivalent to answer (?, predicate, object, timestamps).

Considering instantaneous multi-class loss and the above two classes of regularization term jointly, we train our model by minimizing the following loss function:

$$\mathcal{L} = \mathcal{L}(\theta) + \lambda_1 * \mathcal{L}_r(T) + \lambda_2 * \mathcal{L}_p(E,\mathcal{W}) \tag{16}$$

where λ_1 and λ_2 are tuning parameters to balance the importance of $\mathcal{L}_r(T)$ and $\mathcal{L}_p(E,\mathcal{W})$, respectively.

Table 2
Number of parameters of the proposed method and baseline models. d represents the embedding dimensionality, r denotes the rank of a tensor, and $2r = d$.

De-SimplE	$d((3\gamma + (1-\gamma))\|E\| + \|R\|)$
TComplEx	$2r(\|E\| + \|T\| + 2\|R\|)$
TNTComplEx	$2r(\|E\| + \|T\| + 4\|R\|)$
TIMEPLEX	$d(\|E\| + \|T\| + 6\|R\|)$
TeLM	$d(\|E\| + \|T\| + 2\|R\|)$
TuckERT	$d(\|E\| + \|T\| + 2\|R\|) + d^3$
TuckERTNT	$d(\|E\| + \|T\| + 4\|R\|) + d^3$

4.4. Time complexity and parameter growth

Table 2 presents the numbers of parameters for the proposed models TuckERT, TuckERTNT, and tensor decomposition-based baseline models De-SimplE, TComplEx, TNTComplEx, TIMEPLEX, and TeLM. The number of parameters in baseline models increases linearly with respect to the number of entities and predicates or embedding dimensionality d. While the number of parameters in our model grows three times with embedding dimensionality d as the three-dimension core tensor depends only on the embedding dimensionality. Consequently, the time complexity for the proposed models TuckERT and TuckERTNT is $\mathcal{O}(d^3)$. As for De-SimplE, TComplEx, TNTComplEx, TIMEPLEX, and TeLM, they have a time complexity of $\mathcal{O}(d)$. It can be viewed that our model includes more parameters on the premise of the fixed number of entities, predicates, and embedding dimensionality. However, it has been argued that a model with many parameters tends to arise over-fitting and scalability problems, thus resulting in poor performance. Table 4 suggests that the proposed model can better fit the large-scale discrete temporal data by using more controllable parameters compared with the baseline models. Accordingly, an important challenge in designing a tensor decomposition model for temporal knowledge completion is the trade-off between model parameters and data, as well as the trade-off between expressiveness and model complexity.

4.5. Expressivity analysis

Full expressiveness is a very important property for the KG completion model, which refers to the ability of a model to correctly distinguish positive facts from negatives through learning. The proof of full expressiveness about our model is introduced as follows.

Theorem 1. *TuckERT is fully expressive for temporal knowledge graph completion.*

Proof. Given a 4-tuple (e_s, e_r, e_o, e_t), where $e_s, e_o \in \mathbb{R}^{n_e \times d_e}$ are one-hot binary vector representations of subject and object, $e_r \in \mathbb{R}^{n_r \times d_r}$, $e_t \in \mathbb{R}^{n_t \times d_r}$ are one-hot binary vector representations of predicate and timestamps, respectively. Here, the embedding dimensionality satisfies $d_e = n_e$, $d_r = n_r$. We set the pth element of the binary vector e_s, kth element of e_r, qth element of e_o, rth element of e_t to be 1, all other elements to be 0. Moreover, we set the $pkqr$th element of the tensor $\mathcal{M} \in \mathbb{R}^{d_e \times d_r \times d_e \times d_r}$ to 1 if the temporal fact (e_s, e_r, e_o, e_t) holds and -1 otherwise. According to TuckERT decomposition:

$$\begin{aligned}\phi(E, R, T) &= \langle \mathcal{M}; \mathrm{e}_s, \mathrm{e}_r, \mathrm{e}_o, \mathrm{e}_t \rangle \\ &= \sum_p \sum_k \sum_q \sum_r \mathcal{M}^{(pkqr)} \mathrm{e}_s^{(p)} \circ \mathrm{e}_r^{(k)} \circ \mathrm{e}_o^{(q)} \circ \mathrm{e}_t^{(r)} \end{aligned} \tag{17}$$

the inner product of the entity embeddings, the predicate embedding and time embedding with the core tensor is capable of representing the original temporal tensor accurately. And by modulating the parameters in core tensor, the model can completely distinguish the positive samples from the negative.

From another perspective, we can regard the core tensor $\mathcal{M}$ as a linear classifier in high dimensional space, which possesses the ability to distinguish the positive and negative samples in low dimensional space through learning. □

5. Experiment result

5.1. Datasets

We evaluate the proposed model by utilizing the following three standard benchmarks for temporal KGs completion. The details of these dataset statistics are presented in Table 3.

- ICEWS2014: The ICEWS (Integrated Crisis Early Warning System) [31] dataset is the collection of 4-tuple which is extracted from digital and social news about political events. Intuitively, ICEWS2014 [23] sub-sampling from ICEWS is the temporal facts occurring in 2014, and it contains 7128 entities, 230 predicates, and 365 timestamps.
- ICEWS05-15: Similarly to ICEWS2014, ICEWS05-15 [23] is another subset of ICEWS. This dataset corresponding to the temporal facts from 2005 to 2015 has 10488 entities, 251 predicates, and 4017 timestamps.
- GDELT: GDELT (Global Database of Events, Language, and Tone) [32] is a repository that contains human social relationships. We implement our models and baselines on its subset dataset [33], which corresponds to the facts from 2015 to 2016 and contains 500 entities, 20 predicates, and 366 timestamps.

5.2. Baselines

To evaluate the performance of our proposed TuckERT and TuckERTNT, we compare it with 17 state-of-the-art baseline methods introduced briefly in related work, including static KGs completion methods TransE [14], DistMult [10], ComplEx [11], SimplE [12], and temporal methods t-TransE [20], HyTE [21], TA-DistMult [23], ConT [24], three variants of DE-SimplE [13], TComplEx [7], TNTComplEx [7], ATiSE [25], TIMEPLEX [9], TeLM [26], RTFE-TComplEx [27]. For consistency and fairness, most of baselines are based on tensor decomposition.

5.3. Evaluation protocol

Given the incomplete temporal data, the task of link prediction is to predict the missing entity. More specifically, this task answers the queries of the form (subject, predicate, ?, timestamps) and (?, predicate, object, timestamps). For the above two queries, we employ mean reciprocal rank (MRR) and Hit@n to measure the prediction level of our model. MRR is the average of the reciprocal of the mean rank (MR) assigned to the true triple overall candidate triples, Hits@n measures the percentage of test-set rankings where a true triple is ranked within the top n candidate triples. While MRR is widely used as an evaluation indicator of inference in the literature as MRR is more stable [34] than MR which is highly susceptible to a bad prediction. We denote $k_{f,s}$ and $k_{f,o}$ as the ranking for subject s and object o for the two queries, respectively. MRR and Hit@n are defined as follows. MRR: $\frac{1}{2*|test|} \sum_{f=(s,r,o,t)\in test} (\frac{1}{k_{f,o}} + \frac{1}{k_{f,s}})$. Hit@$n$: $\frac{1}{2*|test|} \sum_{f=(s,r,o,t)\in test} (\mathbb{1}_{k_{f,o} \leqslant n} + \mathbb{1}_{k_{f,s} \leqslant n})$, where $\mathbb{1}_{variable}$ is 1 if *variable* holds and 0 otherwise.

Table 3
Dataset statistics.

DataSets	Entities	Predicates	Timestamps	Training	Validation	Test
ICEWS2014	7,128	230	365	72,826	8,941	8,963
ICEWS05-15	10,488	251	4,017	386,962	46,275	46,092
GDELT	500	20	366	2,735,685	341,961	341,961

Table 4
The results of Link prediction on ICEWS2014, ICEWS05-15 and GDELT datasets. Best results are in bold.

Method	ICEWS2014				ICEWS05-15				GDELT			
	MRR	Hit@1	Hit@3	Hit@10	MRR	Hit@1	Hit@3	Hit@10	MRR	Hit@1	Hit@3	Hit@10
TransE	0.326	0.154	0.430	0.644	0.330	0.152	0.440	0.660	0.155	0.060	0.178	0.335
DistMult	0.441	0.325	0.498	0.668	0.457	0.338	0.515	0.691	0.210	0.133	0.224	0.365
ComplEx	0.442	0.400	0.430	0.664	0.464	0.347	0.524	0.696	0.213	0.133	0.225	0.366
SimplE	0.458	0.341	0.516	0.687	0.478	0.359	0.539	0.708	0.206	0.124	0.220	0.366
t-TransE	0.255	0.074	–	0.601	0.271	0.084	–	0.616	0.115	0.0	0.160	0.318
HyTE	0.297	0.108	0.416	0.655	0.316	0.116	0.445	0.681	0.118	0.0	0.165	0.326
TA-DistMult	0.477	0.363	–	0.686	0.474	0.346	–	0.728	0.206	0.124	0.219	0.365
ConT	0.185	0.117	0.205	0.315	0.163	0.105	0.189	0.272	0.144	0.080	0.156	0.265
De-TransE	0.326	0.124	0.467	0.686	0.314	0.108	0.453	0.685	0.126	0.0	0.181	0.350
De-DistMult	0.501	0.392	0.569	0.708	0.484	0.366	0.546	0.718	0.213	0.130	0.228	0.376
De-simplE	0.526	0.418	0.592	0.725	0.513	0.392	0.578	0.748	0.230	0.141	0.248	0.403
ATiSE*	0.545	0.423	0.632	0.757	0.519	0.378	0.606	0.794	–	–	–	–
TComplEx	0.610	0.530	0.660	0.770	0.660	0.590	0.710	0.800	0.217	0.128	0.231	0.372
TNTComplEx	0.620	0.520	0.660	0.760	0.670	0.590	0.710	0.810	0.224	0.144	0.239	0.381
TIMEPLEX*	0.604	0.515	–	0.771	0.639	0.545	–	0.818	–	–	–	–
TeLM*	0.618	0.535	0.667	0.772	0.673	0.592	0.723	0.819	–	–	–	–
RTFE-TComplEx*	0.592	0.503	0.646	0.758	0.645	0.553	0.706	0.811	0.297	0.212	0.319	0.464
TuckERT	0.607	0.528	0.655	0.751	0.647	0.570	0.694	0.789	**0.448**	**0.352**	**0.492**	**0.630**
TuckERTNT	**0.625**	**0.544**	**0.673**	**0.773**	**0.675**	**0.593**	**0.725**	**0.819**	0.425	0.326	0.470	0.622

5.4. Parameters settings

We implement the proposed model and baseline models in Pytorch [35] framework. The two variants of our model are optimized by Adgard algorithm [36] with batch size of 1000 and learning rate of 0.2. To evaluate the impact of embedding dimensionality on the performance of link prediction, we vary the embedding dimensionality in the range {32, 64, 100, 200, 300, 400}. Considering the efficiency and effectiveness of the model, the embedding dimensionality is eventually set to 300. Time weight coefficient λ_1 and embedding weight coefficient λ_2 are set as 1 and 0.002, respectively. The norm p of the tensor is set to 4, the power q of the tensor norm is set to 2. The kernel width σ is set to 5. For generality and fairness, we consistently apply the above parameters to perform our models on the three datasets. In addition, we reproduce the results of TNTComplEx on GDELT dataset with the tensor rank of 256 for model comparison.

5.5. Results and analysis

We evaluate two variants of our model on ICEWS2014, ICEWS05-15, and GDELT datasets, respectively. The evaluation results against the baseline models are presented in Table 4where results marked (*) are taken from reported results of TIMEPLEX [9], TeLM [26], and RTFE [27]. We have the following observations and analyses. First, from the comparative results, we can obtain that the proposed method TuckERT and TuckERTNT outperform our baseline models on the three evaluation datasets overall, which suggests that the proposed model is effective and achieves considerable results. Second, on ICEWS2014 and ICEWS05-15 datasets, the performance of our method is close to that of TCompleX, we argue that the main reasons are as follows. Compared with GDELT, the amount of training data on ICEWS2014 and ICEWS05-15 is small so that TComplEx and our model can fit them. In addition, both models are designed based on tensor decomposition, thus they have similar performances on ICEWS2014 and ICEWS05-15. However, it is worth noting that even though TComplEx and our model have similar performance on these two datasets, our model has two advantages over TComplEx. On the one hand, our model is convenient and can evade the problem that selecting rank of tensor manually. On the other hand, our model has a large number of parameters so that it can fit more training data than TComplEx and other baselines. Third, our model obtains a promising result on the GDELT dataset, the proposed model TuckERT is almost twice of the baseline model TNTComplEx and is superior to the state-of-the-art method RFTE-TComplEx 15% on MRR. This also suggests the superiority of the extension of Tucker decomposition in temporal KGs completion. In addition, the reasons why the proposed model achieves considerable results on the GDELT dataset are analyzed as follows. (i) As presented in Table 4, GDELT includes a large number of news facts in the global world for two years, and the number of training data of GDELT is nearly 8 times higher than that of ICESW05-15. Thus we argue that the proposed model containing a large number of parameters can fit the training data better than the baseline models. (ii) Taking TNTComplEx as an example, through the experiment results of TNTComplEx trained on GDELT dataset presented in the right part of Fig. 2, we can find that the training of TComplEx and TNTComplEx reach a relatively steady state quickly. In addition, the test results on the training data and test data are low during 50 epochs training phase, thus we argue that TNTComplEx is underfitting on the GDELT. To a certain degree, these results verify our assumption that the baseline models including TNTComplEx may fail to fit the training data very well. On the contrary, the proposed model can be trained well on GDELT and obtain a promising result. Fourth, we observe that TuckERT obtains better performance than TuckERTNT on the GDELT dataset. The reasons of which are analyzed as follows. As described in the Dataset part and Table 4, GDELT includes a large number of news facts within two years, this indicates that

Table 5
The impact of time regularization and embedding regularization on TuckERTNT model trained on ICEWS2014 dataset. Best results are in bold.

TuckERTNT Loss	MRR	Hit@1	Hit@3	Hit@10
$\mathcal{L}(\theta)$	0.582	0.503	0.626	0.738
$\mathcal{L}(\theta)+\mathcal{R}_p(E)$	0.586	0.509	0.631	0.742
$\mathcal{L}(\theta)+\mathcal{R}_p(E,\mathcal{W})$	0.587	0.504	0.634	0.752
$\mathcal{L}(\theta)+\mathcal{L}_s(T)$	0.596	0.509	0.651	0.758
$\mathcal{L}(\theta)+\mathcal{L}_c(T)$	0.594	0.514	0.647	0.753
$\mathcal{L}(\theta)+\mathcal{L}_r(T)$	0.613	0.538	0.660	0.763
$\mathcal{L}(\theta)+\mathcal{L}_s(T)+\mathcal{R}_p(E)$	0.602	0.521	0.652	0.758
$\mathcal{L}(\theta)+\mathcal{L}_s(T)+\mathcal{R}_p(E,\mathcal{W})$	0.605	0.518	0.660	0.767
$\mathcal{L}(\theta)+\mathcal{L}_c(T)+\mathcal{R}_p(E)$	0.592	0.506	0.647	0.753
$\mathcal{L}(\theta)+\mathcal{L}_c(T)+\mathcal{R}_p(E,\mathcal{W})$	0.600	0.515	0.653	0.764
$\mathcal{L}(\theta)+\mathcal{L}_r(T)+\mathcal{R}_p(E)$	0.619	0.541	0.666	0.767
$\mathcal{L}(\theta)+\mathcal{L}_r(T)+\mathcal{R}_p(E,\mathcal{W})$	**0.625**	**0.544**	**0.673**	**0.773**

the facts in GDELT are updated very quickly, thus GDELT has a stronger temporal property than ICEWS2014 and ICEWS05-15. Therefore, TuckERT may be more able to model the facts in GDELT than TuckERTNT.

5.6. Ablation study

To further evaluate the proposed model, we study the impact of regularization and embedding dimensionality on our model, respectively. In addition, we also compare the training curve of our models with baseline models.

As presented in Table 5, we study the impact of different time regularization and embedding regularizations on the TuckERTNT model trained on the ICEWS2014. From the results, we also have the following observations and analyses. First, we can find the embedding regularization exert a positive impact on the proposed model, $\mathcal{R}_p(E)$ and $\mathcal{R}_p(E,\mathcal{W})$ contributes 0.4% and 1.4% to the basic model, respectively. These results verify our assumption that exerting a constraint on the core tensor has an appreciable impact on our model. Second, the proposed three kinds of time regularization improve the basic model explicitly, especially the reconstruction-based method, it helps the basic model improve by 2.5%. In addition, we also find that contrastive learning-based does not perform well as the other two time regularization methods. We argue that the constraint that the representations between the non-adjacent timestamps are dissimilar may be too strong, thus it fails to model the prior knowledge about timestamps very well. Third, when considering the time regularization and embedding regularization jointly, combining reconstruction-based time regularization and the embedding regularization containing the constraint on the core tensor with the basic model achieves the best performance.

5.6.1. Impact of regularization

5.6.2. Impact of embedding dimensionality

To evaluate the impact of embedding dimensionality on our model, we vary the embedding dimensionality of the entity and predicate from 32 to 400 and report MRR and Hits@n results of the TuckERTNT model on the GDELT dataset in Table 6. Fig. 1 presents the Hits@10 performance trend with respect to embedding dimensionality. From the results, we can conclude that the performance increases with embedding dimensionality growth until reaching a relatively steady state. While the higher embedding dimensionality means that the core tensor contains more parameters, thus leading to a sharp drop in efficiency while improving performance. Therefore, considering the efficiency and effectiveness of the proposed model jointly, the embedding dimensionality is eventually set to 300.

Table 6
The performance of TuckERTNT model with different embedding dimensionalities trained on the GDELT dataset.

Dimensionality	MRR	Hit@1	Hit@3	Hit@10
32	0.206	0.132	0.220	0.347
64	0.243	0.163	0.262	0.398
100	0.275	0.190	0.298	0.440
200	0.358	0.261	0.392	0.549
300	0.425	0.326	0.470	0.622
400	0.446	0.349	0.490	0.639

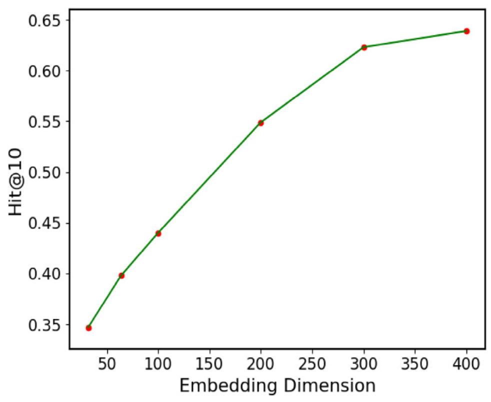


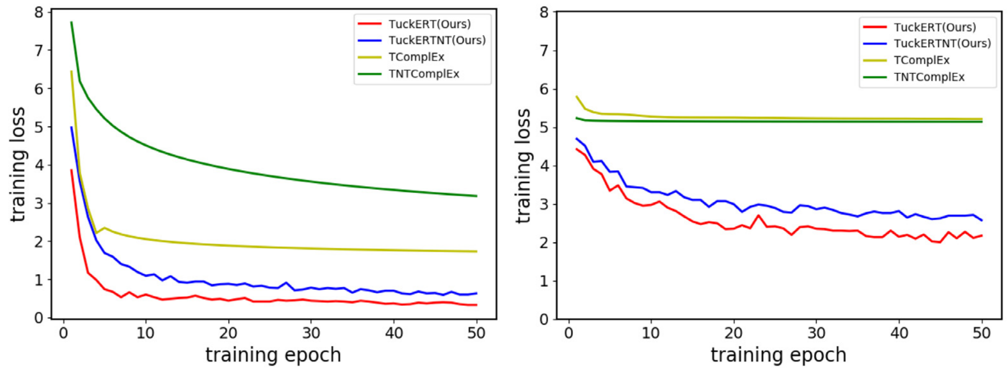
Fig. 1. The Hits@10 performance of TuckERTNT model with different embedding dimensionalities trained on the GDELT dataset.

5.6.3. Training curve

Fig. 2 shows the curve of the training loss for the proposed models TuckERT, TuckERTNT, and baseline models TComplEx, TNTComplEx on the ICEWS2014 and GDELT datasets. According to the results of the curve comparison, we have the following two observations: (1) The training loss (empirical loss) of the proposed models is lower than the baseline models TComplEx and TNTComplEx in the period of stabilization on both ICEWS2014 and GDELT datasets, which suggests that the proposed models fit the training data than baseline models TComplEx and TNTComplEx. This result can further show that the proposed model is more expressive. (2) On the GDELT dataset, the training of TComplEx and TNTComplEx reach a relatively steady state quickly, and the empirical error is relatively large in the steady state. Thus we argue that TComplEx and TNTComplEx do not fit the training data well on the GDELT. In addition, from the above training curve and the experiment results of TComplEx and TNTComplEx, we further argue that TComplEx and TNTComplEx are underfitting on the GDELT dataset.

6. Conclusion and outlook

Tensor decomposition has been widely used in knowledge completion tasks, either on static KGs or temporal. In this work, we developed a new decomposition model that bridges the gap between Tucker decomposition and temporal KGs completion from a generalized perspective. Based on this, to improve the basic model, we provide three methods including cosine similarity, contrastive learning, and reconstruction-based to incorporate the prior knowledge about timestamps into the proposed model. In addition, because the core tensor contains a large number of parameters on the proposed model, thus we present two embedding regularization schemes to avoid the overfitting problem. We proved that our method is fully expressive and achieves outstanding performance on three benchmarks compared with the existing state-of-the-art works. However, our models still



a : Models trained on the ICEWS2014

b : Models trained on the GDELT

Fig. 2. a is the training curve for the proposed models and baseline models on the ICEWS2014. b is the training curve for the proposed models and baseline models on the GDELT.

include more parameters compared with previously presented models and have restrictions on efficiency when the embedding dimensionality exceeds the threshold. Future work might design a lightweight but powerful model, further work might consider that exploring a new paradigm for temporal KGs completion.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

We gratefully acknowledge the help and assistance from Bin Liu, Xiao Wang, and Xinke Wang. Moreover, this work is supported by the National Key Research & Development Plan of China (No. 2017YFC0820602), the National Natural Science Foundation of China (NSFC) (No. 61831022, No. 61771472, No. 61773379, No. 61901473).

References

[1] Y. Koren, R. Bell, C. Volinsky, Matrix factorization techniques for recommender systems, Proc. Comput. (2009) 30–37.
[2] L. Dong, F. Wei, M. Zhou, K. Xu, Question answering over freebase with multi-column convolutional neural networks, in: Proceedings of the Annual Meeting of the Association for Computational Linguistics and the International Joint Conference on Natural Language Processing, 2015, pp. 260–269.
[3] C. Xiong, J.P. Callan, Query expansion with freebase, in: Proceedings of the International Conference on the Theory of Information Retrieval, 2015, pp. 111—120.
[4] Z. Wang, C. Chen, W. Li, Predictive network representation learning for link prediction, in: Proceedings of International ACM SIGIR Conference on Research and Development in Information Retrieval, 2017, pp. 969—972.
[5] T. Lacroix, N. Usunier, G. Obozinski, Canonical tensor decomposition for knowledge base completion, in: Proceedings of the International Conference on Machine Learning, 2018.
[6] I. Balažević, C. Allen, T.M. Hospedales, TuckER: Tensor factorization for knowledge graph completion, in: Proceedings of the Conference on Empirical Methods in Natural Language Processing and the International Joint Conference on Natural Language Processing, 2019, pp. 5185–5194.
[7] T. Lacroix, G. Obozinski, N. Usunier, Tensor decompositions for temporal knowledge base completion, in: Proceedings of International Conference on Learning Representations, 2020.
[8] J. Håstad, Tensor rank is NP-complete, J. Algorithms (1990) 644–654.
[9] P. Jain, S. Rathi, Mausam, S. Chakrabarti, Temporal knowledge base completion: New algorithms and evaluation protocols, in: Proceedings of the Conference on Empirical Methods in Natural Language Processing, 2020, pp. 3733–3747.
[10] B. Yang, S.W.-t. Yih, X. He, J. Gao, L. Deng, Embedding entities and relations for learning and inference in knowledge bases, in: Proceedings of the International Conference on Learning Representations, 2015.
[11] T. Trouillon, J. Welbl, S. Riedel, E. Gaussier, G. Bouchard, Complex embeddings for simple link prediction, in: Proceedings of the International Conference on Machine Learning, 2016, pp. 2071—2080.
[12] S.M. Kazemi, D. Poole, SimplE embedding for link prediction in knowledge graphs, in: Proceedings of the International Conference on Neural Information Processing Systems, 2018, pp. 4289—4300.
[13] R. Goel, S.M. Kazemi, M. Brubaker, P. Poupart, Diachronic embedding for temporal knowledge graph completion, in: Proceedings of AAAI Conference on Artificial Intelligence, 2020, pp. 3988–3995.
[14] A. Bordes, N. Usunier, A. Garcia-Durán, J. Weston, O. Yakhnenko, Translating embeddings for modeling multi-relational data, in: Proceedings of the International Conference on Neural Information Processing Systems, 2013, pp. 2787—2795.
[15] G. Ji, S. He, L. Xu, K. Liu, J. Zhao, Knowledge graph embedding via dynamic mapping matrix, in: Proceedings of Annual Meeting of the Association for Computational Linguistics, 2015.
[16] T. Dettmers, M. Pasquale, S. Pontus, S. Riedel, Convolutional 2D knowledge graph embeddings, in: Proceedings of AAAI Conference on Artificial Intelligence, 2018, pp. 1811–1818.
[17] M. Schlichtkrull, T.N. Kipf, P. Bloem, R.v.d. Berg, I. Titov, M. Welling, Modeling relational data with graph convolutional networks, in: Proc. Semant. Web, 2017, pp. 593–607.
[18] M. Nickel, V. Tresp, H.-P. Kriegel, A three-way model for collective learning on multi-relational data, in: Proceedings of the International Conference on Machine Learning, 2011, pp. 809—816.
[19] M. Nickel, L. Rosasco, T. Poggio, Holographic embeddings of knowledge graphs, in: Proceedings of AAAI Conference on Artificial Intelligence, 2016, pp. 1955—1961.
[20] T. Jiang, T. Liu, T. Ge, L. Sha, S. Li, B. Chang, Z. Sui, Encoding temporal information for time-aware link prediction, in: Proceedings of the Conference on Empirical Methods in Natural Language Processing, 2016, pp. 2350–2354.
[21] S.S. Dasgupta, S.N. Ray, P. Talukdar, HyTE: Hyperplane-based temporally aware knowledge graph embedding, in: Proceedings of the Conference on Empirical Methods in Natural Language Processing, 2018, pp. 2001–2011.
[22] Z. Wang, J. Zhang, J. Feng, Z. Chen, Knowledge graph embedding by translating on hyperplanes, in: Proceedings of AAAI Conference on Artificial Intelligence, 2014, pp. 1112—1119.
[23] A. García-Durán, S. Dumancic, M. Niepert, Learning sequence encoders for temporal knowledge graph completion, in: Proceedings of the Conference on Empirical Methods in Natural Language Processing, 2018, pp. 4816—4821.
[24] Y. Ma, V. Tresp, E. Daxberger, Embedding models for episodic knowledge graphs, Proc. J. Web Semant. (2019).
[25] C. Xu, M. Nayyeri, F. Alkhoury, H. Yazdi, J. Lehmann, Temporal knowledge graph completion based on time series gaussian embedding, in: Proceedings of the Semantic Web, 2020, pp. 654–671.
[26] C. Xu, Y.-Y. Chen, M. Nayyeri, J. Lehmann, Temporal knowledge graph completion using a linear temporal regularizer and multivector embeddings, in: Proceedings of the Conference of the North American Chapter of the

Association for Computational Linguistics: Human Language Technologies, 2021, pp. 2569–2578.
[27] Y. Xu, H. E, M. Song, W. Song, X. Lv, H. Wang, J. Yang, RTFE: A recursive temporal fact embedding framework for temporal knowledge graph completion, in: NAACL-HLT, 2021.
[28] F.L. Hitchcock, The expression of a tensor or a polyadic as a sum of products, Proc. Stud. Appl. Math. (1927) 164–189.
[29] L.R. Tucker, The extension of factor analysis to three-dimensional matrices, in: Proceedings of Contributions to Mathematical Psychology, 1964, pp. 110–127.
[30] T. Kolda, B. Bader, Tensor decompositions and applications, 2009, pp. 455—500,
[31] E. Boschee, J. Lautenschlager, S. O'Brien, S. Shellman, J. Starz, M. Ward, ICEWS coded event data, 2015,
[32] K. Leetaru, P.A. Schrodt, GDELT: Global data on events, location, and tone, in: Proceedings of ISA Annual Convention.
[33] R. Trivedi, H. Dai, Y. Wang, L. Song, Know-Evolve: Deep temporal reasoning for dynamic knowledge graphs, in: Proceedings of the International Conference on Machine Learning, 2017, pp. 3462—3471.
[34] M. Nickel, K. Murphy, V. Tresp, E. Gabrilovich, A review of relational machine learning for knowledge graphs, 2016, pp. 11–33,
[35] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. Devito, Z. Lin, A. Desmaison, L. Antiga, A. Lerer, Automatic differentiation in PyTorch, in: Proceedings of the International Conference on Neural Information Processing Systems, 2017.
[36] J.C. Duchi, E. Hazan, Y. Singer, Adaptive subgradient methods for online learning and stochastic optimization, in: Proceedings of Machine Learning Research, 2011, pp. 2121—2159.