Diachronic Embedding for Temporal Knowledge Graph Completion

Rishab Goel,* **Seyed Mehran Kazemi,*** **Marcus Brubaker, Pascal Poupart**
Borealis AI
{rishab.goal, mehran.kazemi, marcus.brubaker, pascal.poupart}@borealisai.com

Abstract

Knowledge graphs (KGs) typically contain temporal facts indicating relationships among entities at different times. Due to their incompleteness, several approaches have been proposed to infer new facts for a KG based on the existing ones– a problem known as *KG completion*. KG embedding approaches have proved effective for KG completion, however, they have been developed mostly for static KGs. Developing temporal KG embedding models is an increasingly important problem. In this paper, we build novel models for temporal KG completion through equipping static models with a diachronic entity embedding function which provides the characteristics of entities at *any* point in time. This is in contrast to the existing temporal KG embedding approaches where only static entity features are provided. The proposed embedding function is model-agnostic and can be potentially combined with any static model. We prove that combining it with SimplE, a recent model for static KG embedding, results in a fully expressive model for temporal KG completion. Our experiments indicate the superiority of our proposal compared to existing baselines.

Introduction

Knowledge graphs (KGs) are directed graphs where nodes represent entities and (labeled) edges represent the types of relationships among entities. Each edge in a KG corresponds to a fact and can be represented as a tuple such as $(\mathsf{Mary}, \mathsf{Liked}, \mathsf{God\ Father})$ where Mary and God Father are called the head and tail entities respectively and Liked is a relation. An important problem, known as KG completion, is to infer new facts from a KG based on the existing ones. This problem has been extensively studied for static KGs (see (Nickel et al. 2016) for a summary). KG embedding approaches have offered state-of-the-art results for KG completion on several benchmarks. These approaches map each entity and each relation type to a hidden representation and compute a score for each tuple by applying a score function to these representations. Different approaches differ in how they map the entities and relation types to hidden representations and in their score functions.

To capture the temporal aspect of the facts, KG edges are typically associated with a timestamp or time interval; e.g., $(\mathsf{Mary}, \mathsf{Liked}, \mathsf{God\ Father}, 1995)$. However, KG embedding approaches have been mostly designed for static KGs ignoring the temporal aspect. Recent work has shown a substantial boost in performance by extending these approaches to utilize time (Dasgupta, Ray, and Talukdar 2018; Ma, Tresp, and Daxberger 2018; García-Durán, Dumančić, and Niepert 2018). The proposed extensions are mainly through computing a hidden representation for each timestamp and extending the score functions to utilize timestamp representations as well as entity and relation representations.

In this paper, we develop models for temporal KG completion (TKGC) based on an intuitive assumption: to provide a score for, e.g., $(\mathsf{Mary}, \mathsf{Liked}, \mathsf{God\ Father}, 1995)$, one needs to know Mary's and God Father's features on 1995; providing a score based on their current features may be misleading. That is because Mary's personality and the sentiment towards God Father may have been quite different on 1995 as compared to now. Consequently, learning a static representation for each entity – as is done by existing approaches – may be sub-optimal as such a representation only captures the entity features at the current time, or an aggregation of entity features during time.

To provide entity features at any given time, we define entity embedding as a function which takes an entity and a timestamp as input and provides a hidden representation for the entity at that time. Inspired by diachronic word embeddings, we call our proposed embedding *diachronic embedding (DE)*. DE is model-agnostic: any static KG embedding model can be potentially extended to TKGC by leveraging DE. We prove that combining DE with SimplE (Kazemi and Poole 2018b) results in a fully expressive model for TKGC. To the best of our knowledge, this is the first TKGC model with a proof of fully expressiveness. We show the merit of our model on subsets of ICEWS (Boschee et al. 2015) and GDELT (Leetaru and Schrodt 2013) datasets.

Background and Notation

Notation: Lower-case letters denote scalars, bold lower-case letters denote vectors, and bold upper-case letters denote matrices. $\boldsymbol{z}[n]$ represents the n^{th} element of a vec-

*Equal contribution.

tor $\boldsymbol{z}$, $||\boldsymbol{z}||$ represents its norm, and $\boldsymbol{z}^\intercal$ represents its transpose. $\boldsymbol{z}_1 \otimes \boldsymbol{z}_2$ represents a vector $\boldsymbol{z} \in \mathbb{R}^{d_1 d_2}$ such that $\boldsymbol{z}[(n-1) * d_2 + m] = \boldsymbol{z}_1[n] * \boldsymbol{z}_2[m]$ (i.e. the flattened vector of the tensor/outer product of the two vectors). For k vectors $\boldsymbol{z}_1, \ldots, \boldsymbol{z}_k$ of the same length d, $\langle \boldsymbol{z}_1, \ldots, \boldsymbol{z}_k \rangle = \sum_{n=1}^{d} (\boldsymbol{z}_1[n] * \cdots * \boldsymbol{z}_k[n])$ represents the sum of the element-wise product of the elements of the k vectors.

Temporal Knowledge Graph (Completion): Let $\mathcal{V}$ be a finite set of entities, $\mathcal{R}$ be a finite set of relation types, and $\mathcal{T}$ be a finite set of timestamps. Let $\mathcal{W} \subset \mathcal{V} \times \mathcal{R} \times \mathcal{V} \times \mathcal{T}$ represent the set of all temporal tuples $(\mathsf{v}, \mathsf{r}, \mathsf{u}, \mathsf{t})$ that are facts (i.e. true in a world), where $\mathsf{v}, \mathsf{u} \in \mathcal{V}$, $\mathsf{r} \in \mathcal{R}$, and $\mathsf{t} \in \mathcal{T}$. Let $\mathcal{W}^c$ be the complement of $\mathcal{W}$. A temporal knowledge graph (KG) $\mathcal{G}$ is a subset of $\mathcal{W}$ (i.e. $\mathcal{G} \subset \mathcal{W}$)[1]. Temporal KG completion (TKGC) is the problem of inferring $\mathcal{W}$ from $\mathcal{G}$.

Relation Properties: A relation r is *symmetric* if $(\mathsf{v}, \mathsf{r}, \mathsf{u}, \mathsf{t}) \in \mathcal{W} \iff (\mathsf{u}, \mathsf{r}, \mathsf{v}, \mathsf{t}) \in \mathcal{W}$ and *anti-symmetric* if $(\mathsf{v}, \mathsf{r}, \mathsf{u}, \mathsf{t}) \in \mathcal{W} \iff (\mathsf{u}, \mathsf{r}, \mathsf{v}, \mathsf{t}) \in \mathcal{W}^c$. A relation r_i is the inverse of another relation r_j if $(\mathsf{v}, \mathsf{r}_i, \mathsf{u}, \mathsf{t}) \in \mathcal{W} \iff (\mathsf{u}, \mathsf{r}_j, \mathsf{v}, \mathsf{t}) \in \mathcal{W}$. r_i entails r_j if $(\mathsf{v}, \mathsf{r}_i, \mathsf{u}, \mathsf{t}) \in \mathcal{W} \Rightarrow (\mathsf{v}, \mathsf{r}_j, \mathsf{u}, \mathsf{t}) \in \mathcal{W}$.

KG Embedding: Formally, we define an entity embedding as follows.

Definition 1. An *entity embedding*, $\texttt{EEMB} : \mathcal{V} \rightarrow \psi$, is a function that maps every entity $\mathsf{v} \in \mathcal{V}$ to a hidden representation in ψ where ψ is the class of non-empty tuples of vectors and/or matrices.

A *relation embedding* ($\texttt{REMB} : \mathcal{R} \rightarrow \psi$) is defined similarly. We refer to the hidden representation of an entity (relation) as the embedding of the entity (relation). A KG embedding model defines two things: 1- the EEMB and REMB functions, 2- a score function which takes EEMB and REMB as input and provides a score for a given tuple. The parameters of hidden representations are learned from data.

Existing Approaches

In this section, we describe the existing approaches for static and temporal KG completion that will be used in the rest of the paper. For further detail on temporal KG completion approaches, we refer the reader to a recent survey (Kazemi et al. 2019). We represent the score for a tuple by $\phi(.)$.

TransE (static) (Bordes et al. 2013): In TransE, $\texttt{EEMB}(\mathsf{v}) = (\boldsymbol{z}_\mathsf{v})$ for every $\mathsf{v} \in \mathcal{V}$ where $\boldsymbol{z}_\mathsf{v} \in \mathbb{R}^d$, $\texttt{REMB}(\mathsf{r}) = (\boldsymbol{z}_\mathsf{r})$ for every $\mathsf{r} \in \mathcal{R}$ where $\boldsymbol{z}_\mathsf{r} \in \mathbb{R}^d$, and $\phi(\mathsf{v}, \mathsf{r}, \mathsf{u}) = -||\boldsymbol{z}_\mathsf{v} + \boldsymbol{z}_\mathsf{r} - \boldsymbol{z}_\mathsf{u}||$.

DistMult (static) (Yang et al. 2015): Same EEMB and REMB as TransE but defining $\phi(\mathsf{v}, \mathsf{r}, \mathsf{u}) = \langle \boldsymbol{z}_\mathsf{v}, \boldsymbol{z}_\mathsf{r}, \boldsymbol{z}_\mathsf{u} \rangle$.

Tucker (static) (Tucker 1966): Same EEMB and REMB as TransE but defining $\phi(\mathsf{v}, \mathsf{r}, \mathsf{u}) = \langle \boldsymbol{w}, \boldsymbol{z}_\mathsf{v} \otimes \boldsymbol{z}_\mathsf{r} \otimes \boldsymbol{z}_\mathsf{u} \rangle$ where $\boldsymbol{w} \in \mathbb{R}^{d^3}$ is a weight vector shared for all tuples.

[1]In this paper, we mainly study temporal KGs where each fact has a single timestamp. The models we develop, however, can be extended to facts with time intervals through the approach introduced in Dasgupta, Ray, and Talukdar (2018). We leave a more sophisticated way of handling time intervals or dealing with missing temporal information as future work.

RESCAL (static) (Nickel, Tresp, and Kriegel 2011): Same EEMB as TransE but defining $\texttt{REMB}(\mathsf{r}) = (\boldsymbol{Z}_\mathsf{r})$ for every $\mathsf{r} \in \mathcal{R}$ where $\boldsymbol{Z}_\mathsf{r} \in \mathbb{R}^{d \times d}$, and defining $\phi(\mathsf{v}, \mathsf{r}, \mathsf{u}) = \boldsymbol{z}_\mathsf{v}^\intercal \boldsymbol{Z}_\mathsf{r} \boldsymbol{z}_\mathsf{u}$.

CP (static) (Hitchcock 1927): Same REMB as TransE but defining $\texttt{EEMB}(\mathsf{v}) = (\vec{\boldsymbol{z}}_\mathsf{v}, \overleftarrow{\boldsymbol{z}}_\mathsf{v})$ for every $\mathsf{v} \in \mathcal{V}$ where $\vec{\boldsymbol{z}}_\mathsf{v}, \overleftarrow{\boldsymbol{z}}_\mathsf{v} \in \mathbb{R}^d$. $\vec{\boldsymbol{z}}_\mathsf{v}$ is used when v is the head and $\overleftarrow{\boldsymbol{z}}_\mathsf{v}$ is used when v is the tail. In CP, $\phi(\mathsf{v}, \mathsf{r}, \mathsf{u}) = \langle \vec{\boldsymbol{z}}_\mathsf{v}, \boldsymbol{z}_\mathsf{r}, \overleftarrow{\boldsymbol{z}}_\mathsf{u} \rangle$. DistMult is a special case of CP where $\vec{\boldsymbol{z}}_\mathsf{v} = \overleftarrow{\boldsymbol{z}}_\mathsf{v}$ for every $\mathsf{v} \in \mathcal{V}$.

SimplE (static) (Kazemi and Poole 2018b): Noticing an information flow issue between the two vectors $\vec{\boldsymbol{z}}_\mathsf{v}$ and $\overleftarrow{\boldsymbol{z}}_\mathsf{v}$ of an entity v in CP, SimplE takes advantage of the inverse of the relations to address this issue. SimplE defines $\texttt{REMB}(\mathsf{r}) = (\vec{\boldsymbol{z}}_\mathsf{r}, \overleftarrow{\boldsymbol{z}}_\mathsf{r})$ for every $\mathsf{r} \in \mathcal{R}$, where $\vec{\boldsymbol{z}}_\mathsf{r}$ is used as in CP and $\overleftarrow{\boldsymbol{z}}_\mathsf{r} \in \mathbb{R}^d$ is considered the embedding of r^{-1}, the inverse of r. In SimplE, $\phi(\mathsf{v}, \mathsf{r}, \mathsf{u})$ is defined as the average of two CP scores: 1- $\langle \vec{\boldsymbol{z}}_\mathsf{v}, \vec{\boldsymbol{z}}_\mathsf{r}, \overleftarrow{\boldsymbol{z}}_\mathsf{u} \rangle$ corresponding to the score for $(\mathsf{v}, \mathsf{r}, \mathsf{u})$ and 2- $\langle \vec{\boldsymbol{z}}_\mathsf{u}, \overleftarrow{\boldsymbol{z}}_\mathsf{r}, \overleftarrow{\boldsymbol{z}}_\mathsf{v} \rangle$ corresponding to the score for $(\mathsf{u}, \mathsf{r}^{-1}, \mathsf{v})$.

TTransE (temporal) (Jiang et al. 2016): An extension of TransE by adding one more embedding function mapping timestamps to hidden representations: $\texttt{TEMB}(\mathsf{t}) = (\boldsymbol{z}_\mathsf{t})$ for every $\mathsf{t} \in \mathcal{T}$ where $\boldsymbol{z}_\mathsf{t} \in \mathbb{R}^d$. In TTransE, $\phi(\mathsf{v}, \mathsf{r}, \mathsf{u}, \mathsf{t}) = -||\boldsymbol{z}_\mathsf{v} + \boldsymbol{z}_\mathsf{r} + \boldsymbol{z}_\mathsf{t} - \boldsymbol{z}_\mathsf{u}||$.

HyTE (temporal) (Dasgupta, Ray, and Talukdar 2018): Same EEMB, REMB and TEMB as TTransE but defining $\phi(\mathsf{v}, \mathsf{r}, \mathsf{u}, \mathsf{t}) = -||\overline{\boldsymbol{z}}_\mathsf{v} + \overline{\boldsymbol{z}}_\mathsf{r} - \overline{\boldsymbol{z}}_\mathsf{u}||$ where $\overline{\boldsymbol{z}}_\mathsf{x} = \boldsymbol{z}_\mathsf{x} - \boldsymbol{z}_\mathsf{t}^\intercal \boldsymbol{z}_\mathsf{x} \boldsymbol{z}_\mathsf{t}$ for $\mathsf{x} \in \{\mathsf{v}, \mathsf{r}, \mathsf{u}\}$. Intuitively, HyTE first projects the head, relation, and tail embeddings to the space of the timestamp and then applies TransE on the projected embeddings.

ConT (temporal) (Ma, Tresp, and Daxberger 2018): Ma, Tresp, and Daxberger extend several static KG embedding models to TKGC. Their best performing model, ConT, is an extension of Tucker defining $\texttt{TEMB}(\mathsf{t}) = (\boldsymbol{z}_\mathsf{t})$ for every $\mathsf{t} \in \mathcal{T}$ where $\boldsymbol{z}_\mathsf{t} \in \mathbb{R}^{d^3}$ and changing the score function to $\phi(\mathsf{v}, \mathsf{r}, \mathsf{u}, \mathsf{t}) = \langle \boldsymbol{z}_\mathsf{t}, \boldsymbol{z}_\mathsf{v} \otimes \boldsymbol{z}_\mathsf{r} \otimes \boldsymbol{z}_\mathsf{u} \rangle$. Intuitively, ConT replaces the vector $\boldsymbol{w}$ in Tucker with timestamp embeddings $\boldsymbol{z}_\mathsf{t}$.

TA-DistMult (temporal) (García-Durán, Dumančić, and Niepert 2018): An extension of DistMult where each character c in the timestamps is mapped to a vector ($\texttt{CEMB}(c) = \boldsymbol{z}_c$) where $\boldsymbol{z}_c \in \mathbb{R}^d$. Then, for a tuple $(\mathsf{v}, \mathsf{r}, \mathsf{u}, \mathsf{t})$, a temporal relation is created by considering r and the characters in t as a sequence and an embedding $\boldsymbol{z}_{\mathsf{r},\mathsf{t}}$ is computed for this temporal relation by feeding the embedding vectors for each element in the sequence to an LSTM and taking its final output. Finally, the score function of DistMult is employed: $\phi(\mathsf{v}, \mathsf{r}, \mathsf{u}, \mathsf{t}) = \langle \boldsymbol{z}_\mathsf{v}, \boldsymbol{z}_{\mathsf{r},\mathsf{t}}, \boldsymbol{z}_\mathsf{u} \rangle$ (TransE was employed as well but DistMult performed better).

Diachronic Embedding

According to Definition 1, an entity embedding function takes an entity as input and provides a hidden representation as output. We propose an alternative entity embedding function which, besides entity, takes time as input as well. Inspired by diachronic word embeddings, we call such an embedding function a *diachronic entity embedding*. Below is a formal definition of a diachronic entity embedding.

Definition 2. A *diachronic entity embedding*, $\texttt{DEEMB} : (\mathcal{V}, \mathcal{T}) \rightarrow \psi$, is a function that maps every pair (v, t), where

$\mathsf{v} \in \mathcal{V}$ and $\mathsf{t} \in \mathcal{T}$, to a hidden representation in ψ where ψ is the class of non-empty tuples of vectors and/or matrices.

One may take their favorite static KG embedding score function and make it temporal by replacing their entity embeddings with diachronic entity embeddings. The choice of the DEEMB function can be different for various temporal KGs depending on their properties. Here, we propose a DEEMB function which performs well on our benchmarks. We give the definition for models where the output of the DEEMB function is a tuple of vectors but it can be generalized to other cases as well. Let $\boldsymbol{z}_{\mathsf{v}}^{\mathsf{t}} \in \mathbb{R}^d$ be a vector in $\texttt{DEEMB}(\mathsf{v}, \mathsf{t})$ (i.e. $\texttt{DEEMB}(\mathsf{v}, \mathsf{t}) = (\dots, \boldsymbol{z}_{\mathsf{v}}^{\mathsf{t}}, \dots)$). We define $\boldsymbol{z}_{\mathsf{v}}^{\mathsf{t}}$ as follows:

$$\boldsymbol{z}_{\mathsf{v}}^{\mathsf{t}}[n] = \begin{cases} \boldsymbol{a}_{\mathsf{v}}[n]\sigma(\boldsymbol{w}_{\mathsf{v}}[n]\mathsf{t} + \boldsymbol{b}_{\mathsf{v}}[n]), & \text{if } 1 \leq n \leq \gamma d. \\ \boldsymbol{a}_{\mathsf{v}}[n], & \text{if } \gamma d < n \leq d. \end{cases} \quad (1)$$

where $\boldsymbol{a}_{\mathsf{v}} \in \mathbb{R}^d$ and $\boldsymbol{w}_{\mathsf{v}}, \boldsymbol{b}_{\mathsf{v}} \in \mathbb{R}^{\gamma d}$ are (entity-specific) vectors with learnable parameters and σ is an activation function. Intuitively, entities may have some features that change over time and some features that remain fixed. The first γd elements of the vector in Equation (1) capture temporal features and the other $(1-\gamma)d$ elements capture static features. $0 \leq \gamma \leq 1$ is a hyper-parameter controlling the percentage of temporal features. While in Equation (1) static features can be potentially obtained from the temporal ones if the optimizer sets some elements of $\boldsymbol{w}_{\mathsf{v}}$ to zero, explicitly modeling static features helps reduce the number of learnable parameters and avoid overfitting to temporal signals (see ablation studies).

Intuitively, by learning $\boldsymbol{w}_{\mathsf{v}}$s and $\boldsymbol{b}_{\mathsf{v}}$s, the model learns how to turn entity features on and off at different points in time so accurate temporal predictions can be made about them at any time. $\boldsymbol{a}_{\mathsf{v}}$s control the importance of the features. We mainly use *sine* as the activation function for Equation (1) because one sine function can model several on and off states. Our experiments explore other activation functions as well and provide more intuition.

Model-Agnosticism: The proposals in existing temporal KG embedding models can only extend one (or a few) static models to temporal KGs. As an example, it is not trivial how RESCAL can be extended to temporal KGs using the proposal in (García-Durán, Dumančić, and Niepert 2018) (except for the naive approach of expecting the LSTM to output large $\boldsymbol{Z}_{\mathsf{r}}$ matrices) or in (Jiang et al. 2016; Dasgupta, Ray, and Talukdar 2018). Same goes for models other than RESCAL where the relation embeddings contain matrices (see, e.g., (Nguyen et al. 2016; Socher et al. 2013; Lin et al. 2015)). Using our proposal, one may construct temporal versions of TransE, DistMult, SimplE, Tucker, RESCAL, or other models by replacing their EEMB function with DEEMB in Equation (1). We refer to the resulting models as *DE-TransE*, *DE-DistMult*, *DE-SimplE* and so forth, where *DE* is short for **D**iachronic **E**mbedding.

Learning: The facts in a KG $\mathcal{G}$ are split into $train$, $validation$, and $test$ sets. Model parameters are learned using stochastic gradient descent with mini-batches. Let $B \subset train$ be a mini-batch. For each fact $f = (\mathsf{v}, \mathsf{r}, \mathsf{u}, \mathsf{t}) \in B$, we generate two queries: 1- $(\mathsf{v}, \mathsf{r}, ?, \mathsf{t})$ and 2- $(?, \mathsf{r}, \mathsf{u}, \mathsf{t})$. For the first query, we generate a candidate answer set $\mathsf{C}_{f,\mathsf{v}}$ which contains v and n (hereafter referred to as negative ratio) other entities selected randomly from $\mathcal{V}$. For the second query, we generate a similar candidate answer set $\mathsf{C}_{f,\mathsf{u}}$. Then we minimize the cross entropy loss which has been used and shown good results for both static and temporal KG completion (see, e.g., (Kadlec, Bajgar, and Kleindienst 2017; García-Durán, Dumančić, and Niepert 2018)):

$$\mathcal{L} = -\Big(\sum_{f=(\mathsf{v},\mathsf{r},\mathsf{u},\mathsf{t})\in B} \frac{\exp(\phi(\mathsf{v},\mathsf{r},\mathsf{u},\mathsf{t}))}{\sum_{\mathsf{u}' \in \mathsf{C}_{f,\mathsf{u}}} \exp(\phi(\mathsf{v},\mathsf{r},\mathsf{u}',\mathsf{t}))} + \frac{\exp(\phi(\mathsf{v},\mathsf{r},\mathsf{u},\mathsf{t}))}{\sum_{\mathsf{v}' \in \mathsf{C}_{f,\mathsf{v}}} \exp(\phi(\mathsf{v}',\mathsf{r},\mathsf{u},\mathsf{t}))} \Big)$$

Expressivity

Expressivity is an important property and has been the subject of study in several recent works on static (knowledge) graphs (Trouillon et al. 2017; Kazemi and Poole 2018b; Xu et al. 2019; Fatemi, Ravanbakhsh, and Poole 2019). If a model is not expressive enough, it is doomed to underfitting for some applications. A desired property of a model is fully expressiveness:

Definition 3. A model with parameters θ is *fully expressive* if given any world with true tuples $\mathcal{W}$ and false tuples $\mathcal{W}^c$, there exists an assignment for θ that correctly classifies the tuples in $\mathcal{W}$ and $\mathcal{W}^c$.

For static KG completion, several models have been proved to be fully expressive. For TKGC, however, a proof of fully expressiveness does not yet exist for the proposed models. The following theorem establishes the fully expressiveness of DE-SimplE.

Theorem 1 (**Expressivity**). *DE-SimplE is fully expressive for temporal knowledge graph completion.*

Proof. The embedding functions of SimplE map each entity to two vectors and each relation also to two vectors. For every entity $\mathsf{v}_i \in \mathcal{V}$, let $\texttt{DEEMB}(\mathsf{v}_i, \mathsf{t}) = (\vec{\boldsymbol{z}}_{\mathsf{v}_i}^{\mathsf{t}}, \overleftarrow{\boldsymbol{z}}_{\mathsf{v}_i}^{\mathsf{t}})$ where, according to Equation (1) with sine activations, $\vec{\boldsymbol{z}}_{\mathsf{v}_i}^{\mathsf{t}} \in \mathbb{R}^d$ and $\overleftarrow{\boldsymbol{z}}_{\mathsf{v}_i}^{\mathsf{t}} \in \mathbb{R}^d$ are defined as follows:

$$\vec{\boldsymbol{z}}_{\mathsf{v}_i}^{\mathsf{t}}[n] = \begin{cases} \vec{\boldsymbol{a}}_{\mathsf{v}_i}[n]\sin(\vec{\boldsymbol{w}}_{\mathsf{v}_i}[n]t + \vec{\boldsymbol{b}}_{\mathsf{v}_i}[n]), & \text{if } n \leq \gamma d. \\ \vec{\boldsymbol{a}}_{\mathsf{v}_i}[n], & \text{if } n > \gamma d. \end{cases} \quad (2)$$

$$\overleftarrow{\boldsymbol{z}}_{\mathsf{v}_i}^{\mathsf{t}}[n] = \begin{cases} \overleftarrow{\boldsymbol{a}}_{\mathsf{v}_i}[n]\sin(\overleftarrow{\boldsymbol{w}}_{\mathsf{v}_i}[n]t + \overleftarrow{\boldsymbol{b}}_{\mathsf{v}_i}[n]), & \text{if } n \leq \gamma d. \\ \overleftarrow{\boldsymbol{a}}_{\mathsf{v}_i}[n], & \text{if } n > \gamma d. \end{cases} \quad (3)$$

where $\vec{\boldsymbol{a}}_{\mathsf{v}_i} \in \mathbb{R}^d$ and $\vec{\boldsymbol{w}}_{\mathsf{v}_i}, \vec{\boldsymbol{b}}_{\mathsf{v}_i} \in \mathbb{R}^{\gamma d}$ are used when v_i is the head and $\overleftarrow{\boldsymbol{a}}_{\mathsf{v}_i} \in \mathbb{R}^d$ and $\overleftarrow{\boldsymbol{w}}_{\mathsf{v}_i}, \overleftarrow{\boldsymbol{b}}_{\mathsf{v}_i} \in \mathbb{R}^{\gamma d}$ are used when v_i is the tail of a relation. We provide the proof for a specific case of DE-SimplE where the elements of $\vec{\boldsymbol{z}}_{\mathsf{v}}^{\mathsf{t}}$s are all temporal and the elements of $\overleftarrow{\boldsymbol{z}}_{\mathsf{v}}^{\mathsf{t}}$s are all non-temporal. This specific case can be achieved by setting $\gamma = d$, and $\overleftarrow{\boldsymbol{w}}_{\mathsf{v}}[n] = 0$ and $\overleftarrow{\boldsymbol{b}}_{\mathsf{v}}[n] = \frac{\pi}{2}$ for all $\mathsf{v} \in \mathcal{V}$ and for all $1 \leq n \leq d$. If this specific case of DE-SimplE is fully expressive, so is DE-SimplE. In this specific case, $\vec{\boldsymbol{z}}_{\mathsf{v}_i}^{\mathsf{t}}$ and $\overleftarrow{\boldsymbol{z}}_{\mathsf{v}_i}^{\mathsf{t}}$ can be re-written as follows:

$$\vec{\boldsymbol{z}}_{\mathsf{v}_i}^{\mathsf{t}}[n] = \vec{\boldsymbol{a}}_{\mathsf{v}_i}[n]\sin(\vec{\boldsymbol{w}}_{\mathsf{v}_i}[n]t + \vec{\boldsymbol{b}}_{\mathsf{v}_i}[n]) \quad (4)$$

$$\overleftarrow{\boldsymbol{z}}_{\mathsf{v}_i}^{\mathsf{t}}[n] = \overleftarrow{\boldsymbol{a}}_{\mathsf{v}_i}[n] \quad (5)$$

Recall that in SimplE, $\texttt{REMB}(\mathsf{r}_j) = (\vec{\boldsymbol{z}}_{\mathsf{r}_j}, \overleftarrow{\boldsymbol{z}}_{\mathsf{r}_j})$ for every relation $\mathsf{r}_j \in \mathcal{R}$. To further simplify the proof, following (Kazemi and Poole 2018b), we only show how the embedding values can be set such that $\langle \vec{\boldsymbol{z}}^{\,\mathsf{t}}_{\mathsf{v}_i}, \vec{\boldsymbol{z}}_{\mathsf{r}_j}, \overleftarrow{\boldsymbol{z}}^{\,\mathsf{t}}_{\mathsf{v}_k} \rangle$ becomes a positive number if $(\mathsf{v}_i, \mathsf{r}_j, \mathsf{v}_k, \mathsf{t}) \in \mathcal{W}$ and a negative number if $(\mathsf{v}_i, \mathsf{r}_j, \mathsf{v}_k, \mathsf{t}) \in \mathcal{W}^c$. Extending the proof to the case where the score contains both components ($\langle \vec{\boldsymbol{z}}^{\,\mathsf{t}}_{\mathsf{v}_i}, \vec{\boldsymbol{z}}_{\mathsf{r}_j}, \overleftarrow{\boldsymbol{z}}^{\,\mathsf{t}}_{\mathsf{v}_k} \rangle$ and $\langle \vec{\boldsymbol{z}}^{\,\mathsf{t}}_{\mathsf{v}_k}, \overleftarrow{\boldsymbol{z}}_{\mathsf{r}_j}, \overleftarrow{\boldsymbol{z}}^{\,\mathsf{t}}_{\mathsf{v}_i} \rangle$) can be done by doubling the size of the embedding vectors and following a similar procedure as the one explained below for the second half of the vectors.

Assume $d = |\mathcal{R}| \cdot |\mathcal{V}| \cdot |\mathcal{T}| \cdot L$ where $L \in \mathbb{N}$ is a natural number. These vectors can be viewed as $|\mathcal{R}|$ blocks of size $|\mathcal{V}| \cdot |\mathcal{T}| \cdot L$. For the j^{th} relation r_j, let $\vec{\boldsymbol{z}}_{\mathsf{r}_j}$ be zero everywhere except on the j^{th} block where it is 1 everywhere. With such a value assignment to $\vec{\boldsymbol{z}}_{\mathsf{r}_j}$s, to find the score for a fact $(\mathsf{v}_i, \mathsf{r}_j, \mathsf{v}_k, \mathsf{t})$, only the j^{th} block of each embedding vector is important. Let us now focus on the j^{th} block.

The size of the j^{th} block (similar to all other blocks) is $|\mathcal{V}| \cdot |\mathcal{T}| \cdot L$ and it can be viewed as $|\mathcal{V}|$ sub-blocks of size $|\mathcal{T}| \cdot L$. For the i^{th} entity v_i, let the values of $\vec{\boldsymbol{a}}_{\mathsf{v}_i}$ be zero in all sub-blocks except the i^{th} sub-block. With such a value assignment, to find the score for a fact $(\mathsf{v}_i, \mathsf{r}_j, \mathsf{v}_k, \mathsf{t})$, only the i^{th} sub-block of the j^{th} block is important. Note that this sub-block is unique for each tuple $(\mathsf{v}_i, \mathsf{r}_j)$. Let us now focus on the i^{th} sub-block of the j^{th} block.

The size of the i^{th} sub-block of the j^{th} block is $|\mathcal{T}| \cdot L$ and it can be viewed as $|\mathcal{T}|$ sub-sub-blocks of size L. According to the Fourier sine series, with a large enough L, we can set the values for $\vec{\boldsymbol{a}}_{\mathsf{v}_i}$, $\vec{\boldsymbol{w}}_{\mathsf{v}_i}$, and $\vec{\boldsymbol{b}}_{\mathsf{v}_i}$ in a way that the sum of the elements of $\vec{\boldsymbol{z}}^{\,\mathsf{t}}_{\mathsf{v}_i}$ for the p^{th} sub-sub-block becomes 1 when $\mathsf{t} = \mathsf{t}_p$ (where t_p is the p^{th} timestamp in $\mathcal{T}$) and 0 when t is a timestamp other than t_p. Note that this sub-sub-block is unique for each tuple $(\mathsf{v}_i, \mathsf{r}_j, \mathsf{t}_p)$.

Having the above value assignments, if $(\mathsf{v}_i, \mathsf{r}_j, \mathsf{v}_k, \mathsf{t}_p) \in \mathcal{W}$, we set all the values in the p^{th} sub-sub-block of the i^{th} sub-block of the j^{th} block of $\overleftarrow{\boldsymbol{a}}_{\mathsf{v}_k}$ to 1. With this assignment, $\langle \vec{\boldsymbol{z}}^{\,\mathsf{t}}_{\mathsf{v}_i}, \vec{\boldsymbol{z}}_{\mathsf{r}_j}, \overleftarrow{\boldsymbol{z}}^{\,\mathsf{t}}_{\mathsf{v}_k} \rangle = 1$ at $\mathsf{t} = \mathsf{t}_p$. If $(\mathsf{v}_i, \mathsf{r}_j, \mathsf{v}_k, \mathsf{t}_p) \in \mathcal{W}^c$, we set all the values for the p^{th} sub-sub-block of the i^{th} sub-block of the j^{th} block of $\overleftarrow{\boldsymbol{a}}_{\mathsf{v}_k}$ to -1. With this assignment, $\langle \vec{\boldsymbol{z}}^{\,\mathsf{t}}_{\mathsf{v}_i}, \vec{\boldsymbol{z}}_{\mathsf{r}_j}, \overleftarrow{\boldsymbol{z}}^{\,\mathsf{t}}_{\mathsf{v}_k} \rangle = -1$ at $\mathsf{t} = \mathsf{t}_p$. □

Time Complexity

Extending a static model to temporal KGs through our diachronic embedding does not change its time complexity. Therefore, making a single prediction in DE-TransE, DE-DistMult and DE-SimplE has a time complexity of $O(d)$ where d is the size of the embeddings. As for the existing approaches, TTransE and HyTE also have a time complexity of $O(d)$. ConT has a time complexity of $O(d^3)$ due to its timestamp embeddings and TA-DistMult has a time complexity of $O(d^2)$ due to the internal LSTM operations.

Domain Knowledge

For several static KG embedding models, it has been shown how certain types of domain knowledge (if exists) can be incorporated into the embeddings through parameter sharing (aka tying) and how it helps improve model performance (see, e.g., (Kazemi and Poole 2018b; Sun et al. 2019; Minervini et al. 2017; Fatemi, Ravanbakhsh, and Poole 2019)). Incorporating domain knowledge for these static models can be ported to their temporal version when they are extended to temporal KGs through our diachronic embeddings. As a proof of concept, we show how incorporating domain knowledge into SimplE can be ported to DE-SimplE. We chose SimplE for our proof of concept as several types of domain knowledge can be incorporated into it.

Consider $\mathsf{r}_i \in \mathcal{R}$ with $\texttt{REMB}(\mathsf{r}_i) = (\vec{\boldsymbol{z}}_{\mathsf{r}_i}, \overleftarrow{\boldsymbol{z}}_{\mathsf{r}_i})$ (according to SimplE). If r_i is known to be symmetric or anti-symmetric, this knowledge can be incorporated into the embeddings by tying $\vec{\boldsymbol{z}}_{\mathsf{r}_i}$ to $\overleftarrow{\boldsymbol{z}}_{\mathsf{r}_i}$ or negation of $\overleftarrow{\boldsymbol{z}}_{\mathsf{r}_i}$ respectively. If r_i is known to be the inverse of r_j, this knowledge can be incorporated into the embeddings by tying $\vec{\boldsymbol{z}}_{\mathsf{r}_i}$ to $\overleftarrow{\boldsymbol{z}}_{\mathsf{r}_j}$ and $\vec{\boldsymbol{z}}_{\mathsf{r}_j}$ to $\overleftarrow{\boldsymbol{z}}_{\mathsf{r}_i}$.

Proposition 1. *Symmetry, anti-symmetry, and inversion can be incorporated into DE-SimplE in the same way as SimplE.*

If r_i is known to entail r_j, Fatemi, Ravanbakhsh, and Poole (2019) prove that if entity embeddings are constrained to be non-negative, then this knowledge can be incorporated by tying $\vec{\boldsymbol{z}}_{\mathsf{r}_j}$ to $\vec{\boldsymbol{z}}_{\mathsf{r}_i} + \vec{\boldsymbol{\delta}}_{\mathsf{r}_j}$ and $\overleftarrow{\boldsymbol{z}}_{\mathsf{r}_j}$ to $\overleftarrow{\boldsymbol{z}}_{\mathsf{r}_i} + \overleftarrow{\boldsymbol{\delta}}_{\mathsf{r}_j}$ where $\vec{\boldsymbol{\delta}}_{\mathsf{r}_j}$ and $\overleftarrow{\boldsymbol{\delta}}_{\mathsf{r}_j}$ are vectors with non-negative elements. We give a similar result for DE-SimplE.

Proposition 2. *By constraining $\boldsymbol{a}_{\mathsf{v}}$s in Equation* (1) *to be non-negative for all $\mathsf{v} \in \mathcal{V}$ and σ to be an activation function with a non-negative range (such as ReLU, sigmoid, or squared exponential), entailment can be incorporated into DE-SimplE in the same way as SimplE.*

Compared to the result in (Fatemi, Ravanbakhsh, and Poole 2019), the only added constraint for DE-SimplE is that the activation function in Equation (1) is also constrained to have a non-negative range. The proofs of Propositions 1 and 2 can be found in our extended version at: https://arxiv.org/abs/1907.03143.

Experiments & Results

Datasets: Our datasets are subsets of two temporal KGs that have become standard benchmarks for TKGC: ICEWS (Boschee et al. 2015) and GDELT (Leetaru and Schrodt 2013). For ICEWS, we use the two subsets generated by (García-Durán, Dumančić, and Niepert 2018): 1- *ICEWS14* corresponding to the facts in 2014 and 2- *ICEWS05-15* corresponding to the facts between 2005 to 2015. For GDELT, we use the subset extracted by (Trivedi et al. 2017) corresponding to the facts from April 1, 2015 to March 31, 2016. We changed the train/validation/test sets following a similar procedure as in (Bordes et al. 2013) to make the problem into a TKGC rather than an extrapolation problem. Table 1 provides a summary of the dataset statistics.

Baselines: Our baselines include both static and temporal KG embedding models. From the static KG embedding models, we use TransE and DistMult and SimplE where the timing information are ignored. From the temporal KG embedding models, we use the ones introduced in the Existing Approaches section.

Table 1: Statistics on ICEWS14, ICEWS05-15, and GDELT.

Dataset	$\|\mathcal{V}\|$	$\|\mathcal{R}\|$	$\|\mathcal{T}\|$	$\|train\|$	$\|validation\|$	$\|test\|$	$\|\mathcal{G}\|$
ICEWS14	7,128	230	365	72,826	8,941	8,963	90,730
ICEWS05-15	10,488	251	4017	386,962	46,275	46,092	479,329
GDELT	500	20	366	2,735,685	341,961	341,961	3,419,607

Metrics: For each fact $f = (\mathsf{v}, \mathsf{r}, \mathsf{u}, \mathsf{t}) \in test$, we create two queries: 1- $(\mathsf{v}, \mathsf{r}, ?, \mathsf{t})$ and 2- $(?, \mathsf{r}, \mathsf{u}, \mathsf{t})$. For the first query, the model ranks all entities in $\mathsf{u} \cup \overline{\mathsf{C}}_{f,\mathsf{u}}$ where $\overline{\mathsf{C}}_{f,\mathsf{u}} = \{\mathsf{u}' : \mathsf{u}' \in \mathcal{V}, (\mathsf{v}, \mathsf{r}, \mathsf{u}', \mathsf{t}) \notin \mathcal{G}\}$. This corresponds to the filtered setting commonly used in the literature (Bordes et al. 2013). We follow a similar approach for the second query. Let $k_{f,\mathsf{u}}$ and $k_{f,\mathsf{v}}$ represent the ranking for u and v for the two queries respectively. We report *mean reciprocal rank (MRR)* defined as $\frac{1}{2*|test|}\sum_{f=(\mathsf{v},\mathsf{r},\mathsf{u},\mathsf{t})\in test}(\frac{1}{k_{f,\mathsf{u}}} + \frac{1}{k_{f,\mathsf{v}}})$. Compared to its counterpart *mean rank* which is largely influenced by a single bad prediction, MRR is more stable (Nickel et al. 2016). We also report Hit@1, Hit@3 and Hit@10 measures where Hit@k is defined as $\frac{1}{2*|test|}\sum_{f=(\mathsf{v},\mathsf{r},\mathsf{u},\mathsf{t})\in test}(\mathbb{1}_{k_{f,\mathsf{u}}\leq k} + \mathbb{1}_{k_{f,\mathsf{v}}\leq k})$, where $\mathbb{1}_{cond}$ is 1 if $cond$ holds and 0 otherwise.

Implementation[2] We implemented our model and the baselines in PyTorch (Paszke et al. 2017). We ran our experiments on a node with four GPUs. For the two ICEWS datasets, we report the results for some of the baselines from García-Durán, Dumančić, and Niepert (2018). For the other experiments on these datasets, for the fairness of results, we follow a similar experimental setup as in (García-Durán, Dumančić, and Niepert 2018) by using the ADAM optimizer (Kingma and Ba 2014) and setting learning rate $= 0.001$, batch size $= 512$, negative ratio $= 500$, embedding size $= 100$, and validating every 20 epochs selecting the model giving the best validation MRR. Following the best results obtained in (Ma, Tresp, and Daxberger 2018) (and considering the memory restrictions), for ConT we set embedding size $= 40$, batch size $= 32$ on ICEWS14 and GDELT and 16 on ICEWS05-15. We validated dropout values from $\{0.0, 0.2, 0.4\}$. We tuned γ for our model from the values $\{16, 32, 64\}$. For GDELT, we used a similar setting but with a negative ratio $= 5$ due to the large size of the dataset. Unless stated otherwise, we use $sine$ as the activation function for Equation (1). Since the timestamps in our datasets are dates rather than single numbers, we apply the temporal part of Equation (1) to year, month, and day separately (with different parameters) thus obtaining three temporal vectors. Then we take an element-wise sum of the resulting vectors obtaining a single temporal vector. Intuitively, this can be viewed as converting a date into a timestamp in the embedded space.

Comparative Study

We compare the baselines with three variants of our model: 1- DE-TransE, 2- DE-DistMult, and 3- DE-SimplE. The obtained results in Table 2 indicate that the large number of parameters per timestamp makes ConT perform poorly on ICEWS14 and ICEWS05-15. On GDELT, it shows a somewhat better performance as GDELT has many training facts in each timestamp. Besides affecting the predictive performance, the large number of parameters makes training ConT extremely slow. According to the results, the temporal versions of different models outperform the static counterparts in most cases, thus providing evidence for the merit of capturing temporal information.

DE-TransE outperforms the other TransE-based baselines (TTransE and HyTE) on ICEWS14 and GDELT and gives on-par results with HyTE on ICEWS05-15. This result shows the superiority of our diachronic embeddings compared to TTransE and HyTE. DE-DistMult outperforms TA-DistMult, the only DistMult-based baseline, showing the superiority of our diachronic embedding compared to TA-DistMult. Moreover, DE-DistMult outperforms all TransE-based baselines. Finally, just as SimplE beats TransE and DistMult due to its higher expressivity, our results show that DE-SimplE beats DE-TransE, DE-DistMult, and the other baselines due to its higher expressivity.

Previously, each of the existing models was tested on different subsets of ICEWS and GDELT and a comprehensive comparison of them did not exist. As a side contribution, Table 2 provides a comparison of these approaches on the same benchmarks and under the same experimental setting. The results reported in Table 2 may be directly used for comparison in future works.

Model Variants & Ablation Study

We run experiments on ICEWS14 with several variants of the proposed models to provide a better understanding of them. The results can be found in Table 3 and Figure 1. Table 3 includes DE-TransE and DE-DistMult with no variants as well so other variants can be easily compared to them.

Activation Function: So far, we used *sine* as the activation function in Equation (1). The performance for other activation functions including *Tanh*, *sigmoid*, *Leaky ReLU* (with 0.1 leakage), and *squared exponential* are presented in Table 3. From the table, it can be viewed that other activation functions also perform well. Specifically, squared exponential performs almost on-par with sine. We believe one reason why sine and squared exponential give better performance is because a combination of sine or square exponential features can generate more sophisticated features than a combination of Tanh, sigmoid, or ReLU features. While a temporal feature with Tanh or sigmoid as the activation corresponds to a smooth off-on (or on-off) temporal switch, a temporal feature with sine or squared exponential activation corresponds to two (or more) switches (e.g., off-on-off) which can potentially model relations that start at some time and end after

[2]Code and datasets: https://github.com/BorealisAI/DE-SimplE.

Table 2: Results on ICEWS14, ICEWS05-15, and GDELT. Best results are in bold.

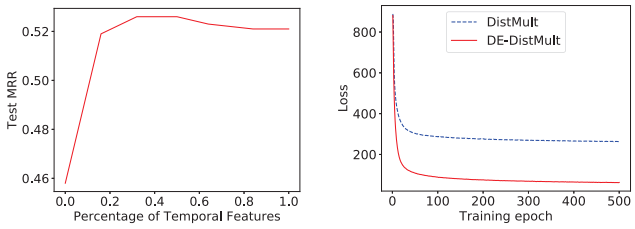
	ICEWS14				ICEWS05-15				GDELT			
Model	MRR	Hit@1	Hit@3	Hit@10	MRR	Hit@1	Hit@3	Hit@10	MRR	Hit@1	Hit@3	Hit@10
TransE	0.280	9.4	-	63.7	0.294	9.0	-	66.3	0.113	0.0	15.8	31.2
DistMult	0.439	32.3	-	67.2	0.456	33.7	-	69.1	0.196	11.7	20.8	34.8
SimplE	0.458	34.1	51.6	68.7	0.478	35.9	53.9	70.8	0.206	12.4	22.0	36.6
ConT	0.185	11.7	20.5	31.5	0.163	10.5	18.9	27.2	0.144	8.0	15.6	26.5
TTransE	0.255	7.4	-	60.1	0.271	8.4	-	61.6	0.115	0.0	16.0	31.8
HyTE	0.297	10.8	41.6	65.5	0.316	11.6	44.5	68.1	0.118	0.0	16.5	32.6
TA-DistMult	0.477	36.3	-	68.6	0.474	34.6	-	72.8	0.206	12.4	21.9	36.5
DE-TransE	0.326	12.4	46.7	68.6	0.314	10.8	45.3	68.5	0.126	0.0	18.1	35.0
DE-DistMult	0.501	39.2	56.9	70.8	0.484	36.6	54.6	71.8	0.213	13.0	22.8	37.6
DE-SimplE	**0.526**	**41.8**	**59.2**	**72.5**	**0.513**	**39.2**	**57.8**	**74.8**	**0.230**	**14.1**	**24.8**	**40.3**

a while (e.g., PresidentOf). These results also provide evidence for the effectiveness of diachronic embedding across several DEEMB functions.

Diachronic Embedding for Relations: Compared to entities, we hypothesize that relations may evolve at a very lower rate or, for some relations, evolve only negligibly. Therefore, modeling relations with a static (rather than a diachronic) representation may suffice. To test this hypothesis, we ran DE-TransE and DE-DistMult on ICEWS14 where relation embeddings are also a function of time. From the obtained results in Table 3, one can see that the model with diachronic embeddings for both entities and relations performs on-par with the model with diachronic embedding only for entities. We conducted the same experiment on ICEWS05-15 (which has a longer time horizons) and GDELT and observed similar results. These results show that at least on our benchmarks, modeling the evolution of relations may not be helpful. Future work can test this hypothesis on datasets with other types of relations and longer horizons.

Generalizing to Unseen Timestamps: We created a variant of the ICEWS14 dataset by including every fact except those on the 5^{th}, 15^{th}, and 25^{th} day of each month in the train set. We split the excluded facts randomly into validation and test sets (removing the ones including entities not observed in the train set). This ensures that none of the timestamps in the validation or test sets has been observed by the model in the train set. Then we ran DistMult and DE-DistMult on the resulting dataset. The obtained results in Table 3 indicate that DE-DistMult gains almost 10% MRR improvement over DistMult thus showing the effectiveness of our diachronic embedding to generalize to unseen timestamps.

Importance of Model Parameters: In Equation (1), the temporal part of the embedding contains three components: $\boldsymbol{a}_v$, $\boldsymbol{w}_v$, and $\boldsymbol{b}_v$. To measure the importance of each component, we ran DE-DistMult on ICEWS14 under three settings: 1- when $\boldsymbol{a}_v$s are removed (i.e. set to 1), 2- when $\boldsymbol{w}_v$s are removed (i.e. set to 1), and 3- when $\boldsymbol{b}_v$s are removed (i.e. set to 0). From the obtained results presented in Table 3, it can be viewed that all three components are important for the temporal features, especially $\boldsymbol{a}_v$s and $\boldsymbol{w}_v$s. Removing $\boldsymbol{b}_v$s does not affect the results as much as $\boldsymbol{a}_v$s and $\boldsymbol{w}_v$s. Therefore, if one needs to reduce the number of parameters, removing $\boldsymbol{b}_v$ may be a good option as long as they can tolerate a slight reduction in accuracy.



(a) (b)

Figure 1: (a) Test MRR of DE-SimplE on ICEWS14 as a function of γ. (b) The training curve for DistMult and DE-DistMult.

Static Features: Figure 1(a) shows the test MRR of DE-SimplE on ICEWS14 as a function of γ, the percentage of temporal features. According to Figure 1(a), as soon as some features become temporal (i.e. γ changes from 0 to a non-zero number), a substantial boost in performance can be observed. This observation sheds more light on the importance of learning temporal features and having diachronic embeddings. As γ becomes larger, MRR reaches a peak and then slightly drops. This slight drop in performance can be due to overfitting to temporal cues. This result demonstrates that modeling static features explicitly can help reduce the number of learnable parameters and avoid overfitting. Such a design choice may be even more important when the embedding dimensions are larger. However, it comes at the cost of adding one hyper-parameter to the model. If one prefers a slightly less accurate model with fewer hyper-parameters, they can make all vector elements temporal.

Training Curve: Figure 1(b) shows the training curve for DistMult and DE-DistMult on ICEWS14. While it has been argued that using sine activation functions may complicate training in some neural network architectures (see, e.g., (Giambattista Parascandolo 2017)), it can be viewed that when using sine activations, the training curve for our model is quite stable.

Table 3: Results for different variations of our model on ICEWS14.

Model	Variation	MRR	Hit@1	Hit@3	Hit@10
DE-TransE	No variation (Activation function: *Sine*)	0.326	12.4	46.7	68.6
DE-DistMult	No variation (Activation function: *Sine*)	0.501	39.2	56.9	70.8
DE-DistMult	Activation function: *Tanh*	0.486	37.5	54.7	70.1
DE-DistMult	Activation function: *Sigmoid*	0.484	37.0	54.6	70.6
DE-DistMult	Activation function: *Leaky ReLU*	0.478	36.3	54.2	70.1
DE-DistMult	Activation function: *Squared Exponential*	0.501	39.0	56.8	70.9
DE-TransE	Diachronic embedding for both entities and relations	0.324	12.7	46.1	68.0
DE-DistMult	Diachronic embedding for both entities and relations	0.502	39.4	56.6	70.4
DistMult	Generalizing to unseen timestamps	0.410	30.2	46.2	62.0
DE-DistMult	Generalizing to unseen timestamps	0.452	34.5	51.3	65.4
DE-DistMult	$\boldsymbol{a}_{\mathsf{v}}[n] = 1$ for $1 \leq n \leq \gamma d$ for all $\mathsf{v} \in \mathcal{V}$	0.458	34.4	51.8	68.3
DE-DistMult	$\boldsymbol{w}_{\mathsf{v}}[n] = 1$ for $1 \leq n \leq \gamma d$ for all $\mathsf{v} \in \mathcal{V}$	0.470	36.4	53.1	67.1
DE-DistMult	$\boldsymbol{b}_{\mathsf{v}}[n] = 0$ for $1 \leq n \leq \gamma d$ for all $\mathsf{v} \in \mathcal{V}$	0.498	38.9	56.2	70.4

Related Work

StaRAI: Statistical relational AI (Raedt et al. 2016) approaches are mainly based on soft (hanf-crafted or learned) rules where the probability of a world is proportional to the number of rules that are satisfied/violated in that world and the confidence for each rule. These approaches have been combined with embeddings (Kazemi and Poole 2018a) and also extended to temporal KGs (see, e.g., (Dylla, Miliaraki, and Theobald 2013; Chekol and Stuckenschmidt 2018)).

Static KG Embedding: A large number of models have been developed for static KG embedding. A class of these models are the translational approaches corresponding to variations of TransE (see, e.g., (Lin et al. 2015; Wang et al. 2014; Nguyen et al. 2016)). Another class of approaches are based on a bilinear score function $\boldsymbol{z}_{\mathsf{v}}^{\top} \boldsymbol{Z}_{\mathsf{r}} \boldsymbol{z}_{\mathsf{u}}$ each imposing a different sparsity constraint on the $\boldsymbol{Z}_{\mathsf{r}}$ matrices (see, e.g., (Nickel, Tresp, and Kriegel 2011; Trouillon et al. 2016; Kazemi and Poole 2018b)). A third class of models are based on deep learning approaches using feed-forward or convolutional layers on top of the embeddings (see, e.g., (Socher et al. 2013; Dettmers et al. 2018)). These models can be potentially extended to TKGC through our diachronic embedding.

Temporal KG Embedding: Several works have extended the static KG embedding models to temporal KGs. (Jiang et al. 2016) extend TransE by adding a timestamp embedding into the score function. (Dasgupta, Ray, and Talukdar 2018) extend TransE by projecting the embeddings to the timestamp hyperplain and then using the TransE score on the projected space. (Ma, Tresp, and Daxberger 2018) extend several models by adding a timestamp embedding to their score functions. These models may not work well when the number of timestamps is large. Furthermore, since they only learn embeddings for observed timestamps, they cannot generalize to unseen timestamps. (García-Durán, Dumančić, and Niepert 2018) extend TransE and DistMult by combining the relation and timestamp through a character LSTM. These models have been described in the paper and their performances have been reported in Table 2.

KG Embedding for Extrapolation: TKGC is an *interpolation* problem where given a set of temporal facts in a time frame, the goal is to predict the missing facts. A related problem is the *extrapolation* problem where future interactions are to be predicted (see, e.g., (Trivedi et al. 2017; Kumar, Zhang, and Leskovec 2018; Trivedi et al. 2019)). Despite some similarities in the employed approaches, KG extrapolation is fundamentally different from TKGC in that a score for an interaction $(\mathsf{v}, \mathsf{r}, \mathsf{u}, \mathsf{t})$ is to be computed given only the past (i.e. facts before t) whereas in TKGC the score is to be computed given past, present, and future. A comprehensive analysis of the existing models for interpolation and extrapolation can be found in (Kazemi et al. 2019).

Diachronic Word Embeddings: The idea behind our proposed embeddings is similar to diachronic word embeddings where a corpus is typically broken temporally into slices (e.g., 20-year chuncks of a 200-year corpus) and embeddings are learned for words in each chunk thus providing word embeddings that are a function of time (see, e.g., (Hamilton, Leskovec, and Jurafsky 2016; Bamler and Mandt 2017)). The main goal of diachronic word embeddings is to reveal how the meanings of the words have evolved over time. Our work can be viewed as an extension of diachronic word embeddings to continuous-time KG completion.

Conclusion

We developed a diachronic embedding function for temporal KG completion which provides a hidden representation for the entities of a temporal KG at any point in time. Our embedding is generic and can be combined with any score function. We proved that combining our diachronic embedding with SimplE results in a fully expressive model – the first temporal KG embedding model for which such a result exists. We showed the superior performance of our model compared to existing work on several benchmarks. Future work includes designing functions other than the one proposed in Equation (1), a comprehensive study of which functions are favored by different types of KGs, and using our proposed embedding for diachronic word embedding.

References

Bamler, R., and Mandt, S. 2017. Dynamic word embeddings. In *ICML*, 380–389.

Bordes, A.; Usunier, N.; Garcia-Duran, A.; Weston, J.; and Yakhnenko, O. 2013. Translating embeddings for modeling multi-relational data. In *NeurIPS*, 2787–2795.

Boschee, E.; Lautenschlager, J.; O'Brien, S.; Shellman, S.; Starz, J.; and Ward, M. 2015. Icews coded event data. *Harvard Dataverse* 12.

Chekol, M. W., and Stuckenschmidt, H. 2018. Rule based temporal inference. In *ICLP*.

Dasgupta, S. S.; Ray, S. N.; and Talukdar, P. 2018. Hyte: Hyperplane-based temporally aware knowledge graph embedding. In *EMNLP*, 2001–2011.

Dettmers, T.; Minervini, P.; Stenetorp, P.; and Riedel, S. 2018. Convolutional 2d knowledge graph embeddings. In *AAAI*.

Dylla, M.; Miliaraki, I.; and Theobald, M. 2013. A temporal-probabilistic database model for information extraction. *VLDB Endowment* 6(14):1810–1821.

Fatemi, B.; Ravanbakhsh, S.; and Poole, D. 2019. Improved knowledge graph embedding using background taxonomic information. In *AAAI*.

García-Durán, A.; Dumančić, S.; and Niepert, M. 2018. Learning sequence encoders for temporal knowledge graph completion. In *EMNLP*.

Giambattista Parascandolo, Heikki Huttunen, T. V. 2017. Taming the waves: sine as activation function in deep neural networks.

Hamilton, W. L.; Leskovec, J.; and Jurafsky, D. 2016. Diachronic word embeddings reveal statistical laws of semantic change. *arXiv preprint arXiv:1605.09096*.

Hitchcock, F. L. 1927. The expression of a tensor or a polyadic as a sum of products. *Journal of Mathematics and Physics* 6(1-4):164–189.

Jiang, T.; Liu, T.; Ge, T.; Sha, L.; Chang, B.; Li, S.; and Sui, Z. 2016. Towards time-aware knowledge graph completion. In *COLING*, 1715–1724.

Kadlec, R.; Bajgar, O.; and Kleindienst, J. 2017. Knowledge base completion: Baselines strike back. *arXiv preprint arXiv:1705.10744*.

Kazemi, S. M., and Poole, D. 2018a. RelNN: A deep neural model for relational learning. In *AAAI*.

Kazemi, S. M., and Poole, D. 2018b. SimplE embedding for link prediction in knowledge graphs. In *NeurIPS*.

Kazemi, S. M.; Goel, R.; Jain, K.; Kobyzev, I.; Sethi, A.; Forsyth, P.; and Poupart, P. 2019. Relational representation learning for dynamic (knowledge) graphs: A survey. *arXiv preprint arXiv:1905.11485*.

Kingma, D. P., and Ba, J. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Kumar, S.; Zhang, X.; and Leskovec, J. 2018. Learning dynamic embedding from temporal interaction networks. *arXiv preprint arXiv:1812.02289*.

Leetaru, K., and Schrodt, P. A. 2013. Gdelt: Global data on events, location, and tone, 1979–2012. In *ISA annual convention*, volume 2, 1–49. Citeseer.

Lin, Y.; Liu, Z.; Sun, M.; Liu, Y.; and Zhu, X. 2015. Learning entity and relation embeddings for knowledge graph completion. In *AAAI*, 2181–2187.

Ma, Y.; Tresp, V.; and Daxberger, E. A. 2018. Embedding models for episodic knowledge graphs. *Journal of Web Semantics*.

Minervini, P.; Costabello, L.; Muñoz, E.; Nováček, V.; and Vandenbussche, P.-Y. 2017. Regularizing knowledge graph embeddings via equivalence and inversion axioms. In *ECML PKDD*, 668–683. Springer.

Nguyen, D. Q.; Sirts, K.; Qu, L.; and Johnson, M. 2016. Stranse: a novel embedding model of entities and relationships in knowledge bases. In *NAACL-HLT*.

Nickel, M.; Murphy, K.; Tresp, V.; and Gabrilovich, E. 2016. A review of relational machine learning for knowledge graphs. *Proceedings of the IEEE* 104(1):11–33.

Nickel, M.; Tresp, V.; and Kriegel, H.-P. 2011. A three-way model for collective learning on multi-relational data. In *ICML*, volume 11, 809–816.

Paszke, A.; Gross, S.; Chintala, S.; Chanan, G.; Yang, E.; DeVito, Z.; Lin, Z.; Desmaison, A.; Antiga, L.; and Lerer, A. 2017. Automatic differentiation in pytorch. In *NIPS-W*.

Raedt, L. D.; Kersting, K.; Natarajan, S.; and Poole, D. 2016. Statistical relational artificial intelligence: Logic, probability, and computation. *Synthesis Lectures on Artificial Intelligence and Machine Learning* 10(2):1–189.

Socher, R.; Chen, D.; Manning, C. D.; and Ng, A. 2013. Reasoning with neural tensor networks for knowledge base completion. In *AAAI*, 926–934.

Sun, Z.; Deng, Z.-H.; Nie, J.-Y.; and Tang, J. 2019. RotatE: Knowledge graph embedding by relational rotation in complex space. In *ICLR*.

Trivedi, R.; Dai, H.; Wang, Y.; and Song, L. 2017. Know-evolve: Deep temporal reasoning for dynamic knowledge graphs. In *ICML*, 3462–3471.

Trivedi, R.; Farajtabar, M.; Biswal, P.; and Zha, H. 2019. DyRep: Learning representations over dynamic graphs. In *ICLR*.

Trouillon, T.; Welbl, J.; Riedel, S.; Gaussier, É.; and Bouchard, G. 2016. Complex embeddings for simple link prediction. In *ICML*, 2071–2080.

Trouillon, T.; Dance, C. R.; Gaussier, É.; Welbl, J.; Riedel, S.; and Bouchard, G. 2017. Knowledge graph completion via complex tensor factorization. *JMLR* 18(1):4735–4772.

Tucker, L. R. 1966. Some mathematical notes on three-mode factor analysis. *Psychometrika* 31(3):279–311.

Wang, Z.; Zhang, J.; Feng, J.; and Chen, Z. 2014. Knowledge graph embedding by translating on hyperplanes. In *AAAI*.

Xu, K.; Hu, W.; Leskovec, J.; and Jegelka, S. 2019. How powerful are graph neural networks? In *ICLR*.

Yang, B.; Yih, W.-t.; He, X.; Gao, J.; and Deng, L. 2015. Embedding entities and relations for learning and inference in knowledge bases. *ICLR*.