Relational Learning on Temporal Knowledge Graphs

Zhen Han

Dissertation
an der Fakulität für Mathematik, Informatik, und Statistik
der Ludwig–Maximilians–Universität München

eingereicht von
Zhen Han

München, den Mai 19, 2022

Erstgutachter: Prof. Volker Tresp

Zweitgutachter: Prof. Dr. Sepp Hochreiter

Drittgutachter: Prof. Dr. Barbara Plank

Tag der mündlichen Prüfung: September 16, 2022

Eidesstattliche Versicherung

(Siehe Promotionsordnung vom 12.07.11, § 8, Abs. 2 Pkt. .5.)

Hiermit erkläre ich an Eidesstatt, dass die Dissertation von mir selbstständig, ohne unerlaubte Beihilfe angefertigt ist.

Han, Zhen
Name, Vorname

München, Mai 16, 2022	Zhen Han
Ort, Datum	Unterschrift Doktorand/in

Formular 3.2

Contents

Acknowledgement

I would like to take this opportunity to thank the people who supported and motivated me during my Ph.D. study, which is a joint program between the Ludwig Maximilian University of Munich and Siemens AG.

First of all, I would like to express my deepest gratitude to my supervisor, Prof. Dr. Volker Tresp, for his great support. Whenever I faced difficulties in my Ph.D. study, I can always get helpful and friendly guidance from him. Volker gave me great freedom to explore various research ideas and provided abundant resources to conduct the research. I enjoy every discussion with Volker and have been motivated deeply. Without his help, this thesis would not have come to such a good end. In addition, I am very honored that Prof. Dr. Sepp Hochreiter and Prof. Dr. Barbara Plank agreed to be external examiners of my thesis.

Besides, I would like to extend my gratitude to Dr. Yunpu Ma, who guided me at the beginning of my Ph.D. study. He provided insightful comments and helpful discussions and kept me on the right track in the research. I am also very grateful to my colleagues Hang Li, Jindong Gu, Dr. Zhiliang, Dr. Rui Zhao, Dr. Marcel Hinderbrand, Yushan Liu. I want to thank them for their advice and encouragement. Moreover, I would like to thank my collaborators Peng Chen, Zifeng Ding, Gengyuan Zhang, and Ruotong Liao. Thank you for your invaluable contribution to this thesis.

The study shown in the thesis was carried out at the Department of Artificial Intelligence of Siemens Corporate Technology, which is headed by Dr. Micheal May. I would like to thank him for providing me with the freedom and resources to realize my research ideas.

Finally, I would like to thank my family, who accompanied me throughout my entire Ph.D. study with ups and downs. Nobody was more important than you!

Abstract

Over the last decade, there has been an increasing interest in relational machine learning (RML), which studies methods for the statistical analysis of relational or graph-structured data. Relational data arise naturally in many real-world applications, including social networks, recommender systems, and computational finance. Such data can be represented in the form of a graph consisting of nodes (entities) and labeled edges (relationships between entities). While traditional machine learning techniques are based on feature vectors, RML takes relations into account and permits inference among entities. Recently, performing prediction and learning tasks on knowledge graphs has become a main topic in RML. Knowledge graphs (KGs) are widely used resources for studying multi-relational data in the form of a directed graph, where each labeled edge describes a factual statement, such as (*Munich*, *locatedIn*, *Germany*).

Traditionally, knowledge graphs are considered to represent stationary relationships, which do not change over time. In contrast, event-based multi-relational data exhibits complex temporal dynamics in addition to its multi-relational nature. For example, the political relationship between two countries would intensify because of trade fights; the president of a country may change after an election. To represent the temporal aspect, temporal knowledge graphs (tKGs) were introduced that store a temporal event as a quadruple by extending the static triple with a timestamp describing when this event occurred, i.e. (*Barack Obama*, *visit*, India, *2010-11-06*). Thus, each edge in the graph has temporal information associated with it and may recur or evolve over time.

Among various learning paradigms on KGs, knowledge representation learning (KRL), also known as knowledge graph embedding, has achieved great success. KRL maps entities and relations into low-dimensional vector spaces while capturing semantic meanings. However, KRL approaches have mostly been done for static KGs and lack the ability to utilize rich temporal dynamics available on tKGs. In this thesis, we study state-of-the-art representation learning techniques for temporal knowledge graphs that can capture temporal dependencies across entities in addition to their relational dependencies. We discover representations for two inference tasks, i.e., tKG forecasting and completion. The former is to forecast future events using historical observations up to the present time, while the latter predicts missing links at observed timestamps. For tKG forecasting, we show how to make the reasoning process interpretable while maintaining performance by employing a sequential reasoning process over local subgraphs. Besides, we propose a continuous-depth multi-relational graph neural network with a novel graph neural ordinary differential equation. It allows for learning continuous-time representations of tKGs, especially in cases with observations in irregular time intervals, as encountered in online analysis. For tKG completion, we systematically review multiple benchmark models. We thoroughly investigate the significance of the proposed temporal encoding technique in each model and provide the first unified open-source framework, which gathers the implementations of well-known tKG completion models. Finally, we discuss the power of geometric learning and show that learning evolving entity representations in a product of Riemannian manifolds can better reflect geometric structures on tKGs and achieve better performances than Euclidean embeddings while requiring significantly fewer model parameters.

Zusammenfassung

In den letzten zehn Jahren hat das Interesse am relationalen maschinellen Lernen (RML) zugenommen, das Methoden zur statistischen Analyse relationaler oder graphenstrukturierter Daten untersucht. Relationale Daten entstehen in vielen realen Anwendungen, einschließlich soziale Netzwerke, Empfehlungssysteme, und Computational Finance. Solche Daten können in Form eines Graphen dargestellt werden, der aus Knoten (Entitäten) und beschrifteten Kanten (Beziehungen zwischen Entitäten) besteht. Während traditionelle maschinelle Lerntechniken basieren auf Merkmalsvektoren, berücksichtigt RML Beziehungen und erlaubt Inferenz zwischen Entitäten. Prognose und Lernen auf Wissensgraphen sind zur Zeit zu Hauptthemen in RML geworden. Wissensgraphen sind weit verbreitete Ressourcen zum Analysis multirelationaler Daten in Form eines gerichteten Graph, bei dem jede beschriftete Kante eine Tatsachenaussage beschreibt, wie z.B. (*München*, *befindet_sich_in*, *Deutschland*).

Traditionell werden Wissensgraphen als Darstellung stationärer Beziehungen, die sich im Laufe der Zeit nicht ändern. Im Gegensatz dazu weisen ereignisbasierte multirelationale Daten neben ihrer multirelationalen Natur auch eine komplexe zeitliche Dynamik auf. Zum Beispiel, die politische Beziehung zwischen zwei Ländern kann sich beispielsweise aufgrund vom Handelsstreit intensivieren. Ebenso kann sich der Präsident eines Landes nach einer Wahl ändern. Zur Darstellung des temporalen Aspekt, wurden temporale Wissensgraphen eingeführt, die ein zeitliches Ereignis als Quadrupel speichern, indem sie das statische Tripel um einen Zeitstempel erweitern, der beschreibt, wann dieses Ereignis eingetreten ist, zum Beispiel (*Barack Obama*, *besucht*, *Indien*, *06.11.2010*). Somit ist jede Kante des Graphen mit zeitlichen Informationen verknüpft und kann sich im Laufe der Zeit entwickeln.

Unter verschiedenen Lernparadigmen für Wissensgraphen hat das Knowledge Representation Learning (KRL), auch bekannt als Knowledge Graph Embedding, einen großen Erfolg erzielt. KRL bildet Entitäten und Beziehungen in niedrigdimensionalen Vektorräumen ab und erfasst dabei ihre semantischen Bedeutungen. Die meisten KRL-Ansätze wurden jedoch für statische Wissensgraphen entwickelt und sind nicht in der Lage, die reichhaltige zeitliche Dynamik von temporalen Wissensgraphen zu nutzen. In dieser Arbeit untersuchen wir modernste Repräsentationslerntechniken für temporale Wissensgraphen, die zusätzlich zu den relationalen Abhängigkeiten auch temporale Abhängigkeiten zwischen Entitäten erfassen können. Wir lernen Repräsentationen für zwei Inferenzaufgaben, und zwar temporale Wissensgraphen Vorhersage und Vervollständigung. Die vordere Aufgabe vorhersagt zukünftiger Ereignisse anhand historischer Beobachtungen bis zur Gegenwart, während die letzte Aufgabe fehlende Kanten zu bekannten Zeitpunkten prognostiziert. Für die vordere Vorhersageaufgabe zeigen wir, wie der Schlussfolgerungsprozess interpretierbar gemacht werden kann, ohne die Leistung zu beeinträchtigen, indem wir einen sequentiellen Schlussfolgerungsprozess über lokale Teilgraphen vorschlagen. Außerdem schlagen wir ein multirelationales neuronales Netzwerk mit einer neuartigen gewöhnlichen Differentialgleichung für Graphen vor. Es ermöglicht das Lernen von zeitkontinuierlichen Repräsentationen von temporalen Wissensgraphen, insbesondere in Fällen mit Beobachtungen in unregelmäßigen Zeitintervallen, wie sie in der Online-Analyse vorkommen. Zur Vervollständigung von temporalen Wissensgraphen machen wir eine systematische Literaturrecherche mit mehrerer Benchmark-Modelle. Wir untersuchen gründlich den Beitrag der vorgeschlagenen temporalen Kodierungstechnik in jedem Modell und stellen das erste einheitliche Open-Source-Framework zur Verfügung, das die Implementierungen bekannter temopralen Wissensgraphen Vervollständigungsmodelle zusammenfasst. Schließlich erörtern wir die Leistungsfähigkeit des geometrischen Lernens und zeigen, dass das Lernen von dynamischen Entitätsrepräsentationen in einem Produkt von Riemannschen Mannigfaltigkeiten geometrische Strukturen von temporalen Wissensgraphen besser widerspiegeln und bessere Leistungen als euklidische Einbettungen erzielen kann, während deutlich weniger Modellparameter erforderlich sind.

List of Publications and Declaration of Authorship

- Zhen Han, Peng Chen, Yunpu Ma, Volker Tresp. Explainable Subgraph Reasoning for Forecasting on Temporal Knowledge Graphs. In *Proceedings of the International Conference on Learning Representations (ICLR)*, Virtual Conference, May 2021. Openreview.net: pdf?id=pGIHq1m7PU.

  I conceived of the original research contributions. My co-author Peng Chen and I performed most implementations and evaluations. I wrote the initial draft of the manuscript and did most of the subsequent corrections. I regularly discussed this work with the co-author, Yunpu Ma, and my supervisor, Volker Tresp, who also assisted me in improving the manuscript.

  This published work serves as Chapter 3.

- Zhen Han, Zifeng Ding, Yunpu Ma, Jiayu Gu, Volker Tresp. Learning Neural Ordinary Equations for Forecasting Future Links on Temporal Knowledge Graphs. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Online and in Dominican Republic, Nov. 2021. DOI: 10.18653/v1/2021.emnlp-main.658.

  My co-author Yunpu Ma and I conceived of the original research contributions and designed the model architecture. Zifeng Ding performed most of the implementations and experiments, with smaller contributions by Jiayu Gu. Zifeng Ding and I designed the experimental protocol and analyzed the results. Zifeng Ding and I wrote the manuscript, and all authors revised it. I regularly discussed this work with my supervisor Volker Tresp.

  This published work serves as Chapter 4.

- Zhen Han, Gengyuan Zhang, Yunpu Ma, Volker Tresp. Time-dependent Entity Embedding is not All You Need: A Re-Evaluation of Temporal Knowledge Graph Completion Models under a Unified Framework. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Online and in Dominican Republic, Nov. 2021. DOI: 10.18653/v1/2021c.emnlp-main.639.

  I conceived of the original research contributions. My co-author Gengyuan Zhang and I performed most implementations and evaluations. I wrote the initial draft of the manuscript and did most of the subsequent corrections. I regularly discussed this work with the co-author, Yunpu Ma, and my supervisor, Volker Tresp, who also assisted me in improving the manuscript.

  This published work serves as Chapter 5.

- Zhen Han, Peng Chen, Yunpu Ma, Volker Tresp. DyERNIE: Dynamic Evolution of Riemannian Manifold Embeddings for Temporal Knowledge Graph Completion. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Virtual Conference, Nov 2020. DOI: 10.18653/v1/2020.emnlp-main.593.

  Yunpu and I conceived of the original research contributions. I performed all implementations and evaluations. I wrote the initial draft of the manuscript and did most of the subsequent corrections. Yunpu Ma, Peng Chen, and my supervisor, Volker Tresp assisted me in improving the manuscript. I regularly discussed this work with the co-author, Yunpu Ma, and my supervisor, Volker Tresp.

  This published work serves as Chapter 6.

Other Publications

- Zhen Han, Yunpu Ma, Yuyi Wang, Stephan Günnemann, Volker Tresp. Graph Hawkes Neural Network for Forecasting on Temporal Knowledge Graphs. In *Proceedings of the Conference on Automated Knowledge Base Construction (AKBC)*, Virtual Conference, June 2020. arXiv:2003.13432.

- Zhen Han, Ruotong Liao, Beiyan Liu, Yao Zhang, Zifeng Ding, Heinz Köppl, Hinrich Schütze, Volker Tresp. Enhanced Temporal Knowledge Embeddings with Contextualized Language Representations. Under review at *the Advances in Neural Information Processing Systems (NeurIPS)*, 2022. arXiv:2203.09590.

- Jin Guo, Zhen Han, Zhou Su, Jiliang Li, Volker Tresp, Yuyi Wang. Continuous Temporal Graph Networks for event-based graph data. Under review at *30th ACM International Conference on Information and Knowledge Management (CIKM)*, 2022.

- Ronggui Fu, Zhao Meng, Zhen Han, Zifeng Ding, Yunpu Ma, Matthias Schubert, Volker Tresp, Roger Wattenhofer. TempCaps: A Capsule Network-based Embedding Model for Temporal Knowledge Graph Completion. In *Proceedings of the workshop on Structured Prediction for NLP at Annual Meeting of the Association for Computational Linguistics (ACL)*, 2022.

- Haohai Sun, Jialun Zhong, Yunpu Ma, Zhen Han, Kun He. TimeTraveler: Reinforcement Learning for Temporal Knowledge Graph Forecasting. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Online and Dominican Republic, Nov. 2021. DOI: 10.18653/v1/2021.emnlp-main.655.

- Yue Feng, Zhen Han, Mingming Sun, Ping Li. Multi-Hop Open-Domain Question Answering over Structured and Unstructured Knowledge. In *Findings of Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*, 2022.

Chapter 1

Introduction

1.1 Motivation

Human beings instinctively model their surrounding environment in order to understand it and take actions to maximize the chance of achieving their goals. Artificial Intelligence (AI) aims to develop intelligent agents that mimic the ability of human beings. In AI research, machine learning (ML) has found the most interest in achieving these goals over the past decades and therefore stays at the core of modern AI among different technical fields [62]. Machine Learning addresses problems in a heavily data-driven fashion, where it models complex systems by searching through prefabricated hypothesis spaces. Specifically, traditional machine learning algorithms learn a mapping from an input feature vector that represents an object with numeric or categorical attributes to an output prediction, which could be class labels or regression scores [65]. However, large quantities of data often exhibit complex structures and inter-dependencies that cannot be captured by feature vectors. For example, billions of users view, purchase, and rate products on online shopping stores. Users are not independent and may build friendships as well as share favorite items. Thus, there is a need for an intuitive and concise modeling scheme to model complex systems with relations. A prominent solution is graphs that represent the system's components as nodes and the relations between nodes as edges. Graphs naturally appear in biomedical interactions (protein-protein interaction) and user-item interactions. Relational (machine) learning studies methods for the statistical analysis of relational graph-structured data [65]. While traditional machine learning techniques are solely based on feature vectors, relational learning takes relations into account that provide valuable information and permits inference among entities. An object's representation in relational learning contains information about its relationships to other objects. The canonical tasks of relational learning include link prediction and node and graph classification, which arise in many highly relevant real-world scenarios, such as recommender systems and social network analysis.

Knowledge graphs (KGs) are an intriguing variant of graphs to deal with multi-relational factual data. A knowledge graph consists of a collection of triples (s, p, o), where s (subject) and o (object) correspond to nodes, each labeled edge between entities describes a factual statement, such as (*Munich*, *locatedIn*, *Germany*). In recent decades, a large amount of large-scale KGs has emerged. For example, YAGO [81] is a semantic knowledge graph with one million entities and five million facts extracted from Wikipedia. It contains general knowledge about individuals, such as persons, organizations, and products, with their semantic relationships and helps build the Watson cognitive platform [22]. Similarly, Freebase [10] is a community-oriented knowledge graph for general purpose and mainly composed of its community members and merged into Wikidata [92] in 2016. Besides, Himmelstein and Baranzini [41] and Walsh et al. [93] proposed biomedical knowledge graphs, i.e., Hetionet and BioKG, to model the interactions between biomedical entities and their effects on the biomedical system, providing direct and precise biomedical knowledge. Moreover, Rotmensch et al. [75] and Gyrard et al. [28] learn health knowledge graphs from electronic medical records for clinical decision support in medicine and self-diagnostic symptom checker.

Common knowledge graphs epresent stationary relationships, which do not change over time. However, the recent availability of event-based multi-relational data exhibits complex temporal dynamics in addition to its multi-relational nature. For example, the political relationship between two countries would intensify because of trade fights; the president of a country may change after an election. To declaratively represent the temporal aspect, temporal knowledge graphs (tKGs) were introduced that store a temporal event as a quadruple by extending the static triple with a timestamp describing when this event occurred, i.e. (*Barack Obama*, *visit*, *India*, *2010-11-06*). Thus, each edge in the graph has temporal information associated with it and may recur over time. The integrated crisis early warning system (ICEWS) [12] and the global database of events, language, and tone (GDELT) [57] have established themselves in the research community as benchmarks of temporal knowledge graphs that contain events across the globe connected people, organizations, and news sources.

Early research on knowledge graphs focuses on manually curated or automatically mined rules to infer new knowledge from observed facts. Such symbolic approaches suffer from scalability and generalizability issues. In recent years, knowledge representation learning (KRL), also known as knowledge graph embedding (KGE), has gained great interest success. KRL maps entities and relations into low-dimensional vector spaces while capturing their semantic meanings and is feasible to generalize on large-scale knowledge graphs with millions of entities. However, KRL approaches have mostly been done for static KGs and cannot utilize rich temporal dynamics available on tKGs. In this thesis, we study state-of-the-art relational representation learning techniques for temporal knowledge graphs that can capture temporal dependencies across entities in addition to their relational dependencies. Specifically, we investigate relational learning on tKGs for link prediction with two settings, i.e., tKG forecasting and tKG completion. The tKG forecasting task is to predict links in the future, given all the observations up to the present time. Forecasting on tKGs can help many downstream applications such as decision support in personalized health care and finance. In comparison, the tKG completion task is designed to the inherent incompleteness of tKGs in the sense that some interactions between entities are missing. Similar to the smoothing problem in traditional time series analysis [63], the tKG completion task requires estimating missing links at past timestamps, given all the evidence up to the current time. We developed several approaches for both tKG forecasting and tKG completion in terms of explainability, continuous-time modeling, and non-Euclidean embedding. Moreover, we systematically studied existing temporal knowledge graph completion models and investigated the contribution of the respective temporal encoding module.

1.2 Contributions

This section provides an overview of the contributions of included publications in this thesis and positions them within the research area.

The first part of this thesis is concerned with the temporal knowledge graph forecasting task, which predicts links in the future given all the observations up to the present time. Forecasting on tKGs can help many downstream applications such as decision support in personalized health care and finance. The use cases often require the predictions made by the learning models to be interpretable, such that users can understand and trust the predictions. However, existing approaches always operate in a black-box fashion and cannot clearly show which evidence contributes to a prediction, making them less suitable for many real-world applications. In this context, we have developed the first explainable reasoning framework for forecasting future links on tKGs by employing a sequential reasoning process over local subgraphs. This part is covered in **Chapter 3**. Given a query in the form of (subject, predicate, ?, timestamp), the proposed model starts from the query subject and iteratively samples relevant edges from the given tKG to construct a query-dependent subgraph. After several rounds of expansion and pruning, the missing object is predicted from entities in the subgraph. To capture the temporal aspect of tKGs, we have proposed a temporal relational graph attention (TRGA) mechanism, which poses temporal constraints on message passing to preserve the causal nature of the temporal data, and learn time-dependent entity embedding. Specifically, the entity embedding concatenates a stationary entity embedding and a functional time encoding. The temporal dynamics are then modeled by the interactions between the functional time encoding and entity features as well as underlying topological structures of tKGs. To understand the contribution of the time embedding, we replace the time-dependent part from entity representations with static embedding, resulting in significant degradation of the model's performance. Compared to existing tKG models, the proposed model has the following advantages: 1) The extracted subgraph can visualize the reasoning process and provide an interpretable graphical explanation to emphasize important evidence supporting the prediction. A survey with 53 respondents demonstrates the extracted evidence is aligned with human understanding. 2) The dynamical pruning procedure enables the model to reason on large-scale tKGs with millions of edges. 3) Extensive experiments on benchmark datasets of tKG forecasting show that our method outperforms state-of-the-art approaches by a large margin while being more interpretable.

Preserving the continuous-time nature of temporal knowledge graphs is considered another major challenge in the community. tKG forecasting approaches mostly model graph dynamics in a discrete-time domain, and thus, cannot model observations in irregular time intervals, which convey essential information for analyzing dynamics on tKGs, e.g., the dwelling time of a user on a website becomes shorter, indicating that the user's interest in the website decreases. To this end, we have proposed a continuous-depth multi-relational graph neural network in **Chapter 4**, which utilizes a novel graph neural ordinary differential equation (ODE) to encode the continuous dynamics on tKGs. Specifically, we integrate the hidden representations over time using an ODE solver and output continuous-time representations of each entity. Unlike many existing tKG models that learn the graph dynamics by employing discrete recurrent model structures, our model lets the time domain coincide with the depth of the graph neural network and takes advantage of neural ODE to steer the latent entity features between two timestamps smoothly. Thus, the model is able to compute the probability of an event at an arbitrary timestamp, which considerably enhances the flexibility of prediction. Additionally, we have proposed a graph transition layer to let the model pay more attention to edge formation and dissolution of tKGs, which leads to a significant improvement in the model's performance. Through experiments on five benchmark tKG forecasting datasets, it is shown that the proposed model delivered significantly better performance than baseline methods while requiring considerably less training cost.

Similar to static KGs, tKGs usually suffer from incompleteness in the sense that they do not include all events related to their entities and relations. To this end, the second part of this thesis is concerned with the temporal knowledge graph completion task, which requires estimating missing links in the past given all the evidence up to the current time. **Chapter 5** systematically studied existing temporal knowledge graph completion models and classified them into two classes based on how they model the dynamics of tKGs, i.e., timestamp embedding (TE) approaches and time-dependent entity embedding (TEE) approaches. As a highlight contribution to the community, we empirically found out that TE approaches perform better on sparse tKGs, which is in contrast to the results in prior studies, and reveal the weakness of TEE approaches. Additionally, introducing new temporal embedding approaches always accompanied new training strategies. Although ablation studies were provided, the significance of the proposed temporal embedding was not thoroughly investigated. To foster further research, we have performed an extensive benchmark study of existing approaches and provided the first unified open-source framework. This framework gathers well-known tKG completion models and enables an insightful assessment for future research on the tKG completion task.

As the last contribution, we studied the influence of the underlying geometry on embedding temporal multi-relational data and introduced non-Euclidean temporal knowledge embedding in **Chapter 6**. Many entities of multi-relational data induce geometric structures. For example, organizations, e.g., public sectors, usually have a hierarchical structure, where the number of departments grows exponentially with their distance to the root (headquarter). Embedding methods in Euclidean space have limitations and suffer from significant distortion when representing large-scale hierarchical or cyclical-structured data. To this end, geometric learning has been exploited in recent knowledge graph embedding models. As a result, hierarchical data, i.e., trees, can be efficiently modeled in a two-dimensional hyperbolic disc since the hyperbolic area grow exponentially with its radius. However, existing non-Euclidean embedding approaches for KGs lack the ability to capture temporal dynamics on tKGs. The difficulty with representing the evolution of tKGs lies in finding a way to integrate temporal information into the non-Euclidean representations of entities. To address this challenge, we have proposed a non-Euclidean representation learning model named DyERNIE that characterizes the time-dependent representation of each entity as movements on manifolds. For each entity, we define an initial embedding on manifolds and a velocity vector residing in the tangent space of the initial embedding to generate a temporal representation at each timestamp. In particular, the initial embedding captures the stationary structural dependencies across facts, while the velocity vectors model the time-varying properties of the entity. Moreover, most graph-structured data has a wide variety of inherent geometric structures, e.g., partially tree-like and partially cyclical. Nevertheless, existing non-Euclidean KG embedding approaches model the latent structures in a single geometry with a constant curvature, limiting the flexibility of the model to match the hypothetical intrinsic manifold. In comparison, our proposed method learns evolving entity representations in a product of Riemannian manifolds to better reflect a wide variety of geometric structures on tKGs. To better capture a broad range of structures in temporal KGs, we show how the product space can be approximately identified from sectional curvatures of temporal KGs, e.g., how to choose the dimensionality of component spaces and their curvatures. To understand the contribution of the product space of Riemannian manifolds (DyERNIE-Prod), we compare the proposed model with its Euclidean counterpart (DyERNIE-Euclid) and the variant in a manifold with a constant curvature (DyERNIE-Sgl). Extensive experiments conclude that both DyERNIE-Prod and DyERNIE-Sgl require five times fewer embedding dimensions to achieve similar performance as DyERNIE-Euclid's, demonstrating the merits of temporal non-Euclidean embeddings. Besides, we observed that DyERNIE-Prod generally performs better than DyERNIE-Sgl on all three tKG completion datasets. To conclude, DyERNIE is the first method that explores the usage of the temporal evolution of geometric embedding and opens a research direction for studying geometric embedding on temporal relational data.

In summary, this thesis is dedicated to relational learning on temporal knowledge graphs and proposes several temporal embedding approaches in terms of different aspects, i.e., **explainability**, **continuous-time modeling**, and **geometric learning**. The proposed approaches can potentially serve a variety of applications, e.g., time-aware recommender systems, which take into account the evolving nature of users and temporal popularity of items, and temporal question answering. To contribute towards reproducibility as well as providing usable research artifacts to the community, we provide openly accessible implementations of all presented works . Overall, we believe the works would stimulate progress in relational learning on temporal knowledge graphs, which has gained increasing attention in recent years.

1.3 Overview

The remainder of this thesis is organized as follows. Chapter 2 gives an overview of the broader research area of relational learning on (temporal) knowledge graphs. It reviews existing work and lays the foundations for the techniques used in the remainder of this thesis. Concretely, we detail the notations used in this work in Section 2.1. In Section 2.2 we introduce knowledge graph fundamentals and review benchmark relational learning techniques on KGs. Section 2.3 and 2.4 introduce temporal knowledge graphs, two link prediction tasks on tKGs, and approaches for learning temporal representations on tKGs. In Section 2.5 we introduce non-Euclidean embedding space and review representation learning models that embed knowledge graphs in non-Euclidean spaces.

Chapters 3 and 4 contain our published work on the tKG forecasting task. Specifically, Chapter 3 presents a subgraph reasoning approach for explainable forecasting on tKGs [31]. Chapter 4 introduces a continuous-depth multi-relational graph neural network for learning continuous-time representations of tKGs [32]. Chapters 5 and 6 present our publications on the tKG completion task. In particular, Chapter 5 provides a systematical study [35] of existing temporal knowledge graph completion models to analyze the effectiveness of different temporal encoding modules. Chapter 6 introduces a novel tKG completion model [30] that learns evolving entity representations in a product of Riemannian manifolds. We conclude in Chapter 7 and discuss directions for future works.

Chapter 2

Background

In this chapter, we introduce the fundamentals of the thesis in greater detail than a single paper allows. In addition, we review existing works to demonstrate the thesis' contributions better.

2.1 Notation

This section first defines the mathematical notation that we use throughout this chapter, which is mostly consistent with the individual publications in the other chapters.

Sets are denoted by calligraphic letters, e.g., the vertex set of a graph is given by $\mathcal{V}$ and the edge set by $\mathcal{E}$. Scalars and elements of sets are given by lowercase letters, e.g., $v \in \mathcal{V}$. Vectors are denoted by bold lowercase letters $\mathbf{x}$ with elements x_i. By $\mathbb{R}$ we denote real numbers, and $\mathbb{R}^n$ indicates n-dimensional real space. If not noted otherwise, we assume each variable to be real. Matrices are represented by bold uppercase letters $\mathbf{X}$, where X_{ij} denotes the entry in the i-th row and j-th column.

We denote the inner product between vectors $\mathbf{x}, \mathbf{y} \in \mathbb{R}^d$ by $\mathbf{x} \cdot \mathbf{y}$, which is equivalent to $\mathbf{x}^T\mathbf{y}$. $\mathbf{x}^T$ denotes the transpose of $\mathbf{x}$. By $\mathbf{x}||\mathbf{y} \in \mathbb{R}^{d+d'}$ we denote the concatenation of the column vectors $\mathbf{x} \in \mathbb{R}^d$ and $\mathbf{y} \in \mathbb{R}^{d'}$. Moreover, $||\mathbf{x}||_p$ denotes the p-norm of $\mathbf{x}$ given by $||\mathbf{x}||_p = \left(\sum_{i=1}^{d} |x_i|^p\right)^{\frac{1}{p}}$. In particular, $||\mathbf{x}||_2$ denotes the Euclidean norm of $\mathbf{x}$. By $\odot$ we denote the Hadamard product, i.e., the elementwise multiplication.

For some interaction models in Section 2.5, we need basic notions of non-Euclidean geometries, particularly the hyperbolic geometry and the hypersphere space. In [88], the vector addition $\oplus_K$, exponential map $\exp_{\mathbf{x}}^K(\cdot)$ at $\mathbf{x}$, logarithmic map $\log_{\mathbf{x}}^K(\cdot)$ at $\mathbf{x}$, matrix-vector multiplication $\otimes_K$, and distance d_K are defined in the non-Euclidean geometry using:

$$\mathbf{x} \oplus_K \mathbf{y} = \frac{(1 - 2K \langle \mathbf{x}, \mathbf{y} \rangle_2 - K||\mathbf{y}||_2^2)\mathbf{x} + (1 + K||\mathbf{x}||_2^2)\mathbf{y}}{1 - 2K \langle \mathbf{x}, \mathbf{y} \rangle_2 + K^2||\mathbf{x}||_2^2||\mathbf{y}||_2^2} \tag{2.1}$$

$$\exp_{\mathbf{x}}^K(\mathbf{v}) = \mathbf{x} \oplus (\tan_K(\frac{\sqrt{|K|}\lambda_{\mathbf{x}}^K||\mathbf{v}||_2}{2})\frac{\mathbf{v}}{\sqrt{K}||v||_2}) \tag{2.2}$$

$$\log_{\mathbf{x}}^K(\mathbf{v}) = \frac{2}{\sqrt{|K|}\lambda_{\mathbf{x}}^K} \tan_K^{-1}(\sqrt{|K|}|| - \mathbf{x} \oplus_K \mathbf{v}||_2)\frac{-\mathbf{x} \oplus_K \mathbf{v}}{||\mathbf{x} \oplus_K \mathbf{v}||_2} \tag{2.3}$$

$$\mathbf{M} \otimes_K \mathbf{x} = \exp_{\mathbf{0}}^K(\mathbf{M} \log_{\mathbf{0}}^K(\mathbf{x})) \tag{2.4}$$

$$d_{\mathcal{M}_K}(\mathbf{x}, \mathbf{y}) = \frac{2}{\sqrt{|K|}} \tan_K^{-1}(\sqrt{|K|}|| - \mathbf{x} \oplus_K \mathbf{y}||_2) \tag{2.5}$$

, where K denotes the curvature of the geometry, $\tan_K = \tan$ if $K > 0$ and $\tan_K = \tanh$ if $K < 0$.

2.2 Knowledge Graphs

There has long been the idea of organizing the world's knowledge in a graphical format. Richens [74] developed the semantic network in the 1950s to represent semantic relations between concepts in a network for machine translation of natural languages. Other researchers, such as Quillian and Collins, further contribute to the semantic network and apply it in different projects. The term knowledge graph itself was coined in the 1970s to discuss how to build modular instructional systems for courses [79]. Later, the distinction between knowledge graphs and semantic networks was diluted. In 2012, Google presented the Google Knowledge Graph [80] that achieved great success in enhancing the Google search engine. After that, the term KG gained tremendous attention. In this section, we provide the fundamentals of knowledge graphs and the relational learning techniques on KGs. This section is not meant to be a survey but rather to introduce important concepts, which will be extended for temporal knowledge graphs in later sections.

2.2.1 Fundamentals of Knowledge Graphs

A knowledge graph is a directed graph with labeled edges, where each edge corresponds to a semantic fact. In this work, we focus on KGs with the following definition:

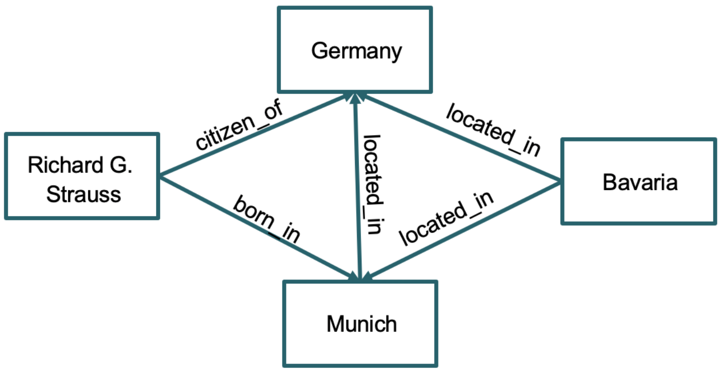


Figure 2.1: A fragment of a semantic knowledge graph (TBD).

Definition 1 *(Knowledge Graph). Let $\mathcal{E}$ and $\mathcal{R}$ represent a finite set of entities and relations, respectively. An entity in $\mathcal{E}$ may correspond to an instance or a class. A KG is given by $\mathcal{KG} \subset \mathcal{E} \times \mathcal{R} \times \mathcal{E}$, which is a collection of semantic facts written as triples. A triple $q = (e_s, p, e_o)$ represents a labeled edge between a subject entity $e_s \in \mathcal{E}$ and an object entity $e_o \in \mathcal{E}$, where $p \in \mathcal{R}$ denotes the edge type (relation).*

As specified in the above definition, the collection of triples is represented as a directed multi-relational graph. The entities correspond to nodes, and directed edges represent the relations between entities, where a subject entity is the source node of a directed edge and the object entity the target node. For example, we can encode that Munich is located in Germany as a triple *(Munich, located in, Germany)* with the subject entity *Munich*, the relation *located in*, and the object entity *Germany*. Note that a knowledge graph may contain multiple edges with different relation types between two entities. Fig. 2.1 shows a small fragment of a KG, including the fact mentioned above.

Large KGs are constructed either manually by the crowd, e.g., Wikidata [92] and Freebase [10], or automatically using information extraction methods, e.g., DBpedia [2] and NELL [14]. However, no matter how a KG is built, it usually suffers from incompleteness in the sense that it does not contain all golden facts but only a subset. Thus, many relational learning approaches are proposed to predict missing links based on observed facts, also known as KG completion.

Definition 2 *(Knowledge Graph Completion) Let $\mathcal{F}$ represent the set of all quadruples that are facts, i.e., true triples. The knowledge graph completion is the problem of inferring $\mathcal{F}$ based on a set of observed facts $\mathcal{O} \subset \mathcal{F}$. Specifically, the task of KG completion is to predict entities $e \in \mathcal{E}$ given either a subject-relation-pair $(e_s, p) \in \mathcal{E} \times \mathcal{R}$ (object prediction) or a relation-object-pair $(p, e_o) \in \mathcal{E} \times \mathcal{R}$ (subject prediction).*

Knowledge graph completion is commonly framed as a ranking task. Taking the object prediction as an example, the relation learning algorithms consider all entities in $\mathcal{E}$ and learn a score function $\phi : \mathcal{E} \times \mathcal{P} \times \mathcal{E} \rightarrow \mathbb{R}$. The score function assigns a score to each triple t, which indicates the plausibility that t corresponds to a fact. Thus, the proper object can be inferred by ranking the scores of the triple $\{(e_s, p, e_{o_i}), e_{o_i} \in \mathcal{E}\}$ that consist of candidate entities and the given subject-predicate pair. As a real example, we consider finding the citizenship of Angela Merkel, i.e., the task (*Angela Merkel*, *citizen of*, *?*). We would expect from relational learning model to score the triple (*Angela Merkel*, *citizen of*, *Germany*) higher than others such as (*Angela Merkel*, *citizen of*, *Schleswig-Holstein*). Besides extending existing KGs, a plethora of AI tasks such as recommendation [40] and question answering [82] can be framed as predicting links in a KG.

A common practice in relational learning on knowledge graphs is to add inverse relations. For any relation $p \in \mathcal{R}$, we denote with p^{-1} the corresponding inverse relation, i.e., (e_s, p, e_o) *is true* $\iff$ (e_o, p^{-1}, e_s) *is true*. Adding inverse triples doubles the number of relations and triples and allow learning two separate representations for each original relation, which shows a beneficial impact in recent publications [51, 26].

2.2.2 Relational Learning on Knowledge Graphs

This section provides an overview of relational learning methods of knowledge graphs. Specifically, we focus on **representation learning approaches** in this thesis, which has become the dominant paradigm for relational learning on KGs over the past years. Representation learning aims to learn compact and numerical representations of the data that contains useful information, such as structural properties, for training classifiers or other predictors [6]. An embedding is a low-dimensional, learnable continuous vector representation and can be trained from scratch via backpropagation. Thus, learning embedding does not require any features and can be applied broadly in various settings. The problem of learning embeddings of entities $\mathcal{E}$ and relations $\mathcal{R}$ can be formulated as follows:

Definition 3 *(Embedding of Entities and Relations). Given a knowledge graph* $\mathcal{KG} = (\mathcal{V}, \mathcal{E}, \mathcal{O})$*, the problem of learning entity embeddings and relation embeddings is concerned with learning a mapping that provides a vector representation* $\mathbf{e} \in \mathbb{R}^d$ *and* $\mathbf{p} \in \mathbb{R}^d$ *for each entity* $e \in \mathcal{E}$ *and relation* $p \in \mathcal{R}$*, respectively, where* $d \ll |\mathcal{E}|$*.*

Overall, the KG representation learning approaches consist of an encoder that learns entity and relation representations and a decoder, which passes these representations to a score function to score triples. Usually, the representations of entities and relations are initialized randomly and optimized during training. As a preparation for the following chapters, we review tensor decomposition techniques, translation-based embedding models, and relational graph neural networks, which are essential representation learning approaches and play an important role in our published work.

Tensor Decomposition

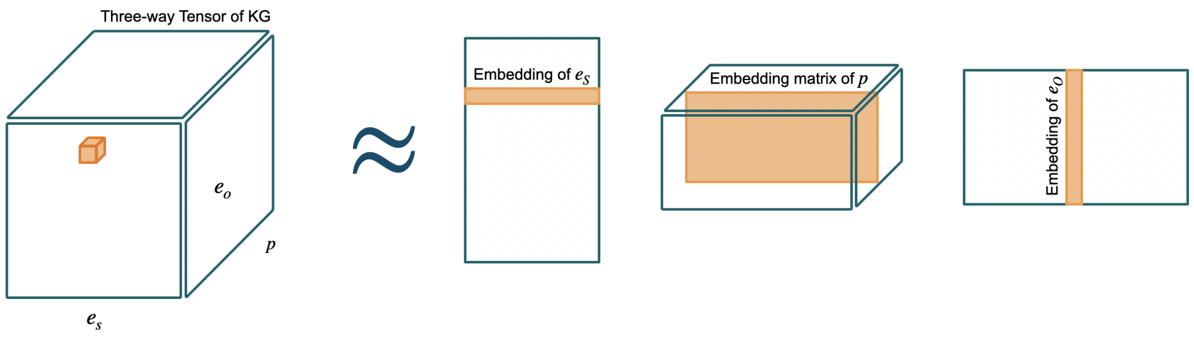


Figure 2.2: A visualization of the low-rank tensor decomposition performed by RESCAL (depiction based on a figure in [39]).

A knowledge graph can be represented as a binary three-way tensor $\mathbf{X} \in \{0,1\}^{N_e \times N_p \times N_e}$, where each entry indicates the truth (1) or falseness (0) of a triple. The main idea of tensor decomposition is to compute a low-rank decomposition of this tensor that can well capture the global patterns in the knowledge graph. One of the first tensor factorization approaches is Canonical Polyadic (CP) decomposition, proposed in 1927 [42]. When applied to KGs, CP learns one embedding for each relation $p \in \mathcal{R}$ and two embeddings for each entity $e \in \mathcal{E}$. One captures the entity's behavior when being the subject of a triple, and one captures the entity's behavior when being the object of a triple. However, CP learns the two embeddings of each entity independently of each other, leading to poor performance on KG completion [86].

In contrast to CP, the RESCAL model [66] associates each entity $e \in \mathcal{E}$ with a single embedding vector $\mathbf{e} \in \mathbb{R}^d$ no matter at what position of a triple they appear. As shown in Figure 2.2, RESCAL employs a matrix $\mathbf{R} \in \mathbb{R}^{d \times d}$ for each relation $p \in \mathcal{R}$ and computes a bi-linear form induced by the relation matrix with a subject entity $e_s \in \mathcal{E}$ and an object entity $e_o \in \mathcal{E}$ as follows,

$$\phi_{RESCAL}(e_s, p, e_o) = \mathbf{e}_s^T \mathbf{R} \mathbf{e}_o \tag{2.6}$$

Hence, the latent dimension of relation embeddings is quadratic to the dimensionality of entity embeddings, such that RESCAL inherently inclines to overfit. DistMult [101] addressed this problem by constraining the embedding matrices of relations to be diagonal and achieved great performance. However, DistMult cannot model asymmetric relations as the score function of DistMult is symmetric. SimplE [51] offers a solution to this limitation by presenting a simple enhancement of CP decomposition. As mentioned previously, CP generally performs poorly on KG completion as it learns two independent embeddings $\mathbf{e}_{i,s}, \mathbf{e}_{i,o}$ for each entity $e_i \in \mathcal{E}$, whereas they are tied in fact. SimplE allows these two embeddings of each entity to be learned dependently by taking advantage of the inverse relations. Specifically, SimplE considers two vectors $\mathbf{p}, \mathbf{p}^{-1} \in \mathbb{R}^d$ for each relation $p \in \mathcal{R}$, which leads to

$$\phi_{SimplE}(e_i, p, e_j) = \frac{1}{2}\big((\mathbf{e}_{i,s} \odot \mathbf{p}) \cdot \mathbf{e}_{j,o} + (\mathbf{e}_{j,s} \odot \mathbf{p}^{-1}) \cdot \mathbf{e}_{i,o}\big) \tag{2.7}$$

By addressing the independence of the entity vectors using the inverse relations, SimplE achieves superior performance on the KG completion task despite its simplicity.

Translation-based Embedding Models

Another line of work is translation-based approaches that define additive functions over embeddings. Bordes et al. [11] proposed the first translation-based model, i.e., TransE, which projects both entities and relations into the same vector space and interprets relations as translations between subject entity embedding and object entity embedding. In other words, if (e_s, p, e_o) holds, the embedding of the object entity e_o should be close to the embedding of the subject entity e_s plus the embedding of the relationship p as follows,

$$f_{TransE}(e_s, p, e_o) = d^2(\mathbf{e}_s + \mathbf{p}, \mathbf{e}_o), \tag{2.8}$$

where $\mathbf{e}_s \in \mathbb{R}^d$ and $\mathbf{e}_o \in \mathbb{R}^d$ are embedding vector for e_s and e_o, respectively, $\mathbf{p}$ denotes the embedding vector of the predicate p, $d(\cdot)$ is the Euclidean distance function. Note that $f_{TransE}(e_s, p, e_o)$ is expected to be lower for a golden triple and higher for a invalid triple, which is different to a score function $\phi(\cdot)$ that is lower for a invalid triple and higher for a golden triple. To distinguish them, we use $f(\cdot)$ to denote the interaction function of translation-based models, e.g., TransE. The motivation behind the translation-based modeling is that relation between entity pairs manifests a common vector offset, such as "*man* is to *woman* as *king* is to *queen*". However, TransE is not able to deal with symmetric as well as one-to-many/many-to-one/many-to-many relations. In other words, the relations cannot be well parameterized if there are multiple relations between two entities. To this end, Wang et al. [94] proposed TransH that associates each entity with distributed representations when involved in different relations. Specifically, if there is a relation p between subject entity e_s and object entity e_o, the embedding of e_s and e_o are first projected to a relation-specific hyperplane $\mathbf{w}_p$ (the normal vector), and then connected by a translation vector $\mathbf{d}_p$ on the hyperplane as follows,

$$f_{TransH}(e_s, p, e_o) = ||(\mathbf{e}_s - \mathbf{w}_p^T \mathbf{e}_s \mathbf{w}_p) + \mathbf{d}_p - (\mathbf{e}_o - \mathbf{w}_p^T \mathbf{e}_o \mathbf{w}_p))||_2. \tag{2.9}$$

Thus, TransH enables different roles of an entity in different relations by introducing relation-specific hyperplanes.

Graph Neural Network

Besides traditional graph embedding models that employ a simple embedding lookup, substantial efforts have been devoted to applying deep learning techniques to learning expressive graph representations. There have been several attempts in the literature and can be categorized into two major classes, i.e., spectral-based graph neural networks (GNNs) and spatial-based graph neural networks. The former focuses on developing graph convolutions based on graph Fourier transforms [13, 38, 19, 52], and the latter employs message-passing heuristics between neighboring nodes based on spatial convolutions [77, 1, 69].

Most studies focus on modeling uni-relational graphs, and thus, cannot be directly applied to modeling abundant relations on KGs. To transfer graph neural networks to KGs, **R-GCN** [78] introduced relation-specific weight matrices into the message-passing mechanism as follows,

$$\mathbf{h}_{e_i}^{l+1} = \sigma\left(\sum_{p\in\mathcal{R}} \sum_{e_j\in\mathcal{N}_{e_i}^p} \frac{1}{c_{e_i,p}} \mathbf{W}_p^l \mathbf{h}_{e_j}^l + \mathbf{W}_0^l \mathbf{h}_{e_i}^l\right), \tag{2.10}$$

where $\mathbf{h}_{e_i}^{l+1} \in \mathbb{R}^{d^{l+1}}$, $\mathbf{h}_{e_i}^l \in \mathbb{R}^{d^l}$ denotes the hidden representation of the entity e_i in the l-th layer and $(l+1)$-th layer of the neural network, respectively, with d^l being the dimensionality of the representations in this layer. $\mathcal{N}_{e_i}^p$ represents the set of neighbors of the entity e_i under relation $p \in \mathcal{R}$. $h_{e_j}^l$ denotes the representation of the entity e_j at the l-th layer. $\mathbf{W}_p^l \in \mathbb{R}^{d^l\times d^{l+1}}$ corresponds to a relation-specific linear transformation in terms of the relation p, and $\mathbf{W}_0^l \in \mathbb{R}^{d^l\times d^{l+1}}$ corresponds to the self-connection, a special relation type, to ensure that the hidden representation of an entity at the $(l+1)$-th layer is informed by the corresponding representation at the l-th layer. $c_{e_i,p}$ is a problem-specific normalization constant that can either be learned or chosen in advance. $\sigma(\cdot)$ denotes an element-wise activation function. Intuitively, Equation 2.10 sums up transformed representation vectors of neighboring entities (nodes), followed by normalization. In practice, multiple layers can be stacked to allow for receiving messages from multi-hop neighbors.

An essential issue of Equation 2.10 is the rapid growth in the number of parameters with the number of relations in the KG, easily leading to overfitting. Schlichtkrull et al. [78] introduced basis- and block-diagonal-decomposition to regularize the weights of R-GCN layers. Additionally, CompGCN [89] handled the over-parameterization issue by leveraging entity-relation composition operations inspired by knowledge graph embedding techniques. The message passing function of **CompGCN** is given as

$$\mathbf{h}_{e_i}^{l+1} = \sigma\left(\sum_{p\in\mathcal{R}}\sum_{e_j\in\mathcal{N}_{e_i}^{p}} \mathbf{W}_O^l \phi\left(\mathbf{h}_{e_j}^l, \mathbf{h}_p^l\right) + \sum_{p\in\mathcal{R}}\sum_{e_j\in\mathcal{N}_{e_i}^{p^{-1}}} \mathbf{W}_I^l \phi\left(\mathbf{h}_{e_j}^l, \mathbf{h}_{p^{-1}}^l\right) + \mathbf{W}_0^l \phi\left(\mathbf{h}_{e_i}^l, \mathbf{h}_0^l\right)\right), \tag{2.11}$$

where $\mathbf{h}_p^l \in \mathbb{R}^{d^l}$ denotes the hidden representation of relation p at the l-th layer. $\mathcal{N}_{e_i}^p$ represents the set of neighbors of entity e_i under ordinary relation $p \in \mathcal{R}$. $\mathcal{N}_{e_i}^{p^{-1}}$ represents the set of neighbors of entity e_i under inverse relation $p^{-1} \in \mathcal{R}_{inv}$. $\mathbf{W}_O^l$ and $\mathbf{W}_I^l$ are shared among ordinary and inverse relations, respectively, making the model more parameter efficient than R-GCN and can scale with the increasing number of relations. $\mathbf{h}_0^l$ represents the hidden representation of the self-loop relation at the l-th layer. $\phi : \mathbb{R}^{d^l} \times \mathbb{R}^{d^l} \to \mathbb{R}^{d^l}$ is a non-parameterized composition operator, such as subtraction [11] and multiplication [101].

Recently, attention mechanisms have succeeded in various tasks, e.g., language modeling [20], image recognition [21], and graph learning [91]. Specifically, Veličković et al. [91] leveraged masked self-attentional layers to enable graph neural networks to weigh the importance of messages from different neighbors, called graph attention networks (GAT). Xu et al. [100] extended the graph attention mechanism to knowledge graph representation learning by constructing attention-induced subgraphs. In particular, the proposed framework, called DPMPN, consists of an inattentive GNN (IGNN) that runs full message passing over the entire KG to acquire features from a global view, and an attentive GNN (AGNN) runs on each input-dependent subgraph. **IGNN** applies a standard message passing mechanism [25],

$$\mathbf{h}_{e_i,IGNN}^{l+1} = \sum_{(e_j,p)\in\mathcal{N}_{e_i}} \frac{1}{\sqrt{N(e_i)}} \delta_{IGNN}(\psi_{IGNN}(\mathbf{h}_{e_j,IGNN}^l || \mathbf{p} || \mathbf{h}_{e_i,IGNN}^l) || \mathbf{h}_{e_i,IGNN}^l || \mathbf{e}_i) + \mathbf{h}_{e_i,IGNN}^l \tag{2.12}$$

where $\mathcal{N}_{e_i}$ is the set of immediate neighbors of e_i for its outgoing edges. $N(e_i)$ represents the number of neighbors that send messages to e_i. Functions δ_{IGNN} and ψ_{IGNN} are implemented by a two-layer MLP. In contrast to IGNN, AGNN is input-dependent, where the message passing is running on subgraphs, each conditioned on an input query. To construct an input-dependent subgraph for query $(e_q, p_q, ?)$, **AGNN** starts from the query subject e_q and uses a temporal sampling procedure to add neighbors of e_q into the subgraph. The hidden representations of entities are computed as follows,

$$\tilde{\mathcal{M}}^l_{e_i,AGNN} = \sum_{(e_j,p)\in\mathcal{N}^{l'}_{e_i}} \frac{1}{\sqrt{N^{l'}_{e_i}}} \psi_{AGNN}\left(\mathbf{h}^l_{e_i,AGNN}||\mathbf{c}_p(e_i)||\mathbf{h}^l_{e_j,AGNN})\right) \quad (2.13)$$

$$\mathbf{h}^{l+1}_{e_i,AGNN} = \mathbf{h}^l_{e_i,AGNN} + \delta_{AGNN}\left(\mathbf{h}^l_{e_i,AGNN}||\tilde{\mathcal{M}}^l_{e_i,AGNN}||a^{l+1}_{e_i}\mathbf{W}\mathbf{h}^{l+1}_{e_i,IGNN}||\mathbf{e}_q||\mathbf{p}_q\right), \quad (2.14)$$

where $\tilde{\mathcal{M}}^l_{e_i,AGNN}$ denotes the aggregated message from neighbors of e_i on the subgraph conditioned on the input query. $N^{l'}_{e_i}$ represents the number of neighbors in the subgraph that send messages to e_i. Functions δ_{AGNN} and ψ_{AGNN} are implemented by MLPs. $\mathbf{c}_p(e_i) = \mathbf{p}||\mathbf{e}_q||\mathbf{p}_q$ represents a relation-specific context vector of the input query and is defined by the query subject and relation embeddings, i.e., $\mathbf{e}_q$ and $\mathbf{p}_q$. Besides, the hidden representation passed to AGNN from IGNN is weighted by a scalar attention score $a^{l+1}_{e_i}$ that guides and prunes the message passing, making it scalable for large-scale knowledge graphs. The iteratively and selectively constructed input-dependent subgraph models a sequential reasoning process and can be seen as a graphical interpretation of the final prediction.

Our published works employ relational graph neural networks to perform link prediction on temporal knowledge graphs. In particular, in Chapter 3, we develop an interpretable subgraph reasoning approach inspired by DPMPN [100]; in Chapter 4, we extend CompGCN [89] to modeling temporal knowledge graphs by proposing a multi-relational graph neural ordinary differential equation.

2.3 Fundamentals of Temporal Knowledge Graphs

Common knowledge graphs assume that the relations between entities are time-invariant and store facts in their current state. In reality, however, multi-relational data may evolve over time, e.g., (*Alice*, *live in*, *New York*) becomes invalid after Alice moves to Munich. Besides, the recent availability of a large amount of event-based interaction data exhibits eventive relationships that are only valid at certain timestamps, i.e., the economical relationship between South Korea and Japan intensified in 2019 due to political conflicts. To accommodate such time-dependent multi-relational data, temporal knowledge graphs (tKGs) have been introduced, where a triple is extended with a timestamp or time range indicating when the triple is valid, e.g., (*South Korea*, *downgrades trade ties with*, *Japan*, *2019-08-12*). Figure 2.3 shows an example of tKGs. In this section, we introduce the fundamentals of tKGs. To distinguish a knowledge graph with static facts from temporal knowledge graphs, we refer to the former as *semantic knowledge graphs*.

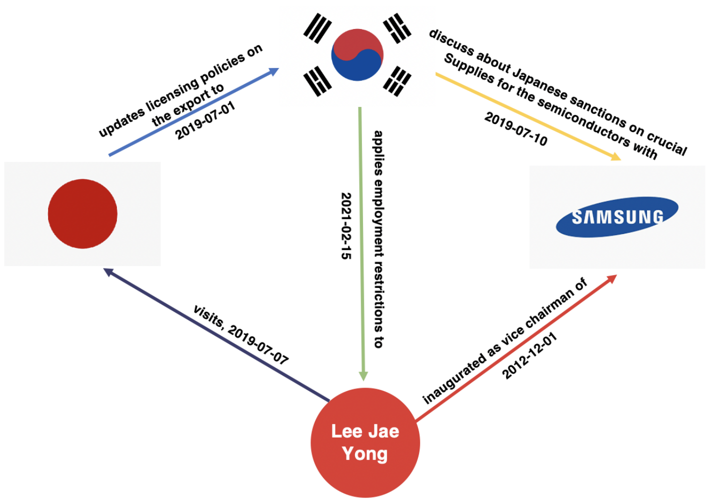

updates licensing policies on the export to
2019-07-01
discuss about Japanese sanctions on crucial Supplies for the semiconductors with
2019-07-10
applies employment restrictions to
2021-02-15
SAMSUNG
visits, 2019-07-07
inaugurated as vice chairman of
2012-12-01
Lee Jae Yong

Figure 2.3: A fragment of a temporal knowledge graph

In contrast to semantic knowledge graphs, a temporal knowledge graph is a directed graph with *timestamped* labeled edges. Each edge characterizes a temporally valid relation between two entities (nodes). In literature, temporal knowledge graphs can be modeled in two ways, i.e., continuous-time temporal knowledge graphs and discrete-time temporal knowledge graphs [50]. A continuous-time temporal knowledge graph consists of a static graph $\mathcal{G}_{init}$ representing an initial state of the multi-relational data at time t_0 and an observation set containing tuples in the form of *(event type, event, timestamp)*, where the event type can be an edge addition/deletion, node addition/deletion, etc. A discrete-time temporal knowledge graph is a sequence of snapshots from time-evolving multi-relational data sampled at regularly-spaced times. Since available event databases, e.g., Integrated Crisis Early Warning System (ICEWS) [95] and Global Database of Events, Language, and Tone (GDELT) [57], are collected with regularly-spaced time annotations, we focus on the discrete-time temporal knowledge graphs with the following definition:

Definition 4 *(Temporal Knowledge Graph). Let $\mathcal{E}$ and $\mathcal{R}$ represent a finite set of entities and relations, respectively. A temporal knowledge graph is a sequence of graph snapshots, i.e., $tKG = \{\mathcal{KG}_1, \mathcal{KG}_2, ..., \mathcal{KG}_t, ..., \mathcal{KG}_T\}$, where $\mathcal{KG}_t \subset \mathcal{E} \times \mathcal{R} \times \mathcal{O} \times \mathcal{T}\}$. A quadruple*

$q = (e_s, p, e_o, t)$ *represents a timestamped and labeled edge between a subject entity* $e_s \in \mathcal{E}$ *and an object entity* $e_o \in \mathcal{E}$ *with a relation* $p \in \mathcal{R}$ *in graph snapshot* $\mathcal{KG}_t$. *Thus, a temporal knowledge graph can also be represented by a collection of quadruples.*

Like semantic knowledge graphs, real-world tKGs are often inherently incomplete. Specifically, typical temporal knowledge graphs, such as GDELT and ICEWS, were built from unstructured textual data using automated information extraction methods. A considerable amount of information was lost in the extraction procedure. To this end, the task of temporal knowledge graph completion has gained growing interest.

Definition 5 *(Temporal Knowledge Graph Completion) Let* $\mathcal{F}$ *represents the set of all quadruples that are golden facts, i.e., true quadruples. The temporal knowledge graph completion is the problem of inferring* $\mathcal{F}$ *based on a set of observed facts* $\mathcal{O} \subset \mathcal{F}$. *Specifically, the task of tKG completion is to predict either a missing subject entity* $(?, p, e_o, t)$ *given the other three components or a missing object entity* $(e_s, r, ?, t)$.

Besides, humans are always interested in looking into the past to predict the future. This is aligned with the temporal knowledge graph forecasting task that aims to predict unknown links at future timestamps based on observed past events.

Definition 6 *(Temporal Knowledge Graph Forecasting). Let* $\mathcal{F}$ *represents the set of all ground-truth quadruples, and let* $(e_q, p_q, e_o, t_q) \in \mathcal{F}$ *denote the target quadruple. Given a query* $(e_q, p_q, ?, t_q)$ *derived from the target quadruple and a set of observed prior facts* $O = \{(e_i, p_k, e_j, t_l) \in \mathcal{F} | t_l < t_q\}$, *the temporal KG forecasting task is to predict the missing object entity* e_o. *Specifically, we consider all entities in* $\mathcal{E}$ *as candidates and rank them by their likelihood of forming a golden quadruple together with the given subject-predicate-pair at timestamp* t_q.

Similar to knowledge graph completion, temporal knowledge graph completion and forecasting are also framed as ranking tasks. Taking the object prediction $(e_s, p, ?, t)$ as an example, the relation learning algorithms consider all entities in $\mathcal{E}$ and learn a score function that assigns a plausibility score to each quadruple. Thus, the proper object can be inferred by ranking the scores of the quadruples $\{(e_s, p, e_{o_i}, t), e_{o_i} \in \mathcal{E}\}$ consisting of a candidate entity and the given subject-predicate-timestamp triple. As a real example, we consider finding which country Catherine Ashton will visit on Nov. 09, 2014, i.e., (*Catherine Ashton*, *make a visit*, ?, *2014-11-09*). We expect temporal relational learning models to score the golden quadruple (*Catherine Ashton*, *make a visit*, *Oman*, *2014-11-09*) higher than others, such as (*Catherine Ashton*, *make a visit*, *Germany*, *2014-11-09*). Besides tKG completion and forecasting, tKG embedding models can power a plethora of downstream tasks, such as time-aware recommendation [71] and temporal question answering [45], by improving the performance of existing models using structured external knowledge.

2.4 Relational Learning on Temporal KGs

Temporal knowledge graph models can be described from an encoder-decoder framework. The encoder produces time-dependent embeddings that capture the evolving features of temporal knowledge graphs, while the decoder is usually a score function from canonical semantic KG models introduced in Section 2.2.2 to examine the plausibility of a given quadruple. In the following, we first review techniques for encoding sequential or time information relevant to our publications and then review benchmark temporal knowledge graph embedding models for both the completion task and forecasting task.

2.4.1 Temporal Encoding

In problems involving time, the input can be viewed as a sequence of observations sampled at regular intervals (synchronous time-series) or irregular intervals (asynchronous time-series). Modeling sequential data has a long history and achieved great success in many fields, ranging from finance to traffic prediction. Classic approaches in the time-series analysis are auto-regressive models that predict the possible future observations based on a finite window into the past, such as ARIMA [29]. The classical approaches are not able to model long-distance dependencies. Besides, they have difficulties dealing with multi-dimensional data. Another line of work in modeling sequential data is so-called state-space models, e.g., Hidden Markov Model [5], Kalman filter [48], Dynamic Bayesian Networks [63], where the model's output is generated from time-evolving hidden states. State-space models have been widely used for sequence modeling in the past decades. They are superior to classical approaches in many respects, such as handling multi-variate inputs and easily adding prior knowledge [63].

However, state space models also have difficulties in modeling long-range dependencies and complex temporal behaviors. Another popular class is recurrent neural networks (RNNs). RNNs parameterize the distribution of each observation by a neural network, where the hidden state at the current step is derived from both the input and the previous hidden state at the last step. However, RNNs often suffer from the problem of vanishing gradients when using the backpropagation through time algorithm to compute the gradient, and thus, lack the ability to capture long-term temporal dependencies. Long Short-Term Memory (LSTM) [43] solved this issue by introducing three gate functions to control the information flow and became one of the most powerful sequence models. In particular, the gate functions consist of a linear layer and a sigmoid activation function. Thus, the outputs are between zero and one and are used to update and control the cell state by deciding what information to forget, input, and output. LSTMs made significant progress in what we can accomplish with RNNs and achieved remarkable results on a plethora of sequential tasks.

Recently, Transformer [90] achieved outstanding success in various sequence modeling problems. It can model dependencies without regard to their distance in the input sequence by applying the self-attention mechanism. However, either Transformer or recurrent neural networks do not explicitly treat time itself as a feature and thus capture sequential signals rather than temporal patterns. In the following, we introduce neural ordinary differential equations and two time-embedding approaches that explicitly account for the time span between data points and have been used in our publications.

Neural Ordinary Differential Equations

Most existing neural networks for time-series analysis models are entirely or partially discrete, resulting in discontinuous latent states and considerable errors in modeling latent temporal dynamics. In contrast to canonical neural networks, neural ordinary differential equations (ODEs) [16] enable neural networks to have continuous depth by parameterizing the derivative of the latent states instead of specifying a discrete sequence of hidden layers. It can be considered a continuous analog of the residual neural network [36], which is an Euler discretization of ordinary differential equations. By coinciding the depth domain of neural ODEs with the time domain, neural ODEs can be adapted to learn the continuous fine-grained temporal dynamics of time series such as medical records and network traffic.

In neural ODEs, the continuous dynamics of hidden states in a neural network is parameterized using an ordinary differential equation (ODE)

$$\frac{d\mathbf{h}(t)}{dt} = f(\mathbf{h}(t), t, \boldsymbol{\theta}), \tag{2.15}$$

where $\mathbf{h}(t) \in \mathbb{R}^d$ denotes the hidden state of a dynamic system at time t, f represents a neural network that models the derivative of the hidden state regarding time, and $\boldsymbol{\theta}$ denotes the parameters in the neural network that are gradually updated during training. Starting from the input layer $\mathbf{h}(t = 0)$, the output layer $\mathbf{h}(T)$ is defined as the solution to this ODE initial value problem at time T. The output can be calculated using an ordinary differential equation solver:

$$\mathbf{h}(T) = \mathbf{h}(0) + \int_{t=0}^{T} f(\mathbf{h}(t), t, \boldsymbol{\theta})dt = \text{ODESolver}(\mathbf{h}(0), f, 0, T, \boldsymbol{\theta}). \tag{2.16}$$

The essential challenge of training continuous-depth neural networks is performing backpropagation through the ODE solver while preserving scalability and keeping low memory costs. To address this challenge, Chen et al. [16] computed gradients using the adjoint sensitivity method [72], which computes gradients by solving a second ODE backward in time. An adjoint is defined as the derivative of the loss concerning the hidden state $\mathbf{h}(t)$, whose dynamics is given by another ODE as follows,

$$\frac{d\mathbf{a}(t)}{dt} = -\mathbf{a}(t)^T \frac{\partial f(\mathbf{h}(t), t, \boldsymbol{\theta})}{\partial \mathbf{h}}. \tag{2.17}$$

We can compute $\partial L/\partial \mathbf{h}(0)$ by a backward ODE solver with the initial value $\partial L/\partial \mathbf{h}(T)$. Thus, the gradients of the loss function with respect to parameter $\boldsymbol{\theta}$ can be calculated as follows,

$$\frac{dL}{d\boldsymbol{\theta}} = -\int_{t=T}^{0} \mathbf{a}(t)^T \frac{\partial f(\mathbf{h}(t), t, \boldsymbol{\theta})}{\partial \boldsymbol{\theta}} dt. \tag{2.18}$$

The vector-Jacobian products $\mathbf{a}(t)^T \frac{\partial f}{\partial \mathbf{h}}$ and $\mathbf{a}(t)^T \frac{\partial f}{\partial \boldsymbol{\theta}}$ can be efficiently evaluated by automatic differentiation, whose time cost is similar to that of f.

Gholami et al. [24] argued that the adjoint method might lead to catastrophic numerical instabilities. To this end, they proposed ANODE, which stores input activations at intermediate timestamps during the forward pass in memory. In the backpropagation, the method first performs a forward pass in each interval between a pair of input activations and save intermediate results in memory. Then the results are used to compute the derivatives backward in time. ANODE uses the discretization scheme to solve the reverse ODEs, and thus, does not suffer from possible numerical instability by solving Equation 2.16 backward in time. Besides, Daulbaev et al. [18] proposed an interpolated reverse dynamic method (IRDM) that approximates the hidden state $\mathbf{h}(t)$ through the barycentric Lagrange interpolation (BLI) on a Chebyshev grid [8]. IRDM requires $\mathbf{h}(t)$ can be approximated by BLI with sufficient accuracy, which can only be verified experimentally.

Our published work [32] in Chapter 4 extends neural ODEs to modeling temporal knowledge graphs and proposes a continuous-depth multi-relational graph neural network for forecasting future links. In contrast to canonical tKG models, neural ODEs enable the model to deal with irregular-sampled continuous-time data and estimate the hidden state of tKGs at any timestamp of interest in future.

Time Embedding

Suppose time is a relevant feature; in that case, many recent studies embed time into high-dimensional spaces and feed it into a model as additional input dimensions by concatenating the time embeddings with the input [56, 55, 98, 49, 71]. Since time is a continuous variable, some works discretize it into a set of timestamps and learn an embedding for each discrete timestamp, while some other works learn encoding functions that take continuous time-variables as input and embed them into a vector space. In the following, we introduce the time encoding techniques of both classes.

Embedding Lookup Many works learn shallow time embeddings by employing a simple embedding lookup. The timestamps of all events build a set $\mathcal{T}$. And each timestamp $t \in \mathcal{T}$ is associated with an embedding vector $\mathbf{t} \in \mathbb{R}^d$ that is usually learned from scratch. Leblay et al. [56] incorporated time embedding into the KG models, i.e., TransE [11] and RESCAL [66], and learned their representations in the same vector space as entities and relations. Experiments show that the time embedding-based approach outperforms approaches that use time as a coefficient (scalar) by a large margin. Lacroix et al. [55] enforced the smoothness of temporal embeddings by penalizing the discrete derivative of the time embedding. Thus, neighboring timestamps have similar representations, leading to considerable performance improvements. However, the above time embedding approaches are inherently limited to transductive tasks since they cannot deal with unseen timestamps at the inference time. Thus, some works embed the time interval between two data points instead of embedding absolute time information. For example, Zhu et al. [71] embedded time intervals between target items and selected user behaviors and incorporated temporal distance information into the click-through rate prediction of the recommendation system. Thus, the model is able to perform future predictions at unseen timestamps if the corresponding time interval embedding has been learned in the training phase.

Time Encoding Function The embedding lookup method is suitable for sequential data sampled at regularly-spaced times. However, for sequential data sampled at irregular-spaced times with high sampling frequency, some timestamps are only associated with a small number of data samples, leading that these time embeddings are hardly trained properly. To this end, Xu et al. [98] proposed a mapping function $\Phi : T \to \mathbb{R}^d$ justified by Bochner's Theorem to embed continuous variables from time domain into vector spaces as follows

$$\Phi(t) = \sqrt{\frac{1}{d}}[\cos(\omega_1 t), \sin(\omega_1 t), ..., \cos(\omega_d t), \sin(\omega_d t)] \tag{2.19}$$

, where $T = [t_{min}, t_{max}]$ is an interval of time. In fact, the mapping function takes continuous time-variables as input and generates time representations whose relative positions reflect their temporal difference. This mapping function is tailored for the self-attention mechanism [90] and can be seen as replacing the positional encoding. Extensive experiments on real-world datasets of continuous-time event sequence prediction demonstrate the effectiveness of the proposed mapping function. In addition, Kazemi et al. [49] developed a model-agnostic representation for time, which is

$$\mathbf{t2v}(\tau)[i] = \begin{cases} \omega_i \tau + \phi_i & \text{if } \ i = 0 \\ \sin(\omega_i \tau + \phi_i) & \text{if } \ 1 \leq i \leq d, \end{cases} \tag{2.20}$$

where $\mathbf{t2v}(\tau) \in R^{d+1}$ denotes the embedding of time τ, $\mathbf{t2v}(\tau)[i]$ is the ith element of $\mathbf{t2v}(\tau)$, and $\sin(\cdot)$ is the sine function helping the time encoding to have periodicity. ω_i and ϕ_i are learnable parameters. This encoding method is similar to the Fourier transform approach for decomposing a temporal signal into a set of frequencies. But instead of using a fixed set of frequencies, **t2v** lets frequencies be learned freely.

2.4.2 Temporal Knowledge Graph Models

Temporal knowledge graphs can be seen as a sequence of graph snapshots and exhibit rich temporal dynamics. Thus, the temporal encoding approaches introduced in Section 2.4.1 can be utilized to capture the temporal properties available on temporal knowledge graphs. In particular, most existing temporal knowledge graph models combine the decoder, i.e., the score function, from canonical semantic KG models with temporal encoding techniques. In this section, we discuss specifics of temporal knowledge graph models for both completion and forecasting.

Temporal Knowledge Graph Completion Models

To address the inherent incompleteness of temporal KGs, Tresp et al. [84] proposed the first tKG completion model by decomposing a four-way tensor whose element is associated with subject entity, predicate, object entity, and timestamp as shown in Figure 2.4. In particular, each timestamp is assigned a shallow embedding from an embedding lookup. Ma et al. [60] followed [84] and generalized several semantic KG models, e.g., Tucker [4], RESCAL [66], to tKG models by introducing an embedding lookup for time. Similarly, Leblay et al. [56] extended TransE [11] as follows,

$$f_{TTransE}(e_s, p, e_o, t) = -||\mathbf{e}_s + \mathbf{p} + \mathbf{t} - \mathbf{e}_o||_2, \tag{2.21}$$

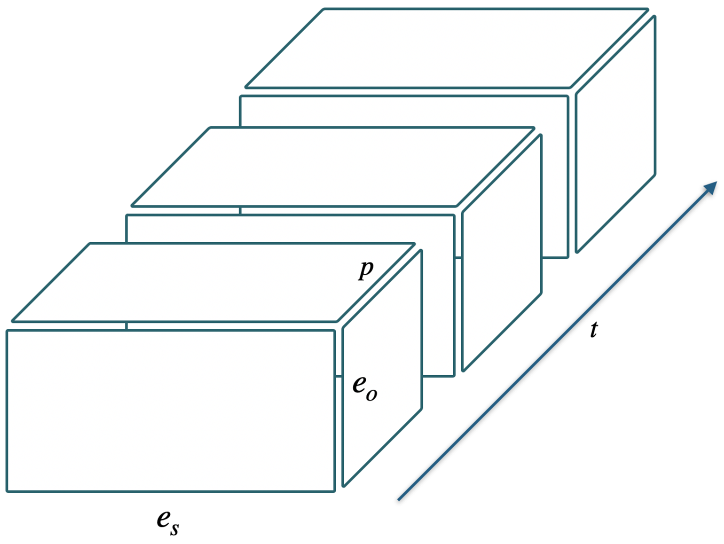


Figure 2.4: The figure shows a four-way tensor representing temporal knowledge graphs (depiction based on a figure in [84]).

where $\mathbf{t}$ denotes time embedding learned from a simple embedding lookup. Similarly, Lacroix et al. [55] applied embedding lookup as the time encoder and extended ComplEx [87] to modeling temporal knowledge graph data, which is

$$\phi_{TComplEx}(e_s, p, e_o, t) = \mathrm{Re}(\langle \mathbf{e}_s, \mathbf{p}, \bar{\mathbf{e}}_o, \mathbf{t} \rangle) \tag{2.22}$$

where $\mathbf{e}_s$, $\mathbf{p}$, $\mathbf{e}_o$, $\mathbf{t} \in \mathbb{C}^d$ are complex-valued vectors, i.e., $\mathbf{e}_s = \mathrm{Re}(\mathbf{e}_s) + i\mathrm{Im}(\mathbf{e}_s)$. $\mathrm{Re}(\mathbf{e}_s) \in \mathbb{R}^d$ and $\mathrm{Im}(\mathbf{e}_s) \in \mathbb{R}^d$ are the real and imaginary parts of $\mathbf{e}_s$, respectively, with $i^2 = -1$. $\bar{\mathbf{e}}_o$ denotes the complex conjugate of $\mathbf{e}_o$, which is $\mathbf{e}_o = \mathrm{Re}(\mathbf{e}_o) - i\mathrm{Im}(\mathbf{e}_o)$. $\langle \cdot \rangle$ denotes the standard componentwise multi-linear dot product $\langle a, b, c, d \rangle = \sum_k a_k b_k c_k d_k$.

Dasgupta et al. [17] associated each timestamp with a hyperplane in the same vector space as entities and relations. For a quadruple (e_s, p, e_o, t), the embedding of e_s, e_o, and p are first projected into the hyperplane of the timestamp t, and then the interaction function of TransE [11] (Equa. 2.8) is applied as the decoder. In fact, different hyperplanes are represented by normal vectors, where each normal vector corresponds to a timestamp and is obtained from an embedding lookup. In particular, the interaction function is as follows

$$\begin{aligned} \mathbf{e}_s(t) &= \mathbf{e}_s - (\boldsymbol{\omega}_t^T \mathbf{e}_s)\boldsymbol{\omega}_t \\ \mathbf{e}_o(t) &= \mathbf{e}_o - (\boldsymbol{\omega}_t^T \mathbf{e}_o)\boldsymbol{\omega}_t \\ \mathbf{p}(t) &= \mathbf{p} - (\boldsymbol{\omega}_t^T \mathbf{p})\boldsymbol{\omega}_t \\ f_{HyTE}(e_s, p, e_o, t) &= ||\mathbf{e}_s(t) + \mathbf{p}(t) - \mathbf{e}_o(t)||_2, \end{aligned} \tag{2.23}$$

where $\mathbf{e}_s$, $\mathbf{e}_o$, and $\mathbf{p} \in \mathbb{R}^d$ are the time-independent embedding of e_s, e_o, and p, respectively. $\boldsymbol{\omega}_t \in \mathbb{R}^d$ denotes the normal vector of the hyperplane corresponding to timestamp t. The translational distance f_{HyTE} of a valid triple (e_s, p, e_o) at time t is minimized. Similarly, Xu et al. [96] embedded temporal knowledge graphs in the complex vector space by defining the temporal evolution of an entity embedding as a rotation. Specifically, entities, predicates, and timestamps are mapped to complex embeddings using a simple lookup. The time-dependent embedding of an entity e_s is defined as

$$\mathbf{e}_s(t) = \mathbf{e}_s \circ \mathbf{t}, \tag{2.24}$$

where $\circ$ denotes the Hermitian dot product between complex vectors, and $\mathbf{e}_s \in \mathbb{C}^d$ represents the time-independent embedding of entity e_s. Then, the translational distance is applied as follows

$$f_{TeRo}(e_s, p, e_o, t) = ||\mathbf{e}_s(t) + \mathbf{p} - \bar{\mathbf{e}}_o(t)||_2. \tag{2.25}$$

Besides, Jung et al. [47] applied a graph attention neural network on the tKG completion task with simple time embedding lookup. In contrast to the approaches mentioned above, Jung et al. encoded the temporal displacement between the query and respective quadruples instead of encoding the timestamps. Given a query $(e_q, p_q, ?, t_q)$ and an observed event $e = (e_s, p, e_o, t_e)$, the temporal displacement is defined as $\delta t_e = t_e - t_q$, whose embedding is taken from an embedding lookup and is learned via gradient-based optimization. With the help of encoding temporal displacement, the proposed model can attend to different types of events that happened before or after the time of interest, depending on specific relations. Taking query relation *member_of_sports_team_until* as an example, the model's attention should be slightly biased toward past events that happened s few years before the time of interest but not too far.

While the above methods use simple time embedding lookup to capture temporal information, there is another line of models that apply temporal encoding functions. Goel et al. [26] proposed a diachronic function to generate entity-specific time embeddings called *DE* and incorporated the time embedding into the entity embedding as additional dimensions. In particular, the entity-specific time embedding $\mathbf{t}(e_i) \in \mathbb{R}^{d_1}$ is defined as

$$\mathbf{t}(e_i) = \mathbf{a}_{e_i} \odot \sin(\boldsymbol{\omega}_{e_i} + \mathbf{b}_{e_i}), \tag{2.26}$$

where $\mathbf{a}_{e_i}, \boldsymbol{\omega}_{e_i}, \mathbf{b}_{e_i}$ are entity-specific amplitude vector, frequency vector, and phase-shift vector, respectively. To obtain a time-dependent embedding of entity e_i at timestamp t, Goel et al. concatenated $\mathbf{t}(e_i) \in \mathbb{R}^{d_1}$ and time-independent entity embedding $\mathbf{e}_i \in \mathbb{R}^{d_2}$, i.e., $\mathbf{e}_i(t) = \mathbf{t}(e_i)||\mathbf{e}_i \in \mathbb{R}^{d_1+d_2}$. This temporal encoding approach is similar to the concept of time2vector (**t2v**) [49] introduced in Section 2.4.1. Besides, it is model-agnostic and can be combined with any score functions, such as SimplE [51], TransE [11], and DistMult [101]. However, since the time embedding is entity-specific, the model parameters scale with the number of entities and may suffer from the overfitting problem when applied to sparse graph data. To this end, our published work in Chapter 5 proposed a time encoding function called *UTEE* that learns a unique time embedding function for all entities, which is

$$\mathbf{t} = \mathbf{a} \odot \sin(\boldsymbol{\omega} + \mathbf{b}), \tag{2.27}$$

where $\mathbf{a}$, $\boldsymbol{\omega}$ and $\mathbf{b}$ are shared among all entities. Thus, the entity embedding at timestamp t is obtained by concatenating the embedding of t and the time-independent entity embedding $\mathbf{e}_i(t) = \mathbf{t}||\mathbf{e}_i$. Notably, the model parameter of *DE* is often more than three times that of *UTEE*. However, experiments show that *UTEE* achieves better performance on sparse datasets.

Inspired from additive time series decomposition, Xu et al. [97] developed time-dependent entity embedding using a trend component vector, a seasonal component vector, and a random component vector, which is

$$\mathbf{e}_i^{ATiSE}(t) = \mathbf{e}_i + \alpha_{e_i}\mathbf{w}_{e_i}t + \boldsymbol{\beta}_{e_i}\sin(2\pi\boldsymbol{\omega}_{e_i}t) + \mathcal{N}(0, \boldsymbol{\Sigma}_{e_i}). \tag{2.28}$$

Specifically, $\mathbf{e}_i + \alpha_{e_i}\boldsymbol{\omega}_{e_i}t$ denotes the trend component, $\boldsymbol{\beta}_{e_i}\sin(2\pi\boldsymbol{\omega}_{e_i}t)$ represents the seasonal component, and the Gaussian noise term $\mathcal{N}(0, \Sigma_{e_i})$ denotes the random component. By applying the multi-dimensional Gaussian distributions, this approach called ATiSE explicitly takes the uncertainty of the temporal evolution modeling. In particular, since the status of an entity at a specific time is not entirely determined by past information, the evolution of entity representations would have randomness. ATiSE uses the translational distance function as the decoder. Since the embeddings are distributions instead of single vectors, the authors used KL divergence as the similarity measure instead of the Euclidean distance.

Temporal Knowledge Graph Forecasting Models

Approaches introduced in the last section are initially tailored for the temporal knowledge graph completion task, which uses observed events to predict missing links at observed timestamps. In comparison, the temporal knowledge graph forecasting task requires models to predict future events based on historical observations, which is more challenging. Trivedi et al. [85] proposed the first tKG forecasting model called KnowEvolve by adapting a multivariate point process. Specifically, they applied the Rayleigh process to model the occurrence of a quadruple (e_s, p, e_o, t) with the following intensity function

$$\lambda_{KnowEvolve}(e_s, p, e_o, t|\bar{t}) = \exp\left(\mathbf{e}_s(t^{e_s}-)^T \mathbf{P} \mathbf{e}_o(t^{e_o}-)\right) * (t - \bar{t}), \tag{2.29}$$

where $t^{e_s}-$ and $t^{e_o}-$ denote the most recent timestamp that e_s and e_o were involved in a quadruple prior to t, respectively, $\bar{t} = \max(t^{e_s}-, t^{e_o}-)$ is defined as the larger one of $t^{e_s}-$ and $t^{e_o}-$. $\mathbf{P} \in \mathbb{R}^{d\times d}$ denotes the weight matrix of relation p. $\mathbf{e}_s, \mathbf{e}_o \in \mathbb{R}^d$ are entity representations obtained from an RNN [76]. The intensity function $\lambda(e_s, p, e_o, t|\bar{t})$ characterizes the expectation of how likely the triple (e_s, p, e_o) would occur at timestamp t conditional on the observations of the point process before t. Since the point process inherently takes the time information into account, there is not necessary to apply other time encoding techniques. At inference time, the model chooses the quadruple with the highest intensity value from all candidates that meet the query requirement as the final prediction.

Unlike classic time series where events happen at different timestamps, a temporal knowledge graph is a sequence of graph slices, and all events (edges) in a graph slice are associated with the same timestamp. However, KnowEolve does not consider such concurrent events that share the same time information. Besides, there is a flaw in their evaluation code [46]. In fact, KnowEvolve underperforms on the tKG forecasting task when applying the fixed evaluation code. Han et al. [34] adapted another point process, i.e., the Hawkes process, to the tKG forecasting task and explicitly took the concurrent events into account. Specifically, Han et al. applied a mean aggregation module to embeddings of all entities involved in the concurrent events and then fed the aggregated embedding into a neural Hawkes process module. Extensive experiments show its superiority compared to KnowEvolve.

Instead of modeling temporal information, Jin et al. [46] took the sequential information into account and employed a recurrent neural network to encode the temporal sequence of past graph slices. Besides, the authors applied Relational Graph Neural Network [78] to model all concurrent events within a graph slice. With the help of a powerful graph neural network and recurrent neural network, the proposed model, RE-Net, becomes a strong baseline for the tKG forecasting task.

Discussion

Temporal knowledge graph models can be categorized into three classes according to the temporal encoder they use, i.e., time embedding (TE), time-dependent entity embedding

(TEE), and deep temporal representation (DTR). TE approaches learn a representation for each discrete timestamp to model the temporal dynamics. In contrast, TEE approaches assume that entities evolve over time, and thus, allow the entity embeddings to drift over time. Additionally, DTR approaches apply various advanced deep learning approaches, i.e., graph neural networks, recurrent neural networks, and transformers, to relational learning on tKGs. The common belief is that TEE approaches are well motivated and more powerful than TE approaches. Thus, many recent works [26, 97, 96] have been devoted to this direction. However, our published work in Chapter 5 found that the primitive TE approaches such as TTransE [56] can outperform the more recent TEE approaches when trained with advanced learning techniques and tuned appropriately. Large-scale empirical experiments with nearly 19000 GPU hours show that training strategies play a significant role in the model's performance and may account for a substantial fraction of the effectiveness of TEE approaches. Thus, future research should raise awareness when developing TEE approaches to ensure whether the time-dependent entity representations are truly helping boost the model's performance.

In the next section, we will introduce the training strategies that are often applied to temporal knowledge graph models for reaching state-of-the-art outcomes.

2.4.3 Training Techniques and Evaluation Metrics

Negative Sampling

Temporal knowledge graphs usually only contain valid quadruples that correspond to true events but do not explicitly incorporate negative information in the form of invalid quadruples. However, learning temporal knowledge graph models requires negative samples. To generate negative samples, the Local Closed World Assumption [7] is often applied, where an observed quadruple (e_s, p, e_o, t) is assumed to be false if (e'_s, p, e_o, t) or (e'_s, p, e_o, t) has been observed. A common implementation is generating a fixed number of negative quadruples by corrupting positive quadruples in a training batch. Borders et al. [11] proposed a procedure to corrupt positive triples, which is extended to corrupt positive quadruples in most tKG representation learning approaches. In particular, It is first randomly decided to corrupt the subject entity or object entity of a positive quadruple (e_s, p, e_o, t). If the subject entity is selected, e_s is replaced with an entity e'_s randomly selected from $\mathcal{E}$ and generate a negative sample (e'_s, p, e_o, t). A similar strategy can be applied to the object entity if selected.

Loss Functions

Several loss functions for training tKG embedding have been introduced in the literature. Here, we review three of them, which are most frequently applied in tKG models.

Pairwise margin ranking (MR) loss is usually used in tKG models with translation-based interaction functions [56, 17], which is

$$\mathcal{L}_{MR} = \sum_{(e_s,p,e_o,t)\in\mathcal{O}} \sum_{(e'_s,p,e'_o,t)\in\mathcal{N}_{(e_s,p,e_o,t)}} \max(\gamma + f(e_s,p,e_o,t) - f(e'_s,p,e'_o,t), 0), \tag{2.30}$$

where $\mathcal{N}_{(e_s,p,e_o,t)}$ denotes the set of corrupted quadruples given a positive quadruple (e_s,p,e_o,t), $f(\cdot)$ represents the interaction function of translation-based models, and $\gamma > 0$ is a margin hyperparameter.

The categorical cross-entropy (CE) loss and the binary cross-entropy (BCE) loss are other two loss functions that are usually applied to bilinear models [55, 23, 30]. The cross-entropy loss applies a softmax function [70] to the score of each quadruple and models the difference between the softmax distribution over all entity candidates and the data distribution, which is

$$\mathcal{L}_{CE} = -\sum_{(e_s,p,e_o,t)\in\mathcal{O}} \log\left(\frac{\exp\left(\phi\left(e_s,p,e_o,t\right)\right)}{\sum_{e'_s\in\mathcal{E}}\exp\left(\phi\left(e'_s,p,e_o,t\right)\right)}\right) + \log\left(\frac{\exp\left(\phi\left(e_s,p,e_o,t\right)\right)}{\sum_{e'_o\in\mathcal{E}}\exp\left(\phi\left(e_s,p,e'_o,t\right)\right)}\right), \tag{2.31}$$

where $\phi(\cdot)$ denotes the score function, $\log(\cdot)$ and $\exp(\cdot)$ represent the logarithmic function and exponential function, respectively.

The binary cross-entropy loss applies a sigmoid function to mapping the score of each quadruple into the interval of $[0,1]$ and uses the cross-entropy between the resulting probability and the quadruple's label as a loss. This is defined as follows,

$$\mathcal{L}_{BCE} = -\sum_{(e_s,p,e_o,t)\in\mathcal{O}} \left(\log\left(\sigma\left(\phi\left(e_s,p,e_o,t\right)\right)\right) + \log\left(\sum_{(e'_s,p,e'_o,t)\in\mathcal{N}_{(e_s,p,e_o,t)}} \left(1-\sigma\left(\phi\left(e'_s,p,e'_o,t\right)\right)\right)\right)\right), \tag{2.32}$$

where $\sigma(\cdot)$ denotes the sigmoid function.

Inverse Relations

Adding inverse relations into tKG embedding training is another training technique that is widely applied in recent works [26, 55]. The idea of inverse relations is to associate each relation with two different embeddings and thus, have different scores for subject entity prediction $(?,p,e_o,t)$ and object entity prediction $(e_s,p,?,t)$. In particular, for a relation $p\in\mathcal{R}$, (e_o,p^{-1},e_s,t) will be true if (e_s,p,e_o,t) holds.

Evaluation Metrics

As introduced in Section 2.3, queries of both tKG completion task and tKG forecasting task are in the form of subject entity prediction and object entity prediction, i.e., $(?, p, e_o, t)$ and $(e_s, p, ?, t)$. Thus, a tKG model needs to predict the subject entity or the object entity for all quadruples in the test set given other three components by ranking all entities from $\mathcal{E}$. Given a test quadruple (e_s, p, e_o, t), let ψ_{e_s} and ψ_{e_o} represent the rank for subject entity e_s and object entity e_o, respectively. The standard evaluation metrics include *mean reciprocal rank* and *Hits@k*$(k \in \{1, 3, 10\})$. MRR is defined as follows,

$$\text{mean reciprocal rank}(MRR) = \frac{1}{2|\mathcal{G}_{\text{test}}|} \sum_{q \in \mathcal{G}_{test}} (\frac{1}{\psi_{e_s}} + \frac{1}{\psi_{e_o}}), \tag{2.33}$$

where $\mathcal{G}_{\text{test}}$ denotes the test set. Hits@k is defined as the percentage of times that the true entity candidate appears in the top k of ranked candidates.

However, the above metrics would be flawed if multiple entity candidates hold true. These entity candidates can be ranked above the ground truth entity from the test quadruple, but this should not be seen as an error. To prevent such a misleading result, Bordes et al. [11] proposed a filtered version of evaluation metrics for *semantic knowledge graphs* that removes from the list of corrupted triples all the **triples** that appear either in the training, validation, or test set. Some recent works, e.g., Trivedi et al. [85] and Jin et al. [46], use this filtering setting for reporting their results on learning *temporal knowledge graphs*. However, our published work in Chapter 3 argued that this filtering strategy is inappropriate for evaluating temporal KG models. For example, there is a test quadruple (Angela Merkel, visit, Turkey, 2021-10-16), and we perform the object prediction (Angela Merkel, visit, ?, 2021-10-16). Besides, we have observed the quadruple (Angela Merkel, visit, US, 2021-07-15) in the training set. According to the filtering strategy proposed by [11], (Angela Merkel, visit, US) will be considered as a genuine triple and be filtered out when computing MRR and Hits@K because the triple is included in the quadruple (Angela Merkel, visit, US, 2021-07-15) in the training set. However, it is only temporally valid on 2021-07-15 but not on 2021-10-16. Thus, this filtering strategy is not proper for tKG models. Therefore, our work [31] proposed another filtering scheme, which is more appropriate for temporal KG models, called *time-aware filtering.* In this case, the temporal information is taken into account, which means only triples that are genuine at the timestamp of the query will be filtered out. Continuing with the above example, since (Angela Merkel, visit, US, 2021-07-15) does not exist in the dataset, it is considered corrupted and will be filtered out.

2.5 Learning Knowledge Graph Representations on Non-Euclidean Spaces

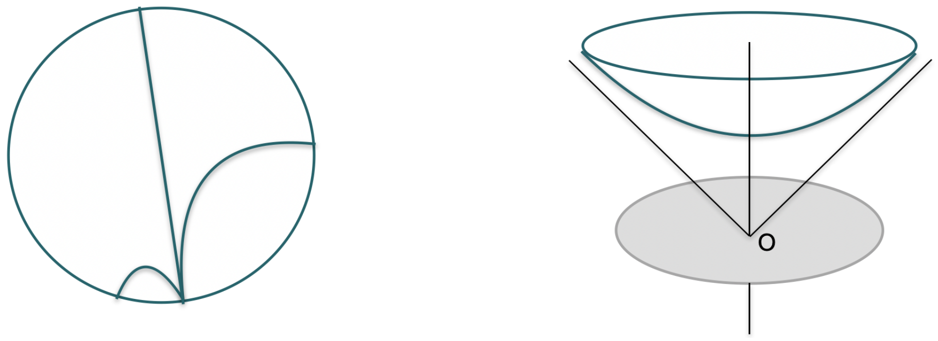


Figure 2.5: The left figure shows the Poincaré disc model with three hyperbolic straight lines (geodesics), which are orthogonal to the disc boundary. The right figure shows the Lorentz model (the hyperboloid model), which is represented by the upper sheet of a two-sheeted hyperboloid.

Like many other networks, knowledge graphs always exhibit power-law or scale-free degree distributions [53], often traced back to hierarchical structures [73]. To provide high-quality embeddings for scale-free networks, their properties have been extensively investigated. In particular, Krioukov et al. [9] showed that scale-free networks naturally emerge in hyperbolic spaces, which was further exploited in various works [67, 27]. Thus, hyperbolic space has the potential to naturally capture the topological information and logical patterns of knowledge graphs. Furthermore, a considerable amount of relations appearing in knowledge graphs exhibit hierarchical or tree-like properties [58]. It is challenging to preserve hierarchical structures in a linear embedding space [64]. In contrast, hyperbolic spaces have shown promise for high-fidelity and parsimonious representations for embedding such hierarchical data [15].

2.5.1 Models of Non-Euclidean Spaces

While the Euclidean geometry fulfills Euclid's axioms [37], Non-Euclidean geometries reject the fifth Euclid's axioms, which states that, for a given point x and a line l_1, there is a unique line l_2 parallel to l_1 passing through x. The curvature of a non-Euclidean geometry describes how much the geometry twist from being flat, where the Euclidean geometry has constant curvature of zero. The hyperbolic geometry is a non-Euclidean geometry with negative curvature. There exist multiple equivalent models of hyperbolic space, where the *Poincaré-ball model* and *Lorentz model* are among the most widely used. We illustrate them in Figure 2.5.

The Poincaré-ball model is a d-dimensional Riemannian manifold $\mathbb{P}^{d,c} = (\mathbb{B}^{d,c}, g_p^c)$, where $\mathbb{B}^{d,c} = \{\mathbf{x} \in \mathbb{R}^d | \; ||x||_2^2 < -\frac{1}{c}\}$ is a d-dimensional unit ball, $|| \cdot ||_2$ denotes the Euclidean norm, and g_p^c is a Riemannian metric tensor

$$g_p^c(\mathbf{x}) = \left(\frac{2}{1 + c||\mathbf{x}||_2^2}\right)^2 g^E, \tag{2.34}$$

where $\mathbf{x} \in \mathbb{B}^{d,c}$ and g^E denote the Euclidean metric tensor [67]. The hyperbolic distance $d^c(\mathbf{x}, \mathbf{y})$ between two points x and y on $\mathbb{B}^{d,c}$ is defined as

$$d(\mathbf{x}, \mathbf{y}) = \operatorname{arccosh}(1 + 2\frac{||\mathbf{x} - \mathbf{y}||_2^2}{(1 - ||\mathbf{x}||_2^2)(1 - ||\mathbf{y}|_2^2|)}). \tag{2.35}$$

Intuitively, the hyperbolic distance on the Poincaré-ball model has locality properties. Specifically, the hyperbolic distance between two points grows very fast when moving the two points from the origin to the boundary of the Poincaré ball, even though their Euclidean distance remains unchanged. Thus, the hyperbolic space is especially suitable for hierarchical relations and tree-like structures, where we can embed the root node near the origin and the leaf node close to the boundary. In a tree structure, the number of child nodes grows exponentially with their distance to the tree's root. While a *high-dimensional* Euclidean space is required to accommodate a large amount of leaf nodes, the hyperbolic space can easily model the tree structure in two dimensions. This is because the hyperbolic disc area and circle length grow exponentially with their radius [54].

The Poincaré ball has the conformal property, which means the angle between two adjacent vectors is the same as that in the Euclidean space. Thus, the Poincaré-ball model is well-suited for gradient-based optimization [68]. However, its distance function may cause numerical instabilities, which is its main drawbacks. In comparison, the Lorentz model avoids such issues. In particular, it defines the hyperbolic space via the Lorentz scalar product $\langle \cdot \rangle_{\mathcal{L}}$ as follows,

$$\mathbb{H}^{d,c} = \{\mathbf{x} \in \mathbb{R}^{d+1} : \langle \mathbf{x}, \mathbf{x} \rangle_{\mathcal{L}} = -(-c)^{\frac{1}{2}}, x_0 > 0\}, \tag{2.36}$$

where $\langle \mathbf{x}, \mathbf{y} \rangle_{\mathcal{L}} = -x_0 y_0 + \sum_{i=1}^d x_i y_i$ and $c > 0$. $\mathbb{H}^{d,c}$ denotes the upper sheet of a two-sheeted hyperboloid in $d+1$ dimensional vector space. The distance function on the Lorentz model is then given as

$$d_l(\mathbf{x}, \mathbf{y}) = \operatorname{arccosh}(-\langle \mathbf{x}, \mathbf{y} \rangle_{\mathcal{L}}) \tag{2.37}$$

2.5.2 Relational Learning in Non-Euclidean Spaces

Hyperbolic space has recently been intensively studied to model relational data. Nickel et al. [67] proposed an approach to learning hyperbolic embeddings of unweighted undirected graphs based on the Poincaré-ball model. Later, the authors find learning hyperbolic embeddings based on the Lorentz model is more suited for Riemannian optimization since the Lorentz model does not have the issue of numerical instabilities [68]. In contrast to simple look-up embeddings learned in [67, 68], Liu et al. [59] used deep learning techniques to learn hyperbolic embeddings by extending graph neural networks (GNN) to operate in hyperbolic space.

Multi-relational data such as knowledge graphs often exhibit hierarchical patterns, which should be considered by learning embeddings. To this end, Balažević et al. [3] extended the hyperbolic representation learning to multi-relational data. The authors proposed a well-suitable score function for learning hyperbolic knowledge graph embeddings by taking both bilinear KG models and translation-based KG models into account. In parallel to [3], Kolyvakis et al. [53] extended the translation-based models into the Poincaré-ball model of hyperbolic space, which improves the performance of translation-based models significantly. Chami et al. [15] argued that previous hyperbolic embedding methods cannot well capture the logical patterns in KGs. Since RotatE [83] showed its full expressiveness to encode relation patterns of symmetry, antisymmetry, inversion, and composition, Chami et al. worked on extending RotatE in hyperbolic space. Specifically, they defined two operations, i.e., *rotation* and *reflection*, in the Poincaré-ball model of hyperbolic space and applied the attention mechanism to the representations resulted from the two operations. The proposed ATTH model achieves new state-of-the-art on several benchmark datasets.

Nevertheless, the above non-Euclidean KG models cannot deal with time-evolving multi-relational data, such as temporal knowledge graphs. Besides, they learn representations in hyperbolic space with a constant negative curvature. But most knowledge graphs exhibit a wide variety of structures instead of a uniform structure. To this end, our published work [30] in Chapter 6 proposed a novel embedding approach in a product manifold combining multiple non-Euclidean spaces. The proposed approach simultaneously captures heterogeneous geometric structures and temporal dynamics on temporal knowledge graphs. Following our work, Montella et al. [61] further investigated time-aware KG representations in non-Euclidean space. They extended the static KG model, ATTH [15], to a time-aware version by learning the curvature of a manifold depending on both relation and time. While our work [30] treats manifold curvatures as hyperparameters, Montella et al. assigned each relation-timestamp pair a corresponding manifold curvature that can be learned via gradient-based optimization, leading to a more efficient training procedure.

This section on non-Euclidean KG representation learning methods completes the background material of this thesis. The remainder of this dissertation continues with our publications on temporal knowledge graph relational learning.

Chapter 3

Explainable Subgraph Reasoning for Forecasting on Temporal Knowledge Graphs

This chapter contains the publication

Explainable Subgraph Reasoning for Forecasting on Temporal Knowledge Graphs. In *Proceedings of the International Conference on Learning Representations (ICLR)*, Virtual Conference, May 2021. Openreview.net: pdf?id=pGIHq1m7PU.

EXPLAINABLE SUBGRAPH REASONING FOR FORECASTING ON TEMPORAL KNOWLEDGE GRAPHS

Zhen Han[*1,2], **Peng Chen**[*2,3], **Yunpu Ma**[†1] , **Volker Tresp**[†1,2]
[1]Institute of Informatics, LMU Munich [2] Corporate Technology, Siemens AG
[3]Department of Informatics, Technical University of Munich
zhen.han@campus.lmu.de, peng.chen@tum.de
cognitive.yunpu@gmail.com, volker.tresp@siemens.com

ABSTRACT

Modeling time-evolving knowledge graphs (KGs) has recently gained increasing interest. Here, graph representation learning has become the dominant paradigm for link prediction on temporal KGs. However, the embedding-based approaches largely operate in a black-box fashion, lacking the ability to interpret their predictions. This paper provides a link forecasting framework that reasons over query-relevant subgraphs of temporal KGs and jointly models the structural dependencies and the temporal dynamics. Especially, we propose a temporal relational attention mechanism and a novel reverse representation update scheme to guide the extraction of an enclosing subgraph around the query. The subgraph is expanded by an iterative sampling of temporal neighbors and by attention propagation. Our approach provides human-understandable evidence explaining the forecast. We evaluate our model on four benchmark temporal knowledge graphs for the link forecasting task. While being more explainable, our model obtains a relative improvement of up to 20 % on Hits@1 compared to the previous best temporal KG forecasting method. We also conduct a survey with 53 respondents, and the results show that the evidence extracted by the model for link forecasting is aligned with human understanding.

1 INTRODUCTION

Reasoning, a process of inferring new knowledge from available facts, has long been considered an essential topic in AI research. Recently, reasoning on knowledge graphs (KG) has gained increasing interest (Das et al., 2017; Ren et al., 2020; Hildebrandt et al., 2020). A knowledge graph is a graph-structured knowledge base that stores factual information in the form of triples (s, p, o), e.g., (*Alice, livesIn, Toronto*). In particular, s (subject) and o (object) are expressed as nodes and p (predicate) as an edge type. Most knowledge graph models assume that the underlying graph is static. However, in the real world, facts and knowledge can change with time. For example, (*Alice, livesIn, Toronto*) becomes invalid after Alice moves to Vancouver. To accommodate time-evolving multi-relational data, temporal KGs have been introduced (Boschee et al., 2015), where a temporal fact is represented as a quadruple by extending the static triple with a timestamp t indicating the triple is valid at t, i.e. (*Barack Obama, visit, India*, 2010-11-06).

In this work, we focus on forecasting on temporal KGs, where we infer future events based on past events. Forecasting on temporal KGs can improve a plethora of downstream applications such as decision support in personalized health care and finance. The use cases often require the predictions made by the learning models to be interpretable, such that users can understand and trust the predictions. However, current machine learning approaches (Trivedi et al., 2017; Jin et al., 2019) for temporal KG forecasting operate in a black-box fashion, where they design an embedding-based score function to estimate the plausibility of a quadruple. These models cannot clearly show which evidence contributes to a prediction and lack explainability to the forecast, making them less suitable for many real-world applications.

[*]Equal contribution.
[†]Corresponding authors.

Explainable approaches can generally be categorized into post-hoc interpretable methods and integrated transparent methods (Došilović et al., 2018). Post-hoc interpretable approaches (Montavon et al., 2017; Ying et al., 2019) aim to interpret the results of a black-box model, while integrated transparent approaches (Das et al., 2017; Qiu et al., 2019; Wang et al., 2019) have an explainable internal mechanism. In particular, most integrated transparent (Lin et al., 2018; Hildebrandt et al., 2020) approaches for KGs employ path-based methods to derive an explicit reasoning path and demonstrate a transparent reasoning process. The path-based methods focus on finding the answer to a query within a single reasoning chain. However, many complicated queries require multiple supporting reasoning chains rather than just one reasoning path. Recent work (Xu et al., 2019; Teru et al., 2019) has shown that reasoning over local subgraphs substantially boosts performance while maintaining interpretability. However, these explainable models cannot be applied to temporal graph-structured data because they do not take time information into account. This work aims to design a transparent forecasting mechanism on temporal KGs that can generate informative explanations of the predictions.

In this paper, we propose an e**x**plainabl**e** **r**easoning framework for forecasting future links on **t**emporal knowl**e**dge graphs, xERTE, which employs a sequential reasoning process over local subgraphs. To answer a query in the form of (subject e_q, predicate p_q, ?, timestamp t_q), xERTE starts from the query subject, iteratively samples relevant edges of entities included in the subgraph and propagates attention along the sampled edges. After several rounds of expansion and pruning, the missing object is predicted from entities in the subgraph. Thus, the extracted subgraph can be seen as a concise and compact graphical explanation of the prediction. To guide the subgraph to expand in the direction of the query's interest, we propose a temporal relational graph attention (TRGA) mechanism. We pose temporal constraints on passing messages to preserve the causal nature of the temporal data. Specifically, we update the time-dependent hidden representation of an entity e_i at a timestamp t by attentively aggregating messages from its temporal neighbors that were linked with e_i prior to t. We call such temporal neighbors as prior neighbors of e_i. Additionally, we use an embedding module consisting of stationary entity embeddings and functional time encoding, enabling the model to capture both global structural information and temporal dynamics. Besides, we develop a novel representation update mechanism to mimic human reasoning behavior. When humans perform a reasoning process, their perceived profiles of observed entities will update, as new clues are found. Thus, it is necessary to ensure that all entities in a subgraph can receive messages from prior neighbors newly added to the subgraph. To this end, the proposed representation update mechanism enables every entity to receive messages from its farthest prior neighbors in the subgraph.

The major contributions of this work are as follows. **(1)** We develop xERTE, the first explainable model for predicting *future links* on temporal KGs. The model is based on a temporal relational attention mechanisms that preserves the causal nature of the temporal multi-relational data. **(2)** Unlike most black-box embedding-based models, xERTE visualizes the reasoning process and provides an interpretable inference graph to emphasize important evidence. **(3)** The dynamical pruning procedure enables our model to perform reasoning on large-scale temporal knowledge graphs with millions of edges. **(4)** We apply our model for forecasting future links on four benchmark temporal knowledge graphs. The results show that our method achieves on average a better performance than current state-of-the-art methods, thus providing a new baseline. **(5)** We conduct a survey with 53 respondents to evaluate whether the extracted evidence is aligned with human understanding.

2 RELATED WORK

Representation learning is an expressive and popular paradigm underlying many KG models. The embedding-based approaches for knowledge graphs can generally be categorized into bilinear models (Nickel et al., 2011; Yang et al., 2014; Ma et al., 2018a), translational models (Bordes et al., 2013; Lv et al., 2018; Sun et al., 2019; Hao et al., 2019), and deep-learning models (Dettmers et al., 2017; Schlichtkrull et al., 2018). However, the above methods are not able to use rich dynamics available on temporal knowledge graphs. To this end, several studies have been conducted for temporal knowledge graph reasoning (García-Durán et al., 2018; Ma et al., 2018b; Jin et al., 2019; Goel et al., 2019; Lacroix et al., 2020; Han et al., 2020a;b; Zhu et al., 2020). The published approaches are largely black-box, lacking the ability to interpret their predictions. Recently, several explainable reasoning methods for knowledge graphs have been proposed (Das et al., 2017; Xu et al., 2019;

Hildebrandt et al., 2020; Teru et al., 2019). However, the above explainable methods can only deal with static KGs, while our model is designed for interpretable forecasting on temporal KGs.

3 Preliminaries

Let $\mathcal{E}$ and $\mathcal{P}$ represent a finite set of entities and predicates, respectively. A temporal knowledge graph is a collection of timestamped facts written as quadruples. A quadruple $q = (e_s, p, e_o, t)$ represents a timestamped and labeled edge between a subject entity $e_s \in \mathcal{E}$ and an object entity $e_o \in \mathcal{E}$, where $p \in \mathcal{P}$ denotes the edge type (predicate). The temporal knowledge graph forecasting task aims to predict unknown links at future timestamps based on observed past events.

Definition 1 *(Temporal KG forecasting). Let $\mathcal{F}$ represent the set of all ground-truth quadruples, and let $(e_q, p_q, e_o, t_q) \in \mathcal{F}$ denote the target quadruple. Given a query $(e_q, p_q, ?, t_q)$ derived from the target quadruple and a set of observed prior facts $\mathcal{O} = \{(e_i, p_k, e_j, t_l) \in \mathcal{F} | t_l < t_q\}$, the temporal KG forecasting task is to predict the missing object entity e_o. Specifically, we consider all entities in the set $\mathcal{E}$ as candidates and rank them by their likelihood to form a true quadruple together with the given subject-predicate-pair at timestamp t_q*[1].

For a given query $q = (e_q, p_q, ?, t_q)$, we build an *inference graph* $\mathcal{G}_{inf}$ to visualize the reasoning process. Unlike in temporal KGs, where a node represents an entity, each node in $\mathcal{G}_{inf}$ is an entity-timestamp pair. The inference graph is a directed graph in which a link points from a node with an earlier timestamp to a node with a later timestamp.

Definition 2 *(Node in Inference Graph and its Temporal Neighborhood). Let $\mathcal{E}$ represent all entities, $\mathcal{F}$ denote all ground-truth quadruples, and let t represent a timestamp. A node in an inference graph $\mathcal{G}_{inf}$ is defined as an entity-timestamp pair $v = (e_i, t), e_i \in \mathcal{E}$. We define the set of one-hop prior neighbors of v as $\mathcal{N}_{v=(e_i,t)} = \{(e_j, t') | (e_i, p_k, e_j, t') \in \mathcal{F} \wedge (t' < t)\}$*[2]*. For simplicity, we denote one-hop prior neighbors as $\mathcal{N}_v$. Similarly, we define the set of one-hop posterior neighbors of v as $\overline{\mathcal{N}}_{v=(e_i,t)} = \{(e_j, t') | (e_j, p_k, e_i, t) \in \mathcal{F} \wedge (t' > t)\}$. We denote them as $\overline{\mathcal{N}}_v$ for short.*

We provide an example in Figure 4 in the appendix to illustrate the inference graph.

4 Our Model

We describe xERTE in a top-down fashion where we provide an overview in Section 4.1 and then explain each module from Section 4.2 to 4.6.

4.1 Subgraph Reasoning Process

Our model conducts the reasoning process on a dynamically expanded inference graph $\mathcal{G}_{inf}$ extracted from the temporal KG. We show a toy example in Figure 1. Given query $q = (e_q, p_q, ?, t_q)$, we initialize $\mathcal{G}_{inf}$ with node $v_q = (e_q, t_q)$ consisting of the query subject and the query time. The inference graph expands by sampling prior neighbors of v_q. For example, suppose that (e_q, p_k, e_j, t') is a valid quadruple where $t' < t_q$, we add the node $v_1 = (e_j, t')$ into $\mathcal{G}_{inf}$ and link it with v_q where the link is labeled with p_k and points from v_q to v_1. We use an embedding module to assign each node and predicate included in $\mathcal{G}_{inf}$ a temporal embedding that is shared across queries. The main goal of the embedding module is to let the nodes access query-independent information and get a broad view of the graph structure since the following temporal relational graph attention (TRGA) layer only performs query-dependent message passing locally. Next, we feed the inference graph into the TRGA layer that takes node embeddings and predicate embeddings as the input, produces a query-dependent representation for each node by passing messages on the small inference graph, and computes a query-dependent attention score for each edge. As explained in Section 4.7, we propagate the attention of each node to its prior neighbors using the edge attention scores. Then we further expand $\mathcal{G}_{inf}$ by sampling the prior neighbors of the nodes in $\mathcal{G}_{inf}$. The expansion will grow

[1]Throughout this work, we add reciprocal relations for every quadruple, i.e., we add (e_o, p^{-1}, e_s, t) for every (e_s, p, e_o, t). Hence, the restriction to predict object entities does not lead to a loss of generality.

[2]Prior neighbors linked with e_i as subject entity, e.g., (e_j, p_k, e_i, t), are covered using reciprocal relations.

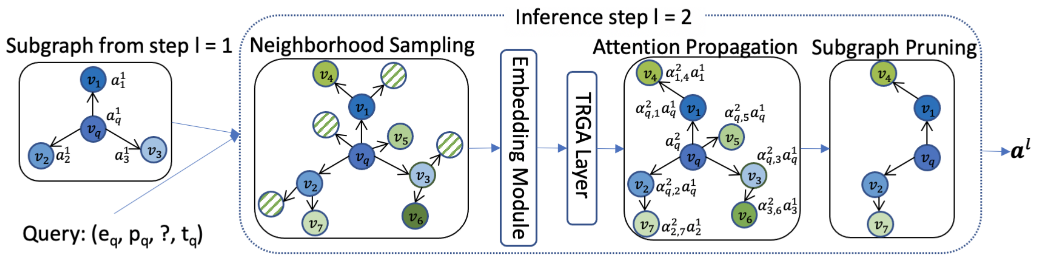


Figure 1: Model Architecture. We take the second inference step ($l = 2$) as an example. Each directed edge points from a source node to its prior neighbor. ▨ denotes nodes that have not been sampled. a_i^l means the attention score of node v_i at the l^{th} inference step. $\alpha_{i,j}^l$ is the attention score of the edge between node i and its prior neighbor j at the l^{th} inference step. Note that all scores are query-dependent. For simplicity, we do not show edge labels (predicates) in the figure.

rapidly and cover almost all nodes after a few steps. To prevent the inference graph from exploding, we reduce the edge amount by pruning the edges that gain less attention. As the expansion and pruning iterate, $\mathcal{G}_{inf}$ allocates more and more information from the temporal KG. After running L inference steps, the model selects the entity with the highest attention score in $\mathcal{G}_{inf}$ as the prediction of the missing query object, where the inference graph itself serves as a graphical explanation.

4.2 Neighborhood Sampling

We define the set of edges between node $v = (e_i, t)$ and its prior neighbors $\mathcal{N}_v$ as $\mathcal{Q}_v$, where $q_v \in \mathcal{Q}_v$ is a *prior edge* of v. To reduce the complexity, we sample a subset of prior edges $\hat{\mathcal{Q}}_v \in \mathcal{Q}_v$ at each inference step. We denote the remaining prior neighbors and posterior neighbors of node v after the sampling as $\hat{\mathcal{N}}_v$ and $\overline{\hat{\mathcal{N}}_v}$, respectively. Note that there might be multiple edges between node v and its prior neighbor u because of multiple predicates. If there is at least one edge that has been sampled between v and u, we add u into $\hat{\mathcal{N}}_v$. The sampling can be uniform if there is no bias, it can also be temporally biased using a non-uniform distribution. For instance, we may want to sample more edges closer to the current time point as the events that took place long ago may have less impact on the inference. Specifically, we propose three different sampling strategies: (1) **Uniform sampling**. Each prior edge $q_v \in \mathcal{Q}_v$ has the same probability of being selected: $\mathbb{P}(q_v) = 1/|\mathcal{Q}_v|$. (2) **Time-aware exponentially weighted sampling**. We temporally bias the neighborhood sampling using an exponential distribution and assign the probability $\mathbb{P}(q_v = (e_i, p_k, e_j, t')) = \exp(t' - t)/\sum_{(e_i, p_l, e_m, t'') \in \mathcal{Q}_v} \exp(t'' - t)$ to each prior neighbor, which negatively correlates with the time difference between node v and its prior neighbor (e_j, t'). Note that t' and t'' are prior to t. (3) **Time-aware linearly weighted sampling**. We use a linear function to bias the sampling. Compared to the second strategy, the quadruples occurred in early stages have a higher probability of being sampled. Overall, we have empirically found that the second strategy is most beneficial to our framework and provide a detailed ablation study in Section 5.2.

4.3 Embedding

In temporal knowledge graphs, graph structures are no longer static, as entities and their links evolve over time. Thus, entity features may change and exhibit temporal patterns. In this work, the embedding of an entity $e_i \in \mathcal{E}$ at time t consists of a static low-dimensional vector and a functional representation of time. The time-aware entity embedding is defined as $\mathbf{e}_i(t) = [\bar{\mathbf{e}}_i || \mathbf{\Phi}(t)]^T \in \mathbb{R}^{d_S + d_T}$. Here, $\bar{\mathbf{e}}_i \in R^{d_S}$ represents the static embedding that captures time-invariant features and global dependencies over the temporal KG. $\mathbf{\Phi}(\cdot)$ denotes a time encoding that captures temporal dependencies between entities (Xu et al., 2020). We provide more details about $\mathbf{\Phi}(\cdot)$ in Appendix I. $||$ denotes the concatenation operator. d_S and d_T represent the dimensionality of the static embedding and the time embedding, which can be tuned according to the temporal fraction of the given dataset. We also tried the temporal encoding presented in Goel et al. (2019), which has significantly more parameters. But we did not see considerable improvements. Besides, we assume that predicate features do not evolve. Thus, we learn a stationary embedding vector $\mathbf{p}_k$ for each predicate p_k.

4.4 Temporal Relational Graph Attention Layer

Here, we propose a temporal relational graph attention (TRGA) layer for identifying the relevant evidence in the inference graph related to a given query q. The input to the TRGA layer is a set of entity embeddings $\mathbf{e}_i(t)$ and predicate embeddings $\mathbf{p}_k$ in the given inference graph. The layer produces a query-dependent attention score for each edge and a new set of hidden representations as its output. Similar to GraphSAGE (Hamilton et al., 2017) and GAT (Veličković et al., 2017), the TRGA layer performs a local representation aggregation. To avoid misusing future information, we only allow message passing from prior neighbors to posterior neighbors. Specifically, for each node v in the inference graph, the aggregation function fuses the representation of node v and the sampled prior neighbors $\hat{\mathcal{N}}_v$ to output a time-aware representation for v. Since entities may play different roles, depending on the predicate they are associated with, we incorporate the predicate embeddings in the attention function to exploit relation information. Instead of treating all prior neighbors with equal importance, we take the query information into account and assign varying importance levels to each prior neighbor $u \in \hat{\mathcal{N}}_v$ by calculating a query-dependent attention score using

$$e^l_{vu}(q, p_k) = \mathbf{W}^l_{sub}(\mathbf{h}^{l-1}_v || \mathbf{p}^{l-1}_k || \mathbf{h}^{l-1}_{e_q} || \mathbf{p}^{l-1}_q)\mathbf{W}^l_{obj}(\mathbf{h}^{l-1}_u || \mathbf{p}^{l-1}_k || \mathbf{h}^{l-1}_{e_q} || \mathbf{p}^{l-1}_q), \tag{1}$$

where $e^l_{vu}(q, p_k)$ is the attention score of the edge (v, p_k, u) regarding the query $q = (e_q, p_q, ?, t_q)$, p_k corresponds to the predicate between node u and node v, $\mathbf{p}_k$ and $\mathbf{p}_q$ are predicate embeddings. $\mathbf{h}^{l-1}_v$ denotes the hidden representation of node v at the $(l-1)^{th}$ inference step. When $l = 1$, i.e., for the first layer, $\mathbf{h}^0_v = \mathbf{W}_v \mathbf{e}_i(t) + \mathbf{b}_v$, where $v = (e_i, t)$. $\mathbf{W}^l_{sub}$ and $\mathbf{W}^l_{obj}$ are two weight matrices for capturing the dependencies between query features and node features. Then, we compute the normalized attention score $\alpha^l_{vu}(q, p_k)$ using the *softmax function* as follows

$$\alpha^l_{vu}(q, p_k) = \frac{\exp(e^l_{vu}(q, p_k))}{\sum_{w \in \hat{\mathcal{N}}_v} \sum_{p_z \in \mathcal{P}_{vw}} \exp(e^l_{vw}(q, p_z))}, \tag{2}$$

where $\mathcal{P}_{vw}$ represents the set of labels of edges that connect nodes v and w. Once obtained, we aggregate the representations of prior neighbors and weight them using the normalized attention scores, which is written as

$$\widetilde{\mathbf{h}}^l_v(q) = \sum_{u \in \hat{\mathcal{N}}_v} \sum_{p_k \in \mathcal{P}_{vu}} \alpha^l_{vu}(q, p_k)\mathbf{h}^{l-1}_u(q). \tag{3}$$

We combine the hidden representation $\mathbf{h}^{l-1}_v(q)$ of node v with the aggregated neighborhood representation $\widetilde{\mathbf{h}}^l_v(q)$ and feed them into a fully connected layer with a LeakyReLU activation function $\sigma(\cdot)$, as shown below

$$\mathbf{h}^l_v(q) = \sigma(\mathbf{W}^l_h(\gamma \mathbf{h}^{l-1}_v(q) + (1-\gamma)\widetilde{\mathbf{h}}^l_v(q) + \mathbf{b}^l_h)), \tag{4}$$

where $\mathbf{h}^l_v(q)$ denotes the representation of node v at the l^{th} inference step, and γ is a hyperparameter. Further, we use the same layer to update the relation embeddings, which is of the form $\mathbf{p}^l_k = \mathbf{W}^l_h \mathbf{p}^{l-1}_k + \mathbf{b}^l_h$. Thus, the relations are projected to the same embedding space as nodes and can be utilized in the next inference step.

4.5 Attention Propagation and Subgraph Pruning

After having the edges' attention scores in the inference graph, we compute the attention score $a^l_{v,q}$ of node v regarding query q at the l^{th} inference step as follows:

$$a^l_{v,q} = \sum_{u \in \overline{\mathcal{N}}_v} \sum_{p_z \in \mathcal{P}_{uv}} \alpha^l_{uv}(q, p_z) a^{l-1}_{u,q}. \tag{5}$$

Thus, we propagate the attention of each node to its prior neighbors. As stated in Definition 2, each node in inference graph is an entity-timestamp pair. To assign each entity a unique attention score, we aggregate the attention scores of nodes whose entity is the same:

$$a^l_{e_i,q} = g(a^l_{v,q} | v(e) = e_i), \quad \text{for } v \in \mathcal{V}_{\mathcal{G}_{inf}}, \tag{6}$$

where $a^l_{e_i,q}$ denotes the attention score of entity e_i, $\mathcal{V}_{\mathcal{G}_{inf}}$ is the set of nodes in inference graph $\mathcal{G}_{inf}$. $v(e)$ represents the entity included in node v, and $g(\cdot)$ represents a score aggregation function. We try two score aggregation functions $g(\cdot)$, i.e., summation and mean. We conduct an ablation study and find that the summation aggregation performs better. To demonstrate which evidence is important for the reasoning process, we assign each edge in the inference graph a contribution score. Specifically, the contribution score of an edge (v, p_k, u) is defined as $c_{vu}(q, p_k) = \alpha^l_{vu}(q, p_k)a^l_{v,q}$, where node u is a prior neighbor of node v associated with the predicate p_k. We prune the inference graph at each inference step and keep the edges with K largest contribution scores. We set the attention score of entities, which the inference graph does not include, to zero. Finally, we rank all entity candidates according to their attention scores and choose the entity with the highest score as our prediction.

4.6 Reverse Representation Update Mechanism

When humans perform a reasoning process, the perceived profile of an entity during the inference may change as new evidence joins the reasoning process. For example, we want to predict the profitability of company A. We knew that A has the largest market portion, which gives us a high expectation about A's profitability. However, a new evidence shows that conglomerate B enters this market as a strong competitor. Although the new evidence is not directly related to A, it indicates that there will be a high competition between A and B, which lowers our expectation about A's profitability. To mimic human reasoning behavior, we should ensure that all existing nodes in inference graph $\mathcal{G}_{inf}$ can receive messages from nodes newly added to $\mathcal{G}_{inf}$. However, since $\mathcal{G}_{inf}$ expands once at each inference step, it might include l-hop neighbors of the query subject at the l^{th} step. The vanilla solution is to iterate the message passing l times at the l^{th} inference step, which means that we need to run the message passing $(1 + L) \cdot L/2$ times in total, for L inference steps. To avoid the quadratic increase of message passing iterations, we propose a novel reverse representation update mechanism. Recall that, to avoid violating temporal constraints, we use prior neighbors to update nodes' representations. And at each inference step, we expand $\mathcal{G}_{inf}$ by adding prior neighbors of each node in $\mathcal{G}_{inf}$. For example, assuming that we are at the fourth inference step, for a node that has been added at the second step, we only need to aggregate messages from nodes added at the third and fourth steps. Hence, we can update the representations of nodes in reverse order as they have been added in $\mathcal{G}_{inf}$. Specifically, at the l^{th} inference step, we first update the representations of nodes added at the $(l-1)^{th}$ inference step, then the nodes added at $(l-2)^{th}$, and so forth until $l = 0$, as shown in Algorithm 1 in the appendix. In this way, we compute messages along each edge in $\mathcal{G}_{inf}$ only once and ensure that every node can receive messages from its farthest prior neighbor.

4.7 Learning

We split quadruples of a temporal KG into *train*, *validation*, and *test* sets by timestamps, ensuring (timestamps of training set)<(timestamps of validation set)<(timestamps of test set). We use the binary cross-entropy as the loss function, which is defined as

$$\mathcal{L} = -\frac{1}{|Q|}\sum_{q\in Q}\frac{1}{|\mathcal{E}^{inf}_q|}\sum_{e_i\in\mathcal{E}^{inf}_q}\left(y_{e_i,q}\log(\frac{a^L_{e_i,q}}{\sum_{e_j\in\mathcal{E}^{inf}_q}a^L_{e_j,q}}) + (1-y_{e_i,q})\log(1-\frac{a^L_{e_i,q}}{\sum_{e_j\in\mathcal{E}^{inf}_q}a^L_{e_j,q}})\right),$$

where $\mathcal{E}^{inf}_q$ represents the set of entities in the inference graph of the query q, $y_{e_i,q}$ represents the binary label that indicates whether e_i is the answer for q, and Q denotes the set of training quadruples. $a^L_{e_i,q}$ denotes the attention score of e_i at the final inference step. We list all model parameters in Table 2 in the appendix. Particularly, we jointly learn the embeddings and other model parameters by end-to-end training.

5 Experiments

5.1 Datasets and Baselines

Integrated Crisis Early Warning System (ICEWS) (Boschee et al., 2015) and YAGO (Mahdisoltani et al., 2013) have established themselves in the research community as benchmark datasets of temporal KGs. The ICEWS dataset contains information about political events with time annotations, e.g., (*Barack Obama, visit, Malaysia,* 2014-02-19). We evaluate our model on three subsets of the ICEWS dataset, i.e., ICEWS14, ICEWS18, and ICEWS05-15, that contain event facts in 2014, 2018, and the facts from 2005 to 2015, respectively. The YAGO dataset is a temporal knowledge base that fuses information from Wikipedia with the English WordNet dataset (Miller, 1995). Following the experimental settings of HyTE (Dasgupta et al., 2018), we use a subset and only deal with year level granularity by dropping the month and date information. We compare our approach and baseline methods by performing the link prediction task on the ICEWS14, ICEWS18, ICEWS0515, and YAGO datasets. The statistics of the datasets are provided in Appendix C.

We compare xERTE with benchmark temporal KG and static KG reasoning models. From the temporal KG reasoning models, we compare our model with several state-of-the-art methods, including TTransE (Leblay & Chekol, 2018), TA-DistMult/TA-TransE (García-Durán et al., 2018), DE-SimplE(Goel et al., 2019), TNTComplEx (Lacroix et al., 2020), CyGNet(Zhu et al., 2020), and RE-Net (Jin et al., 2019). From the static KG reasoning models, we choose TransE (Bordes et al., 2013), DistMult (Yang et al., 2014), and ComplEx (Trouillon et al., 2016).

5.2 Experimental Results and Ablation Study

Datasets	**ICEWS14 - filtered**				**ICEWS05-15 - filtered**				**ICEWS18 - filtered**				**YAGO - filtered**			
Model	MRR	HITS@1	HITS@3	HITS@10	MRR	Hits@1	Hits@3	Hits@10	MRR	Hits@1	Hits@3	Hits@10	MRR	Hits@1	Hits@3	Hits@10
TransE	22.48	13.36	25.63	41.23	22.55	13.05	25.61	42.05	12.24	5.84	12.81	25.10	11.69	10.37	11.96	13.83
DistMult	27.67	18.16	31.15	46.96	28.73	19.33	32.19	47.54	10.17	4.52	10.33	21.25	11.98	10.20	12.31	14.93
ComplEx	30.84	21.51	34.48	49.58	31.69	21.44	35.74	52.04	21.01	11.87	23.47	39.87	12.07	10.42	12.36	14.82
TTransE	13.43	3.11	17.32	34.55	15.71	5.00	19.72	38.02	8.31	1.92	8.56	21.89	5.68	1.42	9.04	11.21
TA-DistMult	26.47	17.09	30.22	45.41	24.31	14.58	27.92	44.21	16.75	8.61	18.41	33.59	11.50	10.21	11.90	13.88
TA-TransE	17.41	0.00	29.19	47.41	19.37	1.81	31.34	50.33	12.59	0.01	17.92	37.38	6.74	2.13	11.01	12.28
DE-SimplE	32.67	24.43	35.69	49.11	35.02	25.91	38.99	52.75	19.30	11.53	21.86	34.80	11.73	10.70	12.10	13.51
TNTComplEx	32.12	23.35	36.03	49.13	27.54	19.52	30.80	42.86	21.23	13.28	24.02	36.91	12.00	11.12	12.13	13.57
CyGNet[3]	32.73	23.69	36.31	50.67	34.97	25.67	39.09	52.94	24.93	15.90	28.28	42.61	12.48	11.00	12.66	14.82
RE-Net	38.28	28.68	41.34	54.52	42.97	31.26	46.85	63.47	28.81	19.05	32.44	**47.51**	**54.87**	47.51	57.84	**65.81**
xERTE	**40.79**	**32.70**	**45.67**	**57.30**	**46.62**	**37.84**	**52.31**	**63.92**	**29.31**	**21.03**	**33.51**	46.48	53.62	**48.53**	**58.42**	60.53

Table 1: Results of future link prediction on four datasets. Compared metrics are **time-aware** filtered MRR (%) and Hits@1/3/10 (%). The best results among all models are in bold.

Comparison results Table 1 summarizes the **time-aware** filtered results of the link prediction task on the ICEWS and YAGO datasets[4]. The time-aware filtering scheme only filters out triples that are genuine at the query time while the filtering scheme applied in prior work (Jin et al., 2019; Zhu et al., 2020) filters all triples that occurred in history. A detailed explanation is provided in Appendix D. Overall, xERTE outperforms all baseline models on ICEWS14/05-15/18 in MRR and Hits@1/3/10 while being more interpretable. Compared to the strongest baseline RE-Net, xERTE obtains a relative improvement of 5.60% and 15.15% in MRR and Hits@1, which are averaged on ICEWS14/05-15/18. Especially, xERTE achieves more gains in Hits@1 than in Hits@10. It confirms the assumption that subgraph reasoning helps xERTE make a sharp prediction by exploiting local structures. On the YAGO dataset, xERTE achieves comparable results with RE-Net in terms of MRR and Hits@1/3. To assess the importance of each component, we conduct several ablation studies and show their results in the following.

Representation update analysis We train a model without the reverse representation update mechanism to investigate how this mechanism contributes to our model. Since the reverse representation update ensures that each node can receive messages from all its prior neighbors in the inference graph, we expect this mechanism could help nodes mine available information. This update mechanism should be especially important for nodes that only have been involved in a small number of events. Since the historical information of such nodes is quite limited, it is very challenging to forecast their future behavior. In Figure 2a and 2b we show the metrics of Hits@1 and Hits@10 against the number of nodes in the inference graph. It can be observed the model with the reverse update mechanism performs better in general. In particular, this update mechanism significantly improves the performance if the query subject only has a small number of neighbors in the subgraph, which meets our expectation.

[3]We found that CyGNet does not perform subject prediction in its evaluation code and does not report time-aware filtered results. The performance significantly drops after fixing the code.

[4]Code and datasets are available at https://github.com/TemporalKGTeam/xERTE

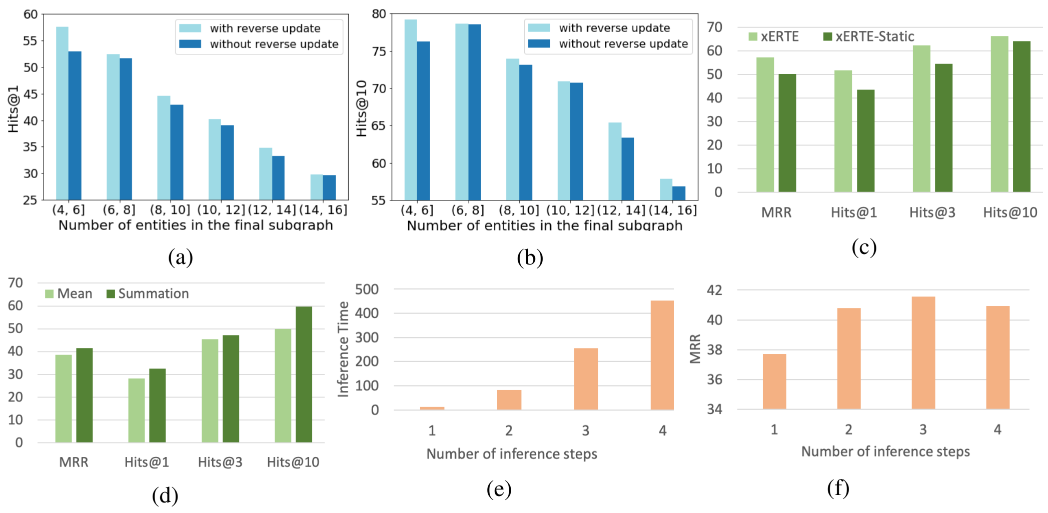


Figure 2: Ablation Study. Unlike in Table 1 that reports results on the whole test set, here we filter out test quadruples that contain unseen entities. (a)-(b) We compare the model with/without the reverse representation update in terms of raw Hits@1(%) and Hits@10(%) on ICEWS14, respectively. (c) Temporal embedding analysis on YAGO. We refer the model without temporal embeddings as xERTE-Static. (d) Attention score aggregation function analysis on ICEWS14: raw MRR (%) and Hits@1/3/10(%). (e) Inference time (seconds) on the test set of ICEWS14 regarding different inference step settings $L \in \{1, 2, 3, 4\}$. (f) Raw MRR(%) on ICEWS14 regarding different inference step settings L.

Time-aware representation analysis and node attention aggregation To verify the importance of the time embedding, we evaluate the performance of a model without time encoding. As shown in Figure 2c, removing the time-dependent part from entity representations sacrifices the model's performance significantly. Recall that each node in inference graph $\mathcal{G}_{inf}$ is associated with a timestamp, the same entity might appear in several nodes in $\mathcal{G}_{inf}$ with different timestamps. To get a unified attention score for each entity, we aggregate the attention scores of nodes whose entity is the same. Figure 2d shows that the summation aggregator brings a considerable gain on ICEWS14.

Sampling analysis We run experiments with different sampling strategies proposed in Section 4.2. To assess the necessity of the time-aware weighted sampling, we propose a deterministic version of the time-aware weighted sampling, where we chronologically sort the prior edges of node v in terms of their timestamps and select the last N edges to build the subset $\hat{\mathcal{Q}}_v$. The experimental results are provided in Table 3 in the appendix. We find that the sampling strategy has a considerable influence on model's performance. Sampling strategies that bias towards recent quadruples perform better. Specifically, the exponentially time-weighted strategy performs better than the linear time-weighted strategy and the deterministic last-N-edges strategy.

Time cost analysis The time cost of xERTE is affected not only by the scale of a dataset but also by the number of inference steps L. Thus, we run experiments of inference time and predictive power regarding different settings of L and show the results in Figures 2e and 2f. We see that the model achieves the best performance with $L = 3$ while the training time significantly increases as L goes up. To make the computation more efficient, we develop a series of segment operations for subgraph reasoning. Please see Appendix G for more details.

5.3 Graphical Explanation and Human Evaluation

The extracted inference graph provides a graphical explanation for model's prediction. As introduced in 4.7, we assign each edge in the inference graph a contribution score. Thus, users can trace back the important evidence that the prediction mainly depends on. We study a query chosen from the test set, where we predict whom will Catherine Ashton visit on Nov. 9, 2014 and show the final

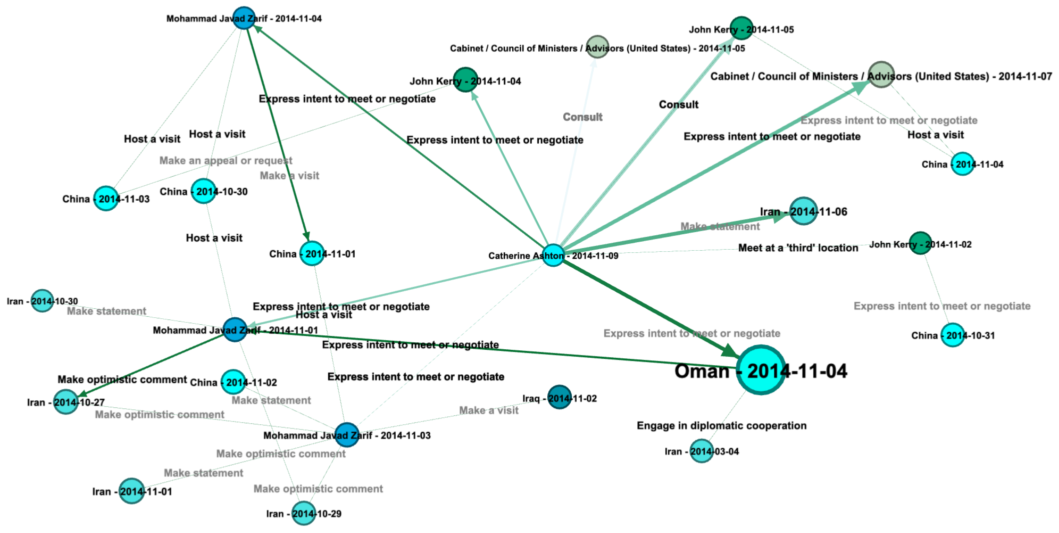

Mohammad Javad Zarif - 2014-11-04
Cabinet / Council of Ministers / Advisors (United States) - 2014-11-05
John Kerry - 2014-11-05
John Kerry - 2014-11-04
Cabinet / Council of Ministers / Advisors (United States) - 2014-11-07
Express intent to meet or negotiate
Consult
Consult
Express intent to meet or negotiate
Express intent to meet or negotiate
Express intent to meet or negotiate
Host a visit
Host a visit
Host a visit
Make an appeal or request
Make a visit
China - 2014-11-04
China - 2014-11-03
China - 2014-10-30
Iran - 2014-11-06
Make statement
Host a visit
Meet at a 'third' location
John Kerry - 2014-11-02
China - 2014-11-01
Catherine Ashton - 2014-11-09
Iran - 2014-10-30
Make statement
Express intent to meet or negotiate
Host a visit
Express intent to meet or negotiate
Mohammad Javad Zarif - 2014-11-01
Express intent to meet or negotiate
Express intent to meet or negotiate
China - 2014-10-31
Oman - 2014-11-04
Make optimistic comment
China - 2014-11-02
Express intent to meet or negotiate
Iran - 2014-10-27
Make statement
Iraq - 2014-11-02
Make optimistic comment
Make a visit
Engage in diplomatic cooperation
Mohammad Javad Zarif - 2014-11-03
Make optimistic comment
Iran - 2014-03-04
Make statement
Iran - 2014-11-01
Make optimistic comment
Iran - 2014-10-29

Figure 3: The inference graph for the query (*Catherine Ashton, Make a visit, ?,* 2014-11-09) from ICEWS14. The biggest cyan node represents the object predicted by xERTE. The cyan node with the entity *Catherine Ashton* and the timestamp 2014-11-09 represents the given query subject and the query timestamp. The node size indicates the value of the node attention score. Also, the edges' color indicates the contribution score of the edge, where darkness increases as the contribution score goes up. The entity at an arrow's tail, the predicate on the arrow, the entity and the timestamp at the arrow's head build a true quadruple.

inference graph in Figure 3. In this case, the model's prediction is Oman. And (*Catherine Ashton, express intent to meet or negotiate, Oman,* 2014-11-04) is the most important evidence to support this answer.

To assess whether the evidence is informative for users in an objective setting, we conduct a survey where respondents evaluate the relevance of the extracted evidence to the prediction. More concretely, we set up an online quiz consisting of 7 rounds. Each round is centered around a query sampled from the test set of ICEWS14/ICEWS05-15. Along with the query and the ground-truth answer, we present the human respondents with two pieces of evidence in the inference graph with high contribution scores and two pieces of evidence with low contribution scores in a randomized order. Specifically, each evidence is based on a chronological reasoning path that connects the query subject with an object candidate. For example, given a query (*police, arrest, ?,* 2014-12-28), an extracted clue is that police made statements to lawyers on 2014-12-08, then lawyers were criticized by citizens on 2014-12-10. In each round, we set up three questions to ask the participants to choose the most relevant evidence, the most irrelevant evidence, and sort the pieces of evidence according to their relevance. Then we rank the evidence according to the contribution scores computed by our model and check whether the relevance order classified by the respondents matches that estimated by our models. We surveyed 53 participants, and the average accuracy of all questions is 70.5%. Moreover, based on a majority vote, 18 out of 21 questions were answered correctly, indicating that the extracted inference graphs are informative, and the model is aligned with human intuition. The complete survey and a detailed evaluation are reported in Appendix H.

6 Conclusion

We proposed an explainable reasoning approach for forecasting links on temporal knowledge graphs. The model extracts a query-dependent subgraph from a given temporal KG and performs an attention propagation process to reason on it. Extensive experiments on four benchmark datasets demonstrate the effectiveness of our method. We conducted a survey about the evidence included in the extracted subgraph. The results indicate that the evidence is informative for humans.

REFERENCES

Ivana Balažević, Carl Allen, and Timothy M Hospedales. Tucker: Tensor factorization for knowledge graph completion. *arXiv preprint arXiv:1901.09590*, 2019.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. Translating embeddings for modeling multi-relational data. In *Advances in neural information processing systems*, pp. 2787–2795, 2013.

Elizabeth Boschee, Jennifer Lautenschlager, Sean O'Brien, Steve Shellman, James Starz, and Michael Ward. Icews coded event data. *Harvard Dataverse*, 12, 2015.

Rajarshi Das, Shehzaad Dhuliawala, Manzil Zaheer, Luke Vilnis, Ishan Durugkar, Akshay Krishnamurthy, Alex Smola, and Andrew McCallum. Go for a walk and arrive at the answer: Reasoning over paths in knowledge bases using reinforcement learning. *arXiv preprint arXiv:1711.05851*, 2017.

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

Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. Convolutional 2d knowledge graph embeddings. *arXiv preprint arXiv:1707.01476*, 2017.

Filip Karlo Došilović, Mario Brčić, and Nikica Hlupić. Explainable artificial intelligence: A survey. In *2018 41st International convention on information and communication technology, electronics and microelectronics (MIPRO)*, pp. 0210–0215. IEEE, 2018.

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

Rishab Goel, Seyed Mehran Kazemi, Marcus Brubaker, and Pascal Poupart. Diachronic embedding for temporal knowledge graph completion. *arXiv preprint arXiv:1907.03143*, 2019.

Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Advances in neural information processing systems*, pp. 1024–1034, 2017.

Zhen Han, Peng Chen, Yunpu Ma, and Volker Tresp. DyERNIE: Dynamic Evolution of Riemannian Manifold Embeddings for Temporal Knowledge Graph Completion. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 7301–7316, Online, November 2020a. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.593.

Zhen Han, Yuyi Wang, Yunpu Ma, Stephan Guünnemann, and Volker Tresp. The graph hawkes network for reasoning on temporal knowledge graphs. *arXiv preprint arXiv:2003.13432*, 2020b.

Junheng Hao, Muhao Chen, Wenchao Yu, Yizhou Sun, and Wei Wang. Universal representation learning of knowledge bases by jointly embedding instances and ontological concepts. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 1709–1719, 2019.

Marcel Hildebrandt, Jorge Andres Quintero Serna, Yunpu Ma, Martin Ringsquandl, Mitchell Joblin, and Volker Tresp. Reasoning on knowledge graphs with debate dynamics. *arXiv preprint arXiv:2001.00461*, 2020.

Woojeong Jin, Changlin Zhang, Pedro Szekely, and Xiang Ren. Recurrent event network for reasoning over temporal knowledge graphs. *arXiv preprint arXiv:1904.05530*, 2019.

Timothee Lacroix, Guillaume Obozinski, and Nicolas Usunier. Tensor decompositions for temporal knowledge base completion. *ICLR preprint https://openreview.net/pdf?id=rke2P1BFwS*, 2020.

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

Xi Victoria Lin, Richard Socher, and Caiming Xiong. Multi-hop knowledge graph reasoning with reward shaping. *arXiv preprint arXiv:1808.10568*, 2018.

Xin Lv, Lei Hou, Juanzi Li, and Zhiyuan Liu. Differentiating concepts and instances for knowledge graph embedding. *arXiv preprint arXiv:1811.04588*, 2018.

Shiheng Ma, Jianhui Ding, Weijia Jia, Kun Wang, and Minyi Guo. Transt: Type-based multiple embedding representations for knowledge graph completion. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pp. 717–733. Springer, 2017.

Yunpu Ma, Marcel Hildebrandt, Volker Tresp, and Stephan Baier. Holistic representations for memorization and inference. In *UAI*, pp. 403–413, 2018a.

Yunpu Ma, Volker Tresp, and Erik A Daxberger. Embedding models for episodic knowledge graphs. *Journal of Web Semantics*, pp. 100490, 2018b.

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

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

Grégoire Montavon, Sebastian Lapuschkin, Alexander Binder, Wojciech Samek, and Klaus-Robert Müller. Explaining nonlinear classification decisions with deep taylor decomposition. *Pattern Recognition*, 65:211–222, 2017.

Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. A three-way model for collective learning on multi-relational data. In *Icml*, volume 11, pp. 809–816, 2011.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, pp. 8024–8035, 2019.

Lin Qiu, Yunxuan Xiao, Yanru Qu, Hao Zhou, Lei Li, Weinan Zhang, and Yong Yu. Dynamically fused graph network for multi-hop reasoning. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 6140–6150, 2019.

Hongyu Ren, Weihua Hu, and Jure Leskovec. Query2box: Reasoning over knowledge graphs in vector space using box embeddings. *arXiv preprint arXiv:2002.05969*, 2020.

Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne Van Den Berg, Ivan Titov, and Max Welling. Modeling relational data with graph convolutional networks. In *European Semantic Web Conference*, pp. 593–607. Springer, 2018.

Zhiqing Sun, Zhi-Hong Deng, Jian-Yun Nie, and Jian Tang. Rotate: Knowledge graph embedding by relational rotation in complex space. *arXiv preprint arXiv:1902.10197*, 2019.

Komal K Teru, Etienne Denis, and William L Hamilton. Inductive relation prediction by subgraph reasoning. *arXiv preprint arXiv:1911.06962*, 2019.

Rakshit Trivedi, Hanjun Dai, Yichen Wang, and Le Song. Know-evolve: Deep temporal reasoning for dynamic knowledge graphs. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70, pp. 3462–3471. JMLR. org, 2017.

Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. Complex embeddings for simple link prediction. International Conference on Machine Learning (ICML), 2016.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

Xiang Wang, Dingxian Wang, Canran Xu, Xiangnan He, Yixin Cao, and Tat-Seng Chua. Explainable reasoning over knowledge graphs for recommendation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pp. 5329–5336, 2019.

Da Xu, Chuanwei Ruan, Evren Korpeoglu, Sushant Kumar, and Kannan Achan. Inductive representation learning on temporal graphs. *arXiv preprint arXiv:2002.07962*, 2020.

Xiaoran Xu, Wei Feng, Yunsheng Jiang, Xiaohui Xie, Zhiqing Sun, and Zhi-Hong Deng. Dynamically pruned message passing networks for large-scale knowledge graph reasoning. *arXiv preprint arXiv:1909.11334*, 2019.

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

Zhitao Ying, Dylan Bourgeois, Jiaxuan You, Marinka Zitnik, and Jure Leskovec. Gnnexplainer: Generating explanations for graph neural networks. In *Advances in neural information processing systems*, pp. 9244–9255, 2019.

Cunchao Zhu, Muhao Chen, Changjun Fan, Guangquan Cheng, and Yan Zhan. Learning from history: Modeling temporal knowledge graphs with sequential copy-generation networks, 2020.

APPENDIX

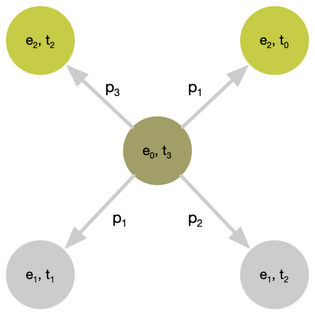


Figure 4: The inference graph for the query $(e_0, p_1, ?, t_3)$. The entity at an arrow's tail, the predicate on the arrow, the entity and the timestamp at the arrow's head build a true quadruple. Specifically, the true quadruples in this graph are as follows: $\{(e_0, p_1, e_1, t_1),\ (e_0, p_2, e_1, t_2), (e_0, p_3, e_2, t_2), (e_0, p_1, e_2, t_0)\}$. Note that t_3 is posterior to t_0, t_1, t_2.

Algorithm 1 Reverse Representation Update at the L^{th} Inference Step

Input: Inference graph $\mathcal{G}_{inf}$, nodes in the inference graph $\mathcal{V}$, nodes that have been added into $\mathcal{G}_{inf}$ at the l^{th} inference step $\mathcal{V}^l$, sampled prior neighbors $\hat{\mathcal{N}}_v$, hidden representation at the $(L\text{-}1)^{th}$ step $\mathbf{h}_v^{L-1}$, entity embeddings $\mathbf{e}_i$, weight matrices $\mathbf{W}_{sub}^L$, $\mathbf{W}_{obj}^L$, and $\mathbf{W}_h^L$, query $q = (e_q, p_q, ?, t_q)$, update ratio γ.

Output: Hidden representations $\mathbf{h}_v^L$ at the L^{th} inference step, $\forall v \in \mathcal{V}$.

```
1:  for $l = L-1, ..., 0$ do
2:      for $v \in \mathcal{V}^l$ do
3:          for $u \in \hat{\mathcal{N}}_v$ do
4:              $e_{vu}^L(q, p_k) = \mathbf{W}_{sub}^L(\mathbf{h}_v^{L-1}||\mathbf{p}_k||\mathbf{h}_{e_q}^{L-1}||\mathbf{p}_q)\mathbf{W}_{obj}^L(\mathbf{h}_u^{L-1}||\mathbf{p}_k||\mathbf{h}_{e_q}^{L-1}||\mathbf{p}_q)$,
5:              $\alpha_{vu}^L(q, p_k) = \frac{\exp(e_{vu}^L(q,p_k))}{\sum_{w \in \hat{\mathcal{N}}_v} \sum_{p_z \in \mathcal{P}_{vw}} e_{vw}^L(q,p_z)}$
6:          end for
7:          $\widetilde{\mathbf{h}}_v^L(q) = \sum_{u \in \hat{\mathcal{N}}_v} \sum_{o_z \in \mathcal{P}_{vu}} \alpha_{vu}^L(q, p_k)\mathbf{h}_u^{L-1}(q)$,
8:          $\mathbf{h}_v^L(q) = \sigma(\mathbf{W}_h^L(\gamma\mathbf{h}_v^{L-1}(q) + (1-\gamma)\widetilde{\mathbf{h}}_v^L(q) + \mathbf{b}_h^L))$
9:      end for
10: end for
11: Return $\mathbf{h}_v^L, \forall v \in \mathcal{V}$.
```

Parameter	Symbol
static entity embeddings	$\bar{\mathbf{e}}_i$
frequencies and phase shift of time encoding	$\mathbf{w}, \boldsymbol{\phi}$
predicate embeddings	$\mathbf{p}_k$
weight matrices of TRGA	$\mathbf{W}_{sub}^l, \mathbf{W}_{obj}^l, \mathbf{W}_h^l$
bias vector of TRGA	$\mathbf{b}_h^l$
weight matrix and bias of node embeddings	$\mathbf{W}_v, \mathbf{b}_v$

Table 2: Model parameters.

Sampling Strategies	MRR	Hits@1	Hits@3	Hits@10
Uniform	36.26	27.66	41.39	53.96
Time-aware exponentially weighted	41.56	32.49	47.27	59.63
Time-aware linearly weighted	38.21	29.25	43.77	56.07
Last-N-edges	39.84	31.31	45.04	57.40

Table 3: Comparison between model variants with different sampling strategies on ICEWS14 : raw MRR (%) and Hits@1/3/10 (%). In this ablation study, we filter out test triples that contain unseen entities.

A Related Work

A.1 Knowledge Graph Models

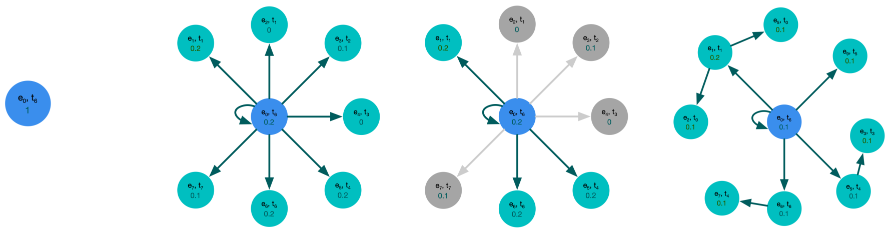
Representation learning is an expressive and popular paradigm underlying many KG models. The key idea is to embed entities and relations into a low-dimensional vector space. The embedding-based approaches for knowledge graphs can generally be categorized into bilinear models (Nickel et al., 2011; Balažević et al., 2019), translational models (Bordes et al., 2013; Sun et al., 2019), and deep-learning models (Dettmers et al., 2017; Schlichtkrull et al., 2018). Besides, several studies (Hao et al., 2019; Lv et al., 2018; Ma et al., 2017) explore the ontology of entity types and relation types and utilize type-based semantic similarity to produce better knowledge embeddings. However, the above methods lack the ability to use rich temporal dynamics available on temporal knowledge graphs. To this end, several studies have been conducted for link prediction on temporal knowledge graphs (Leblay & Chekol, 2018; García-Durán et al., 2018; Ma et al., 2018b; Dasgupta et al., 2018; Trivedi et al., 2017; Jin et al., 2019; Goel et al., 2019; Lacroix et al., 2020). Ma et al. (2018b) developed extensions of static knowledge graph models by adding timestamp embeddings to their score functions. Besides, García-Durán et al. (2018) suggested a straight forward extension of some existing static knowledge graph models that utilize a recurrent neural network (RNN) to encode predicates with temporal tokens derived from given timestamps. Also, HyTE (Dasgupta et al., 2018) embeds time information in the entity-relation space by arranging a temporal hyperplane to each timestamp. However, these models cannot generalize to unseen timestamps because they only learn embeddings for observed timestamps. Additionally, the methods are largely black-box, lacking the ability to interpret their predictions while our main focus is to employ an integrated transparency mechanism for achieving human-understandable results.

A.2 Explainable reasoning on Knowledge Graphs

Recently, several explainable reasoning methods for knowledge graphs have been proposed (Das et al., 2017; Xu et al., 2019; Hildebrandt et al., 2020) . Das et al. (2017) proposed a reinforcement learning-based path searching approach to display the query subject and predicate to the agents and let them perform a policy guided walk to the correct object entity. The reasoning paths produced by the agents can explain the prediction results to some extent. Also, Hildebrandt et al. (2020) framed the link prediction task as a debate game between two reinforcement learning agents that extract evidence from knowledge graphs and allow users to understand the decision made by the agents. Besides, and more related to our work, Xu et al. (2019) models a sequential reasoning process by dynamically constructing an input-dependent subgraph. The difference here is that these explainable methods can only deal with static KGs, while our model is designed for forecasting on temporal KGs.

B Workflow

We show the workflow of the subgraph reasoning process in Figure 5. The model conducts the reasoning process on a dynamically expanding inference graph $\mathcal{G}_{inf}$ extracted from the temporal KG. This inference graph gives an interpretable graphical explanation about the final prediction. Given a query $q = (e_q, p_q, ?, t_q)$, we initialize the inference graph with the query entity e_q and define the tuple of (e_q, t_q) as the first node in the inference graph (Figure 5a). The inference graph expands by



(a) Initialization. (b) Expansion. (c) Pruning. (d) New Expansion.

Figure 5: Inference step by step illustration. Node attention scores are attached to the nodes. Gray nodes are removed by the pruning procedure.

sample neighbors that have been linked with e_q prior to t_q, as shown in Figure 5b. The expansion would go rapidly that it covers almost all nodes after a few steps. To prevent the inference graph from exploding, we constrain the number of edge by pruning the edges that are less related to the query (Figure 5c) . Here, we propose a query-dependent temporal relational attention mechanism in Section 4.4 to identify the nodes' importance in the inference graph for query q and aggregate information from nodes' local neighbors. Next, we sample the prior neighbors of the remaining nodes in the inference graph to expand it further, as shown in Figure 5d. As this process iterates, the inference graph incrementally gains more and more information from the temporal KG. After running L inference steps, the model selects the entity with the highest attention score in $\mathcal{G}_{inf}$ as the prediction of the missing query object, where the inference graph itself serves as a graphical explanation.

2

C Dataset Statistics

Dataset	N_{train}	N_{valid}	N_{test}	N_{ent}	N_{rel}	$N_{timestamp}$	Time granularity
ICEWS14	63685	13823	13222	7128	230	365	day
ICEWS18	373018	45995	49545	23033	256	304	day
ICEWS0515	322958	69224	69147	10488	251	4017	day
YAGO	51205	10973	10973	10038	10	194	year

Table 4: Dataset Statistics

Dataset	$\vert\mathcal{E}_{tr}\vert$	$\vert\mathcal{E}_{tr}\vert/\vert\mathcal{E}\vert$	$\vert\mathcal{E}_{tr+val}\vert$	$\vert\mathcal{E}\vert$
ICEWS14	6180	86.7	6710	7128
ICEWS18	21085	91.5	21995	23033
ICEWS0515	8853	84.4	9792	10488
YAGO	7904	78.7	9008	10038

Table 5: Unseen entities (new emerging entities) in the validation set and test set. $|\mathcal{E}_{tr}|$ denotes the number of entities in the training set, $|\mathcal{E}_{tr+val}|$ represents the number of entities in the training set and validation set, $|\mathcal{E}|$ denotes the number of entities in the whole dataset.

We provide the statistics of datasets in Table 4. Since we split each dataset into subsets by timestamps, ensuring (timestamps of training set) $<$ (timestamps of validation set) $<$ (timestamps of test set), a considerable amount of entities in test sets is unseen. We report the number of entities in each subset in Table 5.

D Evaluation protocol

For each quadruple $q = (e_s, p, e_o, t)$ in the test set $\mathcal{G}_{test}$, we create two queries: $(e_s, p, ?, t)$ and $(e_o, p^{-1}, ?, t)$, where p^{-1} denotes the reciprocal relation of p. For each query, the model ranks all entities $\mathcal{E}_q^{inf}$ in the final inference graph according to their attention scores. If the ground truth entity does not appear in the final subgraph, we set its rank as $|\mathcal{E}|$ (the number of entities in the dataset). Let ψ_{e_s} and ψ_{e_o} represent the rank for e_s and e_o of the two queries respectively. We evaluate our model using standard metrics across the link prediction literature: *mean reciprocal rank (MRR)*: $\frac{1}{2\cdot|\mathcal{G}_{test}|}\sum_{q\in\mathcal{G}_{test}}(\frac{1}{\psi_{e_s}}+\frac{1}{\psi_{e_o}})$ and *Hits*$@k(k \in \{1,3,10\})$: the percentage of times that the true entity candidate appears in the top k of the ranked candidates.

In this paper, we consider two different filtering settings. The first one is following the ranking technique described in Bordes et al. (2013), where we remove from the list of corrupted **triples** all the **triples** that appear either in the training, validation, or test set. We name it *static filtering*. Trivedi et al. (2017), Jin et al. (2019), and Zhu et al. (2020) use this filtering setting for reporting their results on temporal KG forecasting. However, this filtering setting is not appropriate for evaluating the link prediction on temporal KGs. For example, there is a test quadruple (Barack Obama, visit, India, 2015-01-25), and we perform the object prediction (Barack Obama, visit, ?, 2015-01-25). We have observed the quadruple (Barack Obama, visit, Germany, 2013-01-18) in the training set. According to the *static filtering*, (Barack Obama, visit, Germany) will be considered as a genuine triple at the timestamp **2015-01-25** and will be filtered out because the triple (Barack Obama, visit, Germany) appears in the training set in the quadruple (Barack Obama, visit, Germany, 2015-01-18). However, the triple (Barack Obama, visit, Germany) is only temporally valid on 2013-01-18 but not on 2015-01-25. Therefore, we apply another filtering scheme, which is more appropriate for the link forecasting task on temporal KGs. We name it *time-aware filtering*. In this case, we only filter out the triples that are genuine at the timestamp of the query. In other words, if the triple (Barack Obama, visit, Germany) does not appear at the query time of 2015-01-25, the quadruple (Barack Obama, visit, Germany, 2015-01-25) is considered as corrupted and will be filtered out. We report the *time-aware* filtered results of baselines and our model in Table 1.

E Implementation

We implement our model and all baselines in PyTorch (Paszke et al., 2019). We tune hyperparameters of our model using a grid search. We set the learning rate to be 0.0002, the batch size to be 128, the inference step L to be 3. Please see the source code[5] for detailed hyperparameter settings. We implement TTransE, TA-TransE/TA-DistMult, and RE-Net based on the code[6] provided in (Jin et al., 2019). We use the released code to implement DE-SimplE[7], TNTComplEx[8], and CyGNet[9]. We use the binary cross-entropy loss to train these baselines and optimize hyperparameters according to MRR on the validation set. Besides, we use the datasets augmented with reciprocal relations to train all baseline models.

F Reverse representation update mechanism for subgraph reasoning

In this section, we explain an additional reason why we have to update node representations along edges selected in previous inference steps. We show our intuition by a simple query in Figure 6 with two inference steps. For simplicity, we do not apply the pruning procedure here. First, we check the equations without updating node representations along previously selected edges. h_i^l denotes the

[5]https://github.com/TemporalKGTeam/xERTE

[6]https://github.com/INK-USC/RE-Net

[7]https://github.com/BorealisAI/de-simple

[8]https://github.com/facebookresearch/tkbc

[9]https://github.com/CunchaoZ/CyGNet

hidden representation of node i at the l^{th} inference step.

$$\begin{aligned}\textbf{First inference step:}\quad & \mathbf{h}_0^1 = f(\mathbf{h}_0^0, \mathbf{h}_1^0, \mathbf{h}_2^0, \mathbf{h}_3^0) \\ & \mathbf{h}_1^1 = f(\mathbf{h}_1^0) \\ & \mathbf{h}_2^1 = f(\mathbf{h}_2^0) \\ & \mathbf{h}_3^1 = f(\mathbf{h}_3^0)\end{aligned}$$

$$\begin{aligned}\textbf{Second inference step:}\quad & \mathbf{h}_0^2 = f(\mathbf{h}_0^1, \mathbf{h}_1^1, \mathbf{h}_2^1, \mathbf{h}_3^1) \\ & \quad\;\; = f(\mathbf{h}_0^1, f(\mathbf{h}_1^0), f(\mathbf{h}_2^0), f(\mathbf{h}_3^0)) \\ & \mathbf{h}_1^2 = f(\mathbf{h}_1^1, \mathbf{h}_4^1, \mathbf{h}_5^1) \\ & \mathbf{h}_2^2 = f(\mathbf{h}_2^1, \mathbf{h}_7^1, \mathbf{h}_8^1) \\ & \mathbf{h}_3^2 = f(\mathbf{h}_3^1, \mathbf{h}_6^1)\end{aligned}$$

Note that $\mathbf{h}_0^2$ is updated with $\mathbf{h}_1^0, \mathbf{h}_2^0, \mathbf{h}_3^0$ and has nothing to do with $\mathbf{h}_4^1, \mathbf{h}_5^1, \mathbf{h}_6^1, \mathbf{h}_7^1, \mathbf{h}_8^1$, i.e., two-hop neighbors. In comparison, if we update the node representations along previously selected edges, the update in second layer changes to:

$$\begin{aligned}\textbf{Second inference step part a:}\quad & \mathbf{h}_4^2 = f(f(\mathbf{h}_4^0)) \\ & \mathbf{h}_5^2 = f(f(\mathbf{h}_5^0)) \\ & \mathbf{h}_6^2 = f(f(\mathbf{h}_6^0)) \\ & \mathbf{h}_7^2 = f(f(\mathbf{h}_7^0)) \\ & \mathbf{h}_8^2 = f(f(\mathbf{h}_8^0)) \\ \textbf{Second inference step part b:}\quad & \mathbf{h}_1^2 = f(\mathbf{h}_1^1, \mathbf{h}_4^2, \mathbf{h}_5^2) \\ & \mathbf{h}_2^2 = f(\mathbf{h}_2^1, \mathbf{h}_7^2, \mathbf{h}_8^2) \\ & \mathbf{h}_3^2 = f(\mathbf{h}_3^1, \mathbf{h}_6^2)\end{aligned}$$

$$\textbf{Second inference step part c:}\quad \mathbf{h}_0^2 = f(\mathbf{h}_0^1, \mathbf{h}_1^2, \mathbf{h}_2^2, \mathbf{h}_3^2)$$

Thus, the node 1$\sim$3 receive messages from their one-hop prior neighbors, i.e. $\mathbf{h}_1^2 = f(\mathbf{h}_1^1, \mathbf{h}_4^2, \mathbf{h}_5^2)$. Then they pass the information to the query subject (node 0), i.e., $\mathbf{h}_0^2 = f(\mathbf{h}_0^1, \mathbf{h}_1^2, \mathbf{h}_2^2, \mathbf{h}_3^2)$.

G SEGMENT OPERATIONS

The degree of entities in temporal KGs, i.e., ICEWS, varies from thousands to a single digit. Thus, the size of inference graphs of each query is also different. To optimize the batch training, we define an array to record all nodes in inference graphs for a batch of queries. Each node is represented by a tuple of (inference graph index, entity index, timestamp, node index). The node index is the unique index to distinguish the same node in different inference graphs.

Note that the inference graphs of two queries may overlap, which means they have the same nodes in their inference graphs. But the query-dependent node representations would be distinct in different inference graphs. To avoid mixing information across different queries, we need to make sure that tensor operations can be applied separately to nodes in different inference graphs. Instead of iterating through each inference graph, we develop a series of segment operations based on matrix multiplication. The segment operations significantly improve time efficiency and reduce the time cost. We report the improvement of time efficiency on ICEWS14 in Table 6. Additionally, we list two examples of segment operations in the following.

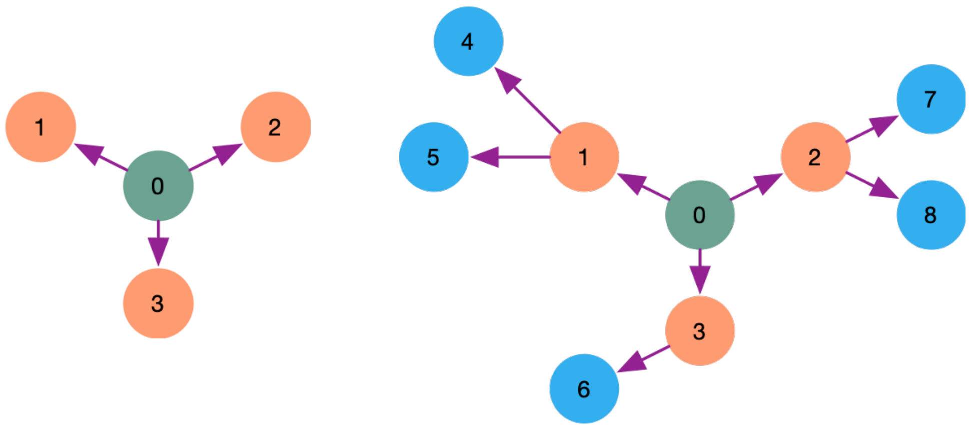


Figure 6: A simple example with two inference steps for illustrating reverse node representation update schema. The graph is initialized with the green node. In the first step (the left figure), orange nodes are sampled; and in the second step (the right figure), blue nodes are sampled. Each directed edge points from a source node to its prior neighbor.

Computations	Time Cost using Iterator	Time Cost using Segment Operation
Aggregation of Node Score	11.75s	0.004s
Aggregation of Entity Score	3.62s	0.026s
Softmax	1.56s	0.017s
Node Score Normalization	0.000738s	0.000047s

Table 6: Reduction of time cost for a batch on ICEWS14

Segment Sum Given a vector $\mathbf{x} \in \mathbb{R}^d$ and another vector $\mathbf{s} \in \mathbb{R}^d$ that indicates the segment index of each element in $\mathbf{x}$, the segment sum operator returns the summation for each segment. For example, we have $\mathbf{x} = [3, 1, 5]^T$ and $\mathbf{s} = [0, 0, 1]^T$, which means the first two element of $\mathbf{x}$ belong to the 0^{th} segment and the last elements belongs to the first segment. The segment sum operator returns $[4, 5]^T$ as the output. It is realized by creating a sparse matrix $\mathbf{Y} \in \mathbb{R}^{n \times d}$, where n denotes the number of segments. We set 1 in positions $\{(\mathbf{s}[i], i), \quad \forall i \in \{0, ..., d\}\}$ of $\mathbf{Y}$ and pad other positions with zeros. Finally, we multiply $\mathbf{Y}$ with $\mathbf{x}$ to get the sum of each segment.

Segment Softmax The standard softmax function $\sigma : \mathbb{R}^K \to \mathbb{R}^K$ is defined as:

$$\sigma(\mathbf{z})_i = \frac{\exp(z_i)}{\sum_{j=1}^{K} \exp(z_j)}$$

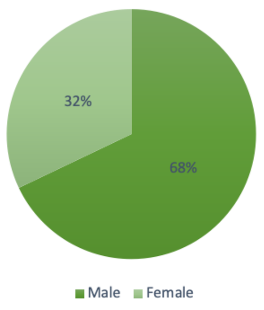
The segment softmax function has two inputs: $\mathbf{z} \in \mathbb{R}^K$ contains elements to normalize and $\mathbf{s} \in \mathbb{R}^K$ denotes the segment index of each element. It is then defined as:

$$\sigma(\mathbf{z})_i = \frac{\exp(z_i)}{\sum_{j \in \{k | s_k = s_i, \forall k \in \{0,...,K\}\}} \exp(z_j)}$$

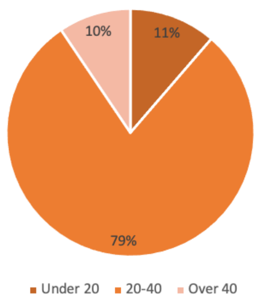
, where s_i denotes the segment that z_i is in.

The segment softmax function can be calculated by the following steps:

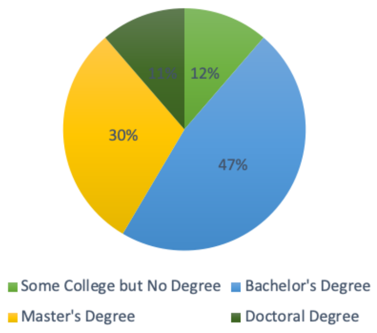
1. We apply the exponential function to each element of $\mathbf{z}$ and then apply the segment sum operator to get a denominator vector $\mathbf{d}$. We need broadcast $\mathbf{d}$ such that it aligns with $\mathbf{z}$, which means $\mathbf{d}[i]$ is the summation of segment $\mathbf{s}[i]$.
2. We apply element-wise division between $\mathbf{d}$ and $\mathbf{z}$.



(a) Gender distribution.



(b) Age distribution.



(c) Education level.

Figure 7: Information about the respondent population.

H SURVEY

In this section, we provide the online survey (see Section 5.3 in the main body) and the evaluation statistics based on 53 respondents. To avoid biasing the respondents, we did not inform them about the type of our project. Further, all questions are permuted at random.

We set up the quiz consisting of 7 rounds. In each round, we sample a query from the test set of ICEWS14/ICEWS0515. Along with the query and the ground-truth object, we present the users with two pieces of evidence extracted from the inference graph with high contribution scores and two pieces of evidence with low contribution scores in randomized order. The respondents are supposed to judge the relevance of the evidence to the query in two levels, namely relevant or less relevant. There are three questions in each round that ask the participants to give the most relevant evidence, the most irrelevant evidence, and rank the four pieces of evidence according to their relevance. The answer to the first question is classified as correct if a participant gives one of the two statements with high contribution scores as the most relevant evidence. Similarly, the answer to the second question is classified as correct if the participant gives one of the two statements with low contribution scores as the most irrelevant evidence. For the relevance ranking task, the answer is right if the participant ranks the two statements with high contribution scores higher than the two statements with low contribution scores.

H.1 POPULATION

We provide the information about gender, age, and education level of the respondents in Figure 7.

H.2 AI QUIZ

You will participate in a quiz consisting of eight rounds. Each round is centered around an international event. Along with the event, we also show you four reasons that explain why the given event happened. While some evidence may be informative and explain the occurrence of this event, others may irrelevant to this event. Your task is to find the most relevant evidence and most irrelevant evidence, and then sort all four evidence according to their relevance. Don't worry if you feel that you cannot make an informed decision: Guessing is part of this game!

Additional Remarks: Please don't look for external information (e.g., Google, Wikipedia) or talk to other respondents about the quiz. But you are allowed to use a dictionary if you need vocabulary clarifications.

Example

Given an event, please rank the followed evidence according to the relevance to the given event. Especially, please select the most relevant reason, the most irrelevant reason, and rank the relevance from high to low.

Event: French government made an optimistic comment about China on 2014-11-24.

A. First, on 2014-11-20, South Africa engaged in diplomatic cooperation with Morocco. Later, on 2014-11-21, a representative of the Morocco government met a representative of the French government.

B. First, on 2014-11-18, the Chinese government engaged in negotiation with the Iranian government. Later, on 2014-11-21, a representative of the French government met a representative of the Chinese government.

C. On 2014-11-23, the French hosted a visit by Abdel Fattah Al-Sisi.

D. A representative of the French government met a representative of the Chinese government on 2014-11-21.

Correct answer

Most relevant: D Most irrelevant: A Relevance ranking: D B C A

Tasks

1. Event: On 2014-12-17, the UN Security Council accused South Sudan.

A. South Africa engaged in diplomatic cooperation with South Sudan on 2014-12-11.

B. First, on 2014-11-17, Uhuru Muigai Kenyatta accused UN Security Council. Later, on 2014-11-26, the UN Security Council provided military protection to South Sudan.

C. On 2014-12-16, UN Security Council threatened South Sudan with sanctions.

D. South Sudan hosted the visit of John Kerry on 2014-12-16.

Most relevant: Most irrelevant: Relevance ranking:

2. Event: Indonesia police arrested and retained an Indonesia citizen at 2014-12-28.

A. The Indonesia police claimed that an attorney denounced the citizen on 2014-12-10.

B. Zaini Abdullah endorsed the Indonesia citizen on 2014-12-25.

C. The Indonesia police made an optimistic comment on the citizen on 2014-12-14.

D. The Indonesia police investigated the citizen on 2014-12-08.

Most relevant: Most irrelevant: Relevance ranking:

3. Event: A citizen from Greece protested violently against the police of Greece on 2014-11-17.

A. The Greek head of government accused the political party “Coalition of the Radical Left” on 2014-05-25.

B. Greek police refused to surrender to the Greek head of government on 2014-10-15.

C. Greek citizens gathered support on behalf of John Kerry on 2014-11-17.

D. Greek police arrested and detained another Greek police officer on 2014-11-04.

Most relevant: Most irrelevant: Relevance ranking:

4. Event: Raúl Castro signed a formal agreement with Barack Obama on 2014-12-17.

A. First, on 2009-01-28, Dmitry A. Medvedev made statements to Barack Obama. Later, on 2009-01-30, Raúl Castro negotiated with Dmitry A. Medvedev.

B. Raúl Castro visited Angola on 2009-07-22.

C. Raúl Castro hosted a visit of Evo Morales on 2011-09-19.

D. First, on 2008-11-05, Evo Morales hosted a visit of Barack Obama. Later, on 2011-09-19, Raúl Castro appeal for de-escalation of military engagement to Evo Morales.

Most relevant: Most irrelevant: Relevance ranking:

5. Event: The head of the government of Ukraine considered to make a policy option with Angela Merkel on 2015-07-10.

A. First, on 2014-07-04, the armed rebel in Ukraine used unconventional violence to the military of Ukraine. Later, on 2014-07-10, the head of government of Ukraine made statements to the armed rebel in Ukraine.

B. The head of the government of Ukraine expressed intent to meet with Angela Merkel on 2014-10-30.

C. First, on 2014-07-04, the armed rebel in Ukraine used unconventional violence to the military of Ukraine. Later, on 2014-07-19, the head of government of Ukraine made statements to the armed rebel in Ukraine.

D. The head of the government of Ukraine consulted with Angela Merkel on 2015-06-06.

Most relevant: Most irrelevant: Relevance ranking:

6. Event: On 2014-08-09, Ukraine police arrested a member of the Ukraine military.

A. First, on 2014-07-23, a member of Ukraine parliament consulted the head of the Ukraine government. Later, on 2014-07-24, the head of government made a statement to the Ukraine police.

B. First, on 2014-06-25, the military of Ukraine used violence to an armed rebel that occurred in Ukraine. Later, on 2014-07-10, the armed rebel used violence to the Ukraine police.

C. First, on 2005-02-20, the military of Ukraine made a statement to the head of the government of Ukraine. Later, on 2005-07-18, the head of government of Ukraine appealed for a change in leadership of the Ukraine police.

D. On 2014-07-31, the head of the Ukraine government praised the Ukraine police.

Most relevant: Most irrelevant: Relevance ranking:

7. Event: The Office of Business Affairs of Bahrain negotiated with the Labor and Employment Ministry of Bahrain on 2015-07-16.

A. First, on 2014-07-27, the undersecretary of Bahrain made statements to the Labor and Employment Ministry of Bahrain. Later, on 2015-01-21, an officer of Business Affairs of Bahrain signed a formal agreement with the undersecretary of Bahrain.

B. On 2012-01-21, the office of Business Affairs of Bahrain expressed intent to provide policy support to the employees in Bahrain.

C. First, on 2006-11-01, the employees in Bahrain made statements with the special Rapporteurs of the United Nation. Later, on 2011-05-11, the office of Business Affairs of Bahrain reduced relations with the employees in Bahrain.

D. A representative of the Labor and Employment Ministry of Bahrain consulted with a representative of the Office of Business Affairs of Bahrain on 2014-01-31.

Most relevant: Most irrelevant: Relevance ranking:

H.3 Ground Truth Answers

Question 1:

Most relevant: B/C Most irrelevant: A/D

Relevance ranking: BCAD/BCDA/CBAD/CBDA

Question 2:

Most relevant: A/D Most irrelevant: B/C

Relevance ranking: ADBC/ADCB/DABC/DACB

Question 3:

Most relevant: B/D Most irrelevant: A/C

Relevance ranking: BDAC/BDCA/DBAC/DBCA

Question 4:

Most relevant: A/D Most irrelevant: B/C

Relevance ranking: ADBC/ADCB/DABC/DACB

Question 5:

Most relevant: B/D Most irrelevant: A/C

Relevance ranking: BDAC/BDCA/DBAC/DBCA

Question 6:

Most relevant: B/C. Most irrelevant: A/D

Relevance ranking: BCAD/BCDA/CBAD/CBDA

Question 7:

Most relevant: A/D Most irrelevant: B/C

Relevance ranking: ADBC/ADCB/DABC/DACB

H.4 EVALUATION

The evaluation results of 53 respondents are shown in Figure 8.

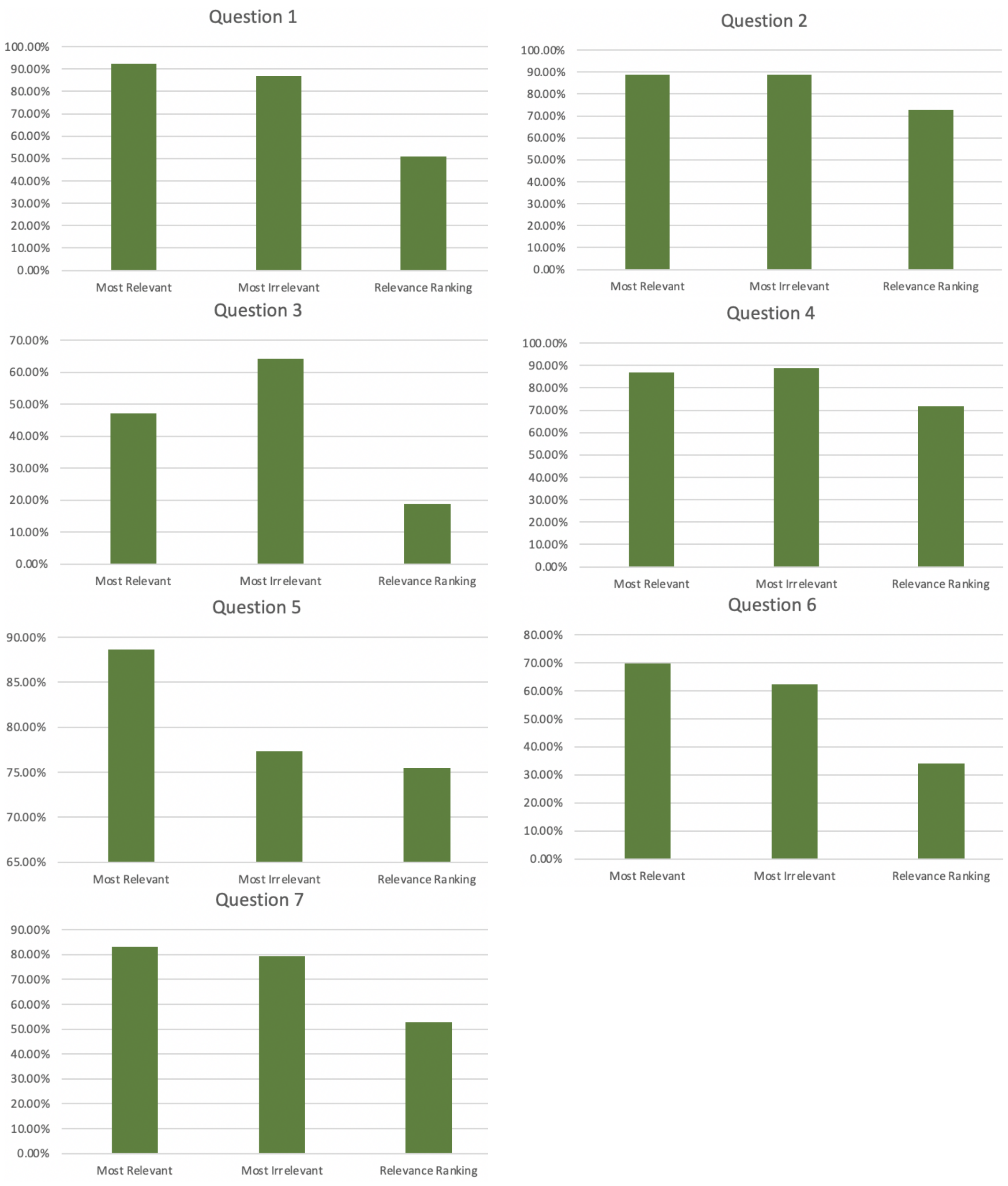


Figure 8: The accuracy of the survey questions.

I Additional analysis of time-aware entity representations

We use a generic time encoding (Xu et al., 2020) defined as $\mathbf{\Phi}(t) = \sqrt{\frac{1}{d}}[\cos(\omega_1 t + \phi_1), \dots, \cos(\omega_d t + \phi_d)]$ to generate the time-variant part of entity representations (please see Section 4.2 for more details). Time-aware representations have considerable influence on the temporal attention mechanism. To make our point, we conduct a case study and extract the edges' attention scores from the final inference graph. Specifically, we study the attention scores of the interactions between *military* and *student* at different timestamps in terms of the query (*student*, *criticize*, ?, Nov. 17, 2014). We list the results of the model with time encoding in Table 7 and the results of the model without time encoding in Table 8.

As shown in Table 7, by means of the time-encoding, quadruples that even have the same subject, predicate, and object have different attention scores. Specifically, quadruples that occurred recently tend to have higher attention scores. This makes our model more interpretable and effective. For example, given three quadruples $\{$(country A, accuse, country B, t_1), (country A, express intent to negotiate with, country B, t_2), (country A, cooperate with, country B, t_3)$\}$, country A probably has a good relationship with B at t if $(t_1 < t_2 < t_3 < t)$ holds. However, there would be a strained relationship between A and B at t if $(t > t_1 > t_2 > t_3)$ holds. Thus, we can see that the time information is crucial to the reasoning, and attention values should be time-dependent. In comparison, Table 8 shows that the triple (military, use conventional military force, student) has randomly different attention scores at different timestamps, which is less interpretable.

Subject	Object	Predicate	Timestamp	Attention Score
Military	Student	Use conventional military force	Jan. 17, 2014	0.0123
Military	Student	Use conventional military force	May 22, 2014	0.0186
Military	Student	Use conventional military force	Aug. 18, 2014	0.0235
Military	Student	Use unconventional violence (reciprocal relation)	Aug. 25, 2014	0.0348

Table 7: Attention scores of the interactions between *military* and *student* at different timestamps (with time encoding).

Subject	Object	Predicate	Timestamp	Attention Score
Military	Student	Use conventional military force	Jan. 17, 2014	0.0152
Military	Student	Use conventional military force	May 22, 2014	0.0122
Military	Student	Use conventional military force	Aug. 18, 2014	0.0159
Military	Student	Use unconventional violence (reciprocal relation)	Aug. 25, 2014	0.0021

Table 8: Attention scores of the interactions between *military* and *student* at different timestamps (without time encoding).

Chapter 4

Learning Neural Ordinary Equations for Forecasting Future Links on Temporal Knowledge Graphs

This chapter contains the publication

Zhen Han, Zifeng Ding, Yunpu Ma, Jiayu Gu, Volker Tresp. Learning Neural Ordinary Equations for Forecasting Future Links on Temporal Knowledge Graphs. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Online and in Dominican Republic, Nov. 2021. DOI: 10.18653/v1/2021.emnlp-main.658.

Learning Neural Ordinary Equations for Forecasting Future Links on Temporal Knowledge Graphs

Zhen Han[*1,2], **Zifeng Ding**[*1,2], **Yunpu Ma**[*1], **Yujia Gu**[3] **Volker Tresp**[†1,2]
[1]Institute of Informatics, LMU Munich [2] Corporate Technology, Siemens AG
[3]Department of Electrical and Computer Engineering, Technical University of Munich
zhen.han@campus.lmu.de, cognitive.yunpu@gmail.com
{zifeng.ding, volker.tresp}@siemens.com, yujia.gu@tum.de

Abstract

There has been an increasing interest in inferring future links on temporal knowledge graphs (KG). While links on temporal KGs vary continuously over time, the existing approaches model the temporal KGs in discrete state spaces. To this end, we propose a novel continuum model by extending the idea of neural ordinary differential equations (ODEs) to multi-relational graph convolutional networks. The proposed model preserves the continuous nature of dynamic multi-relational graph data and encodes both temporal and structural information into continuous-time dynamic embeddings. In addition, a novel graph transition layer is applied to capture the transitions on the dynamic graph, i.e., edge formation and dissolution. We perform extensive experiments on five benchmark datasets for temporal KG reasoning, showing our model's superior performance on the future link forecasting task.

1 Introduction

Reasoning on relational data has long been considered an essential subject in artificial intelligence with wide applications, including decision support and question answering. Recently, reasoning on knowledge graphs has gained increasing interest (Ren and Leskovec, 2020; Das et al., 2018). A Knowledge Graph (KG) is a graph-structured knowledge base to store factual information. KGs represent facts in the form of triples (s, r, o), e.g., (*Bob*, *livesIn*, *New York*), in which s (subject) and o (object) denote nodes (entities), and r denotes the edge type (relation) between s and o. Knowledge graphs are commonly static and store facts in their current state. In reality, however, the relations between entities often change over time. For example, if Bob moves to California, the triple of (*Bob*, *livesIn*, *New York*) will be invalid. To this end, temporal knowledge graphs (tKG) were introduced. A tKG represents a temporal fact as a quadruple (s, r, o, t) by extending a static triple with time t, describing that this fact is valid at time t. In recent years, several sizable temporal knowledge graphs, such as ICEWS (Boschee et al., 2015), have been developed that provide widespread availability of such data and enable reasoning on temporal KGs. While lots of work (García-Durán et al., 2018; Goel et al., 2020; Lacroix et al., 2020) focus on the temporal KG completion task and predict missing links at observed timestamps, recent work (Jin et al., 2019; Trivedi et al., 2017) paid attention to forecast future links of temporal KGs. In this work, we focus on the temporal KG forecasting task, which is more challenging than the completion task.

Most existing work (Jin et al., 2019; Zhu et al., 2020) models temporal KGs in a discrete-time domain where they take snapshots of temporal KGs sampled at regularly-spaced timestamps. Thus, these approaches cannot model irregular time intervals, which convey essential information for analyzing dynamics on temporal KGs, e.g., the dwelling time of a user on a website becomes shorter, indicating that the user's interest in the website decreases. KnowEvolve (Trivedi et al., 2017) uses a neural point process to model continuous-time temporal KGs. However, Know-Evolve does not take the graph's structural information into account, thus losing the power of modeling temporal topological information. Also, KnowEolve is a transductive method that cannot handle unseen nodes. In this paper, we present a graph neural-based approach to learn dynamic representations of entities and relations on temporal KGs. Specifically, we propose a graph neural ordinary differential equation to model the graph dynamics in the continuous-time domain.

Inspired by neural ordinary differential equations (NODEs) (Chen et al., 2018), we extend the idea of continuum-depth models to encode the continuous dynamics of temporal KGs. To apply NODEs

[*]Equal contribution.
[†]Corresponding author.

to temporal KG reasoning, we employ a NODE coupled with multi-relational graph convolutional (MGCN) layers. MGCN layers are used to capture the structural information of multi-relational graph data, while the NODE learns the evolution of temporal KGs over time. Specifically, we integrate the hidden representations over time using an ODE solver and output the continuous-time dynamic representations of entities and relations. Unlike many existing temporal KG models that learn the dynamics by employing recurrent model structures with discrete depth, our model lets the time domain coincide with the depth of a neural network and takes advantage of NODE to steer the latent entity features between two timestamps smoothly. Besides, existing work simply uses the adjacency tensor from previous snapshots of the tKG to predict its linkage structure at a future time. Usually, most edges do not change between two observations, while only a few new edges have formatted or dissolved since the last observation. However, the dissolution and formation of these small amounts of edges always contain valuable temporal information and are more critical than unchanged edges for learning the graph dynamics. For example, we know an edge with the label *economicallyCooperateWith* between two countries x and y at time t, but this dissolves at $t + \Delta t_1$. Additionally, there is another edge with the label *banTradesWith* between these two countries that are formated at $t + \Delta t_2$ ($\Delta t_2 > \Delta t_1$). Intuitively, the dissolution of (x, *economicallyCooperateWith*, y) is an essential indicator of the quadruple (x, *banTradesWith*, y, $t + \Delta t_2$). Thus, it should get more attention from the model. However, suppose we only feed the adjacency tensors of different observation snapshots into the model. In that case, we do not know whether the model can effectively capture the changes of the adjacency tensors and puts more attention on the evolving part of the graph. To let the model focus on the graph's transitions, we propose a graph transition layer that takes a graph transition tensor containing edge formation and dissolution information as input and uses graph convolutions to process the transition information explicitly.

In this work, we propose a model to perform **T**emporal Knowledge Gr**a**ph Forecasti**ng** with Neural **O**rdinary Equations (TANGO💃). The main contributions are summarized as follows:

- We propose a continuous-depth multi-relational graph neural network for forecasting future links on temporal KGs by defining a multi-relational graph neural ordinary differential equation. The ODE enables our model to learn continuous-time representations of entities and relations. We are the first to show that the neural ODE framework can be extended to modeling dynamic multi-relational graphs.

- We propose a graph transition layer to model the edge formation and dissolution of temporal KGs, which effectively improves our model's performance.

- We propose two new tasks, i.e., inductive link prediction and long horizontal link forecasting, for temporal KG models. They evaluate a model's potential by testing the model's performance on previously unseen entities and predicting the links happening in the farther future.

- We apply our model to forecast future links on five benchmark temporal knowledge graph datasets, showing its state-of-the-art performance.

2 Preliminaries and Related Work

2.1 Graph Convolutional Networks

Graph convolutional networks (GCNs) have shown great success in capturing structural dependencies of graph data. GCNs come in two classes: i) spectral methods (Kipf and Welling, 2016; Defferrard et al., 2016) and ii) spatial methods (Niepert et al., 2016; Gilmer et al., 2017). However, common GCNs can only deal with homogeneous graphs. To distinguish between different relations, R-GCN (Schlichtkrull et al., 2017) introduces relation-specific weight matrices for message transformations. However, the number of parameters in R-GCN grows rapidly with the number of relations, easily leading to overfitting. Vashishth et al. (2019) proposed a multi-relational GCN, which is compatible with KGs and leverages various entity-relation composition operations from KG embedding techniques. Additionally, some work combines GCN with temporal graphs (Yan et al., 2018; Li et al., 2020). However, they are designed for homogeneous graphs but not for multi-relational graphs.

2.2 Neural Ordinary Differential Equations

Neural Ordinary Differential Equation (NODE) (Chen et al., 2018) is a continuous-depth deep neural network model. It represents the derivative of the hidden state with a neural network:

$$\frac{d\mathbf{z}(t)}{dt} = f(\mathbf{z}(t), t, \theta), \tag{1}$$

where $\mathbf{z}(t)$ denotes the hidden state of a dynamic system at time t, and f denotes a function parameterized by a neural network to describe the derivative of the hidden state regarding time. θ represents the parameters in the neural network. The output of a NODE framework is calculated using an ODE solver coupled with an initial value:

$$\mathbf{z}(t_1) = \mathbf{z}(t_0) + \int_{t_0}^{t_1} f(\mathbf{z}(t), t, \theta)dt. \tag{2}$$

Here, t_0 is the initial time point, and t_1 is the output time point. $\mathbf{z}(t_1)$ and $\mathbf{z}(t_0)$ represent the hidden state at t_1 and t_0, respectively. Thus, the NODE can output the hidden state of a dynamic system at any time point and deal with continuous-time data, which is extremely useful in modeling continuous-time dynamic systems.

Moreover, to reduce the memory cost in the back-propagation, Chen et al. (2018) introduced the adjoint sensitivity method into NODEs. An adjoint is $\mathbf{a}(t) = \frac{\partial \mathcal{L}}{\partial \mathbf{z}(t)}$, where $\mathcal{L}$ means the loss. The gradient of $\mathcal{L}$ with regard to network parameters θ can be directly computed by the adjoint and an ODE solver:

$$\frac{d\mathcal{L}}{d\theta} = -\int_{t_1}^{t_0} \mathbf{a}(t)^T \frac{\partial f(\mathbf{z}(t), t, \theta)}{\partial \theta} dt. \tag{3}$$

In other words, the adjoint sensitivity method solves an augmented ODE backward in time and computes the gradients without backpropagating through the operations of the solver.

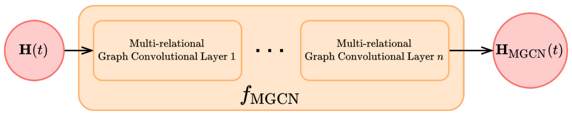
2.3 Temporal Knowledge Graph Reasoning

Let $\mathcal{V}$ and $\mathcal{R}$ represent a finite set of entities and relations, respectively. A temporal knowledge graph (tKG) $\mathcal{G}$ is a multi-relational graph whose edges evolve over time. At any time point, a snapshot $\mathcal{G}(t)$ contains all valid edges at t. Note that the time interval between neighboring snapshots may not be regularly spaced. A quadruple $q = (s, r, o, t)$ describes a labeled timestamped edge at time t, where $r \in \mathcal{R}$ represents the relation between a subject entity $s \in \mathcal{V}$ and an object entity $o \in \mathcal{V}$. Formally, we define the tKG forecasting task as follows. Let (s_q, r_q, o_q, t_q) denote a target quadruple and $\mathcal{F}$ represent the set of all ground-truth quadruples. Given query $(s_q, r_q, ?, t_q)$ derived from the target quadruple and a set of observed events $\mathcal{O} = \{(s, r, o, t_i) \in \mathcal{F} | t_i < t_q\}$, the tKG forecasting task predicts the missing object entity o_q based on observed **past** events. Specifically, we consider all entities in set $\mathcal{V}$ as candidates and rank them by their scores to form a true quadruple together with the given subject-relation-pair (s_q, r_q) at time t_q. In this work, we add reciprocal relations for every quadruple, i.e., adding (o, r^{-1}, s, t) for every (s, r, o, t). Hence, the restriction to predict object entities does not lead to a loss of generality.

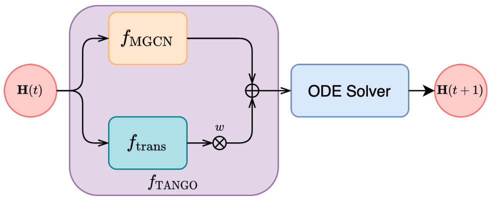
Extensive studies have been done for temporal KG **completion** task (Leblay and Chekol, 2018; García-Durán et al., 2018; Goel et al., 2020; Han et al., 2020a). Besides, a line of work (Trivedi et al., 2017; Jin et al., 2019; Deng et al., 2020; Zhu et al., 2020) has been proposed for the tKG **forecasting** task and can generalize to unseen timestamps. Specifically, Trivedi et al. (2017) and Han et al. (2020b) take advantage of temporal point processes to model the temporal KG as event sequences and learn evolving entity representations.

3 Our Model

Our model is designed to model time-evolving multi-relational graph data by learning continuous-time representations of entities. It consists of a neural ODE-based encoder and a decoder based on classic KG score functions. As shown in Figure 1b, the input of the network will be fed into two parallel modules before entering the ODE Solver. The upper module denotes a multi-relational graph convolutional layer that captures the graph's structural information according to an observation at time t. And the lower module denotes a graph transition layer that explicitly takes the edge transition tensor of the current observation representing which edges have been added and removed since the last observation. The graph transition layer focuses on modeling the graph *transition* between neighboring observations for improving the prediction of link formation and dissolution. For the decoder, we compare two score functions, i.e., DistMult (Yang et al., 2014) and TuckER (Balazevic et al., 2019). In principle, the decoder can be any score function.


$\mathbf{H}(t)$
Multi-relational Graph Convolutional Layer 1
Multi-relational Graph Convolutional Layer n
$\mathbf{H}_{\text{MGCN}}(t)$
f_{MGCN}

(a) f_{MGCN}



(b) f_{TANGO}

Figure 1: (a) The structure of f_{MGCN}: stacked multi-relational graph convolutional layers (the orange block). $\mathbf{H}(t)$ denotes the hidden representations of entities and relations at time t. $\mathbf{H}_{\text{MGCN}}(t)$ denotes the output of the stacked multi-relational graph convolutional layers. (b) The architecture of TANGO that parameterizes the derivatives of the hidden representations $\mathbf{H}(t)$. In addition to f_{MGCN}, a graph transition layer f_{trans} is employed to model the edge formation and dissolution.

3.1 Neural ODE for Temporal KG

The temporal dynamics of a time-evolving multi-relational graph can be characterized by the following neural ordinary differential equation

$$\begin{aligned}\frac{d\mathbf{H}(t)}{dt} =& f_{\text{TANGO}}(\mathbf{H}(t), \mathbf{T}(t), \mathcal{G}(t), t) \\ =& f_{\text{MGCN}}(\mathbf{H}(t), \mathcal{G}(t), t) \\ &+ w f_{\text{trans}}(\mathbf{H}(t), \mathbf{T}(t), \mathcal{G}(t), t),\end{aligned} \tag{4}$$

where $\mathbf{H} \in R^{(|\mathcal{V}|+2|\mathcal{R}|)\times d}$ denotes the hidden representations of entities and relations. f_{TANGO} represents the neural network that parameterizes the derivatives of the hidden representations. Besides, f_{MGCN} denotes stacked multi-relational graph convolutional layers, f_{trans} represents the graph transition layer, and $\mathcal{G}(t)$ denotes the snapshot of the temporal KG at time t. $\mathbf{T}(t)$ contains the information on edge formation and dissolution since the last observation. w is a hyperparameter controlling how much the model learns from edge formation and dissolution. We set $\mathbf{H}(t=0) = \text{Emb}(\mathcal{V}, \mathcal{R})$, where $\text{Emb}(\mathcal{V}, \mathcal{R})$ denotes the learnable initial embeddings of entities and relations on the temporal KG. Thus, given a time window Δt, the representation evolution performed by the neural ODE assumes the following form

$$\begin{aligned}&\mathbf{H}(t+\Delta t) - \mathbf{H}(t) \\ &= \int_t^{t+\Delta t} f_{\text{TANGO}}(\mathbf{H}(\tau), \mathbf{T}(\tau), \mathcal{G}(\tau), \tau)\, d\tau \\ &= \int_t^{t+\Delta t} (f_{\text{MGCN}}(\mathbf{H}(\tau), \mathcal{G}(\tau), \tau) \\ &\qquad + w f_{\text{trans}}(\mathbf{H}(\tau), \mathbf{T}(\tau), \tau))d\tau.\end{aligned} \tag{5}$$

In this way, we use the neural ODE to learn the dynamics of continuous-time temporal KGs.

3.2 Multi-Relational Graph Convolutional Layer

Inspired by (Vashishth et al., 2019) and (Yang et al., 2014), we use the entity-relation composition to model relational information. Specifically, we propose a multi-relational graph convolutional layer as follows. At time t, for every object entity $o \in \mathcal{V}$ with $\mathcal{N}(o) = \{(s,r)|(s,r,o,t) \in \mathcal{G}(t)\}$, its hidden representation evolves as

$$\begin{aligned}\tilde{\mathbf{h}}_o^{l+1}(t) &= \frac{1}{|\mathcal{N}(o)|} \sum_{(s,r)\in\mathcal{N}(o)} \mathbf{W}^l(\mathbf{h}_s^l(t) * \mathbf{h}_r), \\ \mathbf{h}_o^{l+1}(t) &= \mathbf{h}_o^l(t) + \delta\sigma(\tilde{\mathbf{h}}_o^{l+1}(t)),\end{aligned} \tag{6}$$

where $\mathbf{h}_o^{l+1}(t)$ denotes the hidden representation of the object o at the $(l+1)^{th}$ layer, $\mathbf{W}^l$ represents the weight matrix on the l^{th} layer, $*$ denotes element-wise multiplication. $\mathbf{h}_s^l(t)$ means the hidden representation of the subject s at the l^{th} layer. $\mathbf{h}_s^{l=0}(t) = \mathbf{h}_s(t)$ is obtained by the ODE Solver that integrates Equation 4 until t. δ is a learnable weight. In this work, we assume that the relation representations do not evolve, and thus, $\mathbf{h}_r$ is time-invariant. We use $ReLU(\cdot)$ as the activation function $\sigma(\cdot)$. From the view of the whole tKG, we use $\mathbf{H}(t)$ to represent the hidden representations of all entities and relations on the tKG. Besides, we use f_{MGCN} to denote the network consisting of multiple multi-relational graph convolutional layers (Equation 6).

3.3 Graph Transition Layer

To let the model focus on the graph's transitions, we define a transition tensor for tKGs and use graph convolutions to capture the information of edge formation and dissolution. Given two graph snapshots $\mathcal{G}(t-\Delta t)$ and $\mathcal{G}(t)$ at time $t-\Delta t$ and t, respectively, the graph transition tensor $\mathbf{T}(t)$ is defined as

$$\mathbf{T}(t) = \mathbf{A}(t) - \mathbf{A}(t-\Delta t), \tag{7}$$

where $\mathbf{A}(t) \in \{0,1\}^{|\mathcal{V}|\times|\mathcal{R}|\times|\mathcal{V}|}$ is a three-way adjacency tensor whose entries are set such that

$$A_{sro} = \begin{cases} 1, \text{if the triple } (s,r,o) \text{ exists at time } t, \\ 0, \text{otherwise.} \end{cases} \tag{8}$$

Intuitively, $\mathbf{T}(t) \in \{-1,0,1\}^{|\mathcal{V}|\times|\mathcal{R}|\times|\mathcal{V}|}$ contains the information of the edges' formation and dissolution since the last observation $\mathcal{G}(t-\Delta t)$. Specifically, $T_{sro}(t) = -1$ means that the triple (s,r,o) disappears at t, and $T_{sro}(t) = 1$ means that the triplet (s,r,o) is formatted at t. For all unchanged edges, their values in $\mathbf{T}(t)$ are equal to 0. Additionally, we use graph convolutions to extract the information provided by the graph transition tensor:

$$\begin{aligned} \tilde{\mathbf{h}}^{l+1}_{o,\text{trans}}(t) &= \mathbf{W}_{\text{trans}}(T_{sro}(t)(\mathbf{h}^l_s(t) * \mathbf{h}_r)) \\ \mathbf{h}^{l+1}_{o,\text{trans}}(t) &= \sigma\left(\frac{1}{|\mathcal{N}_T(o)|}\sum_{(s,r)\in\mathcal{N}_T(o)} \tilde{\mathbf{h}}^{l+1}_{o,\text{trans}}(t)\right) \end{aligned} \tag{9}$$

Here, $\mathbf{W}_{\text{trans}}$ is a trainable diagonal weight matrix and $\mathcal{N}_T(o) = \{(s,r)|T_{sro}(t) \neq 0)\}$. By employing this graph transition layer, we can better model the dynamics of temporal KGs. We use f_{trans} to denote Equation 9. By combining the multi-relational graph convolutional layers f_{MGCN} with the graph transition layer f_{trans}, we get our final network that parameterizes the derivatives of the hidden representations $\mathbf{H}(t)$, as shown in Figure 1b.

3.4 Learning and Inference

TANGO is an autoregressive model that forecasts the entity representation at time t by utilizing the graph information before t. To answer a link forecasting query $(s,r,?,t)$, TANGO takes three steps. First, TANGO computes the hidden representations $\mathbf{H}(t)$ of entities and relations at the time t. Then TANGO uses a score function to compute the scores of all quadruples $\{(s,r,o,t)|o \in \mathcal{V}\}$ accompanied with candidate entities. Finally, TANGO chooses the object with the highest score as its prediction.

Representation inference The representation inference procedure is done by an ODE Solver, which is $\mathbf{H}(t) = \text{ODESolver}(\mathbf{H}(t-\Delta t), f_{\text{TANGO}}, t-\Delta t, t, \Theta_{\text{TANGO}}, \mathcal{G})$. Adaptive ODE solvers may incur massive time consumption in our work. To keep the training time tractable, we use fixed-grid ODE solvers coupled with the Interpolated Reverse Dynamic Method (IRDM) proposed by Daulbaev et al. (2020). IRDM uses Barycentric Lagrange interpolation (Berrut and Trefethen, 2004) on Chebyshev grid (Tyrtyshnikov, 2012) to approximate the solution of the hidden states in the reverse-mode of NODE. Thus, IRDM can lower the time cost in the backpropagation and maintain good learning accuracy. Additional information about representation inference is provided in Appendix A.

Table 1: Score Functions. $\mathbf{h}_s, \mathbf{h}_r, \mathbf{h}_o$ denote the entity representations of the subject entity s, object entity o, and the representation of the relation r, respectively. d denotes the hidden dimension of representations. $\mathcal{W} \in \mathbb{R}^{d\times d\times d}$ is the core tensor specified in (Balazevic et al., 2019). As defined in (Tucker, 1964), $\times_1, \times_2, \times_3$ are three operators indicating the tensor product in three different modes.

Method	Score Function	
Distmult (Yang et al., 2014)	$< \mathbf{h}_s, \mathbf{h}_r, \mathbf{h}_o >$	$\mathbf{h}_s, \mathbf{h}_r, \mathbf{h}_o \in \mathbb{R}^d$
TuckER (Balazevic et al., 2019)	$\mathcal{W} \times_1 \mathbf{h}_s \times_2 \mathbf{h}_r \times_3 \mathbf{h}_o$	$\mathbf{h}_s, \mathbf{h}_r, \mathbf{h}_o \in \mathbb{R}^d$

Score function Given the entity and relation representations at the query time t_q, one can compute the scores of every triple at t_q. In our work, we take two popular knowledge graph embedding models, i.e., Distmult (Yang et al., 2014) and TuckER (Balazevic et al., 2019). Given triple (s,r,o), its score is computed as shown in Table 1.

Parameter Learning For parameter learning, we employ the cross-entropy loss:

$$\mathcal{L} = \sum_{(s,r,o,t)\in\mathcal{F}} -\log(f(o|s,r,t,\mathcal{V})), \tag{10}$$

where $f(o|s,r,t,\mathcal{V}) = \frac{\exp(score(\mathbf{h}_s(t),\mathbf{h}_r,\mathbf{h}_o(t)))}{\sum_{e\in\mathcal{V}}\exp(score(\mathbf{h}_s(t),\mathbf{h}_r,\mathbf{h}_e(t)))}$. $e \in \mathcal{V}$ represents an object candidate, and $score(\cdot)$ is the score function. $\mathcal{F}$ summarizes valid quadruples of the given tKG.

4 Experiments

4.1 Experimental Setup

We evaluate our model by performing future link prediction on five tKG datasets[1]. We compare TANGO's performance with several existing methods and evaluate its potential with inductive link prediction and long horizontal link forecasting. Besides, an ablation study is conducted to show the effectiveness of our graph transition layer.

[1]Code and datasets are available at https://github.com/TemporalKGTeam/TANGO.

4.1.1 Datasets

We use five benchmark datasets to evaluate TANGO: 1) ICEWS14 (Trivedi et al., 2017) 2) ICEWS18 (Boschee et al., 2015) 3) ICEWS05-15 (García-Durán et al., 2018) 4) YAGO (Mahdisoltani et al., 2013) 5) WIKI (Leblay and Chekol, 2018). Integrated Crisis Early Warning System (ICEWS) (Boschee et al., 2015) is a dataset consisting of timestamped political events, e.g., (*Barack Obama*, *visit*, *India*, 2015-01-25). Specifically, ICEWS14 contains events occurring in 2014, while ICEWS18 contains events from January 1, 2018, to October 31, 2018. ICEWS05-15 is a long-term dataset that contains the events between 2005 and 2015. WIKI and YAGO are two subsets extracted from Wikipedia and YAGO3 (Mahdisoltani et al., 2013), respectively. The details of each dataset and the dataset split strategy are provided in Appendix D.

4.1.2 Evaluation Metrics

We use two metrics to evaluate the model performance on extrapolated link prediction, namely Mean Reciprocal Rank (MRR) and Hits@1/3/10. MRR is the mean of the reciprocal values of the actual missing entities' ranks averaged by all the queries, while Hits@1/3/10 denotes the proportion of the actual missing entities ranked within the top 1/3/10. The filtering settings have been implemented differently by various authors. We report results based on two common implementations: i) time-aware (Han et al., 2021) and ii) time-unaware filtering (Jin et al., 2019). We provide a detailed evaluation protocol in Appendix B.

4.1.3 Baseline Methods

We compare our model performance with nine baselines. We take three static KG models as the static baselines, including Distmult (Yang et al., 2014), TuckER (Balazevic et al., 2019), and COMPGCN (Vashishth et al., 2019). For tKG baselines, we report the performance of TTransE (Leblay and Chekol, 2018), TA-Distmult (García-Durán et al., 2018), CyGNet (Zhu et al., 2020), DE-SimplE (Goel et al., 2020), TNTComplEx (Lacroix et al., 2020), and RE-Net (Jin et al., 2019). We provide implementation details of baselines and TANGO in Appendix C.

4.2 Experimental Results

4.2.1 Time-aware filtered Results

We run TANGO five times and report the averaged results. The time-aware filtered results are presented in Table 2, where 💃 denotes TANGO. As explained in Appendix B, we take the time-aware filtered setting as the fairest evaluation setting. Results demonstrate that TANGO 💃 outperforms all the static baselines on every dataset. This implies the importance of utilizing temporal information in tKG datasets. The comparison between Distmult and TANGO-Distmult shows the superiority of our NODE-based encoder, which can also be observed by the comparison between TuckER and TANGO-TuckER. Additionally, TANGO achieves much better results than COMPGCN, indicating our method's strength in incorporating temporal features into tKG representation learning.

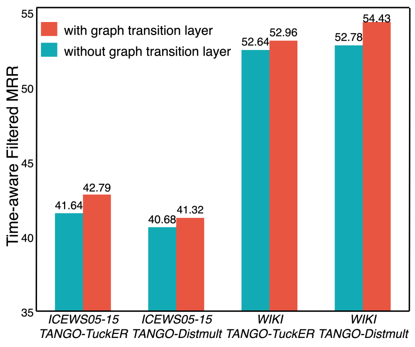


Figure 2: Time-aware filtered MRR of TANGO with or without the graph transition layer on subsets of ICEWS05-15 and WIKI. We split the graph snapshots into two groups, where the transition tensor's norm $||\mathbf{T}(t)||_{L1}$ of each graph snapshot in the first group is larger than that of all graph snapshots in the second group. Since the graph transition layer is tailored to graph changes, we show the results of the first group here. The corresponding result of the ablation study on the whole test sets are presented in Figure 8 in the appendix.

Similarly, TANGO outperforms all the tKG baselines as well. Unlike TTransE and TA-Distmult, RE-Net uses a recurrent neural encoder to capture temporal information, which shows great success on model performance and is the strongest baseline. Our model TANGO implements a NODE-based encoder in the recurrent style to capture temporal dependencies. It consistently outperforms RE-Net on all datasets because TANGO explicitly encodes time information into hidden representations while RE-Net only considers the temporal order between events. Additionally, we provide the raw and time-unaware filtered results in Table 5 and 4 in the appendix.

Datasets	ICEWS05-15 - aware filtered				ICEWS14 - aware filtered				ICEWS18 - aware filtered				WIKI - aware filtered				YAGO - aware filtered			
Model	MRR	Hits@1	Hits@3	Hits@10	MRR	Hits@1	Hits@3	Hits@10	MRR	Hits@1	Hits@3	Hits@10	MRR	Hits@1	Hits@3	Hits@10	MRR	Hits@1	Hits@3	Hits@10
Distmult	24.75	16.10	27.67	42.42	14.49	8.15	15.31	27.66	16.69	9.68	18.12	31.21	49.66	46.17	52.81	54.13	54.84	47.39	59.81	68.52
TuckER	27.13	17.01	29.93	47.81	18.96	11.23	20.77	33.94	20.68	12.58	22.60	37.27	50.01	46.12	53.60	54.86	54.86	47.42	59.63	68.96
CompGCN	29.68	20.72	32.51	47.87	17.81	10.12	19.49	33.11	20.56	12.01	22.96	38.15	49.88	45.78	52.91	55.58	54.35	46.72	59.26	68.29
TTransE	21.24	4.98	31.48	49.88	9.67	1.25	12.29	28.37	8.08	1.84	8.25	21.29	29.27	21.67	34.43	42.39	31.19	18.12	40.91	51.21
TA-DistMult	24.39	14.77	27.80	44.22	10.34	4.72	10.54	21.48	11.38	5.58	12.04	22.82	44.53	39.92	48.73	51.71	54.92	48.15	59.61	66.71
CyGNet	35.79	26.09	40.18	54.48	22.83	14.28	25.36	39.97	24.93	15.90	28.28	42.61	33.89	29.06	36.10	41.86	52.07	45.36	56.12	63.77
DE-SimplE	35.57	26.33	39.41	53.97	21.58	13.77	23.68	37.15	19.30	11.53	21.86	34.80	45.43	42.6	47.71	49.55	54.91	51.64	57.30	60.17
TNTComplEx	35.88	26.92	39.55	53.43	23.81	15.58	26.27	40.12	21.23	13.28	24.02	36.91	45.03	40.04	49.31	52.03	57.98	52.92	61.33	66.69
RE-Net	40.23	30.30	44.83	59.59	25.66	16.69	28.35	43.62	27.90	18.45	31.37	46.37	49.66	46.88	51.19	53.48	58.02	53.06	61.08	66.29
💃-TuckER	**42.86**	**32.72**	**48.14**	**62.34**	**26.25**	**17.30**	**29.07**	**44.18**	**28.97**	**19.51**	**32.61**	**47.51**	51.60	49.61	52.45	54.87	62.50	58.77	64.73	68.63
	$\pm$ 0.2	$\pm$ 0.3	$\pm$ 0.2	$\pm$ 0.2	$\pm$ 0.1	$\pm$ 0.1	$\pm$ 0.1	$\pm$ 0.1	$\pm$ 0.2	$\pm$ 0.1	$\pm$ 0.2	$\pm$ 0.3	$\pm$ 0.3	$\pm$ 0.2	$\pm$ 0.3	$\pm$ 0.3	$\pm$ 0.5	$\pm$ 0.2	$\pm$ 0.1	$\pm$ 0.4
💃-Distmult	40.71	31.23	45.33	58.95	24.70	16.36	27.26	41.35	27.56	18.68	30.86	44.94	**53.04**	**51.52**	**53.84**	**55.46**	**63.34**	**60.04**	**65.19**	**68.79**
	$\pm$ 0.3	$\pm$ 0.4	$\pm$ 0.1	$\pm$ 0.5	$\pm$ 0.1	$\pm$ 0.1	$\pm$ 0.1	$\pm$ 0.1	$\pm$ 0.2	$\pm$ 0.2	$\pm$ 0.2	$\pm$ 0.3	$\pm$ 0.3	$\pm$ 0.4	$\pm$ 0.2	$\pm$ 0.1	$\pm$ 0.4	$\pm$ 0.4	$\pm$ 0.1	$\pm$ 0.2

Table 2: Extrapolated link prediction results on five datasets. Evaluation metrics are time-aware filtered MRR (%) and Hits@1/3/10 (%). 💃 denotes TANGO. The best results are marked in bold.

4.2.2 Ablation Study

To evaluate the effectiveness of our graph transition layer, we conduct an ablation study on two datasets, i.e., ICEWS05-15 and WIKI. We choose these two datasets as the representative of two types of tKG datasets. ICEWS05-15 contains events that last shortly and happen multiple times, i.e., Obama visited Japan. In contrast, the events in the WIKI datasets last much longer and do not occur periodically, i.e., Eliran Danin played for Beitar Jerusalem FC between 2003 and 2010. The improvement of the time-aware filtered MRR brought by the graph transition layer is illustrated in Figure 2, showing that the graph transition layer can effectively boost the model performance by incorporating the edge formation and dissolution information.

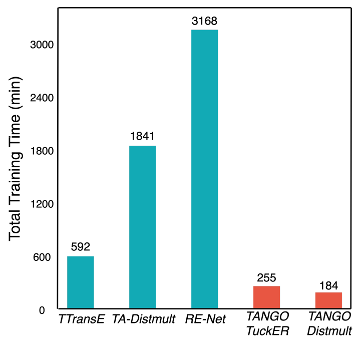


Figure 3: Time cost comparison on ICEWS05-15. Columns marked as orange denote the time consumed by our model.

4.2.3 Time Cost Analysis

Keeping training time short while achieving a strong performance is significant in model evaluation. We report in Figure 3 the total training time of our model and the baselines on ICEWS05-15. We see that static KG reasoning methods generally require less training time than temporal methods. Though the total training time for TTransE is short, its performance is low, as reported in the former sections. TA-Distmult consumes more time than our model and is also beaten by TANGO in performance. RE-Net is the strongest baseline in performance; however, it requires almost ten times as much as the total training time of TANGO. TANGO ensures a short training time while maintaining the state-of-the-art performance for future link prediction, which shows its superiority.

4.3 New Evaluation Tasks

4.3.1 Long Horizontal Link Forecasting

Given a sequence of observed graph snapshots until time t, the future link prediction task infers the quadruples happening at $t + \Delta t$. Δt is usually small, i.e., one day, in standard settings (Trivedi et al., 2017; Jin et al., 2019; Zhu et al., 2020). However, in some scenarios, the graph information right before the query time is likely missing. This arouses the interest in evaluating the temporal KG models by predicting the links in the farther future. In other words, given the same input, the model should predict the links happening at $t+\Delta T$, where $\Delta T >> \Delta t$. Based on this idea, we define a new evaluation task, e.g., long horizontal link forecasting.

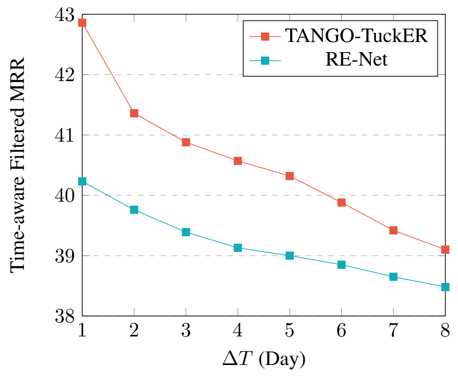


Figure 4: Long horizontal link forecasting: time-aware filtered MRR (%) on ICEWS05-15 with regard to different Δt.

Datasets	ICEWS05-15 - raw				ICEWS05-15 - aware filtered				ICEWS05-15 - unaware filtered			
Model	MRR	Hits@1	Hits@3	Hits@10	MRR	Hits@1	Hits@3	Hits@10	MRR	Hits@1	Hits@3	Hits@10
RE-Net	4.96	2.20	5.39	10.12	5.02	2.29	5.49	10.12	5.50	2.95	5.93	10.26
-TuckER w.o.trans	5.13	2.58	5.67	9.91	5.18	2.64	5.70	9.94	5.98	3.34	6.71	10.67
-Distmult w.o.trans	3.72	2.05	3.80	6.76	3.76	2.09	3.82	6.77	4.09	2.46	4.17	6.99
-TuckER	**5.74**	**3.07**	**6.48**	**10.74**	**5.81**	**3.16**	**6.52**	**10.78**	**6.75**	**4.11**	**7.60**	**11.54**
-Distmult	5.00	2.70	5.67	9.16	5.05	2.78	5.69	9.17	5.69	3.45	6.27	9.69

Table 3: Inductive future link prediction results on ICEWS05-15. Evaluation metrics are raw, time-aware filtered, and time-unaware filtered MRR (%), Hits@1/3/10 (%). *w.o.trans* means without the graph transition layer. The best results are marked in bold.

To perform long horizontal link forecasting, we adjust the integral length according to how far the future we want to predict. As described in Figure 5, the integration length between the neighboring timestamps is short for the first k steps, e.g., integration from $(t - t_k)$ to $(t - t_k + \Delta t)$. However, for the last step, e.g., integration from t to $t + \Delta T$, the integration length becomes significantly large according to how far the future we want to predict. The larger ΔT is, the longer the length is for the last integration step.

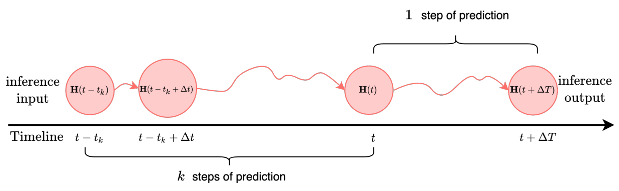


Figure 5: Graphical illustration of long horizontal link forecasting. Given a sequence of graph snapshots $\mathbb{G} = \{\mathcal{G}(t-t_k), ..., \mathcal{G}(t)\}$, whose length is k, test quadruples at $t + \Delta T$ are to be predicted.

We report the results corresponding to different ΔT on ICEWS05-15 and compare our model with the strongest baseline RE-Net. In Figure 4, we observe that our model outperforms RE-Net in long horizontal link forecasting. The gap between the performances of the two models diminishes as ΔT increases. This trend can be explained in the following way. Our model employs an ODE solver to integrate the graph's hidden states over time. Since TANGO takes the time information into account and integrates the ODE in the continuous-time domain, its performance is better than RE-Net, which is a discrete-time model. However, TANGO assumes that the dynamics it learned at t also holds at $t + \Delta T$. This assumption holds when ΔT is small. As ΔT increases, the underlying dynamics at $t+\Delta T$ would be different from the dynamics at t. Thus, the TANGO's performance degrades accordingly, and the advancement compared to RE-Net also vanishes.

4.3.2 Inductive Link Prediction

New graph nodes might emerge as time evolves in many real-world applications, i.e., new users and items. Thus, a good model requires a strong generalization power to deal with unseen nodes. We propose a new task, e.g., inductive link prediction, to validate the model potential in predicting the links regarding *unseen entities* at a future time. A test quadruple is selected for the inductive prediction if either its subject or object or both haven't been observed in the training set. For example, in the test set of ICEWS05-15, we have a quadruple (*Raheel Sharif*, *express intent to meet or negotiate*, *Chaudhry Nisar Ali Khan*, 2014-12-29). The entity *Raheel Sharif* does not appear in the training set, indicating that the aforementioned quadruple contains an entity that the model does not observe in the training set. We call the evaluation of this kind of test quadruples the *inductive link prediction analysis*.

We perform the future link prediction on these inductive link prediction quadruples, and the results are shown in Table 3. We compare our model with the strongest baseline RE-Net on ICEWS05-15. We also report the results achieved by TANGO without the graph transition layer to show the performance boost brought by it. As shown in Table 3, TANGO-TuckER achieves the best results across all metrics. Both TANGO-TuckER and TANGO-Distmult can beat RE-Net, showing the strength of our model in inductive link prediction. The results achieved by the TANGO models are much better than their variants without the graph transition layers, which proves that the proposed graph transition layer plays an essential role in inductive link prediction.

5 Conclusions

We propose a novel representation method, TANGO, for forecasting future links on temporal knowledge graphs (tKGs). We propose a multi-relational graph convolutional layer to capture structural dependencies on tKGs and learn continuous dynamic representations using graph neural ordinary differential equations. Especially, our model is the first one to show that the neural ODE can be extended to modeling dynamic multi-relational graphs. Besides, we couple our model with the graph transition layer to explicitly capture the information provided by the edge formation and deletion. According to the experimental results, TANGO achieves state-of-the-art performance on five benchmark datasets for tKGs. We also propose two new tasks to evaluate the potential of link forecasting models, namely inductive link prediction and long horizontal link forecasting. TANGO performs well in both tasks and shows its great potential.

References

Ivana Balazevic, Carl Allen, and Timothy Hospedales. 2019. TuckER: Tensor factorization for knowledge graph completion. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5185–5194, Hong Kong, China. Association for Computational Linguistics.

Jean-Paul Berrut and Lloyd N Trefethen. 2004. Barycentric lagrange interpolation. *SIAM review*, 46(3):501–517.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Durán, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'13, page 2787–2795, Red Hook, NY, USA. Curran Associates Inc.

Elizabeth Boschee, Jennifer Lautenschlager, Sean O'Brien, Steve Shellman, James Starz, and Michael Ward. 2015. ICEWS Coded Event Data.

Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. 2018. Neural ordinary differential equations. In *Advances in neural information processing systems*, pages 6571–6583.

Rajarshi Das, Shehzaad Dhuliawala, Manzil Zaheer, Luke Vilnis, Ishan Durugkar, Akshay Krishnamurthy, Alex Smola, and Andrew McCallum. 2018. Go for a walk and arrive at the answer: Reasoning over paths in knowledge bases using reinforcement learning.

Talgat Daulbaev, Alexandr Katrutsa, Larisa Markeeva, Julia Gusak, Andrzej Cichocki, and Ivan Oseledets. 2020. Interpolated adjoint method for neural odes. *arXiv preprint arXiv:2003.05271*.

Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. 2016. Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in neural information processing systems*, pages 3844–3852.

Songgaojun Deng, Huzefa Rangwala, and Yue Ning. 2020. Dynamic knowledge graph based multi-event forecasting. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1585–1595.

Alberto García-Durán, Sebastijan Dumančić, and Mathias Niepert. 2018. Learning sequence encoders for temporal knowledge graph completion. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4816–4821, Brussels, Belgium. Association for Computational Linguistics.

Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. 2017. Neural message passing for quantum chemistry. *arXiv preprint arXiv:1704.01212*.

Rishab Goel, Seyed Mehran Kazemi, Marcus Brubaker, and Pascal Poupart. 2020. Diachronic embedding for temporal knowledge graph completion. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 3988–3995.

Zhen Han, Peng Chen, Yunpu Ma, and Volker Tresp. 2020a. Dyernie: Dynamic evolution of riemannian manifold embeddings for temporal knowledge graph completion. *arXiv preprint arXiv:2011.03984*.

Zhen Han, Peng Chen, Yunpu Ma, and Volker Tresp. 2021. xerte: Explainable reasoning on temporal knowledge graphs for forecasting future links.

Zhen Han, Yuyi Wang, Yunpu Ma, Stephan Guünnemann, and Volker Tresp. 2020b. The graph hawkes network for reasoning on temporal knowledge graphs. *arXiv preprint arXiv:2003.13432*.

Woojeong Jin, He Jiang, Meng Qu, Tong Chen, Changlin Zhang, Pedro Szekely, and Xiang Ren. 2019. Recurrent event network: Global structure inference over temporal knowledge graph. *arXiv preprint arXiv:1904.05530*.

Diederik P. Kingma and Jimmy Ba. 2017. Adam: A method for stochastic optimization.

Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*.

Timothee Lacroix, Guillaume Obozinski, and Nicolas Usunier. 2020. Tensor decompositions for temporal knowledge base completion. *ICLR preprint https://openreview.net/pdf?id=rke2P1BFwS*.

Julien Leblay and Melisachew Wudage Chekol. 2018. Deriving validity time in knowledge graph. In *Companion Proceedings of the The Web Conference 2018*, pages 1771–1776.

Jianan Li, Xuemei Xie, Zhifu Zhao, Yuhan Cao, Qingzhe Pan, and Guangming Shi. 2020. Temporal graph modeling for skeleton-based action recognition. *arXiv preprint arXiv:2012.08804*.

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

Mathias Niepert, Mohamed Ahmed, and Konstantin Kutzkov. 2016. Learning convolutional neural networks for graphs. In *International conference on machine learning*, pages 2014–2023.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. Pytorch: An imperative style, high-performance deep learning library. In *Advances in neural information processing systems*, pages 8026–8037.

Hongyu Ren and Jure Leskovec. 2020. Beta embeddings for multi-hop logical reasoning in knowledge graphs.

Michael Schlichtkrull, Thomas N. Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. 2017. Modeling relational data with graph convolutional networks.

Rakshit Trivedi, Hanjun Dai, Yichen Wang, and Le Song. 2017. Know-evolve: Deep temporal reasoning for dynamic knowledge graphs.

L. R. Tucker. 1964. The extension of factor analysis to three-dimensional matrices.

Eugene E Tyrtyshnikov. 2012. *A brief introduction to numerical analysis*. Springer Science & Business Media.

Shikhar Vashishth, Soumya Sanyal, Vikram Nitin, and Partha Talukdar. 2019. Composition-based multi-relational graph convolutional networks. *arXiv preprint arXiv:1911.03082*.

Sijie Yan, Yuanjun Xiong, and Dahua Lin. 2018. Spatial temporal graph convolutional networks for skeleton-based action recognition. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32.

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

Cunchao Zhu, Muhao Chen, Changjun Fan, Guangquan Cheng, and Yan Zhan. 2020. Learning from history: Modeling temporal knowledge graphs with sequential copy-generation networks. *arXiv preprint arXiv:2012.08492*.

Appendix

A Representation Inference

Assume we want to forecast a link at t. We take the graph histories between the timestamp $(t - t_k)$ and the timestamp t into account, where t_k indicates the length of history. To infer the hidden representations $\mathbf{H}(t)$, we first use the initial embeddings $\text{Emb}(\mathcal{V}, \mathcal{R})$ to approximate the hidden representations $\mathbf{H}(t - t_k)$. Then we take $\mathbf{H}(t - t_k)$ as the NODE input at the timestamp $(t - t_k)$, and integrate it with an ODE solver $\text{ODESolver}(\mathbf{H}(t - t_k), f_{\text{TANGO}}, t - t_k, t, \Theta_{\text{TANGO}}, \mathcal{G})$ over time. As the hidden state evolves with time, it learns from different graph observations taken at different time. The whole process is described in Figure 6 and Algorithm 1. In Figure 6, set_graph and set_transition stand for two functions used to feed graph snapshots and the transition tensors into the neural network f_{TANGO}. They are called at every observation time before integration.

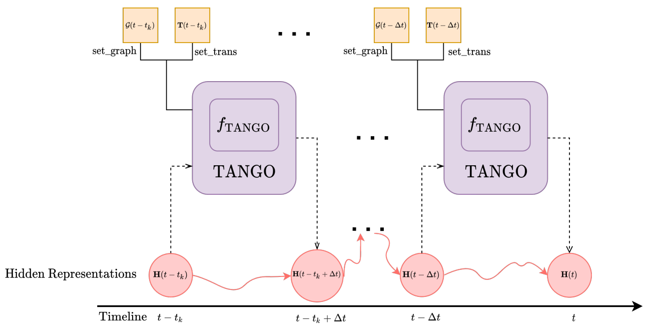


Figure 6: Illustration of the inference procedure. The shaded purple area represents the whole architecture of TANGO. It is a Neural ODE equipped with a GNN-based module f_{TANGO}. Dashed arrows denote the input and the output path of the graph's hidden state. Red solid arrows indicate the continuous hidden state flows learned by TANGO. Black solid lines represent that TANGO calls the function set_graph and set_trans. The corresponding graph snapshots $\mathcal{G}$ and transition tensors $\mathbf{T}$ are input into f_{TANGO} for learning temporal dynamics.

B Evaluation Metrics

We report the results in three settings, namely raw, time-unaware filtered, and time-aware filtered. For time-unaware filtered results, we follow the filtered evaluation constraint applied in (Bordes et al., 2013; Jin et al., 2019), where we remove from the list of corrupted triplets all the triplets that appear either in the training, validation, or test set ex-

Datasets	ICEWS05-15 - raw				ICEWS14 - raw				ICEWS18 - raw				WIKI - raw				YAGO - raw			
Model	MRR	Hits@1	Hits@3	Hits@10	MRR	Hits@1	Hits@3	Hits@10	MRR	HITS@1	HITS@3	HITS@10	MRR	Hits@1	Hits@3	Hits@10	MRR	Hits@1	Hits@3	Hits@10
Distmult	24.55	15.85	27.53	42.17	14.00	7.72	14.65	27.16	16.30	9.25	17.67	30.93	42.08	34.29	48.69	53.25	47.66	36.59	55.89	67.45
TuckER	26.95	16.81	29.69	47.61	18.39	10.69	20.01	33.42	20.20	12.08	21.99	36.91	**42.50**	**34.41**	**49.41**	53.90	47.48	36.20	55.55	68.07
COMPGCN	29.41	20.41	32.17	47.65	17.13	9.36	18.84	32.54	19.98	11.45	22.25	37.73	42.33	34.02	48.65	**54.63**	47.08	65.36	66.90	68.81
TTransE	20.89	4.88	3.11	49.66	9.21	1.12	11.19	27.46	7.92	1.75	8.00	21.02	19.53	12.34	23.11	32.47	26.18	12.36	36.16	48.00
TA-DistMult	24.03	14.37	27.36	44.04	9.92	4.39	9.99	20.90	11.05	5.24	11.72	22.55	27.33	19.94	32.05	39.42	45.54	36.54	51.08	62.15
RE-Net	39.31	28.88	44.40	59.38	23.84	14.60	26.48	42.58	26.62	16.91	30.26	45.82	31.10	25.31	34.13	41.33	46.28	37.52	51.77	61.55
💃-TuckER	**41.82**	**31.10**	**47.55**	**62.19**	**24.36**	**15.12**	**27.15**	**43.07**	**27.59**	**17.77**	**31.40**	**46.92**	31.99	25.74	35.00	42.61	49.31	40.78	55.12	63.73
💃-Distmult	40.23	30.53	44.95	59.05	22.87	14.22	25.43	40.32	26.21	16.92	29.77	44.41	32.53	26.33	35.75	43.17	**49.49**	**40.90**	**55.42**	**63.74**

Table 4: Future link prediction results on benchmark datasets. Evaluation metrics are raw MRR (%) and Hits@1/3/10 (%). 💃 denotes TANGO. The best results are marked in bold.

Datasets	ICEWS05-15 - unaware filtered				ICEWS14 - unaware filtered				ICEWS18 - unaware filtered				WIKI - unaware filtered				YAGO - unaware filtered			
Model	MRR	Hits@1	Hits@3	Hits@10	MRR	Hits@1	Hits@3	Hits@10	MRR	Hits@1	Hits@3	Hits@10	MRR	Hits@1	Hits@3	Hits@10	MRR	Hits@1	Hits@3	Hits@10
Distmult	48.77	43.85	51.22	57.99	33.88	27.86	36.16	45.14	40.28	36.04	41.78	48.36	53.22	52.61	53.41	54.20	67.55	66.76	67.49	69.11
TuckER	58.69	54.74	59.82	66.57	46.51	41.11	49.45	57.34	44.50	38.33	46.11	53.71	53.97	**52.70**	**54.15**	54.94	67.40	66.22	67.62	69.84
COMPGCN	49.60	43.13	52.85	61.59	38.15	31.04	41.00	51.44	35.68	27.87	39.38	49.94	53.54	52.29	53.61	55.76	66.66	65.36	66.90	68.81
TTransE	28.81	5.83	48.67	60.38	15.95	1.57	25.98	42.67	10.52	3.01	11.98	26.16	31.94	24.82	36.91	43.55	33.73	20.99	43.51	52.61
TA-DistMult	38.54	29.94	42.92	54.81	18.74	11.97	20.32	31.95	16.27	10.22	17.39	27.91	50.18	48.65	51.41	52.37	66.06	64.36	66.78	68.74
RE-Net	57.66	51.86	60.40	68.60	45.24	37.82	48.53	58.92	43.02	36.26	45.61	56.03	52.27	50.92	52.73	53.57	64.68	62.94	65.11	67.82
💃-TuckER	**59.93**	**54.99**	**62.65**	**69.64**	46.42	38.94	**50.25**	**59.80**	**44.56**	37.87	**47.46**	**57.06**	53.28	52.21	53.61	54.84	67.21	65.56	67.59	70.04
💃-Distmult	58.89	54.42	60.76	67.47	**46.68**	**41.20**	48.64	57.05	44.00	**38.64**	45.78	54.27	**54.05**	51.52	53.84	**55.46**	**68.34**	**67.05**	**68.39**	**70.70**

Table 5: Future link prediction results on benchmark datasets. Evaluation metrics are time-unaware filtered MRR (%) and Hits@1/3/10 (%). 💃 denotes TANGO. The best results are marked in bold.

cept the triplet of interest. Time-unaware filtering setting is inappropriate for temporal KG reasoning, while the time-aware filtering setting provides fairer results. For time-aware filtered results, we follow the setting proposed by (Han et al., 2021) by only removing from the list of corrupted triplets all the triplets that appear at the query time t_q. The following example illustrates the reason why the time-aware filtered results are fairer than the time-unaware filtered results. Assume we have a test quadruple of interest (*Xi Jinping*, *make a visit*, *New Zealand*, 2014-11-26) in the test set, and we derive an object prediction query (*Xi Jinping*, *make a visit*, ?, 2014-11-26) from this quadruple where the query time is 2014-11-26. Additionally, we have another quadruple (*Xi Jinping*, *make a visit*, *South Korea*, 2014-07-05) in the test set. According to the time-unaware filtering setting (Bordes et al., 2013), (*Xi Jinping*, *make a visit*, *South Korea*) will be filtered out since it appears in the test set. However, it is unreasonable because (*Xi Jinping*, *make a visit*, *South Korea*) is not valid at 2014-11-26. Therefore, we use the time-aware filtered setting, which, in our example, will only filter the triplets (*Xi Jinping*, *make a visit*, o) appearing at 2014-11-26. Here, o denotes all the objects from triplets accompanied with *Xi Jinping*, *Make a visit*, and the date 2014-11-26.

C Implementation Details

We train TANGO with the following settings. We tune the model across a range of hyperparameters as shown in Table 7. We do 432 trials, and each trial runs 20 epochs. We select the best-performing configuration according to filtered MRR on validation data. The best configuration will be further trained until its convergence. We run the selected configuration five times and obtain an averaged results. Specifically, we use a fixed-grid ODE solver, fourth-order Runge-Kutta, as the ODE solver, and implement the interpolated reverse dynamic method (Daulbaev et al., 2020) with 3 Chebyshev nodes to keep training time tractable while maintaining high precision. To improve the ODE solver's precision, we re-scale the time range of each dataset from 0 to 0.01 (or 0.1). This step restricts the length of ODE integration, preventing the high error induced by ODE solvers. For each query, we set the time range of the input history t_k to 4 days for the ICEWS datasets. For WIKI and YAGO, we set t_k to 4 years. Besides, we choose different values for the transition coefficient w for different datasets. Our model is implemented with PyTorch (Paszke et al., 2019), and the experiments are run on GeForce RTX 2080 Ti. A detailed report of the best configuration is provided in Table 8.

We implement Distmult in PyTorch and use the binary cross-entropy loss for learning parameters. We use the official implementation of TuckER[2], COMPGCN[3], and RE-Net[4]. For a fair comparison, we choose to use the variant of RE-Net with ground truth history during multi-step inference, and thus the model knows all the interactions before the time for testing. Besides, we set the history length of RE-Net to 10 and use the max-pooling in the global

[2]https://github.com/ibalazevic/TuckER
[3]https://github.com/malllabiisc/CompGCN
[4]https://github.com/INK-USC/RE-Net

Datasets	ICEWS05-15 - aware filtered				ICEWS18 - aware filtered				WIKI - aware filtered				YAGO - aware filtered			
Model	MRR	Hits@1	Hits@3	Hits@10	MRR	Hits@1	Hits@3	Hits@10	MRR	Hits@1	Hits@3	Hits@10	MRR	Hits@1	Hits@3	Hits@10
💃-TuckER	44.57	34.40	49.94	63.95	30.68	20.75	34.61	50.43	62.29	59.54	63.92	66.63	69.29	64.33	72.40	77.63
💃-Distmult	43.33	33.46	48.45	62.05	29.62	20.18	33.35	48.36	63.93	62.14	64.74	67.06	70.79	66.15	74.04	78.18

Table 6: Validation results on benchmark datasets regarding our model. Evaluation metrics are time-aware filtered MRR (%) and Hits@1/3/10 (%). 💃 denotes TANGO. The best results are marked in bold. ICEWS14 has no validation set.

model. Additionally, we use the implementation of TTransE and TA-Distmult provided in (Jin et al., 2019). For TA-Distmult, the vocabulary of temporal tokens consists of year, month, and day for all the datasets. We use the released code to implement DE-SimplE[5], TNTComplEx[6], and CyGNet[7]. All the baselines are trained with Adam Optimizer (Kingma and Ba, 2017), and the batch size is set to 512.

Table 7: Search space of hyperparameters. w represents the weight controlling how much the model learns from edge formation and dissolution. *Scale* represents the time range re-scaling parameter as introduced in C.

Hyperparameter	Search space
Embedding size	{200, 300}
# MGCN layer	{2, 3}
Decoder	{TuckER, Distmult}
Scale	{0.001, 0.01, 0.1}
w	{0.01, 0.1, 1}
Dropout	{0.3, 0.5}
History length	{4, 6, 10}

Table 8: Best hyperparameter settings on each dataset.

Datasets	ICEWS14	ICEWS18	ICEWS05-15	WIKI	YAGO
Hyperparameter					
Embedding size	200	200	200	200	300
# MGCN layer	2	2	2	2	3
Decoder	TuckER	TuckER	TuckER	Distmult	Distmult
Scale	0.01	0.1	0.1	0.1	0.1
w	0.01	1	0.01	1	1
Dropout	0.3	0.3	0.3	0.3	0.3
History length	4	4	4	4	4

D Datasets

Table 9 We follow the data preprocessing method and the dataset split strategy proposed in (Jin et al., 2019). Specifically, we split each dataset except ICEWS14 in chronological order into three parts, e.g., $80\%/10\%/10\%$ (training/validation/test). For ICEWS14, we split it into the training set and testing set with $50\%/50\%$ since ICEWS14 is not provided with a validation set. As explained in (Jin et al., 2019), the difference between the first type (ICEWS) and the second type (WIKI and YAGO) of tKG datasets is that the first type datasets are events that often last shortly and happen multiple times, i.e., Obama visited Japan four times. In contrast, the events in the second type datasets last much longer and do not occur periodically, i.e., Eliran Danin played for Beitar Jerusalem FC between 2003 and 2010.

Dataset	N_{train}	N_{valid}	N_{test}	$\lvert\mathcal{V}\rvert$	$\lvert\mathcal{R}\rvert$	N_{obs}
ICEWS14 (Trivedi et al., 2017)	$323,895$	–	$341,409$	$12,498$	260	365
ICEWS18 (Boschee et al., 2015)	$373,018$	$45,995$	$49,545$	$23,033$	256	304
ICEWS05-15 (García-Durán et al., 2018)	$369,104$	$46,188$	$46,037$	$10,488$	251	$4,017$
WIKI (Leblay and Chekol, 2018)	$539,286$	$67,538$	$63,110$	$12,554$	24	232
YAGO (Mahdisoltani et al., 2013)	$161,540$	$19,523$	$20,026$	$10,623$	10	189

Table 9: Dataset statistics. N_{train}, N_{valid}, N_{test} represent the number of quadruples in the training set, validation set, and test set, respectively. N_{obs} denotes the number of observations, where we take a snapshot of the tKG at each observation.

E Impact of Past History Length

As mentioned in A, TANGO utilizes the previous histories between $(t - t_k)$ and t to forecast a link at t, where t_k is a hyperparameter. Figure 7 shows the performance with various lengths of past histories along with the corresponding training time. When TANGO uses longer histories, MRR is getting higher. However, a long history requires more forwarding inferences. The choice of history length is a trade-off between the performance and computational cost. We observe that the gain of MRR compared to the training time is not significant when the length of history is four and over. Thus, the history length of four is chosen in our experiments.

F Analysis on Temporal KGs with Irregular Time Intervals

Most existing tKG reasoning models cannot properly deal with temporal KGs with irregular time intervals, while TANGO model them much better due to the nature of Neural ODE. We verify this via experiments on a new dataset. We call it

[5]https://github.com/BorealisAI/de-simple
[6]https://github.com/facebookresearch/tkbc
[7]https://github.com/CunchaoZ/CyGNet

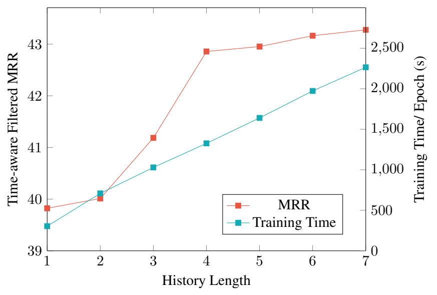


Figure 7: Time-aware filtered MRR (%) and Training Time (seconds) on ICEWS05-15 corresponding to different history length (days).

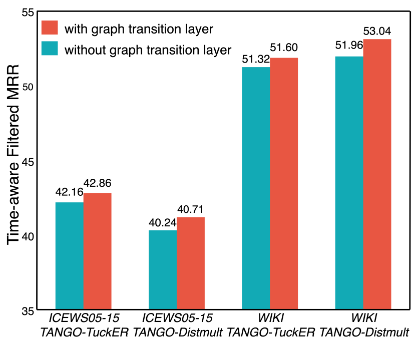


Figure 8: Time-aware filtered MRR of TANGO with or without the graph transition layer on the whole test sets of ICEWS05-15 and WIKI.

ICEWS05-15_continuous. We sample the timestamps in ICEWS05-15 and keep the time intervals between each two of them in a range from 1 to 4. We only keep the temporal KG snapshots at the sampled time and extract a new subset. ICEWS05-15_continuous fits the setting when observations are taken non-periodically in continuous time. The dataset statistics of ICEWS05-15_continuous is reported in Table 11. We train our model and baseline methods on it and evaluate them with time-aware filtered MRR. As shown in Table 10, we validate that TANGO performs well on temporal KGs with irregular time intervals.

G Average runtime for each approach

Table 12 show the average runtime for each model.

Datasets	ICEWS05-15 continuous - aware filtered				ICEWS05-15 - aware filtered			
Model	MRR	Hits@1	Hits@3	Hits@10	MRR	Hits@1	Hits@3	Hits@10
TTransE	20.55	5.36	29.80	47.54	21.24	4.98	31.48	49.88
CyGNet	34.13	25.06	37.85	51.94	35.79	26.09	40.18	54.48
DE-SimplE	33.56	24.79	37.32	50.63	35.57	26.33	39.41	53.97
TNTComplEx	33.96	24.93	37.86	51.30	35.88	26.92	39.55	53.43
💃-TuckER	**37.69**	**28.01**	**45.00**	**59.05**	**42.86**	**32.72**	**48.14**	**62.34**
💃-Distmult	36.91	26.91	40.28	54.34	40.71	31.23	45.33	58.95

Table 10: Future link prediction results on ICEWS05-15 continuous dataset. Evaluation metrics are time-aware filtered MRR (%) and Hits@1/3/10 (%). 💃 denotes TANGO. The best results are marked in bold.

Dataset	N_{train}	N_{valid}	N_{test}	$\lvert\mathcal{V}\rvert$	$\lvert\mathcal{R}\rvert$	N_{obs}
ICEWS05-15 continuous	149,001	17,962	17,902	10,488	251	1,589

Table 11: Dataset statistics. N_{train}, N_{valid}, N_{test} represent the number of quadruples in the training set, validation set, and test set, respectively. N_{obs} denotes the number of observations, where we take a snapshot of the tKG at each observation.

Table 12: Average training time (second) until convergence

Datasets	ICEWS14	ICEWS18	ICEWS05-15	WIKI	YAGO
Model	Runtime	Runtime	Runtime	Runtime	Runtime
Distmult	743	1,365	401	2,245	3,310
TuckER	730	3,147	1,626	5,093	2,795
COMPGCN	9,226	6,432	1,607	5,810	2,233
TTransE	15,840	23,894	35,520	19,337	5,395
TA-Distmult	6,232	112,188	110,460	83,999	27,833
RE-Net	33,313	46,068	190,076	42,983	27,489
💃-TuckER	5,796	3,786	15,301	9,218	2,355
💃-Distmult	3,593	2,883	11,085	15,086	5,106

Chapter 5

Time-dependent Entity Embedding is not All You Need: A Re-Evaluation of Temporal Knowledge Graph Completion Models under a Unified Framework

This chapter contains the publication

Time-dependent Entity Embedding is not All You Need: A Re-Evaluation of Temporal Knowledge Graph Completion Models under a Unified Framework. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Online and in Dominican Republic, Nov. 2021. DOI: 10.18653/v1/2021c.emnlp-main.639.

Time-dependent Entity Embedding is not All You Need: A Re-evaluation of Temporal Knowledge Graph Completion Models under a Unified Framework

Zhen Han[*1,2], **Gengyuan Zhang**[*1], **Yunpu Ma**[†1], **Volker Tresp**[†1,2]
[1]Institute of Informatics, LMU Munich [2] Corporate Technology, Siemens AG
zhen.han@campus.lmu.de, gengyuanmax@gmail.com
cognitive.yunpu@gmail.com, volker.tresp@siemens.com

Abstract

Various temporal knowledge graph (KG) completion models have been proposed in the recent literature. The models usually contain two parts, a temporal embedding layer and a score function derived from existing static KG modeling approaches. Since the approaches differ along several dimensions, including different score functions and training strategies, the individual contributions of different temporal embedding techniques to model performance are not always clear. In this work, we systematically study six temporal embedding approaches and empirically quantify their performance across a wide range of configurations with about 4000 experiments and 19000 GPU hours. We classify the temporal embeddings into two classes: (1) *timestamp embeddings* and (2) *time-dependent entity embeddings*. Despite the common belief that the latter is more expressive, an extensive experimental study shows that timestamp embeddings can achieve on-par or even better performance with significantly fewer parameters. Moreover, we find that when trained appropriately, the relative performance differences between various temporal embeddings often shrink and sometimes even reverse when compared to prior results. For example, TTransE (Leblay and Chekol, 2018), one of the first temporal KG models, can outperform more recent architectures on ICEWS datasets. To foster further research, we provide the first unified open-source framework for temporal KG completion models with full composability, where temporal embeddings, score functions, loss functions, regularizers, and the explicit modeling of reciprocal relations can be combined arbitrarily.

1 Introduction

The Knowledge Graph (KG), a graph-structured knowledge base, has gained increasing interest as a promising way to store factual knowledge. KGs represent facts in the form of triples (s, r, o), e.g., (*Bob*, *livesIn*, *New York*), in which s (subject) and o (object) denote nodes (entities) and r denotes the edge type (relation) between s and o. Knowledge graphs are commonly static and store facts in their current state. In reality, however, the relations between entities often change over time. For example, if Bob moves to California, the triple of (*Bob*, *livesIn*, *New York*) will be invalid. To this end, temporal knowledge graphs (tKGs) have been introduced to capture temporal aspects of facts in addition to their multi-relational nature. A tKG represents a temporal fact as a quadruple (s, r, o, t) by extending a static triple with time t, describing that this fact is valid at time t. Figure 2 in the appendix depicts an exemplary temporal KG. To address the inherent incompleteness of temporal KGs, Tresp et al. (2015) proposed the first tKG model. Afterwards, a line of work emerged that extends static KG completion models by adding temporal embeddings, e.g., TTransE (Leblay and Chekol, 2018), TA-TransE (García-Durán et al., 2018), DE-SimplE (Goel et al., 2019), TNTComplEx (Lacroix et al., 2020), ConT (Ma et al., 2018), and many more. The models generally consist of two parts, a temporal embedding layer to capture the evolving features of tKGs and a score function to examine the plausibility of a given quadruple.

Temporal embeddings are crucial in temporal KG completion models for storing the evolving knowledge; without them, the temporal aspect cannot be captured. The PEs can be generally categorized into three classes: (1) ***t**imestamp **e**mbeddings (TEs)*: the models learn an embedding for each discrete timestamp in the same vector space as entities and relations (Tresp et al., 2017; Leblay and Chekol, 2018; Dasgupta et al., 2018; Lacroix et al., 2020). (2) ***t**ime-dependent **e**ntity **e**mbeddings (TEEs)*: the models define entity embedding as a function that takes an entity and a timestamp as

[*] Equal contribution.
[†] Corresponding author.

input and generates a time-dependent representation for the entity at that time (Goel et al., 2019; Xu et al., 2019; Han et al., 2020a). (3) *deep representation learning (DTRs)*: the models incorporate temporal information into advanced deep learning models, e.g., Recurrent Neural Network and Graph Neural Network, to learn time-aware representations of entities and relations (García-Durán et al., 2018). In many cases, the introduction of new temporal embedding approaches went along with new score functions and new training methods (regularization, the explicit modeling of reciprocal relations, etc.). Ablation studies were provided, but not investigated thoroughly. Besides, some temporal embedding papers introduced new datasets. They commonly tune model architecture and hyperparameters of old temporal embedding approaches on new datasets using grid search on a small grid involving hand-crafted parameter ranges or settings known to work well from prior studies. A grid suitable for one dataset may be suboptimal for another, however. It is often difficult to attribute the incremental improvements in performance reported with each new state-of-the-art (SOTA) model to the proposed temporal embeddings or other components.

In this work, we investigate the significance of previously reported temporal embeddings with several thousands of experiments and 19000 GPU hours. First, we aim to study which temporal embedding approach can generally outperform other temporal embedding approaches regardless of different score functions and different datasets. We choose one representative from bilinear score functions, i.e., SimplE (Kazemi and Poole, 2018), and one from translation-based score functions, i.e., TransE (Bordes et al., 2013). Then we benchmark six temporal embedding approaches on two subsets of ICEWS (Boschee et al., 2015) and a subset of GDELT(Leetaru and Schrodt, 2013) with the two representative score functions through an extensive set of experiments. Second, we performed an extensive benchmark study on well-known temporal KG completion models using popular model architectures and training strategies in a unified experimental setup. Following the work (Ruffinelli et al., 2020), we considered many training strategies as well as a large hyperparameter space, and we performed model selection using a quasi-random search followed by Bayesian optimization, which has been shown to be able to find good model configurations with relatively low effort.

Regarding the first aim, we surprisingly find that the TE proposed by Leblay and Chekol (2018) outperforms other temporal embedding approaches on the ICEWS subsets and achieves on-par results on GDELT. Leblay and Chekol (2018) represent timestamps in the same vector space as entities and relations and learn embeddings for each discrete timestamp. While achieving better results, the TE models only require about half of the model parameters as much of TEEs. However, the common belief is that the TEEs are more expressive and can better capture the evolving knowledge. Recall that models with TEEs learn an embedding function for each entity that takes time as input and provides an entity representation as output. In particular, it has been proven that TEEs are fully expressive for tKG completion in combination with certain score functions (Goel et al., 2019), and thus, they should perform better than TEs, which is in contrast to our findings. We argue that the sparsity of temporal KG data may cause the undesirable empirical performance of TEEs. Every entity has the same dimensionality of time-dependent embeddings, but the majority of entities are only involved in a small number of quadruples. As a result, the TEEs may suffer from the overfitting problem. To verify our assumption, we learn a **u**nique **t**emporal **e**mbedding function for all **e**ntities instead of learning entity-specific embedding functions. We refer to it as UTEE. Empirical study shows that the UTEE achieves similar or even better results than all other TEEs variants, emphasizing the overfitting problem of TEEs.

Besides, we empirically find that the performance of a fine-tuned baseline can by far exceed the performance observed in all previous studies. For example, T-TransE (Leblay and Chekol, 2018), one of the first temporal KG completion models, achieves superior performance metric in our study that is more than **doubled** to that reported in recent papers (García-Durán et al., 2018; Goel et al., 2019; Lacroix et al., 2020; Xu et al., 2019). Thus, it is competitive to or even outperforms current SOTA models such as DE-SimplE (Goel et al., 2019) and TComplEx (Lacroix et al., 2020). This suggests that training strategies significantly affect the performance of temporal KG models and are responsible for a substantial fraction of the progress made in recent years. Thus, to fairly compare the effectiveness of different temporal KG models, it is necessary to evaluate them on a unified framework. To this end, our study realizes the first fair benchmarking by investigating the interplay between temporal KG interaction models, loss functions, regularization methods, the use of reciprocal relations, and other training techniques in a unified open-source framework[1]. To ensure the composability of the framework, the temporal embedding layer, score functions, and various training strategies are implemented as independent submodules. Thus, one can easily assess the individual benefit of a novel temporal embedding approach via our framework. Additionally, we perform an extensive experimental study in which well-known temporal KG models are fine-tuned by popular training strategies and a wide range of hyperparameter settings. The reported results can be directly used for comparison in future work.

2 Preliminaries and Related Work

2.1 Temporal Knowledge Graph Completion

Temporal knowledge graphs (tKGs) are multi-relational, directed graphs with labeled timestamped edges between entities. Let $\mathcal{E}$, $\mathcal{R}$, and $\mathcal{T}$ represent a finite set of entities, relations, and timestamps, respectively. Each fact can be denoted by a quadruple $q = (e_s, r, e_o, t)$, representing a timestamped and labeled edge between a subject entity $e_s \in \mathcal{E}$ and an object entity $e_o \in \mathcal{E}$ regarding a relation $r \in \mathcal{R}$ at a timestamp $t \in \mathcal{T}$. Let $\mathcal{F}$ represents the set of all quadruples that are facts, i.e., real events in the world, the tKG completion (tKGC) is the problem of inferring $\mathcal{F}$ based on a set of observed facts $\mathcal{O}$, which is a subset of $\mathcal{F}$. Specifically, the task of tKGC is to predict either a missing subject entity $(?, r, e_o, t)$ given the other three components or a missing object entity $(e_s, r, ?, t)$.

Our study focuses solely on temporal knowledge graph embedding models for the completion task, which do not exploit temporal knowledge graph embedding models for the forecasting task (Trivedi et al., 2017; Han et al., 2020b, 2021).

2.2 Temporal KG Embedding Models

A tKG embedding (tKGE) model embeds each entity $e \in \mathcal{E}$ and relation $r \in \mathcal{R}$ in a vector space. To capture temporal aspects, each model either embeds discrete timestamps into a vector space or learns time-dependent representations for each entity. Besides, each model has a score function that takes the temporal information and the embeddings of the subject, relation, and object as the input and computes a score for each potential quadruple. The higher the quadruple score, the more plausible it is considered to be true by the model. Taking the object prediction as an example, we consider all entities in $\mathcal{E}$ and learn a score function $\phi(e_s, r, e_o, t) = f(\mathbf{e}_s(t), \mathbf{r}, \mathbf{e}_o(t))$, for models with TEEs and $\phi(e_s, r, e_o, t) = f(\mathbf{e}_s, \mathbf{r}, \mathbf{e}_o, \mathbf{t})$ for models with TEs. The bold symbols denote the embeddings of the corresponding entities, relation, and time.

2.2.1 Temporal Embeddings

tKGE models differ in their temporal embeddings and score functions. Temporal embedding approaches come in three categories: timestamp embeddings (TEs), where the models learn a representation for each discrete timestamp; time-dependent entity embeddings (TEEs), where an entity embedding function takes time and an entity as inputs and provides a hidden representation as output; and deep temporal representations (DTRs), where the models incorporate time information into deep learning frameworks.

The best known TE is the vanilla TE (abbreviated to T by its authors) proposed by Leblay and Chekol (2018) where each timestamp is mapped in the same vector space as entities and relations. Later, Lacroix et al. (2020) introduced a new regularization scheme to smooth the representation of neighboring timestamps. Another well-known TE is HyTE (Dasgupta et al., 2018), which associates each timestamp with a corresponding hyperplane and projects the embeddings of entities and relations onto timestamp-specific hyperplanes to incorporate temporal information in entity embeddings:

$$\mathbf{e}_i(t) = \mathbf{e}_i \perp \boldsymbol{\omega}_t = \mathbf{e}_i - (\boldsymbol{\omega}_t^T \mathbf{e}_i)\boldsymbol{\omega}_t.$$

$\mathbf{e}_i$ represents the global embedding of entity e_i, $\perp$ represents the projection operator, and $\boldsymbol{\omega}_t$ represents the normal vector of the hyperplane associated with timestamp t.

A well-known variants of TEEs is the diachronic entity embeddings (DE) proposed by Goel et al. (2019) that defines the temporal embeddings of entity e_i at timestamp t as

$$\mathbf{e}_i^{DE}(t)[n] = \begin{cases} \mathbf{a}_{e_i}[n] & \text{if } 1 \leq n \leq \gamma d, \\ \mathbf{a}_{e_i}[n] \sin(\boldsymbol{\omega}_{e_i}[n]t + \mathbf{b}_{e_i}[n]) & \text{else.} \end{cases} \quad (1)$$

[1]https://github.com/TemporalKGTeam/A_Unified_Framework_of_Temporal_Knowledge_Graph_Models

$\mathbf{e}_i^{DE}(t)[n]$ denotes the n^{th} element of the embeddings of entity e_i at time t. $\mathbf{a}_{e_i}, \boldsymbol{\omega}_{e_i}, \mathbf{b}_{e_i}$ are entity-specific vectors with learnable parameters. The first γd elements of the vector in Equation 1 capture static features, and the other $(1-\gamma)d$ elements capture temporal features. ATiSE (Xu et al., 2019) is another popular TEE that adds time information into entity/relation representations by using additive time series decomposition, where the entity representation is defined as

$$\begin{aligned}\mathbf{e}_i^{ATiSE}(t) &= \mathbf{e}_i + \alpha_{e_i}\mathbf{w}_{e_i}t \\ &+ \boldsymbol{\beta}_{e_i}\sin(2\pi\boldsymbol{\omega}_{e_i}t) + \mathcal{N}(0, \boldsymbol{\Sigma}_{e_i}).\end{aligned} \tag{2}$$

The term $\mathbf{e}_i + \alpha_{e_i}\boldsymbol{\omega}_{e_i}t$ is the trend component where the coefficient denotes the evolutionary rates, and the vector ω_{e_i} represents the corresponding evolutionary direction. $\boldsymbol{\beta}_{e_i}\sin(2\pi\boldsymbol{\omega}_{e_i}t)$ is the corresponding seasonal component, and the Gaussian noise term $\mathcal{N}(0, \Sigma_{e_i})$ denotes the random component. In principle, other temporal embedding approaches can also be converted into a probabilistic approach by adding Gaussian noise. Thus, to fairly compare with other temporal embeddings and simplify our study, we do not take the noise term into account. The representation of relations in ATiSE is also time-dependent and defined similarly to the entity representation.

A representative of DTRs is the TA-approach (García-Durán et al., 2018) that utilizes recurrent neural networks to learn time-aware representations of relations. Specifically, the relation representation is obtained by $\mathbf{r}^{TA}(t) = LSTM(r, t)$, where the timestamp (date) t is tokenized into digits (year, month, and day). The sequence of temporal tokens and the relation r is used as input to the LSTM. In addition to the five PEs mentioned above, we propose a new TEE where we learn a **u**nique **t**emporal **e**mbedding function for all **e**ntities to investigate the overfitting problem of DE. We refer to it as UTEE, which is defined as follows:

$$\mathbf{e}_i^{UTEE}(t)[n] = \begin{cases}\mathbf{a}[n] & \text{if } 1 \leq n \leq \gamma d, \\ \mathbf{a}[n]\sin(\boldsymbol{\omega}[n]t + \mathbf{b}[n]) & \text{else.}\end{cases}$$

where the amplitude vector $\mathbf{a}$, frequency vector $\boldsymbol{\omega}$, and bias $\mathbf{b}$ are identical for all entities.

2.2.2 Score Functions

A large number of score functions have been developed for the KG completion task. A class of these models is the translation-based approaches corresponding to variations of TransE (Bordes et al., 2013; Wang et al., 2014; Nguyen et al., 2016) that models relations as a translation of subject to object embeddings, i.e., $s^{TransE}(e_s, r, e_o) = -||\mathbf{e}_s + \mathbf{r} - \mathbf{e}_o||_2$. Another line of work is bilinear score functions (Nickel et al., 2011; Yang et al., 2014; Trouillon et al., 2016; Kazemi and Poole, 2018) that define product-based functions over embeddings, i.e., $s^{RESCAL}(e_s, r, e_o) = \mathbf{e}_s^T\mathbf{R}\mathbf{e}_o$, where relation matrix $\mathbf{R} \in \mathbb{R}^{d\times d}$ contain weights $r_{i,j}$ that capture the interaction between the i-th latent factor of $\mathbf{e}_s$ and the j-th latent factor of $\mathbf{e}_o$. Among the bilinear models, SimplE (Kazemi and Poole, 2018) a simple yet fully expressive model that represents each entity $e_i \in \mathcal{E}$ by two vectors $\mathbf{e}_{i,s}$ and $\mathbf{e}_{i,o}$. Depending on whether e_i participates in a triple as the subject or object entity, either $\mathbf{e}_{i,s}$ or $\mathbf{e}_{i,o}$ is used. To address the independence of the two vectors for each entity, SimplE takes advantage of reciprocal relations and uses $\frac{1}{2}(\langle\mathbf{e}_{i,s}, \mathbf{r}, \mathbf{e}_{j,o}\rangle + \langle\mathbf{e}_{j,s}, \mathbf{r}^{-1}, \mathbf{e}_{i,o}\rangle)$ as the score of (e_i, r, e_j), where r^{-1} is the reciprocal relation of r.

In the rest of the paper, we examine the above six temporal embeddings in terms of the two representative score functions (TransE and SimplE) on two benchmark tKG datasets. We refer to a specific combination of temporal embedding approach and score function as an *interaction model*.

2.3 Reciprocal Relations

Lacroix et al. (2018) and Dettmers et al. (2018) introduced the use of reciprocal relation for training knowledge graph embeddings. For every quadruple (e_s, r, e_o, t) in the dataset, we add (e_o, r^{-1}, e_s, t), where r^{-1} denotes the reciprocal relation of r. The idea of reciprocal relations is to use separate scoring functions for object prediction and subject prediction. Reciprocal relations can help translation-based approaches model symmetric patterns and help bilinear approaches model anti-symmetric and inverse patterns (Kazemi and Poole, 2018).

2.4 Related Work

Previous benchmarking studies (Kadlec et al., 2017; Akrami et al., 2020; Rossi et al., 2021) only focus on static knowledge graph models. For example, Ruffinelli et al. (2020) and Ali et al. (2020) realize a fair benchmarking by re-implementing static KGE models and performing an extensive empirical study with a massive search space. However, they do not take temporal knowledge graph models into account. To this end, we provide a unified framework that covers relevant tKGE models and investigate the influence of temporal embeddings on model performance as well as other components. To the best of our knowledge, this is the first benchmarking study for tKGE models.

3 Experimental Study

In this section, we first introduce the design of our unified framework that enables us to evaluate a large set of different combinations of interaction models, loss functions, regularization methods, the usage of of explicitly modeling reciprocal relations and position-aware entity embeddings. Then we split our experimental study into two parts. In the first part, we examine six temporal embedding methods combined with two representative score functions by performing an extensive set of experiments using advanced training strategies and a wide range of hyperparameter settings via the unified framework. In the second part, we re-evaluate various well-known tKG models from prior studies. We provide evidence that several old tKG models can obtain results competitive to or even better than the SOTA when configured carefully. We present the best configuration of each model and report its best performance on each benchmark that future research can directly use for comparison.

3.1 Composable Unified Framework

In the proposed framework, a tKGE model is considered as a composition of six modules that can flexibly be combined: a temporal embedding layer, a static embedding layer, a score function, a loss function, a regularization method, and the usage of reciprocal relations. In particular, the framework can automatically optimize the embedding method: the temporal embeddings can be either combined with entity embeddings or relation embeddings or both; there are different ways to combine static embeddings and temporal embeddings, i.e., addition, concatenation and element-wise multiplication. The framework supports six temporal embedding approaches as introduced in Section 2.2.1, seven score functions, i.e., TransE(Bordes et al., 2013), SimplE(Kazemi and Poole, 2018), DistMult(Yang et al., 2014), three loss functions (MR, CE, and BCE), four regularization methods (L1/L2/L3-norm , and dropout), and two initialization methods (Xavier uniform and Xavier normal). For interaction models with TEs, a smoothness regularization for timestamp embeddings is applied, enforcing neighboring timestamps to have close representations (Lacroix et al., 2020). Additionally, Kazemi and Poole (2018) distinguished an entity between as a head or as a tail entity and learns two embeddings for each entity, which we term *position-aware entity embedding* and extend to all interaction models. Position-aware entity embeddings can enhance the model's expressiveness. For example, it can help Distmult (Yang et al., 2014) to model anti-symmetric relations: without it, all relations are enforced to be symmetric since $\langle h, r, t, \tau \rangle$ and $\langle t, r, h, \tau \rangle$ share the same score regardless of properties of r.

3.2 Experimental Setup

Datasets Integrated Crisis Early Warning System (ICEWS) (Boschee et al., 2015) dataset has established itself in the research community as representative samples of tKGs and has been widely applied in recent tKG studies. The ICEWS dataset contains information about political events with specific time annotations, e.g. (Barack Obama, visit, India, 2010-11-06). We apply our model on two subsets of the ICEWS dataset: ICEWS14 contains events in 2014, and ICEWS11-14 corresponds to the facts between 2011 to 2014. Besides, we also used a subset of the Global Database of Events, Language, and Tone (GDELT) (Leetaru and Schrodt, 2013) dataset as a benchmark. To make the extensive configuration search feasible, we extracted a subset named GDELT-m10 consisting of factual events in October, 2015. The statistics and further details are provided in Appendix C.

Hyperparameters We used a large hyperparameter search space to ensure that suitable hyperparameters for each model can be covered. We consider seven embedding dimensions $\{64, 100, 128, 256, 512, 1024, 2048\}$. The learning rate can be randomly selected from $(0, 0.1]$. We use separate weights for regularization of embeddings of entities, relations, and timestamps. A detailed report of the search space is provided in Appendix B.

Interaction models In the first part, we evaluate six temporal embedding methods combined with two representative score functions. The formulas of these twelve interaction models are listed in Table 1. Additionally, we select DE-SimplE/TransE(Goel et al., 2019), TNTComplEx(Lacroix et al., 2020), ATiSE(Xu et al., 2019), TTransE(Leblay and Chekol, 2018), TA-TransE (García-Durán et al.,

Table 1: Formulas of a given quadruple (e_i, r, e_j, t). $\mathbf{e}_{i,s}$ denotes the embedding of e_i when the entity is the subject while $\mathbf{e}_{i,o}$ denotes the embedding of e_i when the entity is the object. In comparison, $\mathbf{e}_i$ represents the shared embedding of entity e_i for both subject and object. $\mathbf{t}$, $\mathbf{r}$ represent the embedding of timestamp t and relation r, respectively. $\perp$ represents the projection operator. $\mathbf{e}_i(t)$ denotes the temporal embedding of e_i at t.

Temporal Embeddings	TransE	SimplE
T	$\lVert\mathbf{e}_i + \mathbf{r} + \mathbf{t} - \mathbf{e}_j\rVert$	$\frac{1}{2}(\langle \mathbf{e}_{i,s}, \mathbf{r}, \mathbf{t}, \mathbf{e}_{j,o}\rangle + \langle \mathbf{e}_{j,s}, \mathbf{r}^{-1}, \mathbf{t}, \mathbf{e}_{i,o}\rangle)$
DE	$\lVert\mathbf{e}_i^{DE}(t) + \mathbf{r} - \mathbf{e}_j^{DE}(t)\rVert$	$\frac{1}{2}(\langle \mathbf{e}_{i,s}^{DE}(t), \mathbf{r}, \mathbf{e}_{j,o}^{DE}(t)\rangle + \langle \mathbf{e}_{j,s}^{DE}(t), \mathbf{r}^{-1}, \mathbf{e}_{i,o}^{DE}(t)\rangle)$
UTEE	$\lVert\mathbf{e}_i^{UTEE}(t) + \mathbf{r} - \mathbf{e}_j^{UTEE}(t)\rVert$	$\frac{1}{2}(\langle \mathbf{e}_{i,s}^{UTEE}(t), \mathbf{r}, \mathbf{e}_{j,o}^{UTEE}(t)\rangle + \langle \mathbf{e}_{j,s}^{UTEE}(t), \mathbf{r}^{-1}, \mathbf{e}_{i,o}^{UTEE}(t)\rangle)$
HyTE	$\lVert\mathbf{e}_i \perp \boldsymbol{\omega}_t + \mathbf{r} \perp \boldsymbol{\omega}_t - \mathbf{e}_j \perp \boldsymbol{\omega}_t\rVert$	$\frac{1}{2}(\langle \mathbf{e}_{i,s} \perp \boldsymbol{\omega}_t, \mathbf{r} \perp \boldsymbol{\omega}_t, \mathbf{e}_{j,o} \perp \boldsymbol{\omega}_t\rangle + \langle \mathbf{e}_{j,s} \perp \boldsymbol{\omega}_t, \mathbf{r} \perp \boldsymbol{\omega}_t, \mathbf{e}_{i,o} \perp \boldsymbol{\omega}_t\rangle)$
ATiSE	$\lVert\mathbf{e}_i^{ATiSE}(t) + \mathbf{r}^{ATiSE}(t) - \mathbf{e}_j^{ATiSE}(t)\rVert$	$\frac{1}{2}(\langle \mathbf{e}_{i,s}^{ATiSE}(t), \mathbf{r}^{ATiSE}(t), \mathbf{e}_{j,o}^{ATiSE}(t)\rangle + \langle \mathbf{e}_{j,s}^{ATiSE}(t), \mathbf{r}^{-1,ATiSE}(t), \mathbf{e}_{i,o}^{ATiSE}(t)\rangle)$
TA	$\lVert\mathbf{e}_i + \mathbf{r}^{TA}(t) - \mathbf{e}_j\rVert$	$\frac{1}{2}(\langle \mathbf{e}_{i,s}, \mathbf{r}^{TA}(t), \mathbf{e}_{j,o}\rangle + \langle \mathbf{e}_{j,s}, \mathbf{r}^{-1,TA}(t), \mathbf{e}_{i,o}\rangle)$

2018), and HyTE (Dasgupta et al., 2018) for the second part of our study, which are the most famous tKGE models.

Evaluation All models are evaluated on *link prediction task*. For each test quadruple (s, r, o, t), we create a subject prediction query $(?, r, o, t)$ and an object prediction query $(s, r, ?, t)$. Taking the object prediction as an example, all entities $e_i \in \mathcal{E}$ are ranked according to the score $s(s, r, e_i, t)$. We filter from the candidate list all the entities but the ground truth that form a valid quadruple with s, r, and t, i.e., the quadruple occurs either in the training, validation, or test data. We report filtered Mean Reciprocal Ranks (MRR) and Hits@1, 3, 10 averaged over subject prediction and object prediction. For detailed definitions please see Appendix A.

Computational resources and model selection. We perform large-scale benchmarking with about 4000 experiments and 19000 GPU hours of computation time. All experiments are run on NVIDIA Tesla T4. For each dataset and interaction model, we first randomly generate 40 different configurations from the search space using the Ax framework[2]. After the random hyperparameter search, we search 60 new configurations based on Bayesian optimization to tune the numerical hyperparameters further. Each trial runs for 100 epochs, and an early stopping strategy with a patience of 30 epochs is employed. We select the best-performing configuration according to filtered MRR on validation data. The best configuration will be further trained until its convergence.

3.3 Examining Temporal Embeddings

Performance in prior studies vs. in our study. Table 3 shows the filtered MRR and filtered Hits@1/3/10 on test data of various temporal embeddings on ICEWS14 and ICEWS11-14 datasets. We found that the relative performance differences between various temporal embeddings often shrink and sometimes even reverse compared to published results. For example, T-TransE was first run on ICEWS14 by (García-Durán et al., 2018), achieving a filtered MRR of 25.5%. This number is relatively low compared to today's standards. In comparison, T-TransE achieves a superior MRR of **55.3%** in our study, which has been improved significantly. Studies that report the lower performance number of T-TransE (i.e., 25.5%) thus do not fairly compare the temporal embedding approaches. Similar remarks hold for DE-TransE and HyTE-TransE. (Goel et al., 2019) proposed DE-TransE and report an MRR of 32.6% on ICEWS14 while it achieves an MRR of 50.8 % in our study. Similarly, the achieved MRR of HyTE-TransE on ICEWS14 is 42.9% in our study, which significantly improves the reported results (29.7%) in previous studies (Goel et al., 2019; Sadeghian et al., 2021). The results suggest that the performance of old temporal embedding approaches can be largely improved by advanced training strategies and hyperparameter-tuning, which may account for a large fraction of the progress made in recent years. Figure 1 shows the distribution of

Table 2: Selected hyperparameters of best performing configurations of selected tKG models on ICEWS14. A full description of hyperparameters are reported in Table 12 in the appendix.

	TTransE	T-SimplE	DE-TransE	DE-SimplE
Emb. size	512	256	256	128
lr.	7e-3	9e-3	2e-3	4e-3
loss	CE	CE	CE	BCE
Reciprocal	Yes	Yes	Yes	Yes
Position-aware ent. emb.	Yes	Yes	Yes	Yes

[2]https://ax.dev/

Table 3: Link prediction results of six temporal embedding approaches with two representative score functions on ICEWS datasets: MRR (%) and Hits@1/3/10 (%). The best results in group are in bold.

Dataset	ICEWS14								ICEWS11-14							
Score function	TransE				SimplE				TransE				SimplE			
Temporal Embeddings	MRR	Hits@1	Hits@3	Hits@10	MRR	Hits@1	Hits@3	Hits@10	MRR	Hits@1	Hits@3	Hits@10	MRR	Hits@1	Hits@3	Hits@10
T	**55.3**	**43.7**	**62.7**	**76.5**	**53.9**	**43.9**	59.4	73.0	**57.8**	**46.0**	**65.5**	**79.5**	**60.2**	**51.3**	**65.2**	75.5
DE	50.8	38.7	59.0	72.4	53.9	42.5	61.2	74.6	54.1	42.1	60.9	77.1	54.2	42.3	61.0	67.8
UTEE	52.6	40.5	60.3	74.7	53.7	42.5	**60.8**	**74.8**	55.2	43.0	63.3	77.5	56.1	45.2	62.9	**76.4**
HyTE	47.9	35.8	54.1	71.8	52.3	41.9	58.9	71.4	48.2	36.3	54.1	72.0	54.9	43.1	61.7	77.4
ATiSE	47.1	34.7	53.8	71.2	46.6	34.7	53.4	69.7	51.0	38.8	57.7	74.5	49.3	37.5	56.1	72.2
TA	22.3	14.4	25.0	37.5	37.1	25.3	42.2	61.4	26.3	18.3	28.6	43.0	33.4	24.0	37.6	51.2

Table 4: Link prediction results of six temporal embedding approaches with two representative score functions on the GDELT-m10 dataset: MRR (%) and Hits@1/3/10 (%). The best results in group are in bold.

Dataset	GDELT-m10							
Score function	TransE				SimplE			
Temporal Embeddings	MRR	Hits@1	Hits@3	Hits@10	MRR	Hits@1	Hits@3	Hits@10
T	**31.6**	**22.7**	**34.0**	**48.9**	30.8	21.6	33.5	48.8
DE	25.9	17.1	28.1	43.0	**34.4**	**24.9**	**37.5**	**53.0**
UTEE	26.1	16.9	28.3	44.1	28.5	18.9	30.9	47.4
HyTE	30.8	21.9	33.2	48.3	27.4	17.8	29.9	46.9
ATiSE	25.3	16.7	27.3	42.1	29.6	20.6	32.2	47.3
TA	11.6	1.0	16.1	29.8	19.9	12.4	21.1	34.3

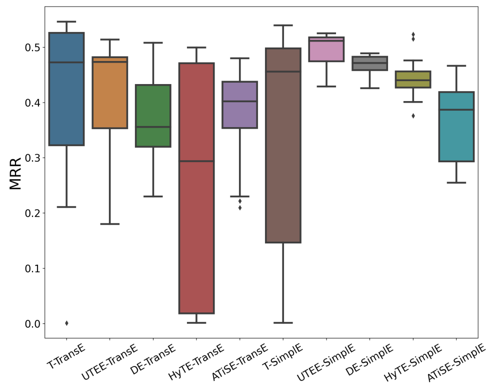


Figure 1: Distribution of filtered MRR (%) on ICEWS14 over the hyperparameter configurations explored in our study.

filtered MRR for each model on ICEWS14. Each distribution consists of 100 different hyperparameter configurations. We can see that some models show a wide dispersion, and only very few configurations achieve good results. Generally, the impact of the hyperparameter choice is more pronounced on TransE-based models (higher variance) than on SimplE-based models. The hyperparameters of the best performing models are reported in Table 2 (selected hyperparameters) and Table 12 in the appendix (all parameters). Perhaps unsurprisingly, we find that the optimum choice of hyperparameters is often model- and dataset-dependent. Thus, a grid search on a small search space is not suitable to compare model performance because the result may be considerably affected by the specific grid points being used. Besides, we find that the use of **r**eciprocal **r**elations (RR) and **p**osition-aware **e**ntity **e**mbeddings (PEE) often improve model performance. To investigate their impacts, we conduct ablation studies where we do not use RR (or PEE) and keep other hyperparameters same to the best configuration. We report the reduction of filtered metrics in Table 5, which confirms our findings.

TE vs. TEE Since the timestamp embeddings (TE) are independent of entities, they can only capture global patterns at each timestamp. In comparison, the time-dependent entity embedding approaches (DE, ATiSE) learn entity-specific temporal functions (e.g., frequency, amplitude, etc.) as shown in Equation 1 and 2. The time-dependent entity embeddings are expected to capture entity-specific temporal features, and thus, being more expressive. However, we see that the simple timestamp embedding approach (T) proposed by (Leblay and Chekol, 2018) achieves overall the best performance. In particular, it outperforms the time-dependent entity embedding approaches (DE, ATiSE), which is in contrast to the common belief. Table 6 provides the number of learnable parameters of each model, showing that the interaction model with timestamp embeddings (T) has significantly fewer model parameters than time-dependent entity embeddings (DE, ATiSE). We argue that the existing time-dependent entity embeddings are overfitting to temporal signals. To this end, we propose the unique time-dependent entity embeddings (UTEE), where we learn a unique (global) entity embedding function for all entities to investigate the overfitting problem of DE. In other words, all entities have the same temporal embedding part. Notably, the model parameter of DE is often more than three times than UTEE. As shown

Table 5: Impact of reciprocal relations and position-aware entity embeddings on ICEWS14. The number in parenthesis shows the performance reduction if the best configuration doesn't use the reciprocal relation or position-aware embeddings.

	with/without reciprocal relation				**with/without position-aware entity embeddings**			
Models	MRR(%)	Hits@1(%)	Hits@3(%)	Hits@10(%)	MRR(%)	Hits@1(%)	Hits@3(%)	Hits@10(%)
T-TransE	51.7 (-3.6)	39.4(-4.3)	59.1(-3.6)	75.0(-1.5)	31.2(-24.1)	10.6(-33.1)	44.8(-17.9)	68.4(-8.1)
HyTE-TransE	40.1(-7.0)	28.4(-9.9)	44.9(-12.2)	64.3(-7.7)	27.7(-19.4)	9.6(-32.3)	38.2(-20.7)	62.2(-7.5)
HyTE-SimplE	41.2(-11.1)	28.9(-13.0)	47.3(-11.6)	65.4(-4.3)	51.2(-1.1)	40.9(-1.0)	57.7(-1.2)	70.2(-1.2)

in Table 3, the UTEE achieves competitive or even better performance with both translation-based and bilinear score functions on both datasets. Additionally, Table 4 shows the evaluation metrics on the GDELT-m10 dataset. Compared to the ICEWS datasets, the number of entities and relations on GDELT-m10 is much fewer while the amount of timestamped edges is about three times more than the ICEWS14 dataset. Thus, the data sparsity issue is alleviated in the GDELT-m10 dataset. Since TEE approaches need dense data for training, their performance has been improved on the GDELT-m10 dataset, which is better than TEs. The results suggest that even though DE has theoretical full expressiveness and provides more freedom degrees of the temporal movements of each entity representation, their performance would deteriorate significantly on sparse data. We tried to add regularization to entity-specific parameters, e.g., amplitude and frequency, and adjust the portion γ of the temporal embeddings. However, there are no significant improvements. Thus, the time-dependent entity embeddings need to be revisited to realize their theoretical expressiveness.

Table 6: Model parameters number: million (M).

Dataset	**ICEWS14**		**ICEWS11-14**		**GDELT-m10**	
Score function	**TransE**	**SimplE**	**TransE**	**SimplE**	**TransE**	**SimplE**
T	7.72M	3.86M	7.89M	3.94M	0.55M	0.55M
UTEE	7.54M	3.77M	28.57M	7.14M	0.55M	0.55M
DE	16.2M	12.01M	18.45M	14.6M	0.82M	2.11M
HyTE	3.86M	3.86M	1.54M	3.94M	0.55M	0.27M
ATiSE	18.9M	4.72M	8.94M	6.98M	0.67M	0.67M
TA	0.78M	0.78M	0.5M	0.5M	0.22M	0.22M

Findings on other temporal embeddings. Besides, we find HyTE is sensitive to the choice of score functions. With the translation-based score function (TransE), HyTE only achieves a relatively low number by today's standards while it obtains a competitive number with the bilinear score function (SimplE). This suggests that the score function has a considerable impact on model performance and may account for a large fraction of the progress. Thus, if a new temporal embedding technique is proposed, it should be evaluated on different score functions to assess its benefits. Additionally, we see that the TA-approach (García-Durán et al., 2018) achieves overall relatively low numbers by today's standards, showing its limited capacity. Moreover, we find that the performance of ATiSE considerably deteriorates in our study compared to the prior study. The difference is that we do not cover the Gaussian noise component in our study. This result suggests that taking temporal uncertainty into account would significantly improve the tKG models. Thus, it is worth extending other deterministic KG models into probabilistic approaches.

3.4 Benchmarking tKGE Models

Table 7: Link prediction results of well-known tKG models on ICEWS14. The number outside the parentheses is the performance achieved in our study. The number in the parentheses is the best performance results obtained in prior studies. We list the references indicate where the performance number was reported: TTransE/TA-TransE (García-Durán et al., 2018), HyTE/DE-TransE/DE-SimplE (Goel et al., 2019), TNTComplEx (Lacroix et al., 2020), ATiSE (Xu et al., 2019). For ATiSE and HyTE, we use the same score function (KL divergence and TransE, respectively) as reported in their original papers.

Models	MRR (%)	Hits@1 (%)	Hits@10 (%)
TTransE	55.3(25.5)	43.7(7.4)	76.5(60.1)
HyTE	47.9(29.7)	35.8(10.8)	71.8(65.5)
DE-TransE	50.8(32.6)	38.7(12.4)	72.4(68.6)
DE-SimplE	53.9(52.6)	42.5(41.8)	74.6(72.5)
ATiSE	55.1(55.0)	42.5(43.6)	75.0(75.0)
TNTComplEx	60.6(62)	51.6(52)	77.3(76)
TA-TransE	26.3(27.5)	18.3(9.5)	43.0(62.5)

Table 7 depicts the best performance of well-known tKGE models from prior studies (numbers in the parentheses) and that found in our study (numbers outside the parentheses). The configuration of the best performing models are reported in Table 13 in the appendix. First, we find that the performance of a single model can vary wildly across studies. For example, DE-TransE, T-TransE, and HyTE have been significantly improved using advanced training strategies and hyperparameter-tuning. Besides, we see that some recent models cannot consistently outperform old models in contrast to the conclusion in prior studies. In particular, T-TransE, which constitutes one of the first tKGE models, achieves results competitive to advanced models, i.e., ATiSE and DE-SimplE, in our study. Even compared to TNTComplEx, which is a very large models with 25.12 million learnable parameters (3 times more than TTransE), the performance difference is not large. We provide explanation for the performance gap between our study and prior study regarding TA-TransE and TNTComplEx in Appendix D.

4 Conclusion

We assess well-known temporal embeddings of tKGE models via an extensive experimental study. We found that when trained appropriately, the naive timestamp embedding approach performs similarly or even outperforms the more advanced time-dependent entity embedding (TEE) approaches, which is in contrast to the results in prior studies. We contribute to the community in at least two ways: $i)$ we provide a unified framework to enable an insightful assessment for novel temporal embedding approaches; $ii)$ reveal the weakness of TEE approaches.

References

Farahnaz Akrami, Mohammed Samiul Saeef, Qingheng Zhang, Wei Hu, and Chengkai Li. 2020. Realistic re-evaluation of knowledge graph completion methods: An experimental study. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, pages 1995–2010.

Mehdi Ali, Max Berrendorf, Charles Tapley Hoyt, Laurent Vermue, Mikhail Galkin, Sahand Sharifzadeh, Asja Fischer, Volker Tresp, and Jens Lehmann. 2020. Bringing light into the dark: A large-scale evaluation of knowledge graph embedding models under a unified framework. *arXiv preprint arXiv:2006.13365*.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Advances in neural information processing systems*, pages 2787–2795.

Elizabeth Boschee, Jennifer Lautenschlager, Sean O'Brien, Steve Shellman, James Starz, and Michael Ward. 2015. Icews coded event data. *Harvard Dataverse*, 12.

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

Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. 2018. Convolutional 2d knowledge graph embeddings. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

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

Rishab Goel, Seyed Mehran Kazemi, Marcus Brubaker, and Pascal Poupart. 2019. Diachronic embedding for temporal knowledge graph completion. *arXiv preprint arXiv:1907.03143*.

Zhen Han, Peng Chen, Yunpu Ma, and Volker Tresp. 2021. Explainable subgraph reasoning for forecasting on temporal knowledge graphs. In *International Conference on Learning Representations*.

Zhen Han, Yunpu Ma, Peng Chen, and Volker Tresp. 2020a. Dyernie: Dynamic evolution of riemannian manifold embeddings for temporal knowledge graph completion. *arXiv preprint arXiv:2011.03984*.

Zhen Han, Yunpu Ma, Yuyi Wang, Stephan Günnemann, and Volker Tresp. 2020b. Graph hawkes neural network for forecasting on temporal knowledge graphs. *arXiv preprint arXiv:2003.13432*.

Woojeong Jin, Changlin Zhang, Pedro Szekely, and Xiang Ren. 2019. Recurrent event network for reasoning over temporal knowledge graphs. *arXiv preprint arXiv:1904.05530*.

Rudolf Kadlec, Ondrej Bajgar, and Jan Kleindienst. 2017. Knowledge base completion: Baselines strike back. *arXiv preprint arXiv:1705.10744*.

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

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Timothee Lacroix, Guillaume Obozinski, and Nicolas Usunier. 2020. Tensor decompositions for temporal knowledge base completion. *ICLR preprint https://openreview.net/pdf?id=rke2P1BFwS*.

Timothée Lacroix, Nicolas Usunier, and Guillaume Obozinski. 2018. Canonical tensor decomposition for knowledge base completion. *arXiv preprint arXiv:1806.07297*.

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

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

Yunpu Ma, Volker Tresp, and Erik A Daxberger. 2018. Embedding models for episodic knowledge graphs. *Journal of Web Semantics*, page 100490.

Sameh K Mohamed, Vít Novácek, Pierre-Yves Vandenbussche, and Emir Muñoz. 2019. Loss functions in knowledge graph embedding models. In *DL4KG@ ESWC*, pages 1–10.

Dat Quoc Nguyen, Kairit Sirts, Lizhen Qu, and Mark Johnson. 2016. Stranse: a novel embedding model of entities and relationships in knowledge bases. *arXiv preprint arXiv:1606.08140*.

Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. 2011. A three-way model for collective learning on multi-relational data. In *Icml*, volume 11, pages 809–816.

Andrea Rossi, Denilson Barbosa, Donatella Firmani, Antonio Matinata, and Paolo Merialdo. 2021. Knowledge graph embedding for link prediction: A comparative analysis. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 15(2):1–49.

Daniel Ruffinelli, Samuel Broscheit, and Rainer Gemulla. 2020. You {can} teach an old dog new tricks! on training knowledge graph embeddings. In *International Conference on Learning Representations*.

Ali Sadeghian, Mohammadreza Armandpour, Anthony Colas, and Daisy Zhe Wang. 2021. Chronor: Rotation based temporal knowledge graph embedding. *arXiv preprint arXiv:2103.10379*.

Volker Tresp, Cristóbal Esteban, Yinchong Yang, Stephan Baier, and Denis Krompaß. 2015. Learning with memory embeddings. *arXiv preprint arXiv:1511.07972*.

Volker Tresp, Yunpu Ma, Stephan Baier, and Yinchong Yang. 2017. Embedding learning for declarative memories. In *European Semantic Web Conference*, pages 202–216. Springer.

Rakshit Trivedi, Hanjun Dai, Yichen Wang, and Le Song. 2017. Know-evolve: Deep temporal reasoning for dynamic knowledge graphs. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70, pages 3462–3471. JMLR. org.

Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. 2016. Complex embeddings for simple link prediction. International Conference on Machine Learning (ICML).

Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014. Knowledge graph embedding by translating on hyperplanes. In *Twenty-Eighth AAAI conference on artificial intelligence*.

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

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

Cunchao Zhu, Muhao Chen, Changjun Fan, Guangquan Cheng, and Yan Zhan. 2020. Learning from history: Modeling temporal knowledge graphs with sequential copy-generation networks. *arXiv preprint arXiv:2012.08492*.

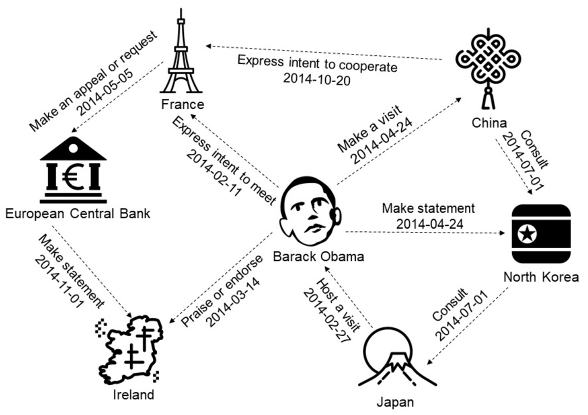


Figure 2: Exemplary temporal KG: nodes represent entities and edges their respective relations.

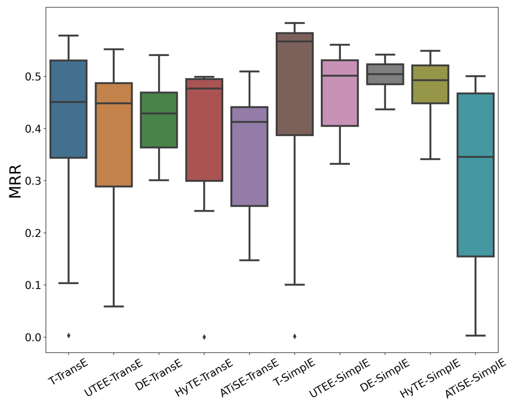


Figure 3: Distribution of filtered MRR (%) on ICEWS11-14 over the hyperparameter configurations explored in our study.

Appendix

Table 8: Link prediction results of well-known temporal KG models on ICEWS11-14: MRR (%) and Hits@1/3/10 (%).

Models	MRR	Hits@1	Hits@3	Hits@10
TTransE	57.8	46.0	65.5	79.5
HyTE	49.8	37.6	56.2	74.0
DE-TransE	54.1	42.1	60.9	77.1
DE-SimplE	54.2	42.3	61.0	67.8
TNTComplEx	63.5	55.4	68.5	78.8
ATiSE	53.3	40.3	61.4	77.9

A Evaluation Metrics

For each test quadruple (e_s, r, e_o, t), we create a subject prediction query $(?, r, e_o, t)$ and an object prediction query $(e_s, r, ?, t)$. Let ψ_{e_s} and ψ_{e_o} represent the rank for ground truth subject e_s and ground truth object e_o of the subject prediction query and object prediction query, respectively. We evaluate our models using standard metrics across the link prediction literature: *mean reciprocal rank (MRR)*: $\frac{1}{2\cdot|\mathcal{G}_{test}|}\sum_{q\in\mathcal{G}_{test}}(\frac{1}{\psi_{e_s}}+\frac{1}{\psi_{e_o}})$ and *Hits*$@k(k \in \{1, 3, 10\})$: the percentage of times that the true entity candidate appears in the top k of ranked candidates.

There are two common filtering settings. The first one is following the ranking technique described in (Bordes et al., 2013), where we remove from the list of corrupted **triples** all the **triples** that appear either in the training, validation, or test set. We name it *static filtering*. Trivedi et al. (2017), Jin et al. (2019), and Zhu et al. (2020) use this filtering setting for reporting their results on temporal KG forecasting. However, this filtering setting is not appropriate for evaluating the link prediction on temporal KGs. For example, there is a test quadruple (Barack Obama, visit, India, 2015-01-25), and we perform the object prediction (Barack Obama, visit, ?, 2015-01-25). We have observed the quadruple (Barack Obama, visit, Germany, 2013-01-18) in training set. According to the *static filtering*, (Barack Obama, visit, Germany) will be considered as a genuine triple at the timestamp **2015-01-25** and will be filtered out because the triple (Barack Obama, visit, Germany) appears in the training set in the quadruple (Barack Obama, visit, Germany, 2015-01-18). However, the triple (Barack Obama, visit, Germany) is only temporally valid on 2013-01-18 but not on 2015-01-25. To this end, another filtering scheme was introduced, which is more appropriate for the link forecasting task on temporal KGs. We name it *time-aware filtering*. In this case, we only filter out the triples that are genuine at the timestamp of the query. In other words, if the triple (Barack Obama, visit, Germany) does not appear at the query time of 2015-01-25, the quadruple (Barack Obama, visit, Germany, 2015-01-25) is considered as corrupted. In this paper, we focus on time-aware filtering.

B Additional Information of Hyperparameter Search Space

Loss functions Various loss functions are used in training temporal knowledge graphs. Dasgupta et al. (2018); Leblay and Chekol (2018) used margin ranking (MR) loss for training, where each pair consists of a positive quadruple and one of its negative quadruple. The margin η is a hyperparameter. Goel et al. (2019); García-Durán et al. (2018) treat the entity prediction task as a categorical classification problem and utilize the cross entropy (CE) loss to align the model distribution and the data distribution. Han et al. (2020a) proposed to use binary cross entropy (BCE) loss that applies a sigmoid function to the score of each positive or negative quadruples and takes the cross entropy between the resulting probability and that quadruple's label as the loss. It has been shown in (Ruffinelli et al., 2020; Mohamed et al., 2019) that the loss function has a significant impact on the performance of static KGE models. To provide additional evidence on temporal KGE models, we search the best choice of loss functions for each model on each dataset.

Regularization methods L2 regularization is widely used in literature (Leblay and Chekol, 2018). Besides, (Dasgupta et al., 2018) proposed to use L1-norm in the regularization term. And (Lacroix et al., 2020) used L3-norm for CP-decomposition. Additionally, (Lacroix et al., 2020) proposed a smoothness regularization for timestamp embeddings that enforce neighboring timestamps to have close representations. Moreover, (Goel et al., 2019) used dropout in its hidden layers. AiTSE normalized the static embeddings $\mathbf{e}_i$, the trend component $\mathbf{w}_{e_i}$ to unit norm after each update.

Other hyperparameters For models with diachronic entity embedding as its temporal encoding heads, we extend the static feature ratio as an extra searchable hyperparameter. The negative sample ratio of the negative sampling policy is 500. Namely, for each positive sample (s, p, o, t), we corrupt the subject and object entity via uniformly sampling from T, where $T = \{(s', p, o, t) | s' \in \mathcal{E} \backslash s\} \cup (\{(s', p, o, t) | t' \in \mathcal{E} \backslash o\}$. We set our batch size to be 512. Besides, since Adam (Kingma and Ba, 2014) optimizer performs well for the majority of the models (Ali et al., 2020), we decided to progress only with Adam in order to reduce the computational costs. Additionally, for translational models, we set the margin γ to be 100 in the score function.

C Datasets

Dataset statistics including subset split information are described in Table 11. We follow the data preprocessing method used in the original papers. For example, DE-SimplE (Goel et al., 2019) takes the date (day/month/year) as timestamp input while AiTSE (Xu et al., 2019) converts dates into consecutive integers.

D Reproducibility Studies

We were not able to reproduce the results of TA-TransE on ICEWS14. A reason might be differences in the implementation details of the frameworks used to train and evaluate the models. Since there exists no official implementation for TA-TransE, it is not possible to check the implementation difference. Also, García-Durán et al. (2018) did not report the full setup, which impedes the reproduction of results. For example, the regularization method and initialization method have not been reported, which can have a significant effect on the results.

Lacroix et al. (2020) provides an official implementation of TNTComplEx. However, we were not able to reproduce the same metric number as reported in their paper. Similarly, Sadeghian et al. (2021) also did not successfully reproduce the results of TNTComplEx. The initialization of the embeddings might be a reason.

E Average Runtime of each Approach

Table 9 shows the average runtime of each training epoch for each interaction model.

Table 9: The average runtime of each training epoch: seconds (s).

Dataset	**ICEWS14**		**ICEWS11-14**		**GDELT-m10**	
Score function	**TransE**	**SimplE**	**TransE**	**SimplE**	**TransE**	**SimplE**
T-	99s	64s	80s	105s	290s	321s
UTEE-	128s	75s	450s	262s	1230s	638s
DE-	196s	145s	375s	302s	438s	518s
HyTE-	208s	212s	146s	105s	360s	382s
ATiSE-	317s	75s	212s	175s	380s	390s
TA-	730s	365s	730s	365s	2696s	3751s

Table 10: Hyperparameter search space used in our study.

Hyperparameter	Search space
Embedding	
Embedding dimension	{64, 100, 128, 256, 512, 1024, 2048}
Embedding initialization	{Xavier Uniform, Xavier Normal}
Training	
Reciprocal relation	{True, False}
Position-aware entity embeddings	{True, False}
Loss function	{CE, BCE, MR}
Learning rate	(0.0, 0.1]
Regularization	
Entity regularization type	{None, L1, L2, L3}
Entity regularization weight	(0.0, 0.1]
Relation regularization type	{None, L1, L2, L3}
Relation regularization weight	(0.0, 0.1]
Timestamp smoothness regularization weight	(0.0, 0.1]
Dropout	[0.0, 0.6]

Data set	N_{train}	N_{valid}	N_{test}	N_{ent}	N_{rel}	$N_{timestamp}$	Time granularity
ICEWS14	72826	8941	8963	7128	230	365	day
ICEWS11-14	118766	14859	14756	6738	235	1461	day
GDELT-m10	221132	27608	27926	50	20	30	day

Table 11: Dataset Statistics

Table 12: Best configurations of six temporal embeddings with two score functions on ICEWS14, ICEWS11-14 and GDELT-m10.

		Emb. dim.	Emb. init.	SE ratio	Recip. rel.	Pos-aware entity	Learning rate	Loss func.	Ent. reg. type	Ent. reg. weight	Rel. reg. type	Rel. reg. weight	Temp. smooth. weight.	Dropout
ICEWS14	T-TransE	512	Xavier Uniform	-	True	True	0.0075324	CE	-	-	-	-	1.9513135e-17	0.5
	UTEE-TransE	512	Xavier Uniform	-	True	True	0.0010000	CE	None	-	None	-	-	0.5
	DE-TransE	256	Xavier Uniform	0.57	True	True	0.0019251	CE	None	-	None	-	-	0.4
	HyTE-TransE	128	Xavier Uniform	-	True	True	0.0080112	CE	L2	0.0368056	L2	0.0762292	0.0381662	0.6
	ATiSE-TransE	256	Xavier Uniform	-	True	True	0.0029397	CE		-	-	-	-	0.2
	T-SimplE	256	Xavier Uniform	-	True	True	0.0090146	CE	None	-	None	-	0.0000881	0.6
	UTEE-SimplE	256	Xavier Uniform	-	True	True	0.0030251	BCE	-	-	-	-	-	0.5
	DE-SimplE	128	Xavier Uniform	0.42	True	True	0.0038826	BCE	-	-	-	-	-	0.4
	HyTE-SimplE	256	Xavier Uniform	-	True	True	0.0089471	CE	None	-	None	-	0.0315673	0.2
	ATiSE-SimplE	64	Xavier Uniform	-	True	True	0.0006407	BCE	L3	0.0165849	L3	0.0148400	-	0.2
ICEWS11-14	T-TransE	512	Xavier Uniform	-	False	True	0.0018173	CE	-	-	-	-	0.0115133	0.4
	UTEE-TransE	2048	Xavier Normal	-	True	True	0.0003236	CE	None	-	L2	0.0968473	-	0.0
	DE-TransE	256	Xavier Uniform	0.46	True	True	0.0026801	CE	-	-	-	-	-	0.6
	HyTE-TransE	100	Xavier Uniform	-	True	True	0.0084563	CE	L2	0.0548569	L3	0.0848771	0.0718383	0.0
	ATiSE-TransE	128	Xavier Uniform	-	True	True	0.0013809	CE	L3	0.0011405	L2	0.0182759	0.0115133	0.2
	T-SimplE	256	Xavier Uniform	-	True	True	0.0015910	CE	-	-	-	-	0.0049301	0.4
	UTEE-SimplE	512	Xavier Normal	-	True	True	0.0048310	CE	-	-	-	-	-	0.6
	DE-SimplE	128	Xavier Normal	0.07	True	True	0.0040708	CE	None	-	None	-	-	0.6
	HyTE-SimplE	100	Xavier Normal	-	True	True	0.0077529	CE	None	-	None	-	0.0751268	0.5
	ATiSE-SimplE	100	Xavier Uniform	-	True	True	0.0006454	CE	L2	0.0397851	L3	0.0262544	-	0.4
GDELT-m10	T-TransE	512	Xavier Normal	-	TRUE	TRUE	0.0005205	CE	None	-	L3	0.0861472	0.0123214	0.2
	UTEE-TransE	512	Xavier Uniform	-	TRUE	TRUE	0.0018018	BCE	None	-	None	-	-	0.4
	DE-TransE	100	Xavier Normal	0.10	TRUE	TRUE	0.0021739	CE	None	-	None	-	-	0.4
	HyTE-TransE	512	Xavier Normal	-	TRUE	TRUE	0.0007655	CE	L3	0.0128709	L2	0.0440400	0.0755946	0.2
	ATiSE-TransE	128	Xavier Normal	-	TRUE	TRUE	0.0001836	CE	L2	0.0322397	L2	0.0854999	-	0.4
	T-SimplE	512	Xavier Normal	-	TRUE	TRUE	0.0086667	CE	L3	0.0039186	L3	0.0466820	0.0730538	0.2
	UTEE-SimplE	512	Xavier Normal	-	TRUE	TRUE	0.0006767	CE	L2	0.0035275	None	-	-	0.4
	DE-SimplE	256	Xavier Normal	0.66	TRUE	TRUE	0.0021578	BCE	None	-	None	-	-	0.4
	HyTE-SimplE	256	Xavier Normal	-	TRUE	TRUE	0.0046384	CE	L3	0.0034211	L2	0.0986175	0.0143311	0.2
	ATiSE-SimplE	128	Xavier Normal	-	TRUE	TRUE	0.0059198	CE	L2	0.0002838	L2	0.0819929	-	0.4

Table 13: Best configurations of well-known temporal knowledge graph models on ICEWS14 and ICEWS11-14.

		Emb. dim.	Emb. init.	SE ratio	Recip. rel.	Pos-aware entity	Learning rate	Loss func.	Ent. reg. type	Ent. reg. weight	Rel. reg. type	Rel. reg. weight	Temp. smooth. weight.	Dropout
ICEWS14	T-TransE	512	Xavier Uniform	-	True	True	0.0075324	CE	-	-	-	-	1.9513135e-17	0.5
	DE-TransE	256	Xavier Uniform	0.57	True	True	0.0019251	CE	None	-	None	-	-	0.4
	DE-SimplE	128	Xavier Uniform	0.42	True	True	0.0038826	BCE	-	-	-	-	-	0.4
	TNTComplEx	1560	-	-	True	-	0.1	CE	n3	0.01	-	-	0.01	-
	ATiSE	500	-	-	False	-	0.00003	LMR	-	-	-	-	-	-
	HyTE	100	Xavier Uniform	-	True	True	0.0084563	CE	L2	0.0548569	L3	0.0848771	0.0718383	0.0
ICEWS11-14	TTransE	512	Xavier Uniform	-	False	True	0.0018173	CE	-	-	-	-	0.0115133	0.4
	DE-TransE	256	Xavier Uniform	0.46	True	True	0.0026801	CE	-	-	-	-	-	0.6
	DE-SimplE	128	Xavier Normal	0.07	True	True	0.0040708	CE	None	-	None	-	-	0.6
	TNTComplEx	1560	-	-	True	-	0.0859516	CE	n3	0.001	-	-	0.1	-
	ATiSE	500	-	-	False	-	0.00003	LMR	-	-	-	-	-	-
	HyTE	100	Xavier Normal	-	True	True	0.0077529	CE	None	-	None	-	0.0751268	0.5
GDELT-m10	TTransE	512	Xavier Normal	-	TRUE	TRUE	0.0005205	CE	None	-	L3	0.0861472	0.0123214	0.2
	DE-TransE	100	Xavier Normal	0.10	TRUE	TRUE	0.0021739	CE	None	-	None	-	-	0.4
	DE-SimplE	256	Xavier Normal	0.66	TRUE	TRUE	0.0021578	BCE	None	-	None	-	-	0.4
	TNTComplEx	1560	-	-	True	-	0.1499999	CE	n3	0.001	-	-	0.1	-
	ATiSE	512	-	-	True	-	0.00003	LMR	-	-	-	-	-	-
	HyTE	512	Xavier Normal	-	TRUE	TRUE	0.0007655	CE	L3	0.0128709	L2	0.0440400	0.0755946	0.2

Chapter 6

DyERNIE: Dynamic Evolution of Riemannian Manifold Embeddings for Temporal Knowledge Graph Completion.

This chapter contains the publication

DyERNIE: Dynamic Evolution of Riemannian Manifold Embeddings for Temporal Knowledge Graph Completion. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Virtual Conference, Nov 2020. DOI: 10.18653/v1/2020.emnlp-main.593.

DyERNIE: Dynamic Evolution of Riemannian Manifold Embeddings for Temporal Knowledge Graph Completion

Zhen Han[1,2], **Peng Chen**[2,3], **Yunpu Ma**[1,2*], **Volker Tresp**[1,2*]
[1]Institute of Informatics, LMU Munich [2] Corporate Technology, Siemens AG
[3]Department of Informatics, Technical University of Munich
zhen.han@campus.lmu.de, peng.chen@tum.de
cognitive.yunpu@gmail.com, volker.tresp@siemens.com

Abstract

There has recently been increasing interest in learning representations of temporal knowledge graphs (KGs), which record the dynamic relationships between entities over time. Temporal KGs often exhibit multiple simultaneous non-Euclidean structures, such as hierarchical and cyclic structures. However, existing embedding approaches for temporal KGs typically learn entity representations and their dynamic evolution in the Euclidean space, which might not capture such intrinsic structures very well. To this end, we propose DyERNIE, a non-Euclidean embedding approach that learns evolving entity representations in a product of Riemannian manifolds, where the composed spaces are estimated from the sectional curvatures of underlying data. Product manifolds enable our approach to better reflect a wide variety of geometric structures on temporal KGs. Besides, to capture the evolutionary dynamics of temporal KGs, we let the entity representations evolve according to a velocity vector defined in the tangent space at each timestamp. We analyze in detail the contribution of geometric spaces to representation learning of temporal KGs and evaluate our model on temporal knowledge graph completion tasks. Extensive experiments on three real-world datasets demonstrate significantly improved performance, indicating that the dynamics of multi-relational graph data can be more properly modeled by the evolution of embeddings on Riemannian manifolds.

1 Introduction

Learning from relational data has long been considered as a key challenge in artificial intelligence. In recent years, several sizable knowledge graphs (KGs), e.g. Freebase (Bollacker et al., 2008) and Wikidata (Vrandečić and Krötzsch, 2014), have been developed that provide widespread availability of such data and enabled improvements to a plethora of downstream applications such as recommender systems (Hildebrandt et al., 2019) and question answering (Zhang et al., 2018). KGs are multi-relational, directed graphs with labeled edges, where each edge corresponds to a fact and can be represented as a triple, such as (John, lives in, Vancouver). Common knowledge graphs are static and store facts at their current state. In reality, however, multi-relational data are often time-dependent. For example, the political relationship between two countries might intensify because of trade fights. Thus, temporal knowledge graphs were introduced, such as ICEWS (Boschee et al., 2015) and GDELT (Leetaru and Schrodt, 2013), that capture temporal aspects of facts in addition to their multi-relational nature. In these datasets, temporal facts are represented as a quadruple by extending the static triplet with a timestamp describing when these facts occurred, i.e. (Barack Obama, inaugurated, as president of the US, 2009). Since real-world temporal KGs are usually incomplete, the task of link prediction on temporal KGs has gained growing interest. The task is to infer missing facts at specific timestamps based on the existing ones by answering queries such as (US, president, ?, 2015).

Many facts in temporal knowledge graphs induce geometric structures over time. For instance, increasing trade exchanges and economic cooperation between two major economies might promote the trade exports and economic growths of a series of countries in the downstream supply chain, which exhibits a tree-like structure over time. Moreover, an establishment of diplomatic relations between two countries might lead to regular official visits between these two countries, which produces a cyclic structure over time. Embedding methods in Euclidean space have limitations and suffer from large distortion when representing large-scale hier-

*Corresponding author.

archical data. Recently, hyperbolic geometry has been exploited in several works (Nickel and Kiela, 2017; Ganea et al., 2018) as an effective method for learning representations of hierarchical data, where the exponential growth of distance on the boundary of the hyperbolic space naturally allows representing hierarchical structures in a compact form. While most graph-structured data has a wide variety of inherent geometric structures, e.g. partially tree-like and partially cyclical, the above studies model the latent structures in a single geometry with a constant curvature, limiting the flexibility of the model to match the hypothetical intrinsic manifold. Thus, using a product of different constant curvature spaces (Gu et al., 2018) might be helpful to match the underlying geometries of temporal knowledge graphs and provide high-quality representations.

Existing non-Euclidean approaches for knowledge graph embeddings (Balazevic et al., 2019; Kolyvakis et al., 2019) lack the ability to capture temporal dynamics available in underlying data represented by temporal KGs. The difficulty with representing the evolution of temporal KGs in non-Euclidean spaces lies in finding a way to integrate temporal information to the geometric representations of entities. In this work, we propose the **dy**namic **e**volution of **R**iemannian ma**ni**fold **e**mbeddings (DyERNIE), a theoretically founded approach to embed multi-relational data with dynamic relationships on a product of Riemannian manifolds with different curvatures. To capture both the stationary and dynamic characteristics of temporal KGs, we characterize the time-dependent representation of an entity as movements on manifolds. For each entity, we define an initial embedding (at t_0) on each manifold and a velocity vector residing in the tangent space of the initial embedding to generate a temporal representation at each timestamp. In particular, the initial embeddings represent the stationary structural dependencies across facts, while the velocity vectors capture the time-varying properties of entities.

Our contributions are the following: (i) We introduce Riemannian manifolds as embedding spaces to capture geometric features of temporal KGs. (ii) We characterize the dynamics of temporal KGs as movements of entity embeddings on Riemannian manifolds guided by velocity vectors defined in the tangent space. (iii) We show how the product space can be approximately identified from sectional curvatures of temporal KGs and how to choose the dimensionality of component spaces as well as their curvatures accordingly. (iv) Our approach significantly outperforms current benchmarks on a link prediction task on temporal KGs in low- and high-dimensional settings. (v) We analyze our model's properties, i.e. the influence of embedding dimensionality and the correlation between node degrees and the norm of velocity vectors.

2 Preliminaries

2.1 Riemannian Manifold

An n-dimensional Riemannian *manifold* $\mathcal{M}^n$ is a real and smooth manifold with locally Euclidean structure. For each point $\mathbf{x} \in \mathcal{M}^n$, the metric tensor $g(\mathbf{x})$ defines a positive-definite inner product $g(\mathbf{x}) = \langle \cdot, \cdot \rangle_{\mathbf{x}} : \mathcal{T}_{\mathbf{x}}\mathcal{M}^n \times \mathcal{T}_{\mathbf{x}}\mathcal{M}^n \rightarrow \mathbb{R}$, where $\mathcal{T}_{\mathbf{x}}\mathcal{M}^n$ is the tangent space of $\mathcal{M}^n$ at $\mathbf{x}$. From the tangent space $\mathcal{T}_{\mathbf{x}}\mathcal{M}^n$, there exists a mapping function $\exp_{\mathbf{x}}(\mathbf{v}) : \mathcal{T}_{\mathbf{x}}\mathcal{M}^n \rightarrow \mathcal{M}^n$ that maps a tangent vector $\mathbf{v}$ at $\mathbf{x}$ to the manifold, also known as the *exponential map*. The inverse of an exponential map is referred to as the *logarithm map* $\log_{\mathbf{x}}(\cdot)$.

2.2 Constant Curvature Spaces

The sectional curvature $K(\tau_{\mathbf{x}})$ is a fine-grained notion defined over a two-dimensional subspace $\tau_{\mathbf{x}}$ in the tangent space at the point $\mathbf{x}$ (Berger, 2012). If all the sectional curvatures in a manifold $\mathcal{M}^n$ are equal, the manifold then defined as a space with a constant curvature K. Three different types of constant curvature spaces can be defined depending on the sign of the curvature: a positively curved space, a flat space, and a negatively curved space. There are different models for each constant curvature space. To unify different models, in this work, we choose the stereographically projected hypersphere $\mathbb{S}_K^n$ for positive curvatures ($K > 0$), while for negative curvatures ($K < 0$) we choose the Poincaré ball $\mathbb{P}_K^n$, which is the stereographic projection of the hyperboloid model:

$$\mathcal{M}_K^n = \begin{cases} \mathbb{S}_K^n = \{\mathbf{x} \in \mathbb{R}^n : \langle \mathbf{x}, \mathbf{x} \rangle_2 > -1/K\} \\ \mathbb{E}^n = \mathbb{R}^n, \text{ if } K = 0 \\ \mathbb{P}_K^n = \{\mathbf{x} \in \mathbb{R}^n : \langle \mathbf{x}, \mathbf{x} \rangle_2 < -1/K\} \end{cases}$$

Both of the above spaces $\mathbb{S}_K$ and $\mathbb{P}_K$ are equipped with the Riemannian metric: $g_{\mathbf{x}}^{\mathbb{S}_K} = g_{\mathbf{x}}^{\mathbb{P}_K} = (\lambda_{\mathbf{x}}^K)^2 g^{\mathbb{E}}$, which is conformal to the Euclidean metric $g^{\mathbb{E}}$ with the conformal factor $\lambda_{\mathbf{x}}^K = 2/(1 + K||\mathbf{x}||_2^2)$ (Ganea et al., 2018). As explained in

(Skopek et al., 2019), $\mathbb{S}_K$ and $\mathbb{P}_K$ have a suitable property, namely the distance and the metric tensors of these spaces converge to their Euclidean counterpart as the curvature goes to 0, which makes both spaces suitable for learning sign-agnostic curvatures.

2.3 Gyrovector Spaces

An important analogy to vector spaces (vector addition and scalar multiplication) in non-Euclidean geometry is the notion of gyrovector spaces (Ungar, 2008). Both the projected hypersphere and the Poincaré ball share the following definition of *Möbius addition*:

$$\mathbf{x} \oplus_K \mathbf{y} = \frac{(1 - 2K\langle \mathbf{x}, \mathbf{y}\rangle_2 - K||\mathbf{y}||_2^2)\mathbf{x} + (1 + K||\mathbf{x}||_2^2)\mathbf{y}}{1 - 2K\langle \mathbf{x}, \mathbf{y}\rangle_2 + K^2||\mathbf{x}||_2^2||\mathbf{y}||_2^2}$$

where we denote the Euclidean norm and inner product by $||\cdot||$ and $\langle \cdot, \cdot \rangle_2$, respectively. Skopek et al. (2019) show that the distance between two points in $\mathbb{S}_K$ or $\mathbb{P}_K$ is equivalent to their variants in gyrovector spaces, which is defined as

$$d_{\mathcal{M}_K}(\mathbf{x}, \mathbf{y}) = \frac{2}{\sqrt{|K|}} \tan_K^{-1}(\sqrt{|K|}||-\mathbf{x} \oplus_K \mathbf{y}||_2),$$

where $\tan_K = \tan$ if $K > 0$ and $\tan_K = \tanh$ if $K < 0$. The same gyrovector spaces can be used to define the exponential and logarithmic maps in the Poincaré ball and the projected hypersphere. We list these mapping functions in Table 8 in the appendix. As Ganea et al. (2018) use the exponential and logarithmic maps to obtain the *Möbius matrix-vector multiplication*: $\mathbf{M} \otimes_K \mathbf{x} = \exp_{\mathbf{0}}^K(\mathbf{M} \log_{\mathbf{0}}^K(\mathbf{x}))$, we reuse them in hyperbolic space. This operation is defined similarly in projected hyperspherical space.

2.4 Product Manifold

We further generalize the embedding space of latent representations from a single manifold to a product of Riemannian manifolds with constant curvatures. Consider a sequence of Riemannian manifolds with constant curvatures, the product manifold is defined as the Cartesian product of k component manifolds $\mathcal{M}^n = \times_{i=1}^k \mathcal{M}_{K_i}^{n_i}$, where n_i is the dimensionality of the $i-$th component, and K_i indicates its curvature, with choices $\mathcal{M}_{K_i}^{n_i} \in \{\mathbb{P}_{K_i}^{n_i}, \mathbb{E}^{n_i}, \mathbb{S}_{K_i}^{n_i}\}$. We call $\{(n_i, k_i)\}_{i=1}^k$ the *signature* of a product manifold. Note that the notation $\mathbb{E}^{n_i}$ is redundant in Euclidean spaces since the Cartesian product of Euclidean spaces with different dimensions can be combined into a single space, i.e. $\mathbb{E}^n = \times_{i=1}^k \mathbb{E}^{n_i}$. However, this equality does not hold in the projected hypersphere and the Poincaré ball. For each point $\mathbf{x} \in \mathcal{M}^n$ on a product manifold, we decompose its coordinates into the corresponding coordinates in component manifolds $\mathbf{x} = (\mathbf{x}^{(1)}, ..., \mathbf{x}^{(k)})$, where $\mathbf{x}^{(i)} \in \mathcal{M}_{K_i}^{n_i}$. The distance function decomposes based on its definition $d_{\mathcal{M}^n}^2(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^k d_{\mathcal{M}_{K_i}^{n_i}}^2(\mathbf{x}^{(i)}, \mathbf{y}^{(i)})$. Similarly, we decompose the metric tensor, exponential and logarithmic maps on a product manifold into the component manifolds. In particular, we split the embedding vectors into parts $\mathbf{x}^{(i)}$, apply the desired operation on that part $f_{K_i}^{n_i}(\mathbf{x}^{(i)})$, and concatenate the resulting parts back (Skopek et al., 2019).

2.5 Temporal Knowledge Graph Completion

Temporal knowledge graphs (KGs) are multi-relational, directed graphs with labeled timestamped edges between entities. Let $\mathcal{E}$, $\mathcal{P}$, and $\mathcal{T}$ represent a finite set of entities, predicates, and timestamps, respectively. Each fact can be denoted by a quadruple $q = (e_s, p, e_o, t)$, where $p \in \mathcal{P}$ represents a timestamped and labeled edge between a subject entity $e_s \in \mathcal{E}$ and an object entity $e_o \in \mathcal{E}$ at a timestamp $t \in \mathcal{T}$. Let $\mathcal{F}$ represents the set of all quadruples that are facts, i.e. real events in the world, the temporal knowledge graph completion (tKGC) is the problem of inferring $\mathcal{F}$ based on a set of observed facts $\mathcal{O}$, which is a subset of $\mathcal{F}$. To evaluate the proposed algorithms, the task of tKGC is to predict either a missing subject entity $(?, p, e_o, t)$ given the other three components or a missing object entity $(e_s, p, ?, t)$. Taking the object prediction as an example, we consider all entities in the set $\mathcal{E}$, and learn a score function $\phi : \mathcal{E} \times \mathcal{P} \times \mathcal{E} \times \mathcal{T} \rightarrow \mathbb{R}$. Since the score function assigns a score to each quadruple, the proper object can be inferred by ranking the scores of all quadruples $\{(e_s, p, e_{o_i}, t), e_{o_i} \in \mathcal{E}\}$ that are accompanied with candidate entities.

3 Related work

3.1 Knowledge Graph Embedding

Static KG Embedding Embedding approaches for static KGs can generally be categorized into bilinear models and translational models. The bilinear approaches are equipped with a bilinear score function that represents predicates as linear transformations acting on entity embeddings (Nickel et al., 2011; Trouillon et al., 2016; Yang et al., 2014; Ma et al., 2018a). Translational approaches measure the plausibility of a triple as the distance between the translated subject and object entity embeddings, including TransE (Bordes et al., 2013) and its variations (Sun et al., 2019; Kazemi and Poole, 2018). Additionally, several models are based on deep learning approaches (Dettmers et al., 2018; Schlichtkrull et al., 2018; Hildebrandt et al., 2020) that apply (graph) convolutional layers on top of embeddings and design a score function as the last layer of the neural network.

Temporal KG Embedding Recently, there have been some attempts of incorporating time information in temporal KGs to improve the performance of link prediction. Ma et al. (2018b) developed extensions of static knowledge graph models by adding a timestamp embedding to the score functions. Also, Leblay and Chekol (2018) proposed TTransE by incorporating time representations into the score function of TransE in different ways. HyTE (Dasgupta et al., 2018) embeds time information in the entity-relation space by arranging a temporal hyperplane to each timestamp. The number of parameters of these models scales with the number of timestamps, leading to overfitting when the number of timestamps is extremely large.

3.2 Graph Embedding Approaches in non-Euclidean Geometries

There has been a growing interest in embedding graph data in non-Euclidean spaces. Nickel and Kiela (2017) first applied hyperbolic embedding for link prediction to the lexical database WordNet. Since then, hyperbolic analogs of several other approaches have been developed (De Sa et al., 2018; Tifrea et al., 2018). In particular, Balazevic et al. (2019) proposed a translational model for embedding multi-relational graph data in the hyperbolic space and demonstrated advancements over state-over-the-art. More recently, Gu et al. (2018) generalized manifolds of constant curvature to a product manifold combining hyperbolic, spherical, and Euclidean components. However, these methods consider graph data as static models and lack the ability to capture temporally evolving dynamics.

4 Temporal Knowledge Graph Completion in Riemannian Manifold

Entities in a temporal KG might form different geometric structures under different relations, and these structures could evolve with time. To capture heterogeneous and time-dependent structures, we propose the DyERNIE model to embed entities of temporal knowledge graphs on a product of Riemannian manifolds and model time-dependent behavior of entities with dynamic entity representations.

4.1 Entity Representation

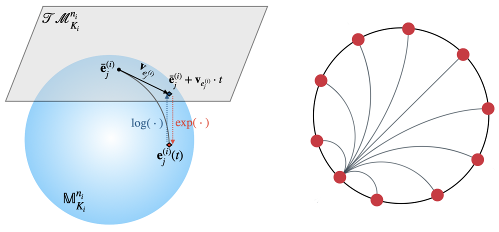
In temporal knowledge graphs, entities might have some features that change over time and some features that remain fixed. Thus, we represent the embedding of an entity $e_j \in \mathcal{E}$ at instance t with a combination of low-dimensional vectors $\mathbf{e}_j(t) = (\mathbf{e}_j^{(1)}(t), ..., \mathbf{e}_j^{(k)}(t))$ with $\mathbf{e}_j^{(i)}(t) \in \mathcal{M}_{K_i}^{n_i}$, where $\mathcal{M}_{K_i}^{n_i} \in \{\mathbb{P}_{K_i}^{n_i}, \mathbb{E}^{n_i}, \mathbb{S}_{K_i}^{n_i}\}$ is the i-th component manifold, K_i and n_i denote the curvature and the dimension of this manifold, respectively. Each component embedding $\mathbf{e}_j^{(i)}(t)$ is derived from an initial embedding and a velocity vector to encode both the stationary properties of the entities and their time-varying behavior, namely

$$\mathbf{e}_j^{(i)}(t) = \exp_{\mathbf{0}}^{K_i}\left(\log_{\mathbf{0}}^{K_i}(\bar{\mathbf{e}}_j^{(i)}) + \mathbf{v}_{e_j^{(i)}} t\right), \quad (1)$$

where $\bar{\mathbf{e}}_j^{(i)} \in \mathcal{M}_{K_i}^{n_i}$ represents the initial embedding that does not change over time. $\mathbf{v}_{e_j^{(i)}} \in \mathcal{T}_{\mathbf{0}}\mathcal{M}_{K_i}^{n_i}$ represents an entity-specific velocity vector that is defined in the tangent space at origin $\mathbf{0}$ and captures evolutionary dynamics of the entity e_j in its vector space representations over time. As shown in Figure 1 (a), we project the initial embedding to the tangent space $\mathcal{T}_{\mathbf{0}}\mathcal{M}_{K_i}^{n_i}$ using the logarithmic map $\log_{\mathbf{0}}^{K_i}$ and then use a velocity vector to obtain the embedding of the next timestamp. Finally, we project it back to the manifold with the exponential map $\exp_{\mathbf{0}}^{K_i}$. Note that in the case of Euclidean space, the exponential map and the logarithmic map are equal to the identity function. By learning both the initial embedding and velocity vector, our model characterizes evolutionary dynamics of entities as movements on manifolds and thus predict unseen entity interactions based on both the stationary and time-varying entity properties.

4.2 Score Function

Bilinear models have been proved to be an effective approach for KG completion (Nickel et al., 2011; Lacroix et al., 2018), where the score function is a bilinear product between subject entity, predicate, and object entity embeddings. However, there is



$\mathcal{T}\mathcal{M}^{n_i}_{K_i}$
$\bar{\mathbf{e}}^{(i)}_j$
$\mathbf{v}_{\bar{e}^{(i)}_j}$
$\bar{\mathbf{e}}^{(i)}_j + \mathbf{v}_{\bar{e}^{(i)}_j} \cdot t$
$\log(\cdot)$
$\exp(\cdot)$
$\mathbf{e}^{(i)}_j(t)$
$\mathbb{M}^{n_i}_{K_i}$

Figure 1: (a) Evolution of an entity embedding on the i-th component manifold (left). For convenience in drawing, the tangent space $\mathcal{T}\mathcal{M}^{n_i}_{K_i}$ is defined at $\bar{\mathbf{e}}^{(i)}_j$. (b) Geodesics in the Poincaré disk (right), where red dots represent nodes on the disk.

no clear correspondence of the Euclidean inner-product in non-Euclidean spaces. We follow the method suggested in Poincaré Glove (Tifrea et al., 2018) to reformulate the inner product as a function of distance, i.e. $\langle \mathbf{x}, \mathbf{y} \rangle = \frac{1}{2}(d(\mathbf{x}, \mathbf{y})^2 + ||x||^2 + ||y||^2)$ and replace squared norms with biases $b_\mathbf{x}$ and $b_\mathbf{y}$. In addition, to capture different hierarchical structures under different relations simultaneously, Balazevic et al. (2019) applied relation-specific transformations to entities, i.e. a stretch by a diagonal predicate matrix $\mathbf{P} \in \mathbb{R}^{n\times n}$ to subject entities and a translation by a vector offset $\mathbf{p} \in \mathbb{P}^n$ to object entities.

Inspired by these two ideas, we define the score function of DyERNIE as

$$\phi(e_s, p, e_o, t) = \sum_{i=1}^{k} -d_{\mathcal{M}^{n_i}_{K_i}} \Big(\mathbf{P}^{(i)} \otimes_{K_i} \mathbf{e}^{(i)}_s(t),$$
$$\mathbf{e}^{(i)}_o(t) \oplus_{K_i} \mathbf{p}^{(i)}\Big)^2 + b^{(i)}_s + b^{(i)}_o$$

where $\mathbf{e}^{(i)}_s(t)$ and $\mathbf{e}^{(i)}_o(t) \in \mathcal{M}^{n_i}_{K_i}$ are embeddings of the subject and object entities e_s and e_o in the i-th component manifold, respectively. $\mathbf{p}^{(i)} \in \mathcal{M}^{n_i}_{K_i}$ is a translation vector of predicate p, and $\mathbf{P}^{(i)} \in \mathbb{R}^{n_i\times n_i}$ represents a diagonal predicate matrix defined in the tangent space at the origin. Since multi-relational data often has different structures under different predicate, we use predicate-specific transformations $\mathbf{P}$ and $\mathbf{p}$ to determine the predicate-adjusted embeddings of entities in different predicate-dependent structures, e.g. multiple hierarchies. The distance between the predicate-adjusted embeddings of e_s and e_o measures the relatedness between them in terms of a predicate p.

4.3 Learning

The genuine quadruples in a temporal KG $\mathcal{G}$ are split into *train*, *validation*, and *test* sets. We add

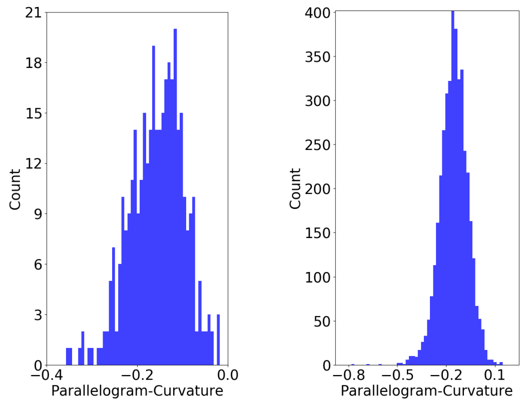


Figure 2: Histogram of sectional curvatures at each timestamps on ICEWS14 (left), and ICEWS05-15 (right).

reciprocal relations for every quadruple, which is a standard data augmentation technique commonly used in literature (Balazevic et al., 2019; Goel et al., 2019), i.e. we add (e_o, p^{-1}, e_s, t) for every (e_s, p, e_o, t). Besides, for each fact (e_s, p, e_o, t) in the training set, we generate n negative samples by corrupting either the object (e_s, p, e'_o, t) or the subject (e_o, p^{-1}, e'_s, t) with a randomly selected entity from $\mathcal{E}$. We use the binary cross-entropy as the loss function, which is defined as

$$\mathcal{L} = \frac{-1}{N}\sum_{m=1}^{N}\left(y_m \log(p_m) + (1 - y_m)\log(1 - p_m)\right),$$

where N is the number of training samples, y_m represents the binary label indicating whether a quadruple q_m is genuine or not, p_m denotes the predicted probability $\sigma(\phi(q_m))$, and $\sigma(\cdot)$ represents the sigmoid function. Model parameters are learned using *Riemannian stochastic gradient descent* (RSGD) (Bonnabel, 2013), where the Riemannian gradient $\Delta_{\mathcal{M}^n} L$ is obtained by multiplying the Euclidean gradient $\Delta_{\mathbb{E}}$ with the inverse of the Riemannian metric tensor.

4.4 Signature Estimation

To better capture a broad range of structures in temporal KGs, we need to choose an appropriate signature of a product manifold $\mathcal{M}^n$, including the number of component spaces, their dimensions, and curvatures. Although we can simultaneously learn embeddings and the curvature of each component during training using gradient-based optimization, we have empirically found that treating curvature as a trainable parameter interferes with the training of other model parameters. Thus, we treat the curvature of each component and the dimension as hyperparameters selected *a priori*. In particular, we use the parallelogram law' deviation (Gu et al., 2018) to estimate both the graph curvature of a given temporal KG and the number of components. Details about this algorithm can be found in Appendix A. Figure 2 shows the curvature histograms on the ICEWS14 and ICEWS05-15 datasets introduced in Section 5.1. It can be noticed that curvatures are mostly non-Euclidean, offering a good motivation to learn embeddings on a product manifold. Taking the ICEWS05-15 dataset as an example, we see that most curvatures are negative. In this case, we can initialize the product manifold consisting of three hyperbolic components with different dimensions. Then we conduct a Bayesian optimization around the initial value of the dimension and the curvature of each component to fine-tune them. Finally, we select the best-performing signature according to performance on the validation set as the final choice.

5 Experiments

5.1 Experimental Set-up

Datasets Global Database of Events, Language, and Tone (GDELT) (Leetaru and Schrodt, 2013) dataset and Integrated Crisis Early Warning System (ICEWS) (Boschee et al., 2015) dataset have established themselves in the research community as representative samples of temporal KGs. The GDELT dataset is derived from an initiative database of all the events across the globe connecting people, organizations, and news sources. We use a subset extracted by Jin et al. (2019), which contains events occurring from 2018-01-01 to 2018-01-31. The ICEWS dataset contains information about political events with specific time annotations, e.g. (Barack Obama, visit, India, 2010-11-06). We apply our model on two subsets of the ICEWS dataset generated by García-Durán et al. (2018): ICEWS14 contains events in 2014, and ICEWS05-15 corresponds to the facts between 2005 to 2015. We compare our approach and baseline methods by performing the link prediction task on the GDELT, ICEWS14 and ICEWS05-15 datasets. The statistics of the datasets are provided in Appendix C.

Baselines Our baselines include both static and temporal KG embedding models. From the static KG embedding models, we use TransE (Bordes et al., 2013), DistMult (Yang et al., 2014), and ComplEx (Trouillon et al., 2016) where we compress temporal knowledge graphs into a static, cumulative graph by ignoring the time information. From the temporal KG embedding models, we compare the performance of our model with several state-of-the-art methods, including TTransE (Leblay and Chekol, 2018), TDistMult/TComplEx (Ma et al., 2018b), and HyTE (Dasgupta et al., 2018).

Evaluation protocol For each quadruple $q = (e_s, p, e_o, t)$ in the test set $\mathcal{G}_{test}$, we create two queries: $(e_s, p, ?, t)$ and $(e_o, p^{-1}, ?, t)$. For each query, the model ranks all possible entities $\mathcal{E}$ according to their scores. Following the commonly filtered setting in the literature (Bordes et al., 2013), we remove all entity candidates that correspond to true triples[1] from the candidate list apart from the current test entity. Let ψ_{e_s} and ψ_{e_o} represent the rank for e_s and e_o of the two queries respectively, we evaluate our models using standard metrics across the link prediction literature: *mean reciprocal rank (MRR)*: $\frac{1}{2\cdot|\mathcal{G}_{test}|}\sum_{q\in\mathcal{G}_{test}}(\frac{1}{\psi_{e_s}}+\frac{1}{\psi_{e_o}})$ and *Hits*$@k(k \in \{1,3,10\})$: the percentage of times that the true entity candidate appears in the top k of ranked candidates.

Implementations We implemented our model and all baselines in PyTorch (Paszke et al., 2019). For fairness of comparison, we use Table 2 in supplementary materials to compute the embedding dimension for each (baseline, dataset) pair that matches the number of parameters of our model with an embedding dimension of 100. Taking HyTE as an example, its embedding dimension is 193 and 151 on the ICEWS14 and GDELT dataset, respectively. Also, we use the datasets augmented with reciprocal relations to train all baseline models. We tune hyperparameters of our models using the quasi-random search followed by Bayesian optimization (Ruffinelli et al., 2020) and report the best configuration in Appendix E. We implement TTransE, TComplEx, and TDistMult based on the implementation of TransE, Distmult, and ComplEx respectively. We use the binary cross-entropy loss and RSGD to train these baselines and optimize hyperparameters by early stopping according to MRR on the validation set. Additionally, we use the implementation of HyTE[2]. We provide the detailed

[1]The triplets that appear either in the train, validation, or test set.

[2]https://github.com/malllabiisc/HyTE

Table 1: Link prediction results: MRR (%) and Hits@1/3/10 (%). The best results among all models are in bold. Additionally, we underline the best results among models with the same embedding dimension.

Datasets		**ICEWS14 - filtered**					**ICEWS05-15 - filtered**					**GDELT - filtered**				
Rank (n)	Model	Manifold	MRR	Hits@1	Hits@3	Hits@10	Manifold	MRR	Hits@1	Hits@3	Hits@10	Manifold	MRR	Hits@1	Hits@3	Hits@10
	TransE		30.0	14.8	42.7	60.1		30.4	13.3	42.4	61.1		17.7	7.9	22.9	36.8
100	DistMult	$\mathbb{E}$	57.5	46.9	64.2	77.9	$\mathbb{E}$	47.1	33.6	55.1	72.5	$\mathbb{E}$	22.6	13.9	26.1	39.2
	ComplEx		49.3	36.6	56.2	74.2		39.0	22.9	49.2	68.4		18.8	10.5	22.2	34.9
	TTransE		34.4	25.7	38.3	51.3		35.6	15.4	51.1	67.6		18.2	0.0	30.7	46.2
	TDistMult		33.1	25.4	36.2	47.8		49.8	41.1	54.3	66.4		28.3	16.2	30.7	47.1
100	TComplEx	$\mathbb{E}$	31.8	12.9	45.7	63.0	$\mathbb{E}$	45.1	36.3	49.2	62.0	$\mathbb{E}$	30.6	21.0	34.7	48.1
	HyTE		33.1	6.8	54.5	73.6		38.1	7.6	65.0	80.4		22.4	0.0	39.5	54.2
	DyERNIE-Prod	$\mathbb{P}^3$	46.2	36.0	51.1	66.3	$\mathbb{P}^3$	58.9	50.5	63.2	75.1	$\mathbb{S}^2$	36.3	29.4	38.3	49.5
10	DyERNIE-Sgl	$\mathbb{P}$	43.3	33.3	47.6	62.9	$\mathbb{P}$	58.0	49.2	62.8	74.5	$\mathbb{S}$	35.7	28.7	37.7	48.9
	DyERNIE-Euclid	$\mathbb{E}$	39.8	30.6	43.6	58.2	$\mathbb{E}$	51.9	43.4	56.1	67.9	$\mathbb{E}$	30.2	23.8	31.8	42.5
	DyERNIE-Prod	$\mathbb{P}^3$	53.9	44.2	58.9	72.7	$\mathbb{P}^3$	64.2	56.5	68.2	79.0	$\mathbb{S}^2$	40.0	33.2	42.0	53.1
20	DyERNIE-Sgl	$\mathbb{P}$	51.3	41.4	56.1	70.3	$\mathbb{P}$	63.8	55.9	67.9	78.7	$\mathbb{S}$	39.2	32.6	41.1	52.1
	DyERNIE-Euclid	$\mathbb{E}$	47.7	38.3	52.0	66.2	$\mathbb{E}$	57.3	49.4	61.1	72.4	$\mathbb{E}$	32.9	26.2	34.7	45.7
	DyERNIE-Prod	$\mathbb{P}^3$	58.8	49.8	63.8	76.1	$\mathbb{P}^3$	68.9	61.8	72.8	82.5	$\mathbb{S}^2$	43.0	36.3	45.1	56.0
40	DyERNIE-Sgl	$\mathbb{P}$	56.6	47.3	61.3	74.6	$\mathbb{P}$	67.3	60.2	71.1	81.1	$\mathbb{S}$	42.5	35.8	44.6	55.6
	DyERNIE-Euclid	$\mathbb{E}$	53.7	44.2	58.6	71.9	$\mathbb{E}$	60.3	52.7	64.1	74.7	$\mathbb{E}$	38.4	31.8	40.4	51.1
	DyERNIE-Prod	$\mathbb{P}^3$	**66.9**	**59.9**	**71.4**	**79.7**	$\mathbb{P}^3$	**73.9**	**67.9**	**77.3**	**85.5**	$\mathbb{S}^2$	**45.7**	**39.0**	**47.9**	**58.9**
100	DyERNIE-Sgl	$\mathbb{P}$	65.7	58.2	70.2	79.4	$\mathbb{P}$	71.2	64.8	74.6	83.4	$\mathbb{S}$	45.4	38.6	47.6	58.4
	DyERNIE-Euclid	$\mathbb{E}$	63.3	54.9	67.9	79.2	$\mathbb{E}$	66.2	59.0	69.9	79.8	$\mathbb{E}$	42.6	36.1	44.5	55.1

settings of hyperparameters of each baseline model in Appendix B.

5.2 Comparative Study

Table 2: Filtered MRR for different choices of the distance function with $K = -1$ and $n = 40$ on ICEWS14.

Distance function	MRR
$d(\mathbf{P} \otimes \mathbf{e}_s(t), \mathbf{e}_o(t) \oplus \mathbf{p})$	**55.87**
$\cosh(d(\mathbf{P} \otimes \mathbf{e}_s(t), \mathbf{e}_o(t) \oplus \mathbf{p}))$	54.00
$d(\mathbf{P} \otimes \mathbf{e}_s(t), \mathbf{P} \otimes \mathbf{e}_o(t))$	52.23
$d(\mathbf{P} \otimes \mathbf{e}_s(t), \mathbf{P} \otimes \mathbf{e}_o(t) \oplus \mathbf{p})$	54.55
$d(\mathbf{P} \otimes \mathbf{e}_s(t), \mathbf{e}_o(t))$	47.24
$d(\mathbf{e}_s(t), \mathbf{e}_o(t) \oplus \mathbf{p})$	51.36

Model variants To compare the performance of non-Euclidean embeddings with their Euclidean counterparts, we implement the Euclidean version of Equation 4.2 with $d_{\mathcal{M}}(\mathbf{x}, \mathbf{y}) = d_{\mathbb{E}}(\mathbf{x}, \mathbf{y})$. We refer to it as DyERNIE-Euclid. Besides, we train our model with a single non-Euclidean component to compare embeddings in a product space and in a manifold with a constant curvature. We refer to them as DyERNIE-Prod and DyERNIE-Sgl, respectively. For DyERNIE-Prod, we generate model configurations with different manifold combinations, i.e. $\mathbb{P} \times \mathbb{S} \times \mathbb{E}, \mathbb{P}^3$. Details about the search space are relegated to Appendix E.

Link prediction results We compare the baselines with three variants of our model: DyERNIE-Prod, DyERNIE-Sgl, and DyERNIE-Euclid. We report the best results on the test set among all model configurations in Table 1. Note that the number of parameters of all baselines matches our model's with an embedding dimension of 100. Thus, we see that both DyERNIE-Prod and DyERNIE-Sgl significantly outperform the baselines and DyERNIE-Euclid on all three datasets with the same number of parameters. Even at a low embedding dimension ($n = 10$), our models still have competitive performance, demonstrating the merits of time-dependent non-Euclidean embeddings. Besides, DyERNIE-Prod generally performs better than DyERNIE-Sgl on all three datasets. On the ICEWS14 and ICEWS05-15 datasets, we can observe that the best performing configuration of DyERNIE-Prod at each dimensionality only contains hyperbolic component manifolds. This observation confirms the curvature estimation shown in Figure 2, where most sectional curvatures on the ICEWS14 and ICEWS05-15 datasets are negative.

Table 3: Filtered MRR for different choices of entity representations with $K = -1$ and $n = 40$ on ICEWS14, where $\mathbf{A}_i$ and $\mathbf{w}_i$ represent the amplitude vector and the frequency vector, respectively. ϕ_i denotes the phase shift.

Entity Representations	MRR
$\exp(\log(\bar{\mathbf{e}}_i) + \mathbf{v}_i t)$	**55.87**
$\exp(\log(\bar{\mathbf{e}}_i) + \mathbf{A}_i \sin(\mathbf{w}_i t + \phi_i))$	52.50
$\exp(\log(\bar{\mathbf{e}}_i) + \mathbf{v}_i t + \mathbf{A}_i \sin(\mathbf{w}_i t + \phi_i))$	53.52

Ablation study We show an ablation study of the distance function and the entity representations in Table 2 and 3, respectively. For the distance

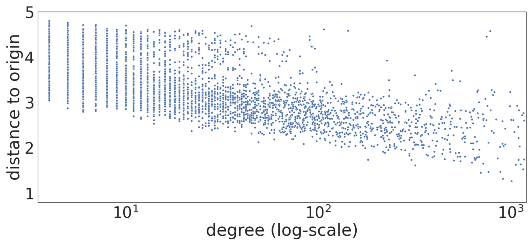


Figure 3: Scatter plot of distances between entity embeddings and the manifold's origin v.s. node degrees on ICEWS05-15. Each point denotes an entity e_j. The x-coordinate gives its degree accumulated over all timestamps, and the y-coordinate represents $d_{\mathcal{M}}(\mathbf{e}_j, \mathbf{0})$.

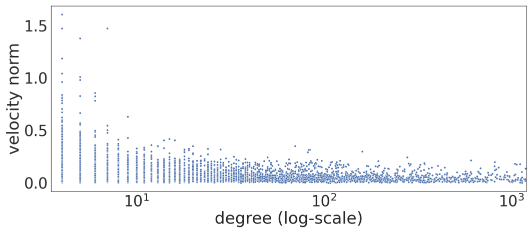


Figure 4: Scatter plot of velocity norms v.s. node degrees on ICEWS05-15. Each point denotes an entity.

function, we use $\mathbf{p}$ and $\mathbf{P}$ to get predicate-adjusted subject and object embeddings and compute the distance between them. We found that any change to distance function causes performance degradation. Especially, removing the translation vector $\mathbf{p}$ most strongly decrease the performance. For the entity representation function, we measure the importance of a linear trend component and a non-linear periodic component. We attempt adding trigonometric functions into entity representations since a combination of trigonometric functions can capture more complicated non-linear dynamics (Rahimi and Recht, 2008). However, experimental results in Table 3 show that using only a linear transformation works the best, which indicates that finding the correct manifold of embedding space is more important than designing complicated non-linear evolution functions of entity embeddings. Additionally, we found the performance degrades significantly if removing the dynamic part of the entity embeddings. For example, on the ICEWS0515 dataset, the Hits@1 metric in the static case is only about half of that in the dynamic case, clearly showing the gain from the dynamism. Details of this ablation study are provided in Appendix G.

Intrinsic hierarchical structures of temporal KGs To illustrate geometric, especially the hierarchical, structures of temporal KGs, we focus on the Poincaré ball model with a dimension of 20 and plot the geodesic distance $d_{\mathcal{M}}(\cdot, \mathbf{0})$ of learned entity embeddings to the origin of the Poincaré ball versus the degree of each entity in Figure 3. Note that the distance is averaged over all timestamps since entity embeddings are time-dependent. We observe that entities with high degrees, which means they got involved in lots of facts, are generally located close to the origin. This makes sense because these entities often lie in the top hierarchical levels. And thus, they should stand close to the root. Under the same settings, we plot the velocity norm of each entity versus the entity degree in Figure 4. Similarly, we see that entities with high degrees have a small velocity norm to stay near the origin of the manifold.

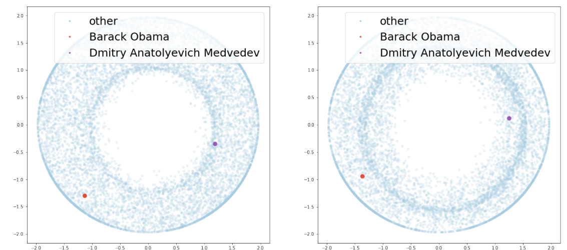


Figure 5: Learned two-dimensional hyperbolic entity embeddings of ICEWS05-15 on the first timestamp 2005-01-01 (left) and the last timestamp 2015-12-31 (right).

Relative movements between a node pair Figure 5 shows two-dimensional hyperbolic entity embeddings of the ICEWS05-15 dataset on two timestamps, 2005-01-01 and 2015-12-31. Specifically, we highlight a former US president (in orange) and a former prime minister of Russia (in purple). We found that the interaction between these two entities decreased between 2005 and 2015, as shown in Figure 9 in the appendix. Accordingly, we observe that the embeddings of these two entities were moving away from each other. More examples of learned embeddings are relegated to Appendix F.

6 Conclusion

In this paper, we propose an embedding approach for temporal knowledge graphs on a product of Riemannian manifolds with heterogeneous curvatures. To capture the temporal evolution of temporal KGs, we use velocity vectors defined in tangent spaces to learn time-dependent entity representations. We show that our model significantly outperforms its Euclidean counterpart and other state-of-the-art approaches on three benchmark datasets of temporal KGs, which demonstrates the significance of geometrical spaces for the temporal knowledge graph completion task.

Acknowledgement

The authors acknowledge support by the German Federal Ministry for Education and Research (BMBF), funding project MLWin (grant 01IS18050).

References

Gregor Bachmann, Gary Bécigneul, and Octavian-Eugen Ganea. 2019. Constant curvature graph convolutional networks. *arXiv preprint arXiv:1911.05076*.

Ivana Balazevic, Carl Allen, and Timothy Hospedales. 2019. Multi-relational poincaré graph embeddings. In *Advances in Neural Information Processing Systems*, pages 4465–4475.

Marcel Berger. 2012. *A panoramic view of Riemannian geometry*. Springer Science & Business Media.

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

Silvere Bonnabel. 2013. Stochastic gradient descent on riemannian manifolds. *IEEE Transactions on Automatic Control*, 58(9):2217–2229.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Advances in neural information processing systems*, pages 2787–2795.

Elizabeth Boschee, Jennifer Lautenschlager, Sean O'Brien, Steve Shellman, James Starz, and Michael Ward. 2015. Icews coded event data. *Harvard Dataverse*, 12.

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

Christopher De Sa, Albert Gu, Christopher Ré, and Frederic Sala. 2018. Representation tradeoffs for hyperbolic embeddings. *Proceedings of machine learning research*, 80:4460.

Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. 2018. Convolutional 2d knowledge graph embeddings. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Octavian Ganea, Gary Bécigneul, and Thomas Hofmann. 2018. Hyperbolic neural networks. In *Advances in neural information processing systems*, pages 5345–5355.

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

Rishab Goel, Seyed Mehran Kazemi, Marcus Brubaker, and Pascal Poupart. 2019. Diachronic embedding for temporal knowledge graph completion. *arXiv preprint arXiv:1907.03143*.

Albert Gu, Frederic Sala, Beliz Gunel, and Christopher Ré. 2018. Learning mixed-curvature representations in product spaces.

Marcel Hildebrandt, Jorge Andres Quintero Serna, Yunpu Ma, Martin Ringsquandl, Mitchell Joblin, and Volker Tresp. 2020. Reasoning on knowledge graphs with debate dynamics. *arXiv preprint arXiv:2001.00461*.

Marcel Hildebrandt, Swathi Shyam Sunder, Serghei Mogoreanu, Mitchell Joblin, Akhil Mehta, Ingo Thon, and Volker Tresp. 2019. A recommender system for complex real-world applications with nonlinear dependencies and knowledge graph context. In *European Semantic Web Conference*, pages 179–193. Springer.

Woojeong Jin, Changlin Zhang, Pedro Szekely, and Xiang Ren. 2019. Recurrent event network for reasoning over temporal knowledge graphs. *arXiv preprint arXiv:1904.05530*.

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

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Prodromos Kolyvakis, Alexandros Kalousis, and Dimitris Kiritsis. 2019. Hyperkg: Hyperbolic knowledge graph embeddings for knowledge base completion. *arXiv preprint arXiv:1908.04895*.

Timothée Lacroix, Nicolas Usunier, and Guillaume Obozinski. 2018. Canonical tensor decomposition for knowledge base completion. *arXiv preprint arXiv:1806.07297*.

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

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

Yunpu Ma, Marcel Hildebrandt, Volker Tresp, and Stephan Baier. 2018a. Holistic representations for memorization and inference. In *UAI*, pages 403–413.

Yunpu Ma, Volker Tresp, and Erik A Daxberger. 2018b. Embedding models for episodic knowledge graphs. *Journal of Web Semantics*, page 100490.

Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. 2011. A three-way model for collective learning on multi-relational data. In *Icml*, volume 11, pages 809–816.

Maximillian Nickel and Douwe Kiela. 2017. Poincaré embeddings for learning hierarchical representations. In *Advances in neural information processing systems*, pages 6338–6347.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, pages 8024–8035.

Ali Rahimi and Benjamin Recht. 2008. Random features for large-scale kernel machines. In *Advances in neural information processing systems*, pages 1177–1184.

Daniel Ruffinelli, Samuel Broscheit, and Rainer Gemulla. 2020. You {can} teach an old dog new tricks! on training knowledge graph embeddings. In *International Conference on Learning Representations*.

Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne Van Den Berg, Ivan Titov, and Max Welling. 2018. Modeling relational data with graph convolutional networks. In *European Semantic Web Conference*, pages 593–607. Springer.

Ondrej Skopek, Octavian-Eugen Ganea, and Gary Bécigneul. 2019. Mixed-curvature variational autoencoders. *arXiv preprint arXiv:1911.08411*.

Zhiqing Sun, Zhi-Hong Deng, Jian-Yun Nie, and Jian Tang. 2019. Rotate: Knowledge graph embedding by relational rotation in complex space. *arXiv preprint arXiv:1902.10197*.

Alexandru Tifrea, Gary Bécigneul, and Octavian-Eugen Ganea. 2018. Poincar\'e glove: Hyperbolic word embeddings. *arXiv preprint arXiv:1810.06546*.

Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. 2016. Complex embeddings for simple link prediction. International Conference on Machine Learning (ICML).

Abraham Albert Ungar. 2008. A gyrovector space approach to hyperbolic geometry. *Synthesis Lectures on Mathematics and Statistics*, 1(1):1–194.

Denny Vrandečić and Markus Krötzsch. 2014. Wikidata: a free collaborative knowledgebase. *Communications of the ACM*, 57(10):78–85.

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

Yuyu Zhang, Hanjun Dai, Zornitsa Kozareva, Alexander J Smola, and Le Song. 2018. Variational reasoning for question answering with knowledge graph. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Appendices

A Graph Curvature Estimation Algorithm

We use Algorithm 1 to estimate the sectional curvatures of a dataset developed by Bachmann et al. (2019).

B Implementation Details of Baselines

Note that the embedding dimension for each (baseline, dataset) pair matches the number of parameters of our models with an embedding dimension of 100. We use Table 4 and 12 to compute the rank for each (baselines, dataset) pair. Besides, for fairness of results, we use the datasets augmented with reciprocal relations to train all baseline models.

Static knowledge graph embedding models We use TransE (Bordes et al., 2013), DistMult (Yang et al., 2014), and ComplEx (Trouillon et al., 2016) as static baselines, where we compress temporal knowledge graphs into a static, cumulative graph by ignoring the time information. We use the cross-entropy loss and Adam optimizer with a batch size of 128 to train the static baselines. Besides, we use uniform sampling to initialize the embeddings of entities and predicates. Other hyperparameters of the above baselines are shown in Table 5.

Temporal knowledge graph embedding models We compare our model's performance with several state-of-the-art temporal knowledge graph embedding methods, including TTransE (Leblay and Chekol, 2018), TDistMult/TComplEx (Ma et al.,

Table 4: Number of parameters for each model considered when using reciprocal relations: d represent the dimension of embeddings.

Model	# Parameters
ComplEx	$(2\|\mathcal{E}\| + 4\|\mathcal{P}\|) \cdot d$
TransE	$(\|\mathcal{E}\| + 2\|\mathcal{P}\|) \cdot d$
DistMult	$(\|\mathcal{E}\| + 2\|\mathcal{P}\|) \cdot d$
TComplEx	$(2\|\mathcal{E}\| + 4\|\mathcal{P}\| + 2\|\mathcal{T}\|) \cdot d$
TTransE	$(\|\mathcal{E}\| + 2\|\mathcal{P}\| + \|\mathcal{T}\|) \cdot d$
TDistMult	$(\|\mathcal{E}\| + 2\|\mathcal{P}\| + \|\mathcal{T}\|) \cdot d$
HyTE	$(\|\mathcal{E}\| + 2\|\mathcal{P}\| + \|\mathcal{T}\|) \cdot d$
DyERNIE	$2(\|\mathcal{E}\| + 2\|\mathcal{P}\|) \cdot d + 2\|\mathcal{E}\|$

2018b), and HyTE (Dasgupta et al., 2018). We use the ADAM optimizer (Kingma and Ba, 2014) and the cross-entropy loss to train the temporal KG models. We set learning rate = 0.001, negative samples pro fact = 500, number of epochs = 500 , batch size = 256, and validate them every 50 epochs to select the model giving the best validation MRR. For the GDELT dataset, we use a similar setting but with negative samples pro fact = 50 due to the large size of the dataset. The embedding dimensions of the above dynamic baselines on each dataset are shown in Table 6.

Table 5: Hyperparameter settings of static baselines.

Model	TransE	DistMult	ComplEx
Embedding dimension			
ICEWS14	202	202	101
ICEWS05-15	202	202	101
GDELT	202	202	101
Negative Sampling	253	657	1529
Learning rate	3e-4	0.16	0.18

C Datasets

Dataset statistics are described in Table 12. Since the timestamps in the ICEWS dataset are dates rather than numbers, we sort them chronologically and encode them into consecutive numbers.

Table 6: Embedding dimensions of dynamic baselines.

Model	TTransE	TDistMult	TComplEx	HyTE
Embedding dimension				
ICEWS14	193	193	96	193
ICEWS05-15	148	148	74	148
GDELT	151	151	76	151

D Evaluation metrics

Let ψ_{e_s} and ψ_{e_o} represent the rank for e_s and e_o of the two queries, respectively. We evaluate our models using standard metrics across the link prediction literature: *mean reciprocal rank (MRR)*: $\frac{1}{2\cdot|\mathcal{G}_{test}|}\sum_{q\in\mathcal{G}_{test}}(\frac{1}{\psi_{e_s}} + \frac{1}{\psi_{e_o}})$ and *Hits@k*$(k \in \{1, 3, 10\})$: the percentage of times that the true entity candidate appears in the top k of ranked candidates.

E Implementation Details of DyERNIE

Signature search On the ICEWS subsets, we try all manifold combinations with the number of components of $\{1, 2, 3\}$. Due to the large size of data samples on the GDELT dataset, we only try manifold combinations with the number of components of $\{1, 2\}$. Specifically, the candidates are $\{\mathbb{P}^n, \mathbb{S}^n, \mathbb{E}^n\}$ for single manifolds, $\{\mathbb{P}^{n_i} \times \mathbb{S}^{n_i}, \mathbb{P}^{n_i} \times \mathbb{P}^{n_i}, \mathbb{S}^{n_i} \times \mathbb{S}^{n_i}, \mathbb{P}^{n_i} \times \mathbb{E}^{n_i}, \mathbb{S}^{n_i} \times \mathbb{E}^{n_i}\}$ for a product of two component manifolds, and $\{\mathbb{P}^{n_i} \times \mathbb{P}^{n_i} \times \mathbb{P}^{n_i}, \mathbb{P}^{n_i} \times \mathbb{S}^{n_i} \times \mathbb{E}^{n_i}, \mathbb{S}^{n_i} \times \mathbb{S}^{n_i} \times \mathbb{S}^{n_i}, \mathbb{P}^{n_i}, \times \mathbb{P}^{n_i} \times \mathbb{S}^{n_i}, \mathbb{P}^{n_i} \times \mathbb{S}^{n_i} \times \mathbb{S}^{n_i}, \mathbb{P}^{n_i} \times \mathbb{P}^{n_i} \times \mathbb{E}^{n_i}, \mathbb{S}^{n_i} \times \mathbb{S}^{n_i} \times \mathbb{E}^{n_i}\}$ for a product of three component manifold. For each combination, we use the Ax-framework[3] to optimize the assignment of dimensions to each component manifold and the curvatures. The assignment of the best-performing models are shown in Table 9, 10, and 11. We report the best results on each dataset in Table 1 in the main body.

Hyperparameter configurations for best-performing models We select the loss function from binary cross-entropy (BCE), margin ranking loss, and cross-entropy (CE). BCE and CE give a similar performance and outperform the margin ranking loss. However, when using the BCE loss, we could use a large learning rate ($lr > 10$) to speed up the training procedure. In contrast, models with the CE loss incline overfitting by large learning rates. Given the BCE loss, we found the learning rate of 50 works the best for all model configurations. Furthermore, increasing negative samples can improve the performance to some extent, while this impact is weakening gradually as the number of negative samples become larger. However, the number of negative samples largely affect the runtime of the training procedure. We empirically found that the negative sample number of 50 is a good compromise

[3]https://ax.dev

between the model performance and the training speed. Besides, there is no statistically significant difference in the model performance when using different optimizers, such as Riemannian Adam (RADAM) and Riemannian stochastic gradient descent (RSGD). Thus, for the model's simplicity, we decide to use RSGD.

Average runtime for each approach & Number of parameters in each model Table 13 shows the number of parameters and the average runtime for each model.

F Visualization

We plot the geodesic distance $d_{\mathcal{M}}(\mathbf{e}_j, \mathbf{0})$ of learned entity embeddings with a dimension of 20 to the manifold's origin versus the degree of each entity in Figure 6, where $d_{\mathcal{M}}(\mathbf{e}_j, \mathbf{0})$ is averaged over all timestamps since $\mathbf{e}_j$ is time-dependent. Also, the degree of each entity is accumulated over all timestamps. Each point in the upper plot represents an entity where the x-coordinate gives their degree, and the y-coordinate gives their average distance to the origin. The plot clearly shows the tendency that entities with high degrees are more likely to lie close to the origin. The bottom plot shows the same content but with a sampling of 20% points. The gray bar around each point shows the variance of the distance between the entity embedding and the origin over time.

Figure 7 shows two-dimensional hyperbolic entity embeddings of the ICEWS05-15 dataset on four timestamps. We highlight some entities to show the relative movements between them. The number of interactions between the selected entities are depicted in Figure 8 and 9, which evolves over time. Specifically, we highlight Nigerian citizens, the Nigerian government, head of the Nigerian government, other authorities in Nigeria, and Nigerian minister in the first row of subplots. Furthermore, we show the relative movements between the entity embeddings of Barack Obama, Xi Jinping, and Dmitry Anatolyevich Medvedev in the second row of subplots. We can see that two entities are getting closer in the Poincare disc if the number of interactions between them increases.

G Additional Ablation Study

To assess the contribution of the dynamic part of entity embeddings, we remove the dynamic part and run the model variant on static knowledge graphs. Specifically, we compress ICEWS05-15 into a static, cumulative graph by ignoring the time information. As shown in Table 7, the performance degrades significantly if the entity embeddings only have the static part. For example, on the ICEWS0515 dataset, the Hits@1 metric of DyERNIE-Sgl in the static case is less than half of that in the dynamic case, clearly showing the gain from the dynamism.

Table 7: Filtered MRR for dynamic/static entity representations with $dim = 20$ on ICEWS05-15. Note that we run the static model variant on static ICEWS05-15.

Entity Representations	MRR	Hits@1	Hits@3	Hits@10
With dynamic part	63.8	55.9	67.9	78.7
Without dynamic part	38.6	28.3	42.8	59.2

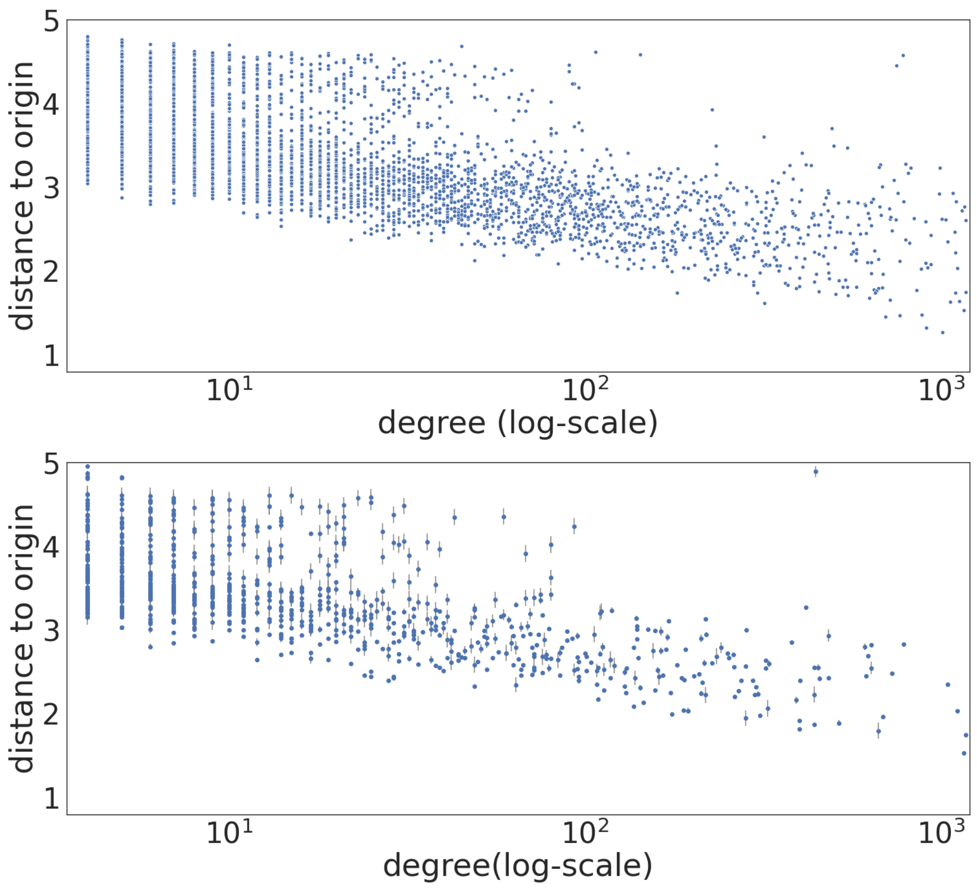
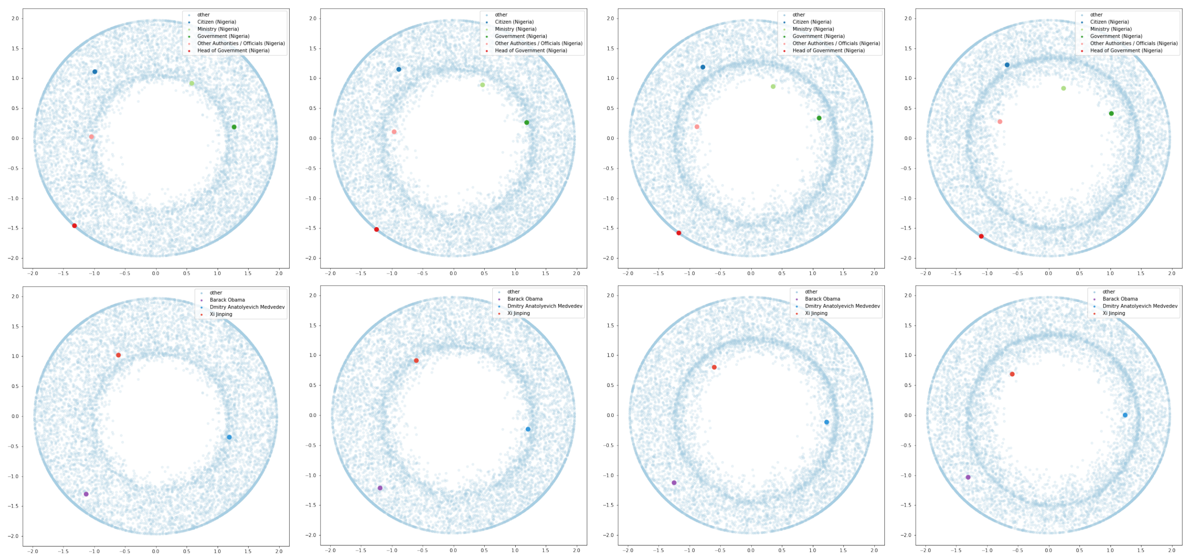


Figure 6: Each point in the upper plot represents an entity whose x-coordinate gives their degree accumulated over all timestamps and y-coordinate gives their distance to the origin averaged over all timestamps. The plot clearly shows the tendency that entities with high degrees are more likely to lie close to the origin. The bottom plot shows the same content but with a sampling of 20% points. The gray bar around each point shows the variance of the distance over all timestamps.



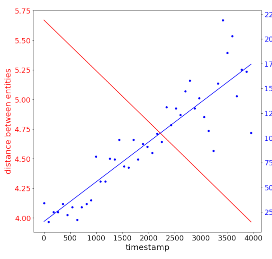
(a) The first timestamp (2005-01-01)

(b) the 1000th timestamp (2007-09-28)

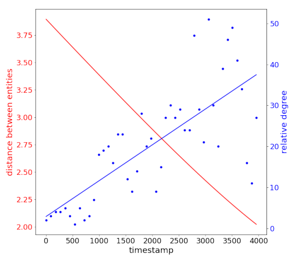
(c) the 2000th timestamp (2010-06-24)

(d) the 3000th timestamp (2013-03-20)

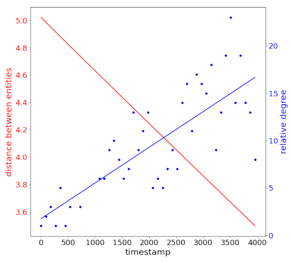
Figure 7: Evolution of entity embeddings over time. We highlight Nigerian citizens, the Nigerian government, the head of Nigerian government, other authorities in Nigeria, and Nigerian minister in the first row; and Barack Obama, Xi Jinping, and Dmitry Anatolyevich Medvedev in the second row.



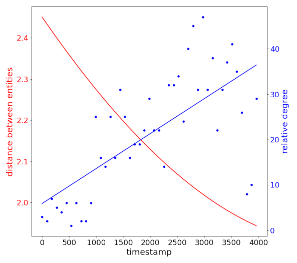
(a) Citizen (Nigeria) Government (Nigeria)



(b) Citizen (Nigeria) Ministry (Nigeria)

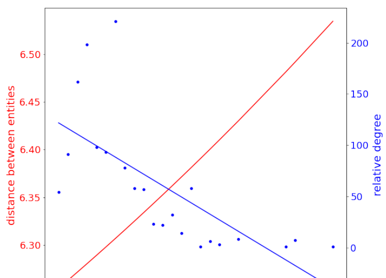


(c) Government (Nigeria) Other Authorities (Nigeria)

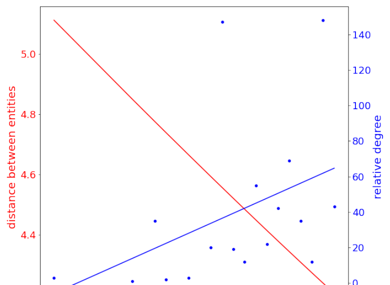


(d) Government (Nigeria) Ministry (Nigeria)

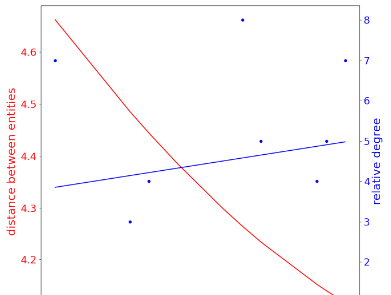
Figure 8: Interaction between Nigerian entities. Subtitles show the names of the given entity pair. Red lines give the geodesic distance between two entities. Blue dots represent the number of interactions between two entities (relative degree) at each timestamp, and blue lines are regression of the relative degree between two entities over time.



(a) Obama, Medvedev



(b) Obama, Xi



(c) Medvedev, Xi

Figure 9: Interaction between Barack Obama, Xi Jinping, and Dmitry Anatolyevich Medvedev. Subtitles show the names of the given entity pair. Red lines give the geodesic distance between two entities. Blue dots represent the number of interactions between two entities (relative degree) at each timestamp, and blue lines are regression of the relative degree between two entities over time.

Algorithm 1: Curvature Estimation

```
Input  : Number of iterations $n_{iter}$, number of timestamps $n_{time}$, Graph Slices $\{G_i\}_{i=1}^{n_{time}}$ of a
         temporal knowledge graph, Neighbor dictionary $\mathcal{N}$.
Output : $\{K_i\}_{i=1}^{n_{time}}$
for $i = 1$ to $n_{time}$ do
    for $m \in G_i$ do
        for $j = 1$ to $n_{iter}$ do
            $b, c \sim \mathcal{U}(\mathcal{N}(m))$ and $a \sim \mathcal{U}(G_i \backslash \{m\})$
            $\psi_j(m, b, c, a) = \frac{1}{2d_{G_i}(a,m)} \left(2d_{G_i}^2(a, m) + d_{G_i}^2(b, c)/4 - d_{G_i}^2(a, b)/2 + d_{G_i}^2(a, c)/2\right)$
        end
        $\psi_i(m) = \sum_{j=1}^{n_{iter}} \psi_j(m, b, c, a)$
    end
    $K_i = \sum_{m \in G_i} \psi_i(m)$
end
```

Table 8: Exponential and logarithmic maps in Poincaré ball and projected hypersphere.

trigonometric functions	$\tan_K(\cdot) = \tan(\cdot)$ if $K > 0$; $\tanh(\cdot)$ if $K < 0$
Exponential map	$\exp_{\mathbf{x}}^K(\mathbf{v}) = \mathbf{x} \oplus (\tan_K(\frac{\sqrt{\lvert K\rvert}\lambda_{\mathbf{x}}^K \lvert\lvert\mathbf{v}\rvert\rvert_2}{2})\frac{\mathbf{v}}{\sqrt{K}\lvert\lvert v\rvert\rvert_2})$
Logarithmic map	$\log_{\mathbf{x}}^K(\mathbf{v}) = \frac{2}{\sqrt{\lvert K\rvert}\lambda_{\mathbf{x}}^K} \tan_K^{-1}(\sqrt{\lvert K\rvert}\lvert\lvert -\mathbf{x} \oplus_K \mathbf{v}\rvert\rvert_2)\frac{-\mathbf{x}\oplus_K\mathbf{v}}{\lvert\lvert\mathbf{x}\oplus_K\mathbf{v}\rvert\rvert_2}$

Table 9: Hyperparameter configurations for best-performing models on the ICEWS14 dataset.

Model	DyERNIE-Sgl				DyERNIE-Prod				DyERNIE-Euclid			
Embedding size	10	20	40	100	10	20	40	100	10	20	40	100
Curvature												
Component A	-0.172	-0.171	-0.171	-0.170	-0.044	-0.114	-0.177	-0.346	0	0	0	0
Component B	-	-	-	-	-0.128	-0.286	-0.281	-0.137	-	-	-	-
Component C	-	-	-	-	-0.371	-0.422	-0.470	-0.855	-	-	-	-
Dimension scale												
Component A	10	20	40	100	3	14	20	20	10	20	40	100
Component B	-	-	-	-	1	4	8	21	-	-	-	-
Component C	-	-	-	-	6	2	12	59	-	-	-	-

Table 10: Hyperparameter configurations for best-performing models on the ICEWS05-15 dataset.

Model	DyERNIE-Sgl				DyERNIE-Prod				DyERNIE-Euclid			
Embedding size	10	20	40	100	10	20	40	100	10	20	40	100
Curvature												
Component A	-0.180	-0.181	-0.179	-0.178	-0.102	-0.122	-0.298	-0.453	0	0	0	0
Component B	-	-	-	-	-0.135	-0.163	-1.243	-0.216	-	-	-	-
Component C	-	-	-	-	-0.214	-0.191	-1.819	-0.938	-	-	-	-
Dimension scale												
Component A	10	20	40	100	7	10	31	32	10	20	40	100
Component B	-	-	-	-	2	8	5	52	-	-	-	-
Component C	-	-	-	-	1	2	4	16	-	-	-	-

Table 11: Hyperparameter configurations for best-performing models on the GDELT dataset.

Model	DyERNIE-Sgl				DyERNIE-Prod				DyERNIE-Euclid			
Embedding size	10	20	40	100	10	20	40	100	10	20	40	100
Curvature												
Component A	0.279	0.336	0.259	0.197	0.213	0.241	0.202	0.342	0	0	0	0
Component B	-	-	-	-	0.291	0.336	0.291	0.336	-	-	-	-
Dimension scale												
Component A	10	20	40	100	8	8	10	68	10	20	40	100
Component B	-	-	-	-	2	12	30	32	-	-	-	-

Table 12: Datasets Statistics

Dataset Name	$\|\mathcal{E}\|$	$\|\mathcal{P}\|$	$\|\mathcal{T}\|$	$\|\mathcal{G}\|$	$\|train\|$	$\|validation\|$	$\|test\|$
ICEWS14	7,128	230	365	90,730	72,826	8,941	8,963
ICEWS05-15	10,488	251	4,017	479,329	386,962	46,275	46,092
GDELT	7,691	240	2,975	2,278,405	1,734,399	238,765	305,241

Table 13: Average runtime and parameter number for each approach: runtime is in seconds.

Datasets		**ICEWS14**			**ICEWS05-15**			**GDELT**		
Rank (d)	Model	Manifold	Runtime	Parameters	Manifold	Runtime	Parameters	Manifold	Runtime	Parameters
100	TransE	$\mathbb{E}$	3,800	1,531,856	$\mathbb{E}$	15,200	2,218,976	$\mathbb{E}$	85,600	1,649,582
	DistMult		9,900	1,531,856		31,500	2,218,976		132,700	1,649,582
	ComplEx		4,300	1,531,856		14.100	2,218,976		76,000	1,649,582
100	TTransE	$\mathbb{E}$	55,000	1,531,856	$\mathbb{E}$	430,000	2,218,976	$\mathbb{E}$	1,500,000	1,649,582
	TDistMult		85,000	1,531,856		680,000	2,218,976		2,040,000	1,649,582
	TComplEx		65,000	1,531,856		520,000	2,218,976		1,500,000	1,649,582
	HyTE		45,000	1,531,856		360,000	2,218,976		1,100,000	1,649,582
100	DyERNIE-Prod	$\mathbb{P}^3$	44,500	1,531,856	$\mathbb{P}^3$	343,800	2,218,900	$\mathbb{S}^2$	1,2 59,400	1,649,582
	DyERNIE-Sgl	$\mathbb{P}$	42,000	1,531,856	$\mathbb{P}$	341,900	2,218,976	$\mathbb{S}$	1,208,300	1,649,582
	DyERNIE-Euclid	$\mathbb{E}$	19,000	1,531,856	$\mathbb{E}$	38,000	2,218,976	$\mathbb{E}$	388,800	1,649,582
40	DyERNIE-Prod	$\mathbb{P}^3$	35,500	621,296	$\mathbb{P}^3$	229,500	900,176	$\mathbb{S}^2$	800,000	669,062
	DyERNIE-Sgl	$\mathbb{P}$	32,000	621,296	$\mathbb{P}$	225,000	900,176	$\mathbb{S}$	740,000	669,062
	DyERNIE-Euclid	$\mathbb{E}$	11,000	621,296	$\mathbb{E}$	25,000	900,176	$\mathbb{E}$	262,000	669,062
20	DyERNIE-Prod	$\mathbb{P}^3$	32,500	317,776	$\mathbb{P}^3$	225,000	460,576	$\mathbb{S}^2$	700,000	342,222
	DyERNIE-Sgl	$\mathbb{P}$	31,500	317,776	$\mathbb{P}$	220,000	460,576	$\mathbb{S}$	676,000	342,222
	DyERNIE-Euclid	$\mathbb{E}$	9,500	317,776	$\mathbb{E}$	22,000	460,576	$\mathbb{E}$	240,000	342,222
10	DyERNIE-Prod	$\mathbb{P}^3$	20,500	166,016	$\mathbb{P}^3$	165,000	240,776	$\mathbb{S}^2$	420,000	178,802
	DyERNIE-Sgl	$\mathbb{P}$	20,500	166,016	$\mathbb{P}$	150,000	240,776	$\mathbb{S}$	400,000	178,802
	DyERNIE-Euclid	$\mathbb{E}$	6,500	166,016	$\mathbb{E}$	15,000	240,776	$\mathbb{E}$	180,000	178,802

Chapter 7

Conclusion

In this thesis, we studied relational machine learning on time-evolving multi-relational data from the perspectives of explainability, continuous-time modeling, and non-Euclidean embedding. The example of time-evolving multi-relational data considered here is temporal knowledge graphs (tKGs) that store temporal facts such as news and events. In the following, we summarize our main contributions.

In Chapter 3, we presented an explainable tKG forecasting model that extracts a query-dependent subgraph from a given tKG and applies a temporal graph attention mechanism over it. The extracted subgraph can be considered a graphical interpretation of the model's prediction. By means of a survey about the evidence included in the extracted subgraph, we found that the evidence is informative for humans. Besides, the model showed its superiority compared to other strong baselines on four benchmark datasets.

In Chapter 4, we proposed another tKG forecasting model that is able to learn continuous-time knowledge embedding using a novel graph neural ordinary differential equation (ODE). In particular, the model is the first one to show that the neural ODE can be used to enhance relational learning on tKGs. Moreover, we showed that using a graph transition layer can effectively boost the model's performance by explicitly taking edge formation and deletion into account.

In Chapter 5, we examined the individual contribution of well-known temporal encoding methods through large-scale experiments with nearly 19,000 GPU hours. We found that some early time encoding approaches achieve competitive performance as advanced techniques when appropriately trained, suggesting the training strategies account for a considerable fraction of the progress made in recent years. To help future research assess whether their temporal encoding is truly helping boost performances, we proposed the first unified open-source framework for tKG completion models with full composability of different training strategies, encoding and decoding techniques.

In Chapter 6, we learned geometric embedding of tKGs on a product of Riemannian manifolds with heterogeneous curvatures. In particular, we assigned each entity an initial embedding in the product manifold and learned velocity vectors in tangent spaces to let entity embedding evolve over time and thus, capture their temporal dynamics. The proposed embedding approach significantly outperforms its Euclidean counterpart and other state-of-the-art baselines on three benchmark datasets of tKG completion, demonstrating the merit of temporal geometric knowledge embedding.

This thesis has only discussed a small portion of relational machine learning on temporal knowledge graphs. However, machine learning for time-evolving multi-relational data is an extensive and active research area that remains a lot to be explored for future research. We outline exciting and challenging future research in the following directions:

- Typical temporal knowledge graphs, e.g., GDELT and ICEWS, were constructed using information extraction techniques that extract structured information from textual data. However, some information would be lost during the extraction procedure due to incomplete schema. Thus, introducing textual information is helpful to enrich the temporal knowledge representations and address the sparseness and incompleteness issues of temporal knowledge graphs. A natural question would be how to inject knowledge from textual data into temporal knowledge embedding. For instance, our ongoing work [33] utilizes pre-trained language representation models to learn contextualized language representation from unstructured textual descriptions and enhance temporal knowledge embedding using informative language representations. In particular, we align the contextualized language representations with temporal knowledge embeddings using a novel knowledge-text prediction task, where we pair quadruples with their textual descriptions and design the prediction task as an extended masked language modeling task by randomly masking words in texts and entity or predicates in quadruples.

- Since joint models often benefit both sides, another interesting research is to investigate whether temporal knowledge embeddings can improve language representations as well as some downstream tasks where time information is needed, such as temporal question answering [44] and time-aware recommender systems [102].

- Existing large-scale temporal knowledge databases often have large noise caused by retrieving structured knowledge from unstructured data. Thus, proposing more advanced construction methods for temporal knowledge graphs is needed in future research. There is a line of work [104, 99, 103] of event extraction that can identify arguments, including actors, locations, and time, from textual data with a pre-defined schema. Such methods can potentially be adapted for constructing temporal knowledge graphs.

- Since temporal knowledge graphs develop over time, it is natural to encounter unseen entities at inference time. However, existing approaches for temporal knowledge graphs do not explicitly tackle this problem, i.e., to be inductive. An interesting research question is how to learn the representations of unseen entities and give an uncertainty measure for queries involving such newly emerged entities, which is crucial for safety-critical tasks, i.e., in the medical domain.

- In many industrial scenarios, both static and dynamic multi-relational data are necessary to support decision-making. Taking predictive maintenance as an example, we need to jointly consider both the static machinery structural dependencies, such as hyponymy relations between different parts and the temporal information from the sensors on the parts to predict which parts should be replaced. However, existing approaches can model either static or temporal knowledge but cannot jointly model both. For example, existing temporal KG approaches explicitly process every graph snapshot. If we adapt them to model static facts, they will consume high computational resources for redundant information since the static facts appear in every snapshot. Thus, further research is needed to be able to effectively model both static and dynamic multi-relational data in a unified framework.

Bibliography

[1] James Atwood and Don Towsley. Diffusion-convolutional neural networks. In *Advances in neural information processing systems*, volume 29, 2016.

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

[3] Ivana Balazevic, Carl Allen, and Timothy Hospedales. Multi-relational poincaré graph embeddings. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

[4] Ivana Balažević, Carl Allen, and Timothy M Hospedales. Tucker: Tensor factorization for knowledge graph completion. *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, 2019.

[5] Leonard E Baum and Ted Petrie. Statistical inference for probabilistic functions of finite state markov chains. *The annals of mathematical statistics*, 37(6):1554–1563, 1966.

[6] Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828, 2013.

[7] Max Berrendorf. *Machine learning for managing structured and semi-structured data.* PhD thesis, lmu, 2022.

[8] Jean-Paul Berrut and Lloyd N Trefethen. Barycentric lagrange interpolation. *SIAM review*, 46(3):501–517, 2004.

[9] Marián Boguná, Fragkiskos Papadopoulos, and Dmitri Krioukov. Sustaining the internet with hyperbolic mapping. *Nature communications*, 1(1):1–8, 2010.

[10] Kurt Bollacker, Robert Cook, and Patrick Tufts. Freebase: A shared database of structured general human knowledge. In *AAAI*, volume 7, pages 1962–1963, 2007.

[11] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. Translating embeddings for modeling multi-relational data. *Advances in neural information processing systems*, 26, 2013.

[12] Elizabeth Boschee, Jennifer Lautenschlager, Sean O'Brien, Steve Shellman, James Starz, and Michael Ward. Icews coded event data. *Harvard Dataverse*, 12, 2015.

[13] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. Spectral networks and locally connected networks on graphs. In Yoshua Bengio and Yann LeCun, editors, *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014.

[14] Andrew Carlson, Justin Betteridge, Bryan Kisiel, Burr Settles, Estevam R Hruschka, and Tom M Mitchell. Toward an architecture for never-ending language learning. In *Twenty-Fourth AAAI conference on artificial intelligence*, 2010.

[15] Ines Chami, Adva Wolf, Da-Cheng Juan, Frederic Sala, Sujith Ravi, and Christopher Ré. Low-dimensional hyperbolic knowledge graph embeddings. *arXiv preprint arXiv:2005.00545*, 2020.

[16] Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. *Advances in neural information processing systems*, 31, 2018.

[17] Shib Sankar Dasgupta, Swayambhu Nath Ray, and Partha Talukdar. Hyte: Hyperplane-based temporally aware knowledge graph embedding. In *Proceedings of the 2018 conference on empirical methods in natural language processing*, pages 2001–2011, 2018.

[18] Talgat Daulbaev, Alexandr Katrutsa, Larisa Markeeva, Julia Gusak, Andrzej Cichocki, and Ivan Oseledets. Interpolation technique to speed up gradients propagation in neural odes. *Advances in Neural Information Processing Systems*, 33:16689–16700, 2020.

[19] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. *Advances in neural information processing systems*, 29, 2016.

[20] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. In Jill Burstein, Christy Doran, and Thamar Solorio, editors, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4171–4186. Association for Computational Linguistics, 2019.

[21] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Uszkoreit Jakob, and Houlsby Neil. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[22] David Ferrucci, Eric Brown, Jennifer Chu-Carroll, James Fan, David Gondek, Aditya A Kalyanpur, Adam Lally, J William Murdock, Eric Nyberg, John Prager, Nico Schlaefer, and Chris Welty. Building watson: An overview of the deepqa project. *AI magazine*, 31(3):59–79, 2010.

[23] Alberto García-Durán, Sebastijan Dumancic, and Mathias Niepert. Learning sequence encoders for temporal knowledge graph completion. In Ellen Riloff, David Chiang, Julia Hockenmaier, and Jun'ichi Tsujii, editors, *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 4816–4821. Association for Computational Linguistics, 2018.

[24] Amir Gholaminejad, Kurt Keutzer, and George Biros. ANODE: unconditionally accurate memory-efficient gradients for neural odes. In Sarit Kraus, editor, *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019*, pages 730–736. ijcai.org, 2019.

[25] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In *International conference on machine learning*, pages 1263–1272. PMLR, 2017.

[26] Rishab Goel, Seyed Mehran Kazemi, Marcus Brubaker, and Pascal Poupart. Diachronic embedding for temporal knowledge graph completion. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 3988–3995, 2020.

[27] Albert Gu, Frederic Sala, Beliz Gunel, and Christopher Ré. Learning mixed-curvature representations in product spaces. In *International Conference on Learning Representations*, 2018.

[28] Amelia Gyrard, Manas Gaur, Saeedeh Shekarpour, Krishnaprasad Thirunarayan, and Amit Sheth. Personalized health knowledge graph. 2018.

[29] James Douglas Hamilton. *Time series analysis*. Princeton university press, 2020.

[30] Zhen Han, Peng Chen, Yunpu Ma, and Volker Tresp. DyERNIE: Dynamic Evolution of Riemannian Manifold Embeddings for Temporal Knowledge Graph Completion. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7301–7316, Online, November 2020. Association for Computational Linguistics.

[31] Zhen Han, Peng Chen, Yunpu Ma, and Volker Tresp. Explainable subgraph reasoning for forecasting on temporal knowledge graphs. In *International Conference on Learning Representations*, 2020.

[32] Zhen Han, Zifeng Ding, Yunpu Ma, Yujia Gu, and Volker Tresp. Learning neural ordinary equations for forecasting future links on temporal knowledge graphs. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 8352–8364, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics.

[33] Zhen Han, Ruotong Liao, Beiyan Liu, Yao Zhang, Zifeng Ding, Heinz Köppl, Hinrich Schütze, and Volker Tresp. Enhanced temporal knowledge embeddings with contextualized language representations. *arXiv preprint arXiv:2203.09590*, 2022.

[34] Zhen Han, Yunpu Ma, Yuyi Wang, Stephan Günnemann, and Volker Tresp. Graph hawkes neural network for forecasting on temporal knowledge graphs. In Dipanjan Das, Hannaneh Hajishirzi, Andrew McCallum, and Sameer Singh, editors, *Conference on Automated Knowledge Base Construction, AKBC 2020, Virtual, June 22-24, 2020*, 2020.

[35] Zhen Han, Gengyuan Zhang, Yunpu Ma, and Volker Tresp. Time-dependent entity embedding is not all you need: A re-evaluation of temporal knowledge graph completion models under a unified framework. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 8104–8118, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics.

[36] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[37] Thomas Little Heath. *The thirteen books of Euclid's Elements.* Courier Corporation, 1956.

[38] Mikael Henaff, Joan Bruna, and Yann LeCun. Deep convolutional networks on graph-structured data. *arXiv preprint arXiv:1506.05163*, 2015.

[39] Marcel Hildebrandt. *Reasoning on graph-structured data with deep-learning, path-based methods, and tensor factorization.* PhD thesis, lmu, 2021.

[40] Marcel Hildebrandt, Swathi Shyam Sunder, Serghei Mogoreanu, Mitchell Joblin, Akhil Mehta, Ingo Thon, and Volker Tresp. A recommender system for complex real-world applications with nonlinear dependencies and knowledge graph context. In *European Semantic Web Conference*, pages 179–193. Springer, 2019.

[41] Daniel S Himmelstein and Sergio E Baranzini. Heterogeneous network edge prediction: a data integration approach to prioritize disease-associated genes. *PLoS computational biology*, 11(7):e1004259, 2015.

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

[43] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[44] Zhen Jia, Abdalghani Abujabal, Rishiraj Saha Roy, Jannik Strötgen, and Gerhard Weikum. Tequila: Temporal question answering over knowledge bases. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, pages 1807–1810, 2018.

[45] Zhen Jia, Soumajit Pramanik, Rishiraj Saha Roy, and Gerhard Weikum. Complex temporal question answering on knowledge graphs. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, pages 792–802, 2021.

[46] Woojeong Jin, Meng Qu, Xisen Jin, and Xiang Ren. Recurrent event network: Autoregressive structure inference over temporal knowledge graphs. *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2020.

[47] Jaehun Jung, Jinhong Jung, and U Kang. T-gap: Learning to walk across time for temporal knowledge graph completion. *The 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2021.

[48] Rudolph Emil Kalman. A new approach to linear filtering and prediction problems. 1960.

[49] Seyed Mehran Kazemi, Rishab Goel, Sepehr Eghbali, Janahan Ramanan, Jaspreet Sahota, Sanjay Thakur, Stella Wu, Cathal Smyth, Pascal Poupart, and Marcus Brubaker. Time2vec: Learning a vector representation of time. *arXiv preprint arXiv:1907.05321*, 2019.

[50] Seyed Mehran Kazemi, Rishab Goel, Kshitij Jain, Ivan Kobyzev, Akshay Sethi, Peter Forsyth, and Pascal Poupart. Representation learning for dynamic graphs: A survey. *J. Mach. Learn. Res.*, 21(1), jan 2020.

[51] Seyed Mehran Kazemi and David Poole. Simple embedding for link prediction in knowledge graphs. *Advances in neural information processing systems*, 31, 2018.

[52] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017.

[53] Prodromos Kolyvakis, Alexandros Kalousis, and Dimitris Kiritsis. Hyperkg: Hyperbolic knowledge graph embeddings for knowledge base completion. *The Semantic Web - 17th International Conference, ESWC 2020, Heraklion, Crete, Greece, May 31-June 4, 2020, Proceedings*, 2020.

[54] Dmitri Krioukov, Fragkiskos Papadopoulos, Maksim Kitsak, Amin Vahdat, and Marián Boguná. Hyperbolic geometry of complex networks. *Physical Review E*, 82(3):036106, 2010.

[55] Timothée Lacroix, Guillaume Obozinski, and Nicolas Usunier. Tensor decompositions for temporal knowledge base completion. *8th International Conference on Learning Representations, ICLR*, 2020.

[56] Julien Leblay and Melisachew Wudage Chekol. Deriving validity time in knowledge graph. In *Companion Proceedings of the The Web Conference 2018*, pages 1771–1776, 2018.

[57] Kalev Leetaru and Philip A Schrodt. Gdelt: Global data on events, location, and tone, 1979–2012. In *ISA annual convention*, volume 2, pages 1–49. Citeseer, 2013.

[58] Manling Li, Yantao Jia, Yuanzhuo Wang, Jingyuan Li, and Xueqi Cheng. Hierarchy-based link prediction in knowledge graphs. In *Proceedings of the 25th International Conference Companion on World Wide Web*, pages 77–78, 2016.

[59] Qi Liu, Maximilian Nickel, and Douwe Kiela. Hyperbolic graph neural networks. *Advances in Neural Information Processing Systems*, 32, 2019.

[60] Yunpu Ma, Volker Tresp, and Erik A Daxberger. Embedding models for episodic knowledge graphs. *Journal of Web Semantics*, 59:100490, 2019.

[61] Sebastien Montella, Lina Rojas-Barahona, and Johannes Heinecke. Hyperbolic temporal knowledge graph embeddings with relational and time curvatures. *Findings of the Association for Computational Linguistics: ACL/IJCNLP*, 2021.

[62] Kevin P Murphy. *Machine learning: a probabilistic perspective.* MIT press, 2012.

[63] Kevin Patrick Murphy. *Dynamic bayesian networks: representation, inference and learning.* University of California, Berkeley, 2002.

[64] Maximilian Nickel, Xueyan Jiang, and Volker Tresp. Reducing the rank in relational factorization models by including observable patterns. *Advances in Neural Information Processing Systems*, 27, 2014.

[65] Maximilian Nickel, Kevin Murphy, Volker Tresp, and Evgeniy Gabrilovich. A review of relational machine learning for knowledge graphs. *Proc. IEEE*, 104(1):11–33, 2016.

[66] Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. A three-way model for collective learning on multi-relational data. In *Icml*, 2011.

[67] Maximillian Nickel and Douwe Kiela. Poincaré embeddings for learning hierarchical representations. *Advances in neural information processing systems*, 30, 2017.

[68] Maximillian Nickel and Douwe Kiela. Learning continuous hierarchies in the lorentz model of hyperbolic geometry. In *International Conference on Machine Learning*, pages 3779–3788. PMLR, 2018.

[69] Mathias Niepert, Mohamed Ahmed, and Konstantin Kutzkov. Learning convolutional neural networks for graphs. In *International conference on machine learning*, pages 2014–2023. PMLR, 2016.

[70] Chigozie Nwankpa, Winifred Ijomah, Anthony Gachagan, and Stephen Marshall. Activation functions: Comparison of trends in practice and research for deep learning. *arXiv preprint arXiv:1811.03378*, 2018.

[71] Qi Pi, Guorui Zhou, Yujing Zhang, Zhe Wang, Lejian Ren, Ying Fan, Xiaoqiang Zhu, and Kun Gai. Search-based user interest modeling with lifelong sequential behavior data for click-through rate prediction. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, pages 2685–2692, 2020.

[72] Lev Semenovich Pontryagin. *Mathematical theory of optimal processes.* CRC press, 1987.

[73] Erzsébet Ravasz and Albert-László Barabási. Hierarchical organization in complex networks. *Physical review E*, 67(2):026112, 2003.

[74] Richard H Richens. Preprogramming for mechanical translation. *Mech. Transl. Comput. Linguistics*, 3(1):20–25, 1956.

[75] Maya Rotmensch, Yoni Halpern, Abdulhakim Tlimat, Steven Horng, and David Sontag. Learning a health knowledge graph from electronic medical records. *Scientific reports*, 7(1):1–11, 2017.

[76] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning internal representations by error propagation. Technical report, California Univ San Diego La Jolla Inst for Cognitive Science, 1985.

[77] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE transactions on neural networks*, 20(1):61–80, 2008.

[78] Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. Modeling relational data with graph convolutional networks. In *European semantic web conference*, pages 593–607. Springer, 2018.

[79] Edward W Schneider. Course modularization applied: The interface system and its implications for sequence control and data analysis. 1973.

[80] Amit Singhal. Introducing the knowledge graph: things, not strings. *Official google blog*, 5:16, 2012.

[81] Fabian M Suchanek, Gjergji Kasneci, and Gerhard Weikum. Yago: a core of semantic knowledge. In *Proceedings of the 16th international conference on World Wide Web*, pages 697–706, 2007.

[82] Haitian Sun, Tania Bedrax-Weiss, and William Cohen. PullNet: Open domain question answering with iterative retrieval on knowledge bases and text. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2380–2390, Hong Kong, China, November 2019. Association for Computational Linguistics.

[83] Zhiqing Sun, Zhi-Hong Deng, Jian-Yun Nie, and Jian Tang. Rotate: Knowledge graph embedding by relational rotation in complex space. *7th International Conference on Learning Representations, ICLR*, 2019.

[84] Volker Tresp, Cristóbal Esteban, Yinchong Yang, Stephan Baier, and Denis Krompaß. Learning with memory embeddings. *NIPS Workshop on Nonparametric Methods for Large Scale Representation Learning*, 2015.

[85] Rakshit Trivedi, Hanjun Dai, Yichen Wang, and Le Song. Know-evolve: Deep temporal reasoning for dynamic knowledge graphs. In *international conference on machine learning*, pages 3462–3471. PMLR, 2017.

[86] Théo Trouillon, Christopher R. Dance, Éric Gaussier, Johannes Welbl, Sebastian Riedel, and Guillaume Bouchard. Knowledge graph completion via complex tensor factorization. *J. Mach. Learn. Res.*, 18(1):4735–4772, jan 2017.

[87] Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. Complex embeddings for simple link prediction. In *International conference on machine learning*, pages 2071–2080. PMLR, 2016.

[88] Abraham Albert Ungar. A gyrovector space approach to hyperbolic geometry. *Synthesis Lectures on Mathematics and Statistics*, 1(1):1–194, 2008.

[89] Shikhar Vashishth, Soumya Sanyal, Vikram Nitin, and Partha P. Talukdar. Composition-based multi-relational graph convolutional networks. *CoRR*, abs/1911.03082, 2019.

[90] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[91] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018.

[92] Denny Vrandečić and Markus Krötzsch. Wikidata: a free collaborative knowledgebase. *Communications of the ACM*, 57(10):78–85, 2014.

[93] Brian Walsh, Sameh K Mohamed, and Vít Nováček. Biokg: A knowledge graph for relational learning on biological data. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, pages 3173–3180, 2020.

[94] Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. Knowledge graph embedding by translating on hyperplanes. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 28, 2014.

[95] Michael D Ward, Andreas Beger, Josh Cutler, Matthew Dickenson, Cassy Dorff, and Ben Radford. Comparing gdelt and icews event data. *Analysis*, 21(1):267–297, 2013.

[96] Chengjin Xu, Mojtaba Nayyeri, Fouad Alkhoury, Hamed Shariat Yazdi, and Jens Lehmann. Tero: A time-aware knowledge graph embedding via temporal rotation. *Proceedings of the 28th International Conference on Computational Linguistics*, 2020.

[97] Chenjin Xu, Mojtaba Nayyeri, Fouad Alkhoury, Hamed Yazdi, and Jens Lehmann. Temporal knowledge graph completion based on time series gaussian embedding. In

The Semantic Web – ISWC 2020: 19th International Semantic Web Conference, Athens, Greece, November 2–6, 2020, Proceedings, Part I, page 654–671, Berlin, Heidelberg, 2020. Springer-Verlag.

[98] Da Xu, Chuanwei Ruan, Evren Korpeoglu, Sushant Kumar, and Kannan Achan. Self-attention with functional time representation learning. *Advances in neural information processing systems*, 32, 2019.

[99] Runxin Xu, Tianyu Liu, Lei Li, and Baobao Chang. Document-level event extraction via heterogeneous graph-based interaction model with a tracker. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 3533–3546, Online, August 2021. Association for Computational Linguistics.

[100] Xiaoran Xu, Wei Feng, Yunsheng Jiang, Xiaohui Xie, Zhiqing Sun, and Zhi-Hong Deng. Dynamically pruned message passing networks for large-scale knowledge graph reasoning. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020.

[101] Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. Embedding entities and relations for learning and inference in knowledge bases. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[102] Yuyue Zhao, Xiang Wang, Jiawei Chen, Wei Tang, Yashen Wang, Xiangnan He, and Haiyong Xie. Time-aware path reasoning on knowledge graph for recommendation. *arXiv preprint arXiv:2108.02634*, 2021.

[103] Shun Zheng, Wei Cao, Wei Xu, and Jiang Bian. Doc2EDAG: An end-to-end document-level framework for Chinese financial event extraction. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 337–346, Hong Kong, China, November 2019. Association for Computational Linguistics.

[104] Tong Zhu, Xiaoye Qu, Wenliang Chen, Zhefeng Wang, Baoxing Huai, Nicholas Jing Yuan, and Min Zhang. Efficient document-level event extraction via pseudo-trigger-

aware pruned complete graph. *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI*, 2022.