Temporal Knowledge Graph Embedding and Reasoning

Dissertation
zur
Erlangung des Doktorgrades (Dr. rer. nat.)
der
Mathematisch-Naturwissenschaftlichen Fakultät
der
Rheinischen Friedrich-Wilhelms-Universität Bonn

von
Chengjin Xu
aus
Hubei, China

Bonn, 2022

Angefertigt mit Genehmigung der Mathematisch-Naturwissenschaftlichen Fakultät der Rheinischen Friedrich-Wilhelms-Universität Bonn

1. Gutachter:	Prof. Dr. Jens Lehmann
2. Gutachter:	Prof. Dr. Stefan Wrobel
Tag der Promotion:	17.01.2023
Erscheinungsjahr:	2023

Abstract

Knowledge Graphs (KGs) have emerged as an efficient way to organize and represent knowledge by storing the underlying relations between entities. Recently, a large amount of research works have been devoted to KG embeddings, aiming at mapping entities and relations in KGs to low-dimensional continuous vector spaces for fast reasoning. KG embedding models have been widely used for different learning tasks over KGs, e.g., KG completion, multi-hop complex reasoning, and KG alignment. Since most structured knowledge is only valid at a specific time point or within a specific interval, a lot of large-scale KGs attach time information into triple facts to capture the temporal dynamics of knowledge in addition to their multi-relational nature. The recent availability of temporal KGs has created a demand for new KG embedding approaches that can model time-aware quadruple facts.

This thesis aims to delve further into the research of temporal KG representation learning and reasoning. Our motivation is to improve the performance of embedding models on temporal KGs by proposing new temporal KG embedding approaches. In this work, we extend three fundamental learning tasks of static KGs to temporal KGs, i.e., temporal KG completion, multi-hop temporal KG reasoning, and temporal entity alignment. We first propose three novel temporal KG embedding models, namely, ATiSE, TeRo, TGeomE, for temporal KG completion tasks. Specifically, ATiSE models the temporal evolution of entity/relation representations using multi-dimensional additive time series decomposition, TeRo defines the evolution of an entity embedding as a rotation over time in the complex vector space and TGeomE performs 4th-order tensor factorization of a temporal KG using multivector embeddings from a multi-dimensional geometric algebra and considers a new linear temporal regularization. Our proposed temporal KG completion models achieved the state-of-the-art at the time of publishing. To tackle the problem of multi-hop temporal KG reasoning, we generate three temporal query datasets from three common temporal KG benchmarks and propose a vector logic-based temporal query embedding framework, TFLEX. TFLEX is the first query embedding framework that can handle first-order logical operations and temporal logical operations at the same time, and answer both multi-hop entity queries and timestamp queries over TKGs. Lastly, we introduce two new temporal KG embedding models based on graph neural networks, TEA-GNN and TREA, for entity alignment between temporal KGs, and propose three new temporal KG datasets as references for evaluating entity alignment methods. TEA-GNN regards timestamps as attentive properties of links between entities and uses a time-aware graph self-attention mechanism to effectively incorporate time information into graph neural networks. Built on the top of TEA-GNN, TREA has a better inductive learning ability to represent new emerging entities and timestamps, and a higher training efficiency on large-scale temporal KGs. We empirically prove that the proposed TEA models significantly outperform the existing static entity alignment methods and temporal KG completion-oriented temporal KG embedding models. Overall, this thesis tackles different challenges of temporal KG embeddings by introducing new tasks, metrics, datasets and models. Experimental results demonstrate that our proposed methods successfully integrate time information into representation learning models of KGs.

Acknowledgements

This thesis was not produced in a vacuum, and there are many people whose ideas and encouragement contributed to the work. It would not be possible without the collaboration with my co-authors, fellow colleagues, supervisors, and the support from my family.

I would like to first express my sincere gratitude to my supervisor Prof. Dr. Jens Lehmann, who provided crucial support and guidance during my Ph.D. study, most importantly provided a research environment and a set of interesting questions where this work could thrive.

I am also very grateful to Dr. Hamed Shariat Yazdi for offering me the chance to interview for this Ph.D. position in the SDA research group and helping me develop my Ph.D. research plan. I am indebted as well to Mojtaba Nayyeri, who has been constantly helping me since the first day I joined the SDA research group, in terms of both my academic research and daily life. I have learned a lot from Mojtaba about how to conduct research as a Ph.D. candidate.

I also would like to extend my sincere thanks to my other co-authors, Dr. Sahar Vahdati, Mirza Mohtashim Alam, Yung-Yu Chen, Fouad Alkhoury, Fenglong Su, Bo Xiong, and Xueyuan Lin for providing support and constructive criticism to improve my research work.

I must thank the SDA research group, for providing an innovative environment for collaboration. I had many positive interactions with my colleagues in the SDA research group that helped me solidify my thinking on many topics related to this thesis. Beside people I have mentioned, Afshin Sadeghi, Mehdi Ali, Shimaa Ibrahim, Firas Kassawat, Dr. Gezim Sejdiu, Dr. Hamid Zafar, Dr. Hajira Jabeen and Dr. Giulio Napolitano have been valued colleagues during my time at the University of Bonn.

Thank you also to the Chinese Scholarship Council (CSC). This work was largely funded by a CSC scholarship.

To my family, particularly my mom, thank you for your love, support, and unwavering belief in me. Without you, I would not be the person I am today.

Above all I would like to thank my wife Xue for her love and constant support, for all the late nights and early mornings, and for keeping me sane over the past few months. Thank you for being my muse, physician, German translator and teacher. But most of all, thank you for being my best friend. I owe you everything. Best wishes for your Dritter Abschnitt der Ärztlichen Prüfung and M.D. defence.

Contents

CHAPTER 1

Introduction

In recent years, knowledge graphs (KGs) have received increasing attention because of their efficient organization of knowledge and powerful semantic processing capabilities. As an important fundamental technology in the field of artificial intelligence, KGs can empower intelligence to analyze, reason and understand, and thus are widely used in tasks such as intelligent search, question answering systems, and personalised recommendations [1]. Currently, KG has become one of the core drivers for the development of artificial intelligence. For instance, Google Knowledge Graph is a next-generation intelligent search feature released by Google in 2012, whose services improve the quality of results returned by the search engine by collecting information from different data sources [2]. As shown in Figure 1.1, a search for "*who is the father of George Walker Bush?*" in Google's search engine returns higher ranking results than a traditional search engine that simply matches the keywords "*George Walker Bush*" and "*father*", the knowledge graph-based Google search is able to understand the semantics of the search question, return the personal name directly, and provide structured information related to the search object at the top right, such as his birth date, presidential term, etc.

Figure 1.1: A search example based on the Google Knowledge Graph

In terms of presentation, a KG is a knowledge base represented by a semantic web that describes real-world things and abstract concepts and their interconnections in symbolic form, using entities

to represent a specific thing or a concept and relations to represent semantic connections between entities. The basic building blocks of a KG are facts shaped like (*subject entity*, *relation*, *object entity*), which are also known as triples (e_s, r, e_o). Entities are linked to each other through relations, forming a web-like knowledge structure, which can also be represented by a directed graph structure, where nodes represent entities and edges represent relations between entities. Figure 1.2a illustrates a subgraph of a KG.

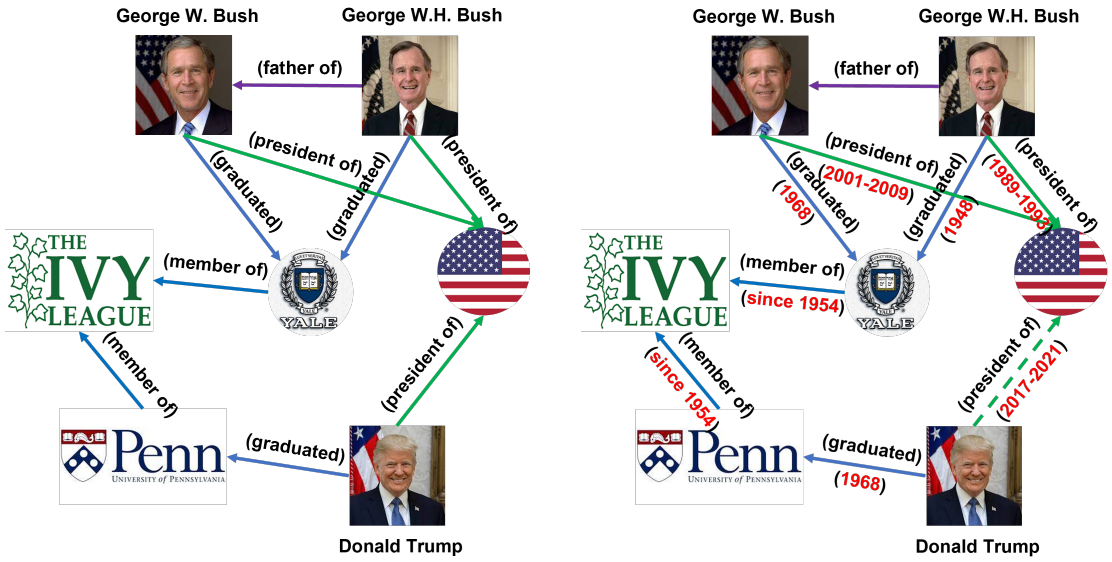


Figure 1.2: Examples of (a) a subgraph of a static KG and (b) its temporal version.

With the fast increment of the volume of structured data on the Internet, the demand for fast and accurate access to high-quality information is growing, and the need for more and more intelligent applications that require knowledge has been driving the birth and development of new KGs. At the early stage, some Linked Data projects have produced high-quality knowledge bases, such as Freebase [3] and DBpedia [4], two of the most iconic general domain knowledge bases, both derived from Wikipedia-based collaborative knowledge resources. After the great success of Google's search engine using Google Knowledge Graph for intelligent search, KG technology has attracted much attention both in academia and industry, and many large Internet companies have started to build their own KGs. For example, Microsoft has built Satori for various intelligence services (e.g. news recommendation [5]), IBM has built its KG for intelligent question and answer, and Facebook has built the Facebook Knowledge Graph to support the intelligent search of friends' services.

The key technologies associated with the development and application of KGs include knowledge graph construction, knowledge graph storage, knowledge reasoning and computation, and knowledge fusion. These key technologies are still in the process of development, and the knowledge graph representation technology is the foundation to help them continue to evolve and improve. Moreover, KG representation technology is also an effective means to apply knowledge graphs to downstream intelligence tasks. Traditional KG representations use the RDF framework technique, where an entity or relationship is represented by a unique identifier number, and knowledge is represented by a triad of

these identifier numbers. To accommodate the applications of KGs on different downstream tasks, it is desired that KG representations can effectively embody semantic information of entities and relations. Inspired by word embedding techniques [6, 7], the researchers proposed knowledge graph embedding (KGE), where entities and relations in the knowledge graph are embedded in a low-dimensional dense continuous vector space, with the corresponding vector representations representing their semantic information. By doing this, entities with similar semantics are embedded in similar regions of the vector space, and the potential relations between entities can be inferred. Compared to one-hot encoding, the embedding technique is a relatively low-dimensional and dense representation technique that avoids the curse of dimensionality and supports fast learning and reasoning on large-scale datasets. The advantages of embedding techniques have led researchers to turn to KGEs in an attempt to learn high-quality representations of KGs and apply them to KG-related technologies and downstream tasks.

Most existing KGE approaches treat the facts in the knowledge graph as common knowledge, based on the assumption that they do not change over time, which is unsuitable for some application scenarios. For example, e.g., (*Donald Trump*, *president of*, *USA*) is valid only from 2017 to 2021. Therefore, such static KGE (SKGE) models tend to lose a large amount of useful temporal information contained in KGs, and cannot discover and restore the missing information in KGs with time-series characteristics to obtain more complete KGs. Temporal knowledge graphs (TKGs) are able to represent temporal dynamics by additionally associating a part of triples in the KG with the corresponding timestamps, such as (*Donald Trump*, *president of*, *USA*, *[2017-2021]*), as shown in Figure 1.2b. A few examples of TKGs include YAGO [8], Wikidata [9], ICEWS [10] and GDELT [11]. Facts in such temporal KGs (TKGs) can be represented as quadruples shaped like (e_s, r, e_o, t), where t denotes the timestamp. In this regard, devising a proper temporal KGE (TKGE) model is the initial step towards successful downstream tasks.

Our objective in this thesis is to develop novel temporal knowledge graph embedding (TKGE) models for improving the existing static KGE (SKGE) models' abilities of learning and reasoning over TKGs. Intuitively, the inclusion of time information can be helpful to learning tasks over TKGs and the relevant time-aware downstream tasks. And TKGE is more challenging than SKGE in the sense that it would require us to model the temporal evolution of KG representation. This thesis mainly focuses on three TKG-related tasks, i.e., temporal knowledge graph completion (TKGC), multi-hop temporal knowledge graph reasoning (MTKGR) and temporal entity alignment (TEA), and develops different TKGE-based models to advance the state-of-the-art on the above tasks.

1.1 Motivation, Problem Statement and Challenges

Although KGs are widely available, they are mostly incomplete. As a result, most KGs face the problem of data sparsity, which means that entities are sparse and the number of facts is relatively small. Thus, finding the missing facts in KGs, i.e., knowledge graph completion (KGC), has become one of the main tasks on KGs. SKGE has been proven to be an effective method to address KGC problems by scoring each triple (e_s, r, e_o) [12–14]. More concretely, for answering a single-hop query (*Donald Trump*, *president of*, *?*), a SKGE model can compute the score of the triple (*Donald Trump*, *president of*, *USA*) with embeddings of the involved entities and relation, and predict the correct answer *USA* by ranking the score against other candidate triples.

KGE has been proven to be a powerful approach for completing incomplete triples to answer single-hop queries on KGs. However, an open challenge in this area is developing techniques that

can go beyond single-hop queries and handle more complex multi-hop logical queries. In general, multi-hop reasoning answers a first-order logic (FOL) query over KGs using logical and relational operators, including relation projection, intersection, union and complement/negation. For example, answering "*which Canadian citizens won the Turing Award?*" requires imputation and prediction of two single-hop queries, (*?*, *won*, *Turing Award*) and (*?*, *citizen of*, *Canada*), while also using logical set operations, e.g., intersection in this case. Based on prior KGE work, some query embedding (QE) approaches [15–24] have been proposed for multi-hop knowledge graph reasoning (MKGR) and have drawn a lot of attention from researchers in the field of KG. Such QE works generally design neural logical operators that simulate the logical set operations, and embed multi-hop queries into a vector space by iteratively executing logical operations according to the conjunctive computation graphs parsed from the complex queries. It is worthy to note that the task of multi-hop knowledge graph reasoning focus on logical queries over KGs, e.g., SQL queries and SPARQL queries, which is different from the task of question answering aiming to answer natural language questions. The task of knowledge graph completion can be regarded as a special case of the multi-hop reasoning task when the query only contains a single relation projection.

Most KGs are independently extracted from various data sources, focusing on different domains or languages. And using a single KG is oftentimes insufficient for the need of downstream applications. Thus, it is essential to fuse heterogeneous knowledge among different KGs where the same entities exist in different surface forms. One important component of multi-source knowledge fusion technology is entity alignment (EA), i.e., matching entities with similar semantics across different KGs. The main purpose of EA is to infer whether two entities in the same or different knowledge graphs jointly point to the same thing in the real world, which plays an important role in the construction of knowledge graphs and the integration of existing knowledge graphs. The creation of EA technology makes the process of fusing massive amounts of knowledge and information data more efficient. Recently, SKGE methods have been successfully applied to EA problems [25–30]. This technique provides a simple way to represent entities in a low-dimensional vector space, where the distance between the individual entity vectors can be calculated directly using distance formulas to obtain the similarity between entities. By using KGEs, EA problems can be solved by finding entities with similar embeddings across KGs.

The recent availability of a large amount of TKGs that exhibits complex temporal dynamics in addition to their multi-relational nature has created the need for approaches that can characterize and reason over temporally evolving knowledge [31]. While SKGE has received a large amount of attention from the research community and is successfully applied to the above-mentioned tasks, i.e., KGC, MKGR and EA, temporal KGE (TKGE) technology is still a relatively unexplored area. SKGE models focus on advancing entity-relation representation learning. But these methods lack the ability to use rich temporal dynamics available in underlying data represented by TKGs, leaving much room for improvement. It is very interesting to investigate whether the incorporation of time information can help KGE models with their performances over TKGs and how it can be done.

In this section, we give the research problem definition for this thesis work; then we look into the challenges posed by the problem definition. We provide examples for each challenge to give a better overview. We recognize three challenges to be tackled while working towards our research problem, based on the different learning tasks related to TKGs as shown in Figure 1.3.

Research Problem Definition

How can we effectively incorporate time information into KGE models, and utilize TKGEs to improve the performance of the existing KGE-based models on different learning tasks over TKGs?

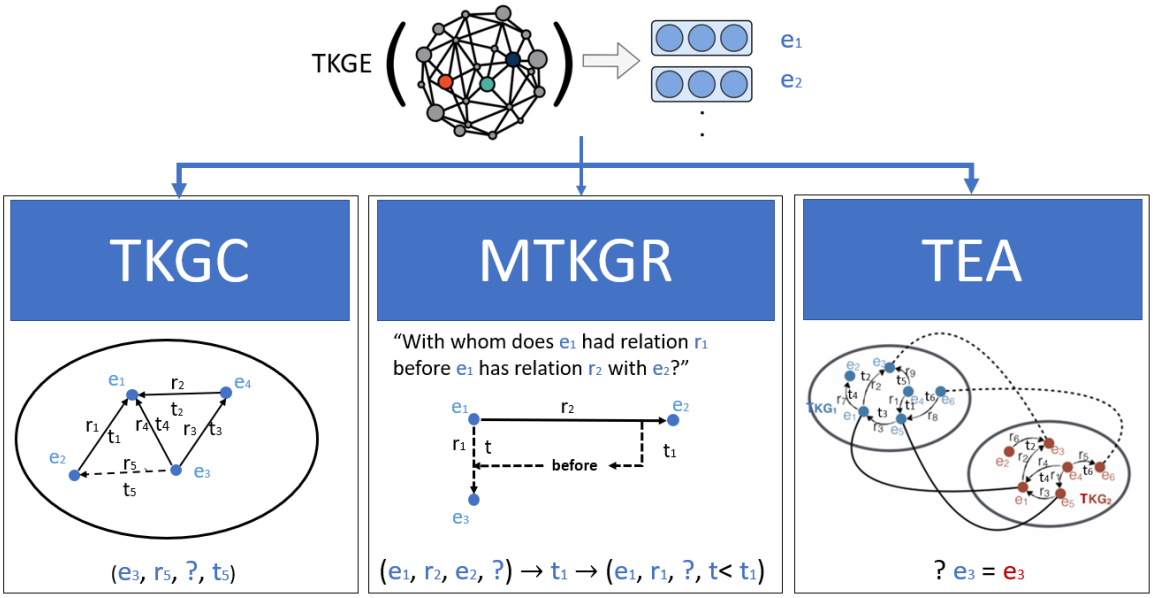


Figure 1.3: Applications of TKGE on different learning tasks over TKGs

Challenge 1: Limitations of Prior TKGC Methods

In temporal knowledge graph completion (TKGC), queries are often related to time information. For example, the query (*?*, *president of*, *USA*) can have different answers at different timestamps. When the attached timestamp is *[2017-2021]*, the predicted answer is expected to be *Donald Trump*. The SKGE-based KGC models do not consider time information and thus probably give wrong predictions. Thus, incorporating time information into KGC models is very crucial to perform TKGC tasks. Generally, there are two different ways of modeling time information for TKGC. First, we can model the temporal evolution of each entity/relation embedding as a time-dependent function and this time-dependent function can be a multi-dimensional time series model or a recurrent neural network [32]. Another solution is to map timestamps into the embedding space, i.e., each timestamp can be represented as an individual embedding [33, 34]. Either way can be helpful for TKGC models more accurately measuring the possibilities of facts in a TKG with the consideration of time information.

However, the prior works on TKGC all have their respective shortcomings, which limit their performances. The limitations of the prior TKGC works can be summarized as the following aspects: 1) temporal uncertainty; 2) temporal interpretability; 3) time representation; 4) time distribution; 5)

time granularity; 6) model expressiveness; 7) temporal regularization. Specifically, the prior TKGC methods disregard the uncertainty during the temporal evolution of entity/relation representations and their ways of modeling temporal evolution of entity/relation representations lack interpretability. Moreover, time data in different TKGs might have different distributions, representation forms and granularities. Few TKGC methods can adapt well to various types of TKGs and study the effects of these time-related factors on the model performance. We also notice that many prior TKGC methods are built on top of flawed SKGE models, which are partly expressive and unable to model some specific relation patterns, and there is a need for effective time regularization techniques to help TKGC methods with retaining physical characteristics between time data while modeling timestamps, e.g., temporal distance and ordering.

To use TKGEs for the task of TKGC, the main challenge is to develop novel TKGC methods to overcome the above-mentioned limitations.

Challenge 2: Modeling Temporal Logic for Multi-hop Reasoning over TKGs

The existing works on multi-hop reasoning over KGs mainly focus on existential positive first-order (EPFO) logical queries, i.e., queries that include conjunction ($\wedge$), disjunction ($\vee$), negation ($\neg$), and existential quantifier ($\exists$). Moreover, TKGC models can only handle single-hop temporal queries, e.g., "*who was the president of the USA in 2018?*". Recently, some path-based TKG forecasting models can perform multi-hop reasoning but do not consider any logical operations. To the best of our knowledge, there is no prior KGE-based work on multi-hop logical queries over TKGs. Besides first-order logic, temporal logic has been broadly used to cover all approaches to reasoning about time and temporal information. Complex temporal queries often contain temporal logical operations. For example, a temporal query "*who have served as president of the USA after George W. Bush?*" can be encoded into a conjunctive computation graph that includes temporal logic operations, e.g., *after* in this case, as shown in Figure 1.4. Therefore, it is of great significance to propose a new multi-hop TKG reasoning (MTKGR) framework that can 1) perform multi-hop FOL reasoning over TKGs and 2) model temporal logical operations at the same time.

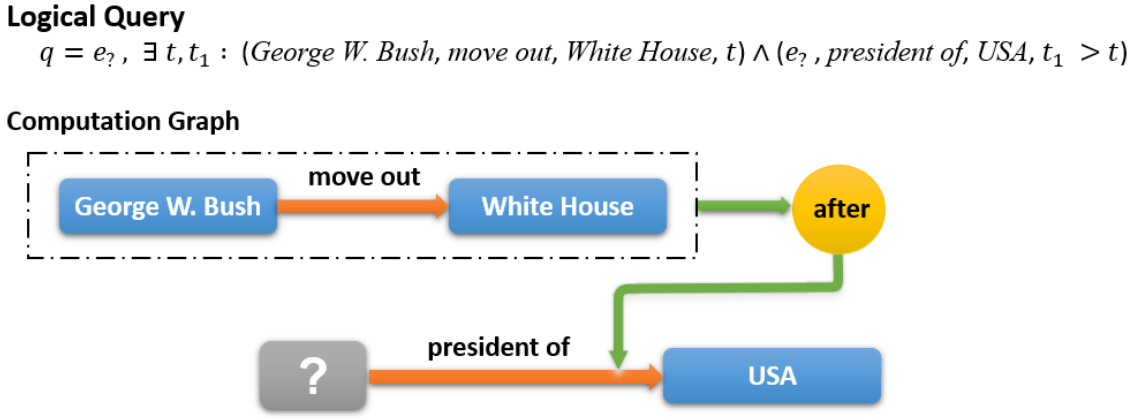


Figure 1.4: A temporal query and its conjunctive computation graph

Challenge 3: Incorporating Time information into Embedding-based EA Models

The existing embedding-based EA models mainly rely on entities' static structure information and disregard time information, leaving much room for improvement. As in the case of Figure 1.5 in which the top and bottom subgraphs are extracted from two separate TKGs respectively, static EA methods are likely to recognize *George Walker Bush* and *George W.H. Bush* as an entity pair to be aligned due to the similarity of their entity names, neighboring nodes and connected relations. On the other hand, these two entities can be easily distinguished by considering the timestamps of links within their neighborhood. Therefore, devising a temporal entity alignment (TEA) method is expected to effectively address the limitations of the existing EA methods. Thanks to the ability of GNNs to model the global information of a graph, a lot of EA approaches [27, 28] introduce GNNs into EA tasks and have been proven to have outstanding performances. However, it is challenging to incorporate time information into GNN models. Most existing temporal GNN models [35, 36] adopt composite architecture combined by GNN and temporal recurrence models, which suffer from long training time. Thus, it is important for TEA to design an efficient temporal GNN structure. Moreover, the existing EA methods are developed based on the closed-world assumption, i.e., where KGs are fixed and new entities cannot be easily added. Thus, the existing embedding-based EA models can not perform inductive learning for new emerging entities and timestamps, which is important for reasoning over dynamic KGs in the real world.

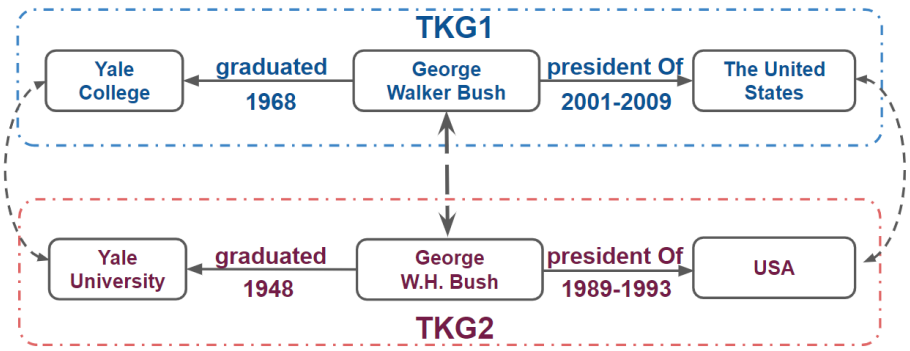


Figure 1.5: Illustration of the limitation of the existing EA approaches

1.2 Research Questions

Research Question 1 (RQ1)

How can we effectively encode time information into KGE models to enhance temporal knowledge graph completion?

Obviously, it is essential to consider time information for performing the task of TKGC. A lot of TKGE models have been proposed for TKGC and proven to outperform traditional SKGE models, supporting this intuition. As mentioned before, the prior works on TKGC have limitations in the

different aspects, 1) temporal uncertainty; 2) temporal interpretability; 3) time representation; 4) time distribution; 5) time granularity; 6) model expressiveness; 7) temporal regularization. To overcome limitations 1-2), we explore how to develop a novel TKGC method which models the temporal evolution of entity/relation representations in an explainable way and considers the uncertainty of temporal evolution. For limitations 3-5), we expect to propose new time representation methods to help TKGC models adapt well to various TKGs which have different time representation forms, time data distributions, and time granularities. With regard to limitations 6-7), we investigate how to develop expressive TKGC models that can model all kinds of relation patterns and a novel temporal regularizer to provide better geometric meanings for time embeddings and improve the performances of TKGC models. These questions form our first research question (RQ1).

Research Question 2 (RQ2)

How can we model temporal logical operations in KGE models to perform multi-hop reasoning over temporal knowledge graphs?

Query embedding (QE) on KGs aims at answering logical queries using neural logical operators instead of traditional databases and query language. Different QE methods have been proposed to model EPFO logical operations for answering multi-hop logical queries over SKGs. Such methods combine the advantages of embedding models and symbolic models, which have high computational efficiency and strong generalization ability as well as great interpretability. Many natural questions include temporal constraints or the target answers are supposed to be timestamps. Therefore, complex queries over TKGs often involve temporal predictions and temporal logic, which express the temporal relationships between facts or events contained in the queries. A commonly used tool to represent temporal relationships for natural language processing is Allen interval algebra [37], which defines 13 temporal relationships, e.g., *before* and *during*, as predicates in the logic. However, few works investigate how to model such temporal relationships in a TKGE model. And to the best of our knowledge, there is no prior work that can model both temporal logical operations and EPFO logical operations for multi-hop reasoning over TKGs in an embedding space. To address this research problem, we need to develop a new temporal QE (TQE) framework for multi-hop TKG reasoning which designs neural logical operators for FOL operations and temporal logical operations, and encodes temporal queries into conjunctive computation graphs. Each step in the temporal query computation graphs is a single-hop prediction or a logical operation, either of which is performed in an embedding space. The temporal query computation graphs iteratively execute these steps to perform multi-hop reasoning over a TKG to obtain a target answer.

Research Question 3 (RQ3)

Can the incorporation of time information be helpful for the performances of KGE models on the task of entity alignment between temporal knowledge graphs?

For this research question, we focus on the task of entity alignment between TKGs. Since the existing static entity alignment (SEA) methods disregard time information in TKGs, the intuitive solution

for temporal entity alignment (TEA) is to incorporate time information into static entity alignment (SEA) models. Besides temporal unawareness, the existing SEA methods also have limitations on the reliance of labeled data, training efficiency and inductive learning inability. Since extensive research works have shown that GNN-based KGE models perform very well on the SEA task, the main challenge of this research problem is to develop effective and efficient temporal GNN models for TEA and investigate whether the incorporation of time information can be helpful for overcoming the limitations of the existing SEA methods.

1.3 Thesis Overview

This chapter highlights the primary contributions of this thesis. We provide references to scientific articles covering these contributions published throughout the whole term.

1.3.1 Contributions

Contributions for RQ1

Three novel TKGE models that overcome the limitations of the prior TKGC methods and achieve state-of-the-art results.

To overcome the limitations of prior TKGC models, we present three novel TKGC models and compare their performances against other state-of-the-art SKGC and TKGC models.

To handle the uncertainty during the temporal evolution of entity/relation representations, we present our first TKGC model, ATiSE, which represents each entity/relation as a time-specific multi-dimensional Gaussian distribution and incorporates time information into entity/relation representations by using additive time series decomposition. We argue that it is of importance to consider randomness for modeling the temporal evolution of entity representations, because the features of an entity at a certain time are not completely determined by the past information. ATiSE outperforms previous SKGC and TKGC models on various TKGC benchmarks and shows good interpretability in modeling different types of entities and relations at a parameter level.

Most of the early TKGC models including ATiSE have issues with modeling specific relation patterns, e.g., symmetric relations. To address this issue, we present our second TKGC model, TeRo, which defines the evolution of an entity embedding as a rotation from the initial time to the current time in the complex vector space. Different from ATiSE, TeRo uses dual relation embeddings to handle the beginning and end of a fact, which is a very efficient way to model facts involving time intervals. TeRo also shows a better expressiveness than previous TKGC models and outperforms ATiSE on various TKGC benchmarks.

Furthermore, we present a fully expressive TKGC model, TGeomE, which performs 4th-order tensor factorization of a TKG using multivector embeddings from a multi-dimensional geometric algebra and considers a new linear temporal regularization for retaining the ordering and distance information between different timestamps. TGeomE subsumes several existing TKGC models and achieves state-of-the-art results on TKGC.

We also develop a time splitting method to address the issue of long-tailed distributions of time data in some TKGs for our presented TKGC models. All of the three presented TKGC models adapt

well to various TKG datasets which have different time representation forms, time data distributions and time granularities.

Contributions for RQ2

A novel temporal query embedding framework for multi-hop reasoning over TKGs that can model both FOL operations and temporal logical operations in a vector space.

Multi-hop logical reasoning over knowledge graph (KG) is a fundamental learning task over KGs. Recent query embedding (QE) methods focus on static KGs, while temporal knowledge graphs (TKGs) have not been fully explored. Logical query over TKGs has two challenges: 1. The query should answer entities or timestamps; 2. The operators should consider both set logic on entity set and temporal logic on timestamp set. To bridge this gap, we define the multi-hop logical reasoning problem over TKGs and then propose the first temporal QE framework named Temporal Feature-Logic Embeddings (TFLEX) to answer the temporal logical queries. Specifically, we utilize vector logic to compute the logic part of the Temporal Feature-Logic embedding, thus naturally modeling all First-Order Logic (FOL) operations on the entity set. In addition, our framework extends vector logic on timestamp set to cope with three extra temporal operators (**After**, **Before** and **Between**). Experiments on numerous query patterns demonstrate the effectiveness of our method.

Contributions for RQ3

Proposing three TKG datasets and two new GNN-based TKGE models for transductive and inductive entity alignment between TKGs.

To verify whether the incorporation of time information can help with EA between TKGs and investigate how it can be done, we first generate three new TKG datasets extracted from ICEWS, YAGO and Wikidata as references for evaluating EA methods. And then we present two new TKGE models for temporal EA.

To incorporate time and relation information into GNN models, we take timestamps as attentive properties of links between entities and develop a temporal graph self-attention mechanism for KGE to specify different weights to different neighboring nodes of each entity with the corresponding link features, i.e., embeddings of the relevant relations and timestamps. Based on this self-attention mechanism, we present a novel GNN model, TEA-GNN, which can effectively and efficiently learn time information for EA between TKGs. TEA-GNN outperforms static EA methods on TKG datasets and shows great robustness against the sizes of labeled data. To the best of our knowledge, this is the first attempt to integrate time information into an embedding-based EA approach.

To further improve TEG-GNN, we present the second temporal EA model, TREA, which has not only better performances but also the ability of inductive graph representation, i.e., representing new emerging entities and timestamps. Besides, we develop a more effective temporal relational attention mechanism for TREA which is based on orthogonal transformation operations and a multi-class log-loss function is adopted for efficient training. Building on top of TEA-GNN, TREA shows better performances on both tasks of transductive EA and inductive EA between TKGs.

1.3.2 Publications

The following articles constitute a scientific basis of this thesis and serve as a reference point for numerous figures, tables and ideas presented in the later chapters. Individual contribution of the author (Chengjin Xu) to each paper is clearly mentioned in the respective chapter. The author will use the "we" pronoun throughout this dissertation, but all of the contributions and materials presented in this work originated from the works of the author solely by himself.

- *Conference Papers* (*peer reviewed*)
    1. **Chengjin Xu**, Mojtaba Nayyeri, Fouad Alkhoury, Hamed Yazdi and Jens Lehmann, "Temporal knowledge graph completion based on time series gaussian embedding", International Semantic Web Conference, pp. 654-671. Springer, Cham, 2020.
    2. **Chengjin Xu**, Mojtaba Nayyeri, Fouad Alkhoury, Hamed Yazdi and Jens Lehmann. "TeRo: A Time-aware Knowledge Graph Embedding via Temporal Rotation", Proceedings of the 28th International Conference on Computational Linguistics, pp. 1583–1593. International Committee on Computational Linguistics, 2020.
    3. **Chengjin Xu**, Yung-Yu Chen, Mojtaba Nayyeri and Jens Lehmann. "Temporal Knowledge Graph Completion using a Linear Temporal Regularizer and Multivector Embeddings", Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pp. 2569–2578. Association for Computational Linguistics, 2021.
    4. **Chengjin Xu**, Fenglong Su and Jens Lehmann. "Time-aware Graph Neural Network for Entity Alignment between Temporal Knowledge Graphs", Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, pp. 8999–9010. Association for Computational Linguistics, 2021.
    5. **Chengjin Xu**, Fenglong Su, Bo Xiong and Jens Lehmann. "Time-Aware Entity Alignment Using Temporal Relational Attention", Proceedings of the ACM Web Conference 2022, pp. 788-797. Association for Computing Machinery, 2022.
- *Journal Papers* (*peer reviewed*)
    6. **Chengjin Xu**, Yung-Yu Chen, Mojtaba Nayyeri and Jens Lehmann. "Geometric Algebra based Embeddings for Static and Temporal Knowledge Graph Completion", IEEE Transactions on Knowledge and Data Engineering. IEEE, 2022.
- *Working Drafts*
    7. Xueyuan Lin, **Chengjin Xu**, Haihong E, Fenglong Su, Gengxian Zhou, Tianyi Hu, Li Ningyuan, Mingzhi Sun and Haoran Luo. "TFLEX: Temporal Feature-Logic Embedding Framework for Complex Reasoning over Temporal Knowledge Graph", The Thirty-Sixth Annual Conference on Neural Information Processing Systems. 2022. (Under Review)
- *Miscellaneous Papers* (*peer reviewed*)
  Following publications originated during and are related to this thesis but are not part of the thesis itself,

8. Mojtaba Nayyeri, **Chengjin Xu**, Sahar Vahdati, Nadezhda Vassilyeva, Emanuel Sallinger, Hamed Shariat Yazdi, and Jens Lehmann. "Fantastic knowledge graph embeddings and how to find the right space for them." In International Semantic Web Conference, pp. 438-455. Springer, Cham, 2020.
9. Mojtaba Nayyeri, **Chengjin Xu**, Yadollah Yaghoobzadeh, Sahar Vahdati, Mirza Mohtashim Alam, Hamed Shariat Yazdi, and Jens Lehmann. "Loss-aware pattern inference: A correction on the wrongly claimed limitations of embedding models." In Pacific-Asia Conference on Knowledge Discovery and Data Mining, pp. 77-89. Springer, Cham, 2021.
10. **Chengjin Xu**, Mojtaba Nayyeri, Sahar Vahdati, and Jens Lehmann. "Multiple Run Ensemble Learning with Low-Dimensional Knowledge Graph Embeddings." In 2021 International Joint Conference on Neural Networks. IEEE, 2021.
11. Mojtaba Nayyeri, **Chengjin Xu**, Mirza Mohtashim Alam, Jens Lehmann, and Hamed Shariat Yazdi. "Logicenn: A neural based knowledge graphs embedding model with logical rules." IEEE Transactions on Pattern Analysis and Machine Intelligence. IEEE, 2021.
12. Mojtaba Nayyeri, **Chengjin Xu**, Franca Hoffmann, Mirza Mohtashim Alam, Jens Lehmann, and Sahar Vahdati. "Knowledge Graph Representation Learning using Ordinary Differential Equations." In Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, pp. 9529-9548. Association for Computational Linguistics, 2021.
13. Bo Xiong, Shichao Zhu, Mojtaba Nayyeri, **Chengjin Xu**, Shirui Pan, Chuan Zhou, and Steffen Staab. "Ultrahyperbolic Knowledge Graph Embeddings." In Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery & Data Mining, 2022.

The full list of publications completed during the PhD term is available in Appendix A.

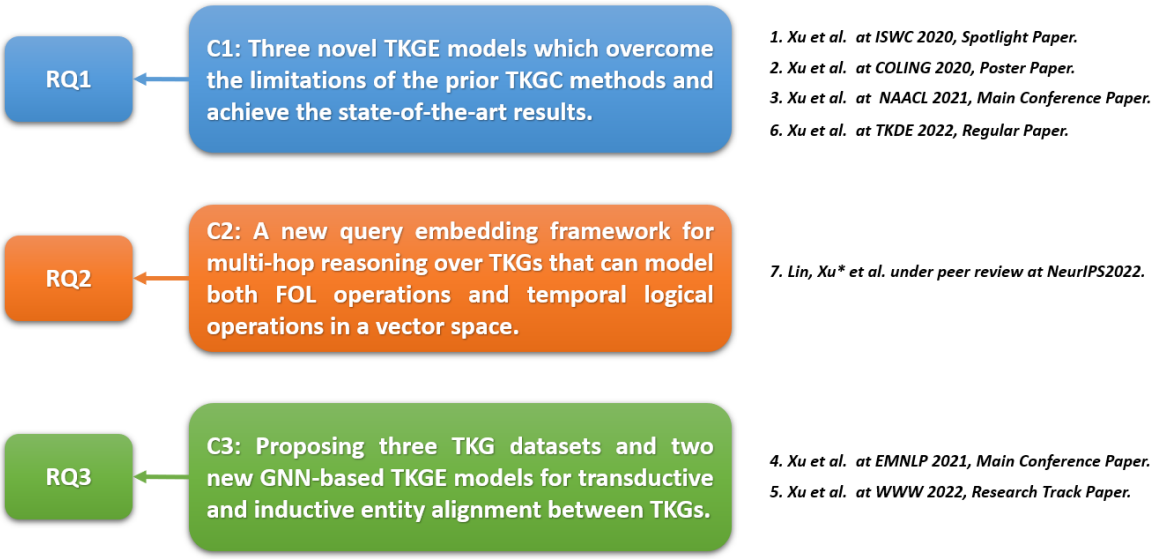


Figure 1.6: Contributions to the Research Questions and the Related Research Papers.

1.4 Thesis Structure

The structure of the thesis consists of seven chapters. Chapter 1 introduces the primary research problem of the thesis and the motivation for the research undertaken. We look into the challenges to be faced for our research goal and drive research questions to overcome the obstacles. We systematically describe the research questions, contributions towards these research questions, and a list of research papers based on the contributions. Chapter 2 provides insights into the preliminary concepts in the area of Knowledge Graphs, Distributed Representation Learning and Temporal Learning, required for understanding the research problem, the proposed approaches to address these research questions, and technical contributions of this thesis. Chapter 3 discusses state-of-the-art community efforts in various domains, e.g., Static Knowledge Graph Completion, Temporal Knowledge Graph Completion, Multi-hop Logical Reasoning over KGs, Entity Alignment between KGs and Dynamic Graph Neural Networks. Chapter 4 formally defines the task of TKGC, introduces the relevant evaluation metrics and benchmark datasets, and presents three novel TKGE models for TKGC. We deeply investigate how our presented TKGC models can address the limitations of the previous TKGC models and provide experimental proof. In Chapter 5, we first provide the problem definition of multi-hop TKG reasoning and then elaborate on a new temporal query embedding framework that can handle both FOL operations and temporal logical operations. To validate the effectiveness of our presented framework, we process three commonly used TKG datasets to sample temporal logical queries for training and evaluation. Chapter 6 describes the problem definition of entity alignment between TKGs and the relevant evaluation metrics. Then, we present three new TKG datasets extracted from ICEWS, YAGO and Wikidata as references for evaluating EA models. Importantly, we introduce two novel GNN-based TKGE models for temporal EA. We elaborate on how the incorporation of time information can be helpful to EA between TKGs in different aspects. Finally, Chapter 7 concludes the thesis with the directions for future work.

CHAPTER 2

Background

To conduct research on the problem of Temporal Knowledge Graph Embedding, it requires a comprehensive understanding of the fundamentals of relevant areas such as Knowledge Graphs, Distributed Representation Learning and Temporal Learning. In this chapter, we first introduce the basic concepts, evolutionary process and key technologies of the Knowledge Graphs. Then we look into distributed representation, which is widely applied in the field of deep learning (DL) and inspires the idea of KGE. Finally, we briefly introduce the development of the time series analysis technology to show the main methods of modeling temporal data at different periods of time.

2.1 Knowledge Graph

With the continuous evolution and development of Web technology, human beings have experienced the ”Web 1.0” era with document interconnection as the main feature and the ”Web 2.0” era with data interconnection as the main feature, and are moving towards a new knowledge-based interconnection, ”Web 3.0” era [38]. The goal of knowledge interconnection is to build a World Wide Web that is understandable to both humans and machines, making people's networks more intelligent. However, due to the heterogeneity and loose organization of the multi-source contents on the World Wide Web, it poses a great challenge to knowledge interconnection in the Big Data environment. Therefore, people need to explore knowledge interconnection methods from a new perspective based on the principles of knowledge organization in the big data environment, which can meet the developmental changes of web information resources and adapt to the cognitive needs of users, in order to reveal the wholeness and relevance of human cognition from a deeper level. With its powerful semantic processing and open interconnection capabilities, the Knowledge Graph can lay a solid foundation for knowledge interconnection on the World Wide Web, making the vision of the ”Web of Knowledge” proposed by Web 3.0 possible.

Google proposed Google Knowledge Graph in 2012, with the original intention of improving the capabilities of search engines and enhancing users' search quality as well as search experience. Currently, with the continuous development of intelligent information service applications, various KGs have been widely used in smart search, smart question answering, personalized recommendation, and other fields. In particular, in intelligent search, users' search requests are no longer limited to simple keyword matching, but will be reasoned according to the context and intent of the user's query to achieve conceptual search. At the same time, the user's search results will have important

features such as hierarchical and structured. KGs enable the computer to understand human verbal communication patterns and thus provide more intelligent feedback on the answers the user needs. At the same time, KGs allow information, data and links on the Web to be aggregated into knowledge, making information resources easier to compute, understand and evaluate, and forming a Web semantic knowledge base.

2.1.1 Development Process of Knowledge Engineering

Looking back at the development of knowledge engineering over the past forty years, and summarizing the evolutionary process and technological progress of knowledge engineering, we can divide knowledge engineering into five landmark stages, the Before Knowledge Engineering period, the Expert System period, the Web 1.0 period, the Rise of Semantic Web period and the Knowledge Graph period, as shown in Figure 2.1.

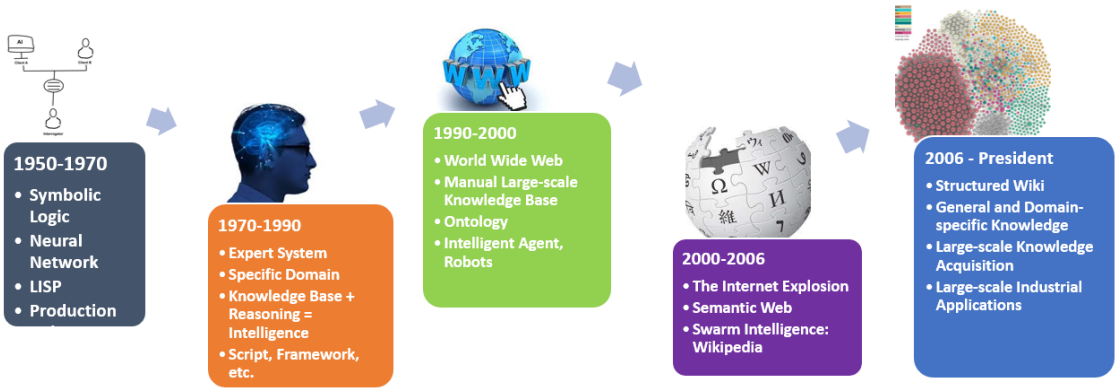


Figure 2.1: The development process of Knowledge Engineering

1950-1970: The Turing Test - The Pre-birth of knowledge Engineering

Artificial intelligence aims at enabling machines to solve complex problems as well as human beings do, and the Turing test is a means of evaluating intelligence. There are two main approaches in this phase: symbolism and connectionism. Symbolism considers the physical symbol system as a sufficient condition for intelligent behavior, while connectionism considers the brain (neurons and their connectivity mechanisms) as the basis for all intelligent activity. A representative work of this phase is the General Problem Solver (GPS): a formal representation of a problem which is searched to obtain a target state from the initial state of the problem, combined with rules or representations. Some of the most successful applications are game theory and machine theorem. The main knowledge representation methods of this period are logical knowledge representation, generative rules, semantic networks, etc.

1970-1990: The Expert Systems - The Boom of Knowledge Engineering

General Problem solver emphasized the use of human problem-solving capabilities to build intelligent systems, while ignoring the support of knowledge for intelligence, making it difficult for artificial

intelligence to work in practical applications. In 1970s, artificial intelligence began to shift to building knowledge-based systems, through the ”knowledge base + reasoning machine” to achieve machine intelligence. Many successful domain-specific expert systems emerged during this period [39]. The concept of knowledge engineering was introduced in a project report [40] in 1980, which has since established the central role of knowledge engineering in artificial intelligence. The late 1980s saw the emergence of a number of platforms for the development of expert systems that could help translate experts' domain knowledge into computer-processable knowledge.

1990-2000: The World Wide Web 1.0

From 1990 to 2000, many manually constructed large-scale knowledge bases emerged, including the widely used WordNet [41] and the Cyc common sense knowledge base [42] which used first-order predicate logic knowledge representation. The creation of the Web 1.0 provided an open platform for people to define the contents of texts using HTML and connect the text through hyperlinks, making it possible for the public to share information. The W3C proposed the Extensible Markup Language XML to structure the content of Internet documents by defining tags, laying the foundation for large-scale knowledge representation. Knowledge representation methods for ontologies were also proposed in this period in knowledge representation research.

2000-2006: The Rise of Semantic Web

The concept of Semantic Web was first proposed in 2000 [43]. The Semantic Web aims at a structured semantic representation of Internet content, using ontologies to describe the semantic structure of Internet content, and getting semantic information of web pages. The W3C further proposed the knowledge description specifications for describing the semantic meanings of the contents of the World Wide Web, such as the semantic markup language RDF (Resource Description Framework) and OWL (World Wide Web Ontology Representation Language) on the World Wide Web. The emergence of the World Wide Web has enabled knowledge to move from closed knowledge to open knowledge, and from centrally constructed knowledge to distributed group intelligence knowledge. The original expert system was the knowledge defined within the system, but now it can realize the interlinking of knowledge sources, and more knowledge can be produced by association rather than entirely by specific people. The most typical representative of this process, in which group intelligence emerges, is Wikipedia, which is actually built by users. It reflects the contribution of Internet mass users to knowledge and has become an important foundation of today's large-scale structured knowledge graph.

2006 - Present: Knowledge Graph - New Development Period of Knowledge Engineering

Transforming World Wide Web content into machine-understandable and computable knowledge capable of powering intelligent applications is the goal of this period. Starting in 2006, the emergences of large-scale Wikipedia-like rich-structure knowledge resources and advances in web-scale information extraction methods have led to tremendous progress in large-scale knowledge acquisition methods. Unlike the pioneering projects of manually developed knowledge bases and ontologies such as Cyc and WordN, knowledge acquisition in this period is automated and operated at web scale. Currently, automatically-built knowledge bases have become powerful assets for semantic search, big data analytics, intelligent recommendations, and data integration, and are being widely used in

large industries and various domains. Typical examples include Google's Knowledge Graph [44], Facebook's Graph Search [45], Microsoft Satori [46], and domain-specific knowledge bases in business, finance, and life sciences. The most representative large-scale web knowledge acquisition efforts include DBpedia [4], Freebase [3], KnowItAll [47], WikiTaxonomy [48], and YAGO [8], as well as BabelNet [49], ConceptNet [50], DeepDive [51], NELL [52], Probase [53], Wikidata [54], XLORE [55], Zhishi.me [56], CNDBpedia [57], etc. These knowledge graphs (KGs) follow the RDF data model and contain tens of billions of entities in scale, and billions or tens of billions of facts (i.e., attribute values and relationships with other entities), and these entities are organized in thousands of conceptual structures of the objective world embodied by semantics. Noteworthily, facts in some KGs contain time information, e.g., YAGO, Wikidata, ICEWS [10], and GDELT [11]. Such KGs are called temporal KGs (TKGs).

2.1.2 Definition and Architecture of Knowledge Graph

Essentially, a knowledge graph (KG) is a semantic network that reveals relationships between entities and allows for a formal description of real-world things and their interrelationships. Knowledge graphs are now used to refer to various large-scale knowledge bases in general.

Triple is a general representation of the knowledge graph. Formally, a TKG is represented as $\mathcal{G} = (\mathcal{E}, \mathcal{R}, \mathcal{F})$ where $\mathcal{E}$, $\mathcal{R}$ and $\mathcal{F} \subseteq \mathcal{E} \times \mathcal{R} \times \mathcal{E}$ are the sets of entities, relations, and triple-based facts, respectively. The basic forms of the triples mainly include (*subject entity*, *relation*, *object entity*) and (*concept*, *attribute*, *attribute value*), etc. Entity is the most basic element in the knowledge graph, and different relationships exist between different entities. Concepts mainly refer to collections, categories, object types, and kinds of things, such as people, geography, etc.; attributes mainly refer to characteristics, features, and parameters that objects may have, such as nationality, birthday, etc.; attribute values mainly refer to the values of specified attributes of objects, such as Germany, 1991-12-29, etc. Each entity (an extension of a concept) can be identified by a globally unique ID, each attribute-value pair (AVP) can be used to characterize the intrinsic properties of an entity, and a relation can be used to connect two entities and characterize the association between them.

Knowledge graphs can also be divided into general KGs and domain-specific KGs. General KGs focus on breadth and emphasize the integration of more entities, which are less accurate than domain-specific KGs and are influenced by the scope of concepts, and it is difficult to regulate their entities, attributes, relations among entities, etc., although with the support ability of ontology libraries for axioms, rules and constraints. General KGs are mainly applied in fields such as intelligent search. Domain-specific KGs usually need to rely on domain-specific data to be constructed with domain-specific significance. In domain-specific KGs, the attributes and data patterns of entities are often rich, and different application scenarios and users need to be considered.

The architecture of a KG mainly includes its logical architecture as well as its technical architecture which are explained as follows respectively.

- Logical structure: A KG can be logically divided into two levels: schema layer and data layer, and the data layer mainly consists of a series of facts, while the knowledge will be stored in facts. If facts are expressed in triples like (*subject entity*, *relation*, *object entity*) or (*entity*, *attribute*, *attribute value*), graph databases can be chosen as storage media, such as the open source Neo4j [58] and Twitter's FlockDB [59]. The schema layer is built on top of the data layer, mainly through an ontology library to regulate a set of factual representations in the data layer.

Ontologies are the conceptual templates of structured knowledge bases, and the knowledge bases formed through ontology libraries are not only stronger in hierarchical structure but also less redundant.

- Technical architecture: The technical architecture of a KG refers to the structure of its construction workflow, as shown in Figure 2.2. The parts in the dashed boxes are the construction process of a KG, which needs to be constantly updated and iterated. There are two main ways to build a knowledge graph, top-down and bottom-up. Top-down refers to defining the ontology and data schema for the knowledge graph first, and then adding the entities to the knowledge base. This approach requires using some existing structured knowledge bases as its basic knowledge base, such as the Freebase project, which gets most of its data from Wikipedia. Bottom-up refers to extracting entities from some open-linked data, selecting the ones with higher confidence to add to the knowledge base, and then constructing the top-level ontology schema. At present, an increasing number of knowledge graphs are constructed using the bottom-up approach, the most typical of which is Google's Knowledge Vault.

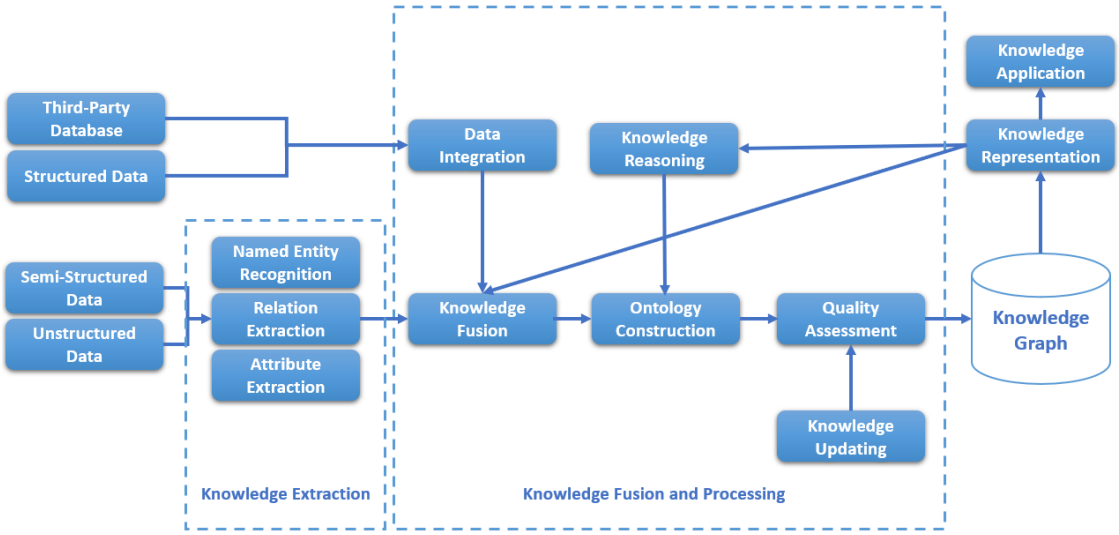


Figure 2.2: The technical architecture of a knowledge graph

2.1.3 Key Technologies of Knowledge Graph

At present, more and more KGs are constructed in a bottom-up manner, and the bottom-up knowledge graph construction technology can be divided into three levels according to the process of knowledge acquisition: knowledge extraction, knowledge fusion and knowledge processing. Another key technology of KG is knowledge representation which can effectively aid knowledge fusion and knowledge processing.

Knowledge Extraction

Knowledge extraction is mainly oriented to open linked data and extracts usable knowledge units through automated technology. The knowledge units mainly include three knowledge elements: entities (extents of concepts), relations and attributes, and based on them, a series of high-quality factual expressions are formed to lay the foundation for the construction of the upper pattern layer.

- Entity Extraction: Entity extraction, also known as named entity recognition, refers to the automatic recognition of named entities from the original corpus. Since entities are the most basic elements in the knowledge graph, the completeness, accuracy and recall of their extraction will directly affect the quality of the KG. Therefore, entity extraction is the most basic and critical step in knowledge extraction. The literature [60] classifies the methods of entity extraction into four categories: rule-based and dictionary-based methods, unsupervised learning methods, feature-based supervised learning methods and deep learning-based methods. Rule-based methods usually require writing templates for target entities and then matching them in the original corpus; unsupervised learning methods extract named entities from the clustered groups based on context similarity; feature-based supervised learning methods mainly train the original corpus by machine learning methods and then use the trained models to identify entities; deep learning-based methods automatically discover representations needed for the classification and/or detection from raw input in an end-to-end manner.

- Relation Extraction: The goal of relation extraction is to solve the problem of semantic links between entities. The early relation extraction is mainly through the manual construction of semantic rules and template methods to identify relations between entities. Subsequently, the relation models between entities gradually replaced the manual predefined syntax and rules. However, it is still necessary to define the types of relationships between entities in advance. The literature [61, 62] proposed an open information extraction (OIE) framework for open domain relation extraction, which is great progress in the extraction model. However, the OIE method has low performance in extracting the implicit relationships of entities, so some researchers [63, 64] have proposed a joint inference-based entity relation extraction method to extract deep implicit relations.

- Attribute Extraction: Attribute extraction is mainly for entities, through which attributes can form a complete sketch of the entity. Since the attributes of an entity can be viewed as a nominal relation between entities and attribute values, it is feasible to convert the entity attribute extraction problem into a relation extraction problem.

Knowledge Fusion

The goal of obtaining entity, relationship, and entity attribute information from unstructured and semi-structured data is achieved through knowledge extraction. However, these results may contain a large amount of redundant and erroneous information, and the relationships between data are flat and lack hierarchy and logic, so it is necessary to clean and integrate them. Knowledge fusion consists of two parts: entity linking and entity alignment.

- Entity Linking: Entity linking refers to the operation of linking the entity objects extracted from the text to the corresponding correct entity objects in the knowledge base. The basic idea of

entity linking is to first select a set of candidate entity objects from the knowledge base based on the given entity mentions, and then link the entity mentions to the correct entity object by similarity calculation. The general process of entity linking is as follows: 1. obtain entity mentions from text by entity extraction; 2. perform entity disambiguation and mention resolution to determine whether the entities with the same names in the knowledge base represent different meanings and whether entities with different names in the knowledge base that represent the same meanings; 3. link the entity mentions to the corresponding entity objects in the knowledge base after confirming the correct corresponding entity object.

- Entity Alignment: When building a KG, knowledge input can be obtained from third-party knowledge bases or existing structured data. Entity alignment, also known as entity matching, is mainly used to eliminate inconsistencies such as conflicting entities and unclear pointers in heterogeneous data, thus helping machines understand heterogeneous data from multiple sources and form high-quality knowledge. The main processes of entity alignment between knowledge bases will include three steps: 1. indexing the data to be aligned in partitions to reduce the computational complexity; 2. finding matching instances using a similarity function or similarity algorithm; 3. performing instance fusion using an entity alignment algorithm; and 4. combining the results of step 2 with step 3 to form the final alignment result.

Knowledge Processing

Through knowledge extraction, knowledge elements such as entities, relations and attributes can be extracted from the original corpus, and then through knowledge fusion, the ambiguity between entity mentions and entity objects can be eliminated, and a series of basic factual expressions can be obtained. However, facts themselves are not equal to knowledge, and to finally obtain a structured and networked knowledge system, it is necessary to go through the process of knowledge processing. Knowledge processing mainly includes four aspects: ontology construction, quality assessment, knowledge updating, and knowledge reasoning.

- Ontology Construction: ontology is the semantic basis for communication and connectivity among different subjects in the same domain [65], which mainly presents a tree-like structure with strict "IsA" relation between adjacent hierarchical nodes or concepts, which is conducive to constraint and reasoning, but not conducive to expressing the diversity of concepts. The knowledge graphs formed by ontology library are not only stronger in hierarchical structure but also less redundant [66]. Ontologies can be constructed manually by human editing or automatically by data-driven construction, and then revised and confirmed by a combination of quality assessment methods and human review. In the face of the huge amount of entity data, the workload of manual editing and construction is extremely huge, so the current mainstream ontology repository products are gradually expanded for specific fields using automatic construction technology.

- Quality Assessment: The task of quality assessment is usually performed together with the knowledge fusion task. This task aims at quantifying the confidence of knowledge, retain those with higher confidence and discard those with lower confidence, which effectively ensures the quality of knowledge.

- Knowledge Updating: Logically, the update of the knowledge base includes the update of the concept layer and the update of the data layer. The update of the concept layer refers to the new concepts obtained after adding new data, and the new concepts need to be automatically added to the concept layer of the knowledge base. The update of the data layer mainly adds or updates the values of entities, relations, and attributes. And the update of the data layer needs to consider the reliability of data sources and the consistency of data (whether there are problems of contradiction or redundancy), and select the facts and attributes that are added to the knowledge base.

- Knowledge Reasoning: Most KGs are incomplete. Knowledge reasoning refers to establishing new associations among entities from the existing entity-relation data in the knowledge base, so as to expand and enrich the knowledge network. Knowledge reasoning is an important technology in the construction of KGs, and through knowledge inference, new knowledge can be discovered from the existing knowledge. The classical methods of knowledge reasoning can be divided into two main categories: logic-based reasoning and graph-based reasoning. Logic-based reasoning mainly includes first-order logic predicates, description logic, and rule-based reasoning. Graph-based reasoning methods are mainly based on neural network (NN) models or Path Ranking algorithms. Recently, more and more researches focus on knowledge reasoning based on KG embeddings.

Distributed Representation Learning for Knowledge Graphs

Although the triple-based knowledge representation form has been widely recognized, it faces many problems in terms of computational efficiency, data sparsity, etc. In recent years, distributed representation learning techniques have made important progress [67, 68]. Distributed representation learning of KGs which represents the semantic information of entities as dense low-dimensional real-valued vectors, can be used to efficiently compute the semantic interactions between entities and relations in the low-dimensional space. Such distributed representation learning is very important for the construction, reasoning, fusion, and applications of KGs.

- Semantic Similarity Measurement: Since entities are formed as a single low-dimensional real-valued vector through distributed representation, the similarity between them can be measured using the entropy weight coefficient method, cosine similarity, or other methods. This similarity portrays the degree of semantic association between entities which provides great convenience for natural language processing, etc.

- Link Prediction: With the distributed representation model, it is possible to predict the relation between any two entities in the graph, as well as the correctness of the relations that already exist between the entities. Especially in the context of large-scale knowledge graphs, the relations of entities in them need to be constantly supplemented, so link prediction is also known as knowledge graph completion (KGC).

2.2 Distributed Representation

Distributed representation, i.e., embedding, is the technique of representing a data sample, such as a word, a sentence, a node in a graph network or an entity in a knowledge graph, as a dense,

low-dimensional vector. Distributed representations are widely used in natural language processing (NLP) and graph learning. In this section, we give a brief introduction to word embedding and graph embedding.

Word Embedding

In the whole history of NLP technology, how to numerically represent words has been a hot research topic. In recent years, low-dimensional word representation vectors trained with massive unannotated text data, also known as word embeddings [69, 70], have shown effectiveness in many tasks including POS tagging [71], syntactic parsing [72], named entity recognition [73], machine translation [74], and so on. However, such word embeddings are static, because they no longer change with new contexts as soon as the training process is completed. Although static word embedding is highly efficient, its static nature property makes it difficult to cope with the problem of multiple meanings of words, because the meaning of a word depends on its context. To deal with this problem, researchers have recently proposed a number of methods to dynamically learn the meaning of words based on their context. For example, given two sentences "A girl sat on the river bank" and "A bank is robbed", static word embeddings cannot distinguish the semantic difference between these two "bank"s, while dynamic word embeddings can give different representations according to the context. Obviously, such dynamic word embeddings extracted from pre-trained language models (LMs) [75–78] can perform better in many natural language processing tasks than the previous static word embeddings.

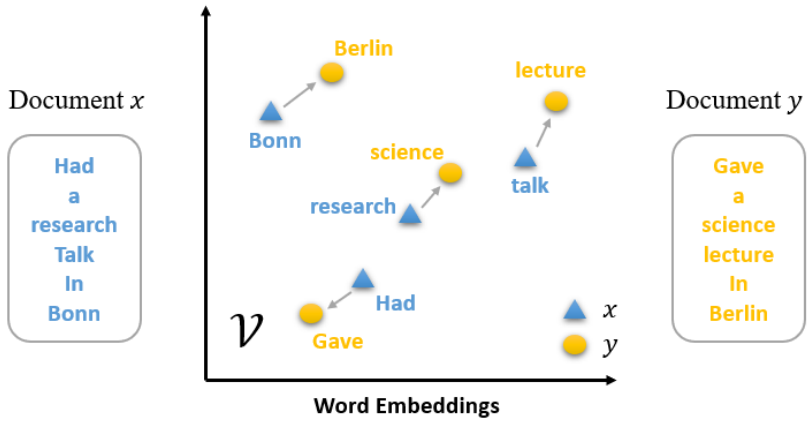


Figure 2.3: An illustration of 2-dimensional word embeddings

In the pre-birth of word embeddings, sparse and high-dimensional vectors are mainly used to represent words. The most classical one is the one-hot representation, where each word corresponds to a high-dimensional vector, and all the bits in the vector are "0" except one is "1". Such word vectors are all orthogonal to each other, so it is naturally impossible to measure the semantic distance between words. This representation system suffers from the problems of sparse data and high dimensionality, and the dimensionality of the word vectors is usually as large as the vocabulary of the system, which makes it difficult to use.

To solve these problems, researchers train dense low-dimensional vectors with large amounts of text data instead of high-dimensional vectors. The following are a few representative word representations in this stage:

- Neural network language models (NNLM) [6]: The use of deep neural networks to generate distributed word vectors has ushered in a new era, and is a good solution to the first stage of the ”data sparsity” problem (i.e., sequences of words may appear in the test set that does not exist in the training set). The training goal of the model is to predict the next possible word given a sequence of words.

- SENNA [79]: SENNA is also a NN-based model, but its training goal is to determine the acceptability of a piece of text. This goal is simpler and more feasible than predicting the probability of a word's occurrence.

- CBOW and Skipgram models [7]: The innovative design and simplification of the network architecture have resulted in a milestone in the history of distributed representations, as the computational complexity of the CBOW and Skipgram models has been significantly reduced. The most popular implementation of CBOW and Skipgram is the familiar Word2Vec.

- GloVe [70] and fastText [80, 81]: have made a big impact in the explosion of word embedding models. The former captures more global information and makes better use of the frequent co-occurrence of certain words; the latter takes into account spelling similarities between different words and once again improves training speed dramatically.

Although these low-dimensional dense distribution representations presented above have been very successful in the NLP domain, they are somewhat powerless for the word polysemy problem. It is clear that a word is represented by a prototype vector, which does not change with context. An intuitive way to solve this problem is to use multiple prototype vectors to represent a word (with different word meanings). Based on this idea, a multi-prototype vector space model [82] was proposed to generate multiple vectors of specific meanings for each word by clustering. This idea of multiple prototypes has also been widely used to learn sense-level embeddings. For example, Huang et al. [83] used multi-prototype representation vectors in the SENNA architecture to achieve good results.

Another way to solve the problem of multiple meanings which is more effective is to use dynamic representations, or so-called ”contextual embeddings”, where the representation changes with the context. The following are a few typical dynamic word representation models:

- CoVe [76]: This is the first attempt to generate different word representations depending on the contextual content. They trained a deep long short-term memory (LSTM) encoder on a sequence-to-sequence machine translation task, and then used it to generate word embeddings that vary according to context, and then applied these word embeddings in a downstream task. The design of this model is simple and straightforward, but it has led to improvements in many tasks and has opened up the route to dynamic representation.

- ELMo [75]: Compared to CoVe, the training of ELMo no longer required bilingual data, allowing it to directly exploit an almost infinite amount of unlabeled text, and its significant success in downstream tasks has sparked attention throughout the NLP research field. From a technical perspective, training a deep bidirectional LM on a large-scale unlabeled corpus and then extracting representations from its internal layers is the EMLo representation.

- ULMFit [84]: ULMFit was also an attempt at improvement based on LSTM. Its technical highlights included discriminative fine-tuning, slanted triangular learning rates, and gradual

unfreezing, which helped the model to better adapt to the target task during the fine-tuning phase, and thus lead other models at that time by a large margin.

- GPT [78]: The learning capability of LSTM is relatively limited. Thus, ELMo and ULMFit, which employ LSTM, cannot handle the dependencies in long sequences well. After the attention-based Transformer [85] model was proposed, the GPT model with Transformer as the core performed well and further demonstrated the effectiveness of LM pre-training and context-based word characterization.

- BERT [77]: BERT, also based on Transformer and considering both left-to-right and right-to-left order pre-training models, is undoubtedly the most frequently compared and discussed model throughout 2019. BERT performed extremely well, and proposed word masking and next sentence prediction (NSP), two new unsupervised pretraining tasks, which have also inspired many later researchers. There are also a large number of improved models based on BERT.

- XLNet [86]: Some researchers argued that BERT's masking approach introduces new drawbacks, and it was also argued that BERT suffers from undertraining (insufficient convergence). XLNet redesigned many details of the pretraining process, used Transformer-XL as its basic model [87], and once again set a new performance record for pretrained word embeddings.

Name	bjective	Model	Technique
NNLM	LM	Feed-forward NN	Directly learn low-dimensional, dense and continuous vectors
SENNA	LM	Feed-forward NN	Pretrain embedding on unlabeled data to benefit downstream tasks
CBOW& Skip-gram	LM	Logistic regression	Accelerate computation by removing hidden layer
GloVe	LM	Logistic regression	Additionally leverage word co-occurrence information
fastText	LM	Logistic regression	Consider morphology by representing words with n-gram characters Leverage tricks to accelerate training
CoVe	Translation	LSTM	Propose contextual embeddings using cross-lingual corpus
ELMo	LM	Bi-LSTM	Consider bidirectional context
ULMFit	LM	LSTM	Propose techniques to improve fine-tuning
GPT	LM	Transformer	Use transformer to model long context
BERT	Masked LM & NSP	Transformer	Consider bidirectional contexts and relation between sentence pairs
XLNet	Permuted LM	Transformer-XL	Use permuted LM to remove pretrain-finetune discrepancy Use Transformer-XL to model long contexts
RoBERTa [88]	Maksed LM	Transformer	Leverage training tricks to exploit potential of BERT
ELECTRA [89]	Replaced LM	Transformer	Improve training efficiency with harder training objectives

Table 2.1: Summary of word embedding models, which are generally arranged in chronological order [90].

Graph Embedding

Many complex systems in the real world can usually be modeled as graph structures [91], with nodes or node attributes in the graph representing entities or entity labels in real network systems, and edges in the graph representing relations between entities in real networks. And the structure and properties of real network systems can be better analyzed by using graph embeddings (GEs). Graph embeddings have two forms:

- Represent nodes in a graph as low-dimensional, real-valued, dense vectors, so that the obtained node vectors can have representation as well as inference capabilities in the vector space, and

such vectors can be used in specific downstream tasks. For example, user social networks get node representations that are representation vectors for each user, which are then used for node classification, etc.

- Represent the entire graph as a low-dimensional, real-valued, dense vector form, which is used to classify the entire graph structure

GE methods can mainly be classified into three categories:

- Matrix Decomposition: The matrix-based decomposition methods express the relationship between nodes in the form of a matrix, and then decompose this matrix to obtain the embedding vector. The matrices that are usually used to represent node relationships include adjacency matrix, Laplace matrix, node transfer probability matrix, node attribute matrix, etc. Different decomposition strategies are applied depending on the nature of the matrices.

- Random Walk: Random walk-based graph embeddings optimize the embeddings of nodes by making the nodes co-occurring in a short-range random walk on the graph have similar representations. DeepWalk [92] is a typical random walk-based GE method that is based on Word2Vec. Word2Vec takes the corpus as input data when training word vectors, while graph embedding takes the whole graph as input. The DeepWalk authors found that the number of occurrences of words in the corpus and the number of random wandering nodes visited on the graph both obeyed a power-law distribution. Therefore, DeepWalk treats nodes as words and the sequence of nodes obtained by random wandering as sentences, and then uses them directly as input to word2vec to obtain the embedding representation of nodes. Also using the embedding representation of nodes as initialization parameters for downstream tasks can be well optimized for downstream tasks and has spawned a lot of related work.

- Graph Neural Network: Networks built based on graphs and deep learning methods are collectively called GNN. GNNs can be applied to graph embedding to get the vector representation of the graph or graph nodes. Since 2015, GNN has attracted a lot of attention and it is widely studied and applied in various fields [93].

Usually, a graph can be denoted as $\mathcal{G}$, the set of nodes (vertices) in the graph can be denoted as $\mathcal{V}$, and the adjacency matrix is denoted as $A \in \mathcal{R}^{|\mathcal{V}\times\mathcal{V}|}$ where $A_{ij} = 1$ if nodes $v_i, v_j \in \mathcal{V}$ are linked, otherwise $A_{ij} = 0$.

Graph Neural Network (GNN) was first proposed by Scarselli et al [94]. A node in a graph can be defined by its features $\mathbf{x}_v$ and related nodes, and the goal of GNN is to learn a state embedding $\mathbf{h}_v$, which is used to represent the neighborhood information of each node. The state embedding can generate the output vector $\mathbf{o}_v$ for use as a prediction of the distribution of node labels, etc. To update the state of a node according to its neighbors, a GNN defines a function for all nodes, called the local transition function and another function to generate the output of a node, called the local output function:

$$\begin{aligned} \mathbf{h}_v &= f(\mathbf{x}_v, \mathbf{x}_{co[v]}, \mathbf{h}_{\mathcal{N}_v}, \mathbf{x}_{\mathcal{N}_v}), \\ \mathbf{o}_v &= g(\mathbf{h}_v, \mathbf{x}_v), \end{aligned} \tag{2.1}$$

where $\mathcal{N}_v^r$ is the set of edges involving the node v and $\mathcal{N}_v$ is the set of nodes neighboring to v. In the case of Figure 2.4, $\mathcal{N}_{v_1}^r$ contains the edges $r_{1,4}$, $r_{6,1}$, $r_{1,2}$ and $r_{3,1}$, $\mathcal{N}_{v_1}$ contains the vertices v_2, v_3, v_4 and v_6.

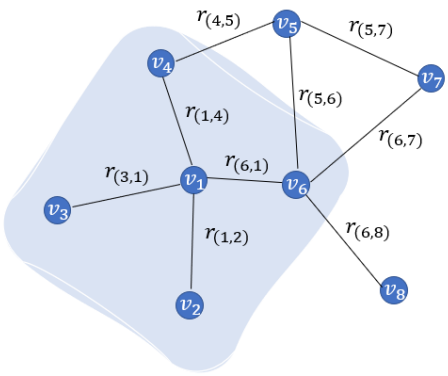


Figure 2.4: An example of a graph network

Let $\mathbf{H}$, $\mathbf{O}$ and $\mathbf{X}$ denote the state vectors, output vectors, feature vectors of all vertices,

$$\begin{aligned} \mathbf{H} &= F(\mathbf{H}, \mathbf{X}), \\ \mathbf{O} &= G(\mathbf{H}, \mathbf{X}), \end{aligned} \tag{2.2}$$

where F is the global transition function, G is the global output function. A GNN updates the state matrix in an iterative way:

$$\mathbf{H}^{l+1} = F(\mathbf{H}^l, \mathbf{X}), \tag{2.3}$$

where $\mathbf{H}^l$ denotes the state matrix in the lth iteration.

The graph convolutional network (GCN) [95] applies the convolutional operations used for traditional data (e.g., images) to graph-structured data. The core idea lies in learning a function f that obtains a representation of a node v_i by aggregating its own features $\mathbf{x}_i$ and those $\mathbf{x}_j$ of its neighbors $v_j \in \mathcal{N}_{v_i}$. The recursive formula between the adjacent layers of a GCN is

$$\mathbf{H}^{l+1} = \sigma(\hat{D}^{-\frac{1}{2}}\hat{A}\hat{D}^{-\frac{1}{2}}\mathbf{H}^l\mathbf{W}^l), \tag{2.4}$$

where $\hat{A} = A + I_{|\mathcal{V}|}$, $\hat{D}$ is the degree matrix of $\hat{A}$, i.e., $\hat{d}_{ii} = \sum_j \hat{a}_{ij}$, $\mathbf{W}^l$ is the lth-layer weight matrix and σ denotes the activation function, e.g., sigmoid or ReLU.

GraphSAGE [96] improves the GCN to generate the embedding vectors of new nodes or new sub-graphs by using the feature information of nodes, graph sampling and aggregation, which can achieve the incremental update of node embeddings, and maintain the feature information and structural information of the graph. The update function in the lth layer is defined as,

$$\begin{aligned} \mathbf{h}^l_{\mathcal{N}_v} &= \text{AGGREGATE}(\{\mathbf{h}^{l-1}_u, \forall u \in \mathcal{N}_v\}), \\ \mathbf{h}^l_v &= \sigma\left(\mathbf{W}^l \cdot \left[\mathbf{h}^{l-1}_v || \mathbf{h}^l_{\mathcal{N}_v}\right]\right), \end{aligned} \tag{2.5}$$

where $||$ denotes the concatenation operator, and the aggregation function can be an Average operator.

In contrast to GCN, which treats all neighbors of a node equally, the attentional mechanism can assign different attention scores to each neighbor and thus identify more important neighbors. Graph attention network (GAT) [97] introduces the attentional mechanism into the propagation process by following the self-attention mechanism and updating the implicit state by paying different attention to

each node's neighbors. GAT defines a graph attentional layer, which constructs a graph attentional network by stacking. For node pairs (v_i, v_j), the coefficients based on the attention mechanism are calculated as follows,

$$\alpha_{ij} = \frac{\exp\big(\text{LeakyReLU}\big(a^\top \big[\mathbf{W}\mathbf{h}_i || \mathbf{W}\mathbf{h}_j\big]\big)\big)}{\sum_{v_m \in \mathcal{N}_{v_i}} \exp\big(LeakyReLU\big(a^\top \big[\mathbf{W}\mathbf{h}_i || \mathbf{W}\mathbf{h}_m\big]\big)\big)}, \tag{2.6}$$

where α_{ij} denotes the attention coefficient of v_j to v_i, $\mathbf{h}_i \in \mathbb{R}^k$ denotes the k-dimensional feature vector of v_i, $\mathbf{W} \in \mathbb{R}^{k' \times k}$ denotes the shared linear transformation matrix, and $a \in \mathbb{R}^{k'}$ is the attention vector which has the same dimension k' as the output feature vector. The output feature vector can be computed as follows,

$$\mathbf{h}'_i = \sigma\Big(\sum_{v_j \in \mathcal{N}_{v_i}} \alpha_{ij} \mathbf{W}\mathbf{h}_j\Big). \tag{2.7}$$

The attention layer uses a multi-headed attention mechanism to stabilize the learning process, and then applies M independent attention mechanisms to compute the hidden states, and finally obtain the output representation by splicing or averaging as follows,

$$\mathbf{h}'_i = ||_{m=1}^{M} \sigma\Big(\sum_{v_j \in \mathcal{N}_{v_i}} \alpha_{ij}^k \mathbf{W}^k \mathbf{h}_j\Big) \quad or \quad \sigma\Big(\frac{1}{M}\sum_{m=1}^{M}\sum_{v_j \in \mathcal{N}_{v_i}} \alpha_{ij}^k \mathbf{W}^k \mathbf{h}_j\Big), \tag{2.8}$$

where α_{ij}^m denotes the normalized attention coefficient computed from the mth attention mechanism.

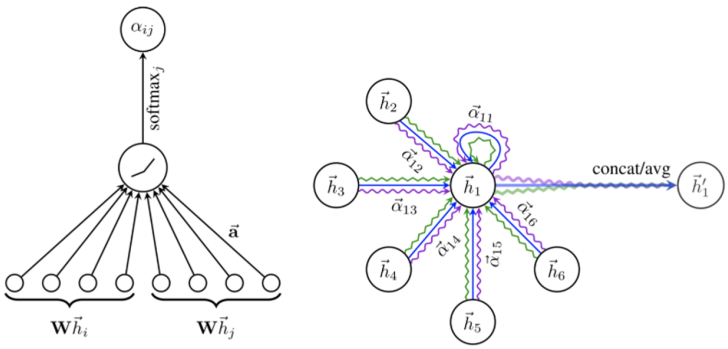


Figure 2.5: The illustration of the graph attention layer, taken from [97]

The structure of the graph attention layer is shown in Figure 2.5. And the attention architecture in GAT has several characteristics:

- The computation against node pairs is parallel, so the computation process is efficient.
- GAT can handle nodes of different degrees and assign corresponding weights to different neighbors.
- GAT can be easily applied to inductive learning problems.

GNNs have been applied in several fields such as image classification, visual question answering, text classification, relation reasoning and so on [98]. Knowledge graphs can be regarded as a type of

relational directed graphs. Thus, relational GNNs are applied to the learning tasks of KGs, e.g., KG completion and entity alignment.

2.2.1 Time Series Analysis

Generally, a time series is a set of random variables ordered by time, which is usually the result of an observation of a potential process at a given sampling rate over an equally spaced period of time [99]. A time series essentially reflects the change patterns of one or multiple random variables over time, and the core of time series analysis is to extract these patterns from the data and use them to make estimates of unobserved data.

Time Series Decomposition

The main difference between time series data and other types of data is that the data values at the current moment are relevant to the data values at the previous moment, and this feature indicates that the past data have implied the change patterns of the present or future data development, and these patterns mainly include trend, seasonality and randomness. The trend reflects the development direction of the time series over a long period of time, which can be expressed as a continuous upward or continuous downward or smooth trend over a considerable period of time. The seasonality reflects that the time series is influenced by various periodic factors to form a fixed length and amplitude of the periodic fluctuations. The randomness reflects the non-trendy and non-periodic irregularity of the time series affected by various unexpected events and chance factors.

The above three patterns are the results of the decomposition of the changes in the values of the time series. Sometimes these changes appear inside a time series at the same time, and sometimes only one or two of them may appear, which is determined by the influencing factors that cause the various changes. The relationship between the three variations and the final change in the indicator values may be additive or multiplicative.

Assuming that a time series is obtained by summing multiple components, it can be described in the following additive form:

$$x_t = T_t + S_t + R_t, \tag{2.9}$$

where $y(t)$ denotes the time series data, T_t, S_t and R_t denote the trend component, seasonal component and random component. In addition, the time series can also be written in a multiplicative form,

$$y(t) = T_t \times S_t \times R_t. \tag{2.10}$$

If the magnitude of seasonal fluctuations does not vary with the level of the time series, then an additive model is most appropriate. When the change in the seasonal component is proportional to the level of the time series, then a multiplicative model is more appropriate.

Figure 2.7 shows an example of the additive time series decomposition of a time series data.

Linear Time Series Model

As seen in the previous section, a time series is generally decomposed into trend and seasonal effects, and a remaining stochastic component, following a decomposition model: additive, multiplicative. In this section, we will focus on how the remaining stochastic component can be modeled.

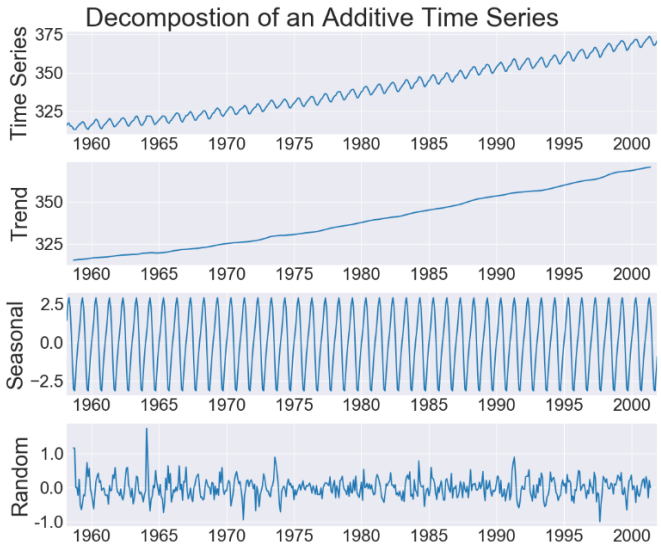


Figure 2.6: The illustration of additive time series decomposition.

Time series data have two properties, i.e, autoregression and stationarity/non-stationarity. Autoregression is a property unique to time series and is expressed as the dependence of the current observation of a time series on its previous observations. The stationarity of a time series indicates that the mean and variance of the time series do not vary systematically across time, while non-stationarity implies that the mean and variance change over time. In other words, the stationarity of a time series ensures that the essential features of the time series do not only exist at the current moment, but also extend into the future.

In a multiple linear regression model, we predicted the target variable by a linear combination of multiple predictor variables. In an autoregressive (AR) model, we predict the target variable based on the combination of historical data of the target variable. Autoregression means that it is a regression on the variables themselves. Thus, a p-order AR model can be expressed as follows,

$$x_t = c + \phi_1 x_{t-1} + \phi_2 x_{t-2} + \cdots + \phi_p x_{t-p} + \epsilon_t, \tag{2.11}$$

where ϵ_t denotes the white noise. This corresponds to a multiple regression replacing the predictor variables with the historical values of the target variables.

Unlike autoregressive models that use historical values of predictor variables for regression, moving average (MA) models use historical forecast errors to build a regression-like model as follows,

$$x_t = c + \epsilon_t + \theta_1 \epsilon_{t-1} + \theta_2 \epsilon_{t-2} + \cdots + \theta_q \epsilon_{t-q}. \tag{2.12}$$

This model is called the MA(q) model. Since we do not make observations of $\epsilon(t)$, this is not really a linear model in the general sense.

When we combine the differential and autoregressive models with the moving average model, we can obtain an autoregressive integrated moving average (ARIMA) model. The ARIMA model is

represented as follows,

$$y'_t = c + \phi_1 y'_{t-1} + + \cdots + \phi_p y'_{t-p} + \epsilon_t + \theta_1 \epsilon_{t-1} + \cdots + \theta_q \epsilon_{t-q}, \tag{2.13}$$

where y'_t is the differential sequence (it may undergo multiple differentiation operations). We refer to this model as ARIMA(p, d, q) model with the following parameter meanings,

- p: The order of the autoregressive term.
- d: The order of differentiation.
- q: The order of the moving average term.

Some special cases of ARIMA model are listed,

- ARIMA$(0, 0, 0)$: White noise.
- ARIMA$(0, 1, 0)$ with $c = 0$: Random walking model.
- ARIMA$(0, 1, 0)$ with $c \neq 0$: Random walking model with shift.
- ARIMA$(p, 0, 0)$: Autoregressive model.
- ARIMA$(0, 0, q)$: Moving average model

Deep Learning Models

Many real-life processes show non-linear characteristics, such as I/O switching. Among nonlinear models, deep neural networks (DNNs) have recently gained considerable attention. Deep learning has emerged as an active research area for the next generation of time series prediction models. Deep learning is particularly well suited for finding suitably complex nonlinear mathematical functions to transform inputs into outputs. Thus, deep learning provides a means of learning temporal dynamics in a purely data-driven manner.

Multi-Layer Perceptron (MLP): The most basic neural network is the multi-layer perceptron, consisting of an input layer, multiple hidden layers, and an output layer. In each hidden layer or the output layer, the output is the weighted sum of the nodes in the previous layer. For a single-layer perceptron, its output can be computed as follows,

$$\begin{aligned} \mathbf{H} &= \phi(\mathbf{X}\mathbf{W}^{(1)} + \mathbf{b}^{(1)}), \\ \mathbf{O} &= \phi(\mathbf{H}\mathbf{W}^{(2)} + \mathbf{b}^{(2)}), \end{aligned} \tag{2.14}$$

where $\mathbf{X}$, $\mathbf{H}$, $\mathbf{O}$ denote the input features, hidden features and output features, $\mathbf{W}^{(1)}$, $\mathbf{W}^{(2)}$, $\mathbf{b}^{(1)}$ and $\mathbf{b}^{(2)}$ denote the weight matrices and bias vectors of the hidden layer and the output layer, $\phi(\cdot)$ denote the non-linear activation function.

MLP can model a set of time-series data $\{x_0, x_1, ..., x_t, ...\}$ in an autoregressive way,

$$x_t = \text{MLP}(x_{t-1}, x_{t-2}, ..., x_{t-n+1}), \tag{2.15}$$

where n is the order of autoregression.

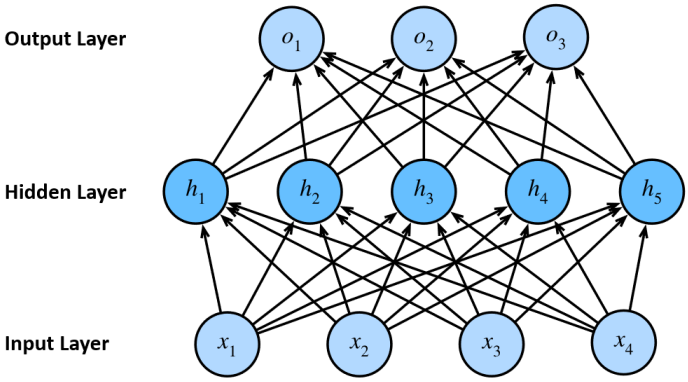


Figure 2.7: A multi-layer perceptron with a single hidden layer

Convolutional Neural Networks (CNNs): CNNs are a powerful class of neural networks designed for processing image data. CNN can be seen as an optimization to reduce the computational complexity of MLP networks. The three paradigms of CNNs are local connectivity, parameter sharing (convolutional kernels) and pooling operations. Although CNNs were originally designed to process two-dimensional image data, one-dimensional CNNs can also be used to model one-dimensional time series. In order to allow CNNs to learn the autoregressive characteristics and the long-term dependencies of time series data, Temporal Convolutional Network (TCN) [100] replaces the normal convolutional operation with a dilated causal convolutional operation as shown in Figure 2.8.

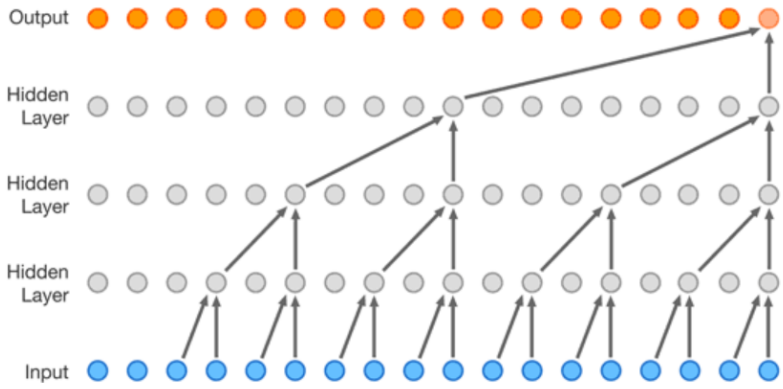


Figure 2.8: The illustration of TCN architecture.

Recurrent Neural Networks (RNNs): RNNs are autoregressive neural networks with hidden states. For a time series $\{x_0, x_1, ..., x_t, ...\}$, we can use the hidden states h_t to model the probability of the prediction $P(x_t|x_{t-1}, x_{t-2}, ..., x_{t-n+1})$ as,

$$\begin{aligned} &P(x_t|x_{t-1}, x_{t-2}, ..., x_{t-n+1}) \approx P(x_t|h_{t-1}), \\ &h_t = f(x_t, h_{t-1}). \end{aligned} \tag{2.16}$$

For a RNN, its output at the tth time step can be computed as follows,

$$\begin{aligned}\mathbf{H}_t &= \phi(\mathbf{X}_t\mathbf{W}_{xh} + \mathbf{H}_{t-1}\mathbf{W}_{hh} + \mathbf{b}_h),\\ \mathbf{O}_t &= \phi(\mathbf{H}_t\mathbf{W}_{ho} + \mathbf{b}_o),\end{aligned} \tag{2.17}$$

where $\mathbf{X}_t$, $\mathbf{H}_t$ and $\mathbf{O}_t$ denote the input features, hidden variables and output features at the tth time step. $\mathbf{W}_{xh}$, $\mathbf{W}_{hh}$ and $\mathbf{W}_{ho}$ are weight matrices, $\mathbf{b}_h$ and $\mathbf{b}_o$ are bias vectors.

Figure 2.9 illustrates the computational logic of a RNN at three adjacent time steps. At any time step t, the computation of the hidden state can be considered as,

- Splicing the input $\mathbf{X}_t$ of the current time step t and the hidden state $\mathbf{H}_{t-1}$ of the previous time step $t-1$;
- The result of the splicing is fed to a fully connected layer with an activation function $\phi(\cdot)$. The output of the fully connected layer is the hidden state $\mathbf{H}_t$ of the current time step t.

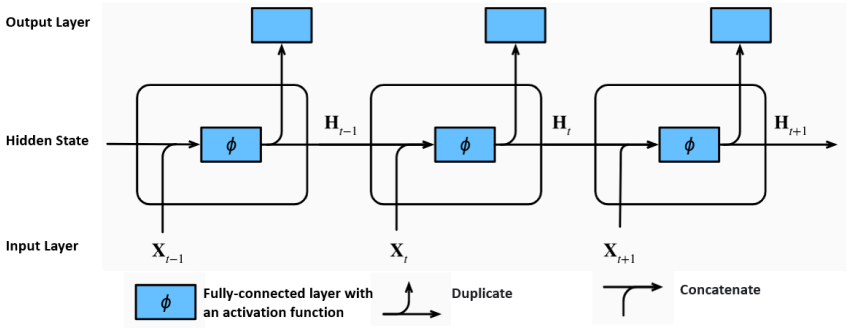


Figure 2.9: The illustration of RNN architecture.

RNNs have the vanishing gradient problems, i.e., the gradients at long distances are too weak, making the sum of the gradients mostly dependent on the gradients at close distances, which causes difficulties in establishing long-term dependence, i.e., long-distance dependence problems. To address this issue, Long Short-Term Memory (LSTM) networks [101] and Gated Recurrent Units (GRUs) [102] used gate mechanisms to store important early information and capture long dependencies.

Transformers: Recurrent models are typically calculated along the positions of the input and output sequences. This inherently sequential nature prevents parallelization of training, which is critical in longer sequence modeling because of memory constraints. Transformers [103] rely entirely on attention mechanisms to map the global dependencies between inputs and outputs, allowing for more significant parallelization. Self-attention enables Transformers to capture both long-term and short-term dependencies, and different attention heads learn different aspects of the temporal model. These advantages make Transformer a good option for time series modeling.

As shown in Figure 2.10, in the self-attention layer, an attention head takes the sequence data and position embeddings as input and converts input features $\mathbf{X}$ into the query matrix $\mathbf{Q} = \mathbf{X}\mathbf{W}^{\mathbf{Q}}$, the key

matrix $\mathbf{K} = \mathbf{X}\mathbf{W}^{\mathbf{K}}$, and the value matrix $\mathbf{V} = \mathbf{X}\mathbf{W}^{\mathbf{V}}$. The output matrix can be computed as,

$$\mathbf{O} = \text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\Big(\frac{\mathbf{Q}\mathbf{K}^{\top}}{\sqrt{d_k}} \cdot \mathbf{M}\Big)\mathbf{V}, \tag{2.18}$$

where d_k is the dimension of key vectors, $\mathbf{M}$ is the masked matrix in which all upper triangular elements are set to $-\infty$ to filter rightward attentions.

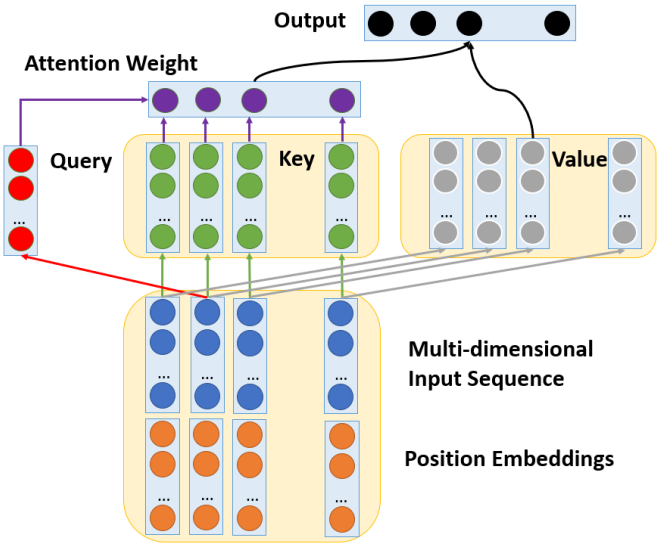


Figure 2.10: The illustration of the self-attention mechanism.

Time Embeddings: Inspired by word embeddings, some research work proposed to use vector representations for time. Time2Vec [104] proposed a representation of time in the form of a vector embedding that could be used by many models instead of proposing a new model for time series analysis. The values of a multi-dimensional time series or the representation of a time-sensitive variable at a specific time can be computed from the embedding of the corresponding time and its original values or time-agnostic representation.

CHAPTER 3

Related Work

In recent years, knowledge graph embedded (KGE) technologies have made significant advances and are widely used for different reasoning tasks over KGs. And the recent availability of temporal KGs has created the need for new KGE approaches which can reason over time. To explore how to extend static KGE approaches to temporal KGs, it is necessary to review the existing KGE techniques. In this chapter, we first summarize the KGE models applied to KG completion, including static KG completion (SKGC) models and temporal KG completion (TKGC) models. Then, we introduce the existing query embedding models used for multi-hop logical reasoning over KGs. We also have a look into the KGE models designed for the entity alignment (EA) task between KGs. Finally, we review the recent work on dynamic graph neural networks (DGNNs), which consider dealing with the temporal evolution of dynamic/temporal graphs.

3.1 Knowledge Graph Completion Models

The task of SKGC is to predict the missing entity in an incomplete triple. Formally, given the query $(e_s, r, ?)$ or $(?, r, e_o)$, a SKGC model is expected to predict the correct entity among the set of entities. Generally, a SKGC model computes the score of a triple with a well-designed scoring function and the learned KGEs. KGEs are learned by maximizing the scores of correct triples with gradient descent and backpropagation. The problem of SKGC can be formally defined as follows,

Problem Definition of Static Knowledge Graph Completion

Let a Knowledge Graph be defined as $\mathcal{G} = (\mathcal{E}, \mathcal{R}, \mathcal{F})$ where $\mathcal{E}$, $\mathcal{R}$ and $\mathcal{F}$ denote the set of entities, relations and observed facts in $\mathcal{G}$. For a triple $(e_s, r, e_o) \in \mathcal{F}_{\text{test}}$ and its related object query $(e_s, r, ?)$ where $\mathcal{F}_{\text{test}}$ denote the testing facts, the goal of SKGC is ranking e_o as high as possible. The goal of answering a subject query $(?, r, e_o)$ is similarly defined.

According to the properties of the scoring functions, most SKGC models can be classified into three categories, i.e., distance-based models, tensor decomposition models and neural network models.

3.1.1 Distance-based Models

The distance-based SKGC models estimate the truthfulness of a triple based on the distance between embeddings of the subject entity and the object entity after a relation-specific translation or rotation. Borders et al. [12] proposed the first translational distance-based model, TransE. The scoring function of TransE is defined as $||\mathbf{e}_s + \mathbf{r} - \mathbf{e}_o||_{1/2}$ where $\mathbf{e}_s$, $\mathbf{e}_s$, $\mathbf{r}$ and $\mathbf{e}_o$ denote embeddings of e_s, r and e_o. The truthfulness of a triple can be defined as the opposite of the value of the scoring function. According to the optimization objective, the scores of true triples should converge to zero. Thus, TransE is less suitable for modeling one-to-many, many-to-one, or many-to-many relations. In response to the limitations of TransE, new distance-based model have emerged since then, e.g.,TransH [105], TransR [106], TransD [107], KG2E [108], TransG [109], RotatE [110] and other models.

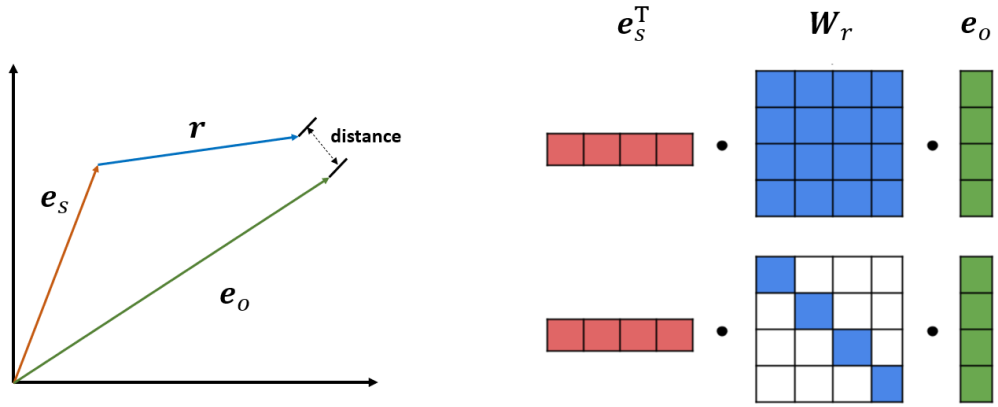


Figure 3.1: The illustration of distance-based model and tensor decomposition model, taking TransE, RESCAL and DistMult as examples.

3.1.2 Tensor Decomposition Models

The tensor decomposition models measure the truthfulness of triples with the scoring functions in the form of $\mathbf{e}_s^\top \mathbf{W}_r \mathbf{e}_o$, where $\mathbf{W}_r$ denotes the relation-dependent matrix. Nickel et al. [111] proposed the first tensor decomposition model, the RESCAL model, for SKGC. After that, they proposed the holographic embedding (HolE) model with fewer parameters [112]. Yang et al. [13] proposed the DistMult model by considering the relation-dependent matrix as a diagonal matrix composed of real numbers. Trouillon et al. [14] used complex numbers instead of real numbers to construct the subject and object entity vectors and considered the relation-dependent matrix as a diagonal matrix composed of complex numbers, and proposed the ComplEx model. Liu et al. [113] proposed the ANALOGY model which introduced normality and exchangeability constraints for relation-dependent matrices to express analogous properties (e.g., the relation between Berlin and Germany is similar to the relation between Beijing and China), and proved that the ANALOGY model was the generalization form of other tensor decomposition models such as HolE and ComplEx. Since then, researchers have also proposed tensor decomposition models such as SimplE [114], TuckER [115], ComplEx-N3 [116] and QuatE [117].

3.1.3 Neural Network Models

Neural network-based SKGC models that take advantage of neural networks in extracting features for representation learning have also achieved good results. SME [118] and NTN [119] use multi-layer perceptrons (MLPs) to jointly encode elements in a triple in different forms. ConvE [120] and ConvKB [121] use two-dimensional convolution and one-dimensional convolution operations, respectively, to learn local features of the triples. CapsE [122] uses a capsule network to model each single triple with a matrix of three columns. RSN [123] takes into account the connectivity information between triples and uses RNN for learning triple paths to better mine the long-range structural information of KGs. GNNs are also applied in SKGC because of their excellent performance in modeling graph networks. Some GNN-based SKGC models [124–126] incorporate relation information between entities into the feature update functions of GNNs, and use GNNs to encode entity features and capture global information, and then use the scoring functions of distance-based models and tensor decomposition models as decoders to score triples.

In addition to the above three types of SKGC methods, there are also other literature discussing SKGC methods using information external to KGs, including methods that combine entity description information, methods that combine entity type information, methods that combine relational path information, and methods that combine logic rules. The characteristic properties underlying different types of SKGC models are summarized in Table 3.1. For more SKGC methods, please see the reference [127].

Category	Characteristics
Distance-based Model	The basic idea is intuitive; Have more room for design improvement; Less number of parameters and high computational efficiency; The expressiveness is limited, and further design is required.
Tensor Decomposition Model	Strong theoretical expressivity; Performances still need to be improved; Larger number of parameters and high training cost.
DNN model	Strong feature learning ability; Interpretability is relatively weak; Performances of some models are not stable.
GNN model	Strong ability of learning global information of KGs; Lack of computational efficiency and scalability.
Model using External Information	Make full use of different types of information in KGs; Further research is needed to explore

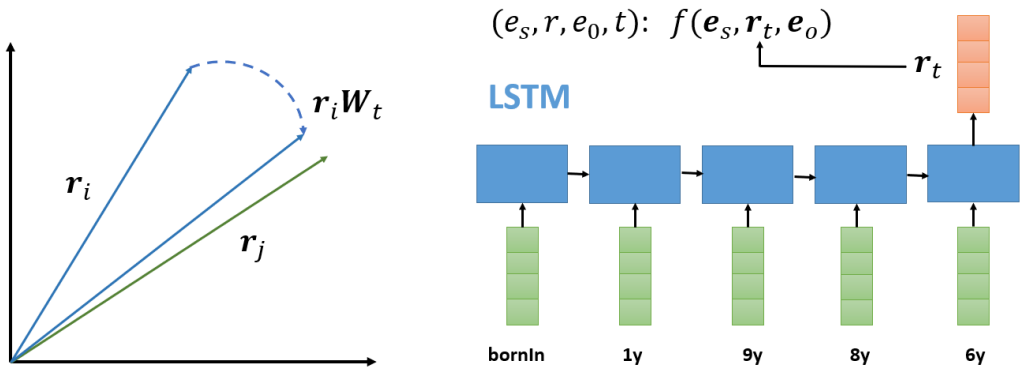
Table 3.1: Summary of basic characteristics of different types of SKGC models.

3.2 Temporal Knowledge Graph Completion Models

Knowledge in static KGs is valid only at a specific time in most cases, while some facts (e.g., evolving events) often appear in a time series. In order to represent the facts in a time series, a number of TKGC models have emerged in recent years which use time-aware scoring functions to measure the truthfulness of quadruples shaped like (e_s, r, e_o, t), where t denotes the timestamp. Based on the treatment of timestamps, these models can be broadly classified into two categories: the first category is time embedding-based models, and the second category is sequence learning models.

3.2.1 Time Embedding Based Models

Time embedding-based models explicitly model timestamps as vectors, matrices or hyperplanes, and then use the timestamp information directly for TKGC. Jiang et al. [128] were the first to propose a TKGC model, consisting of two parts, one of which is embeddings of relations and entities obtained from TransE, and another of which considers three kinds of temporal constraints, i.e., temporal disjointness, temporal ordering and temporal span validity. This model captures the temporal dependencies between different relations through a temporal evolution matrix. Specifically, given two relations r_i and r_j having a temporal ordering dependency, a temporal scoring function is defined as $||\mathbf{r}_i\mathbf{W}_t - \mathbf{r}_j||_{1/2}$, where $\mathbf{W}_t$ is an asymmetric matrix used to capture temporal order information. The basic idea of this scoring function is shown in Figure 3.2a. In Figure 3.2a, r_i and r_j are temporal ordering relation pair, e.g. ¡*wasBornIn, diedIn*¿, which meets the assumption $\mathbf{r}_i\mathbf{W}_t \approx \mathbf{r}_j$. Dasgupta et al. [129] combined the characteristics of the models TransE and TransH and proposed HyTE, a TKGC model. HyTE models the timestamp as time-dependent hyperplanes, then projects the subject and object entities to this time-specific hyperplane and finally scores the triple using a TransE-style scoring function. ConT [130] and TComplEx [33] considered timestamps as the 4th dimension and extended the Tucker and ComplEx tensor decomposition models for TKGC, respectively, and then used the embeddings of timestamps directly for scoring quadruples. Jain et al. [131] proposed TIMEPLEX which was built on Lacroix's work [116], by adding information on sequential and cyclic relations (e.g., Olympic Games held every 3 years) to the scoring function.



(a) (b)

Figure 3.2: Illustrations of (a) the temporal evolving projection in [128] and (b) the time-specific relation embedding in [132]

3.2.2 Sequence Learning Models

Sequence learning models first fuse time information into embeddings of entities or relations, and then use SKGC models to score time-aware facts with time-specific embeddings of entities or relations for TKGC. Given a quadruple (e_s, r, e_o, t), Garcia et al. [132] mapped each token of a timestamp (year, month, day) to an embedding of the same dimension as the relation embedding $\mathbf{r}$, formed a sequence of the relation embedding and embeddings of the timestamp tokens, and then fed the sequence embedding into a long short-term memory (LSTM) network to learn a time-specific relation embedding $\mathbf{r}_t$. As shown in Figure 3.2b, the relation *bornIn* and the timestamp 1986 form the time-specific relation embedding $\mathbf{r}_t$ through an LSTM model. The time-specific triple (e_s, r_t, e_o) are finally scored with the scoring function of DistMult defined as $f(e_s, r_t, e_o) = (\mathbf{e}_s \circ \mathbf{e}_o) \cdot \mathbf{r}_t^\top$, where $\circ$ denotes the Hadamard product, or the scoring function of TransE defined as $||\mathbf{e}_s + \mathbf{r}_t - \mathbf{e}_o||_2$. These time-aware variants of TransE and DistMult are named TA-TransE and TA-DistMult. DE-SimplE [133] divides the entity embeddings into two parts, static and dynamic, and uses the SimplE model to score time-specific triples, in which the static part of entity embeddings expresses the fixed features of the entities during evolution, and the dynamic part utilizes the sinusoidal activation function to regulate the different temporal states and express the features that change during evolution. T-GAP [134] is a temporal GNN for TKGC. This model pretrains the TKG and queries separately, fuses temporal information and entity embeddings, and calculates the attention weights of entities' neighbors, then samples the subgraph structure of each entity and its neighbors related to the query, filters the entities irrelevant to the query, and then uses the GNN model to update embeddings of entities in the subgraph, and utilizes the path traversal-based method to update the attention weights of entities' neighbors and finally infers the hidden relations between entities based on the probabilities of candidates.

3.3 Multi-hop Logical Reasoning over KGs

Methods based on symbolic logic rules have been widely explored in multi-hop KG reasoning (MKGR) research because of their advantages such as high accuracy and interpretability. Among them, first-order logic (FOL), as the main method of logic rules, either combined with probabilistic logic methods or KGE methods, has become a hot research topic in recent years. This section focuses on the ideas of multi-hop KG reasoning approaches based on FOL.

Combining the naturalness of FOL with the uncertainty of probabilistic logic models, Markov logic network methods [135, 136] have been shown to be effective for knowledge graph reasoning. However, the reasoning processes of the above methods are difficult and less efficient on large-scale KGs due to the complex structure between triples. GNNs based on attention mechanisms are good at handling highly complex graph problems. The probabilistic logic graph attention network (pGAT) [137] uses a variational EM algorithm to optimize all possible triples defined by the Markov logic network joint distribution. This helps the model to efficiently combine FOL and GATs.

In recent years, in order to address the issues of the large scales and incompleteness of KGs, researches on multi-hop reasoning over KGs combining logic rules and KGE have received much attention. Query embedding (QE) methods [15, 17, 19, 23] represent queries as directed acyclic computational graphs that specify iterative steps to perform multi-hop reasoning on KGs to obtain the answer entities. These methods treat FOL operators as neural logical operations that can be acquired through training. They all start with the embedding of the anchor entity contained in a query, iteratively use the logical operations to generate an embedding vector for the query, and then predict

the answer entity by computing the distance between the entity embeddings and the query embedding in a vector space.

Let a knowledge graph (KG) be $\mathcal{G} = (\mathcal{E}, \mathcal{R}, \mathcal{F})$ where $\mathcal{E}$, $\mathcal{R}$ and $\mathcal{F}$ denote the set of entities, relations and observed facts in $\mathcal{G}$. Multi-hop reasoning queries include relation traversals as well as several logical operations including conjunction ($\wedge$), disjunction ($\vee$), existential quantification ($\exists$) and negation($\neg$). Here we provide the definition of the multi-hop FOL queries over KGs [19].

Definition of FOL Queries [19]

An FOL query q consists of a non-variable anchor entity set $E_q \subseteq \mathcal{E}$, existentially quantified bound variables E_1, ..., E_k and a single target variable $E_?$ (query answer). The disjunctive normal form of a query q is defined as follows,

$$q[E_?] = E_?,\ \exists E_1, \cdots, E_k : c_1 \vee c_2 \vee \cdots \vee c_n$$

where c_i represents conjunctions of one or more literals a, i.e., $c_i = a_{i1} \wedge a_{i2} \wedge \cdots \wedge a_{im}$ and a literal a represents an atomic formula or its negation, i.e., $a_{ij} = r(e_q, E)$ or $\neg r(e_q, E)$ or $r(E', E)$ or $\neg r(E', E)$, where $e_q \in E_q, E \in \{E_?, E_1, E_2, \cdots, E_k\}, E' \in \{E_1, E_2, \cdots, E_k\}$ and $E \neq E'$.

A query computation graph is a directed acyclic graph whose nodes represent entity sets in the query structure, while directed edges represent logical or relational operations acting on the entity sets. Figure 3.3 illustrates an FOL query "*Where did Canadian citizens with the Turing Award graduate?*" and its computation graph. It can be seen that a query computation graph specifies how the reasoning of the query is proceeded on KGs. Starting from anchor entity sets, we obtain the answer entity set after applying operations on the entity sets according to the directed edges in the query computation graph. These operations are used for implicitly modeling different set operations over the intermediate answer sets. These set operations include relational projection, intersection, union, and complement/Negation.

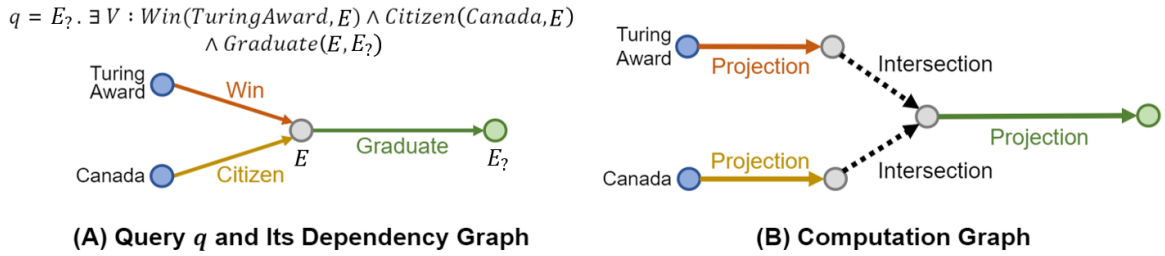


Figure 3.3: Illustrations of (a) an FOL query and (b) its computation graph, taken from [17]

In a query embedding (QE) model, entities and atomic queries are embedded in a continuous vector space, i.e., each node in the computation graph corresponds to an intermediate query embedding, and each edge corresponds to a neural logic operation. Both the input and output of these set operations are query embeddings [138].

To answer a multi-hop logical query, QE models learn embeddings for queries and entities in the KG and then use a score function (e.g. distance function) with these embedding as input to measure

whether entities are in the query answer set. Lots of objects are used to model queries. Some use geometric objects (box, cone) to model queries [15, 17, 23, 24, 139]. Some use probability distribution (Beta, Gaussian) [16, 19]. Some use logic embedding (T-Norms) [21, 140]. Some use other more abstract objects [22, 141, 142]. However, existing embedding-based methods are based on static KG. They cannot utilize temporal information in the temporal KG and therefore cannot handle temporal queries on a temporal KG.

3.4 Entity Alignment

The problem of EA between KGs can be defined as follows,

Problem Definition of Embedding-based Entity Alignment

A KG is a directed relational graph $\mathcal{G} = (\mathcal{E}, \mathcal{R}, \mathcal{F})$ comprising three sets: entities $\mathcal{E}$, relations $\mathcal{R}$ and facts $\mathcal{F} \subseteq \mathcal{E} \times \mathcal{R} \times \mathcal{E}$. A KG stores the real-world information in the form of triples (e_s, r, e_o), where $e_s, e_o \in \mathcal{E}$. Given two KGs $\mathcal{G}_1 = (\mathcal{E}_1, \mathcal{R}_1, \mathcal{F}_1)$, $\mathcal{G}_2 = (\mathcal{E}_2, \mathcal{R}_2, \mathcal{F}_2)$, and a pre-aligned entity pair set $\mathcal{S} = \{(e_i, e_j)|e_i \in \mathcal{E}_1, e_j \in \mathcal{E}_2, e_i \equiv e_j\}$ as alignment seeds where $\equiv$ denotes equivalence. The task of EA between KGs aims at obtaining more potential equivalent entity pairs.

Traditional EA techniques calculate the similarity between entities mainly by statistical methods through manually defined features. Scharffe et al. [143] proposed the entity alignment computation framework RDF-AI, composed of several computational modules such as pre-processing, matching, fusion, and post-processing, among which the matching module implemented algorithms based on string fuzzy matching, string sequence alignment, syntactic similarity, etc. Volz et al. [144] proposed the SILK algorithm, which calculates the similarity from various features such as strings, numbers, dates, and Uniform Resource Identifier (URI) contained in entity attributes. Ngomo et al. [145] proposed the LIMES algorithm, which measures the similarity of entities by the triangular inequality method and returned the similar entity pairs with high similarity as the alignment result. These methods measure the similarity of entity pairs based on manually constructed statistical features and then perform the EA task based on manually set thresholds, but the selection of thresholds relies on manual experience and is closely related to the type of knowledge base and entity types, which lack of scalability.

With the development of machine learning (ML) technology, a large number of researches on EA methods based on ML have emerged, which main ideas are to convert the EA problem into a binary classification problem and automatically discriminate whether the entity pairs match each other by classification models, like decision trees [146], support vector machine (SVM) [147]. These algorithms are based on the features of the description text of the knowledge bases. Bhattacharya et al. [148] and Hall et al. [149] improved these methods by introducing the Latent Dirichlet Allocation (LDA) model [150] to represent the implicit topics of the description text. These methods significantly improve the accuracy compared with the traditional statistical methods. However, these methods require manual construction of feature engineering and different feature engineering methods need to be designed for different kinds of knowledge bases. Moreover, the structural variability of knowledge

bases might lead to the problem of sparse feature data. Thus, such ML-based algorithms have limited applicability.

As mentioned in Section 3.1, with the development of distributed representation learning, KGE models like TransE and its variants, have also emerged in the field of KGs. KGE models capture relations between entities in KGs into a low-dimensional continuous vector space based on a vector space mapping. Based on the early KGE works designed for KGC, researchers have applied KGE technologies to EA tasks. KGE approaches can break through the problem of sparse data in traditional manual feature-based EA methods and enhance the expressiveness of entity representations using richer dimensional information, thus improving the accuracy of entity similarity calculation. Embedding-based EA methods can be roughly classified into two categories, triple-based EA models and GNN-based EA models.

3.4.1 Triple-based Models

Triple-based EA approaches model entities and relations between entities from the perspective of triple learning. These approaches are consistent with most SKGC-oriented KGE models, so it is natural to use the SKGC models as an encoding module for entity alignment. Triple-based EA models usually exploit TransE to learn triples in KGs and an alignment module to match entity pairs.

MTransE [25] learns the entities in each KG individually through the TransE model, and then linearly transforms the embedding space of a KG into another's using the alignment model as shown in Figure 3.4a. IPTransE [151] learns the entities in each KG individually using PTransE [152], and then implements a joint representation between different KGs through a linear transformation and parameter sharing. BootEA [153] proposes a Bootstrapping approach to transform entity alignment into a classification problem, expecting the learned entity embeddings to maximize the likelihood of entity alignment. In addition to relational triples, some triple-based EA models also utilize attribute information and textual information in KGs. JAPE [26] uses the type information of attributes to abstract into several specific types (numeric, character, date, etc.) based on the type of attribute values, and introduces entity attributes into the attribute embedding model, but this model ignores the specific attribute contents in order to represent the attributes uniformly on the discrete feature space. KDCoE [154] addresses the EA problem between cross-lingual KGs by adding the constraint of textual information to the loss function of the EA task, modeling the entity descriptions by gated recurrent units (GRU), and pre-training the model with a cross-lingual corpus so that the model can acquire as much textual information as possible while learning the structure information in KGs. AttrE [155] uses the textual information of entity attributes combined with the long short-term memory (LSTM) model to model the entity descriptions in order to calculate the text similarity of different entities, and then fuse the text similarity as weights into the embeddings learned by TransE.

3.4.2 GNN-based Model

GNN-based EA models consider that triple information in KGs can only capture the one-sided semantics of entities. Unlike triple-based EA models, GNN-based EA models use GNNs as encoders to capture sub-graph structures of entities. GNNs essentially aggregate information from nodes' neighborhoods to target nodes according to message passing rules, allowing entities with similar neighborhoods to approach each other in the embedding space, and excel in capturing global or local structural information of the graph.

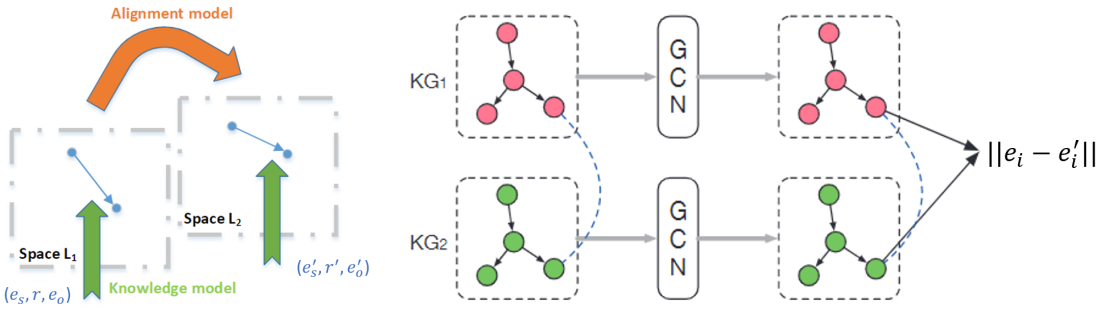


Figure 3.4: Illustrations of (a) triple-based EA models and (b) GNN-based EA models taken from [27]

GCN-Align [27] is the first model that proposes to perform the EA task using GNNs. As shown in Figure 3.4b, GCN-Align uses two GCNs to process two KGs to be aligned separately, and the two GCNs embed entities from different KGs into a unified vector space by sharing a weight matrix to propagate alignment relationships with the help of the structure between entities. In addition, GCN-Align combines attribute information and structure information to jointly learn entity embeddings. NAEA [156] proposed to represent entities by fusing relation-level and neighborhood-level information of the KGs in the encoding module, with TransE and GAT, respectively. Wu et al. [157] proposed a relation-aware dual graph convolutional network RDGCN to make full use of relational information by encoding two KGs to be aligned into the same semantic space and merging $\mathcal{G}_1$ and $\mathcal{G}_2$ through the alignment relationship. To better utilize the entity names in different KGs, RDGCN uses pre-trained English word vectors to construct the input entity representations of the GNNs. MuGNN [28] uses AMIE+ to construct a more dense knowledge graph, using attention mechanisms to model the entire graph features and thus propagate seed alignment information across the graph. AVR-GCN [158] differs from traditional GAT for neighbor feature fusion, but introduces the translation feature of TransE model in the convolution process to merge the different neighbors of an entity into the corresponding relation embeddings for a combined representation, which incorporates relation information into the model in a more direct way, but requires additional pre-aligned relation pairs. To get rid of the reliance on aligned relation pairs, HGCN [159] uses entity embeddings learned by GCNs to approximate relation representations and adds a gate mechanism to control the propagation of noise. HyperKA [160] uses a hyperbolic relational graph neural network for KG alignment. KE-GCN [161] reconciles the problem of structural heterogeneity between KGs by jointly training a GAT-based intersection graph model and a TransE-based KGE model. MRAEA [29] and RREA [30] assign different weight coefficients to entities according to relation types between them, which empowers the models to distinguish the importance between different entities.

For more EA models, please see the reference [162]. So far, there is no existing EA model which considers time information in TKGs.

3.5 Dynamic Graph Neural Network

Different from static graphs, dynamic graphs have two characteristics: 1) a dynamic structure, where connected edges and nodes disappear/appear over time, and 2) dynamic properties, where the states of

nodes and connected edges change over time.

Conventional time-series data are generally represented in two ways: discrete and continuous. The former takes equally spaced windows for the time axis, and the data within the same window are aggregated into a single time step, and finally, the discrete time step is used to represent the continuous timing data. The continuous representation will record the exact time of all data points and is able to preserve the complete time information. Similarly, the representations of dynamic graphs are mainly divided into two forms: discrete and continuous.

The discrete representation of a dynamic graph is essentially an ordered sequence of static graphs to represent a dynamic network, i.e., $\mathcal{DG} = \{\mathcal{G}^1, \mathcal{G}^2, ..., \mathcal{G}^T\}$. The sequence of static graphs can be viewed as a series of snapshots of the dynamic graphs on the time axis in time steps (also expressed as a multi-layer network or tensor), as shown in Figure 3.5. This representation method is simple and intuitive, and can utilize the GNNs to process these snapshots for further completing the processing of dynamic graph data. However, this representation method inevitably loses the timing information of the graph, and the choice of the time granularity can hardly balance computational efficiency and accuracy.

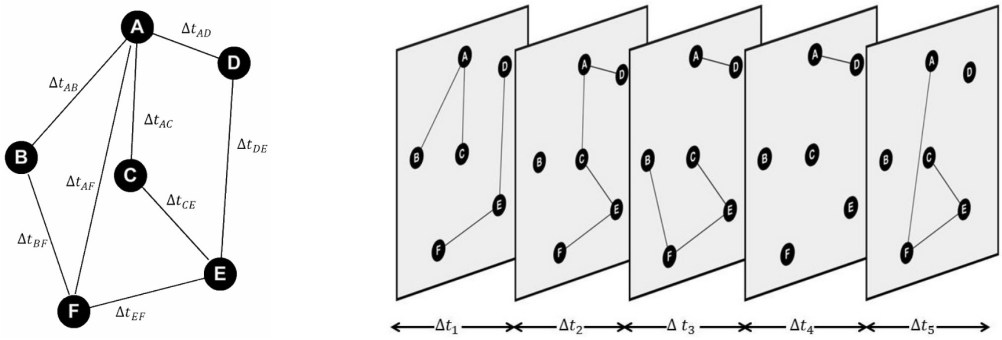


Figure 3.5: The illustration of discrete representation of a dynamic graph, taken from [163]

The continuous representation of a dynamic graph is dedicated to recording the start and end times of all dynamic transformations (or events), thus preserving all temporal information. Although the continuous representation of a dynamic graph can preserve as much temporal information as possible, it is more difficult to process graph data with the continuous representation and the static GNNs can not be simply appropriated to the continuous dynamic graph.

The existing work on dynamic graph neural networks (DGNNs) can be divided into two categories, discrete and continuous, according to the type of dynamic graphs to which they apply, among which the majority aim to deal with discrete data [164].

3.5.1 Discrete Dynamic Graph Neural Network

Discrete DGNN is a framework for processing discrete representations of graph data. This type of framework is divided into two key modules: 1) for static networks with a single time step, static GNN models such as classical GCN and GAT are generally adopted directly; 2) for capturing discrete temporal information, RNN architectures are generally used to be combined with GNNs.

The discrete DGNN-based methods using the combination of GNN and RNN are mainly divided into two categories, one of which is called stacked dynamic GNN and the other is called integrated

dynamic GNN. The main difference between them is how to integrate GNN modules and RNN modules.

The basic idea of stacked dynamic GNN is that GNN is used to encode the information of the graph at a single time point, and then the gating mechanism of RNN (generally using LSTM) is used to complete the transmission and encoding of the graph information in the time axis. The overall structure consists of GNN and RNN stacked sequentially. The simplest form is as follows,

$$\begin{aligned} \mathbf{x}_1^t, ..., \mathbf{x}_n^t &= \text{GNN}(\mathcal{G}^t) \\ \mathbf{h}_j^t &= \text{RNN}(\mathbf{h}_j^{t-1}, \mathbf{x}_j^t) \text{ for } j \in [1, n], \end{aligned} \tag{3.1}$$

where $\mathbf{x}_j^t$ and $\mathbf{h}_j^t$ denote the node feature and hidden state feature of node x_j at the time point t. GNN is used to process the static graph, and the hidden state of the RNN is used to update the representation of the nodes at different time steps. Youngjoo et al. [165] introduced one of the earliest versions of the above framework, utilizing GNN and peehole LSTM. Following the above basic idea, stacked dynamic GNNs can have many variants. These include, but are not limited to, using different GNN or RNN models [166, 167], not sharing RNN parameters between different nodes [168], and using a self-attentive mechanism instead of RNN [169].

The basic idea of integrating dynamic GNNs is to integrate GNN and RNN into a uniform architecture: on the one hand, the linear transform module in RNN can be replaced by graph convolution, or RNN can be used to control the parameters of GNN at different time steps. There are two typical works. One [165] followed the convolutional long short-term memory network convLSTM but replaced linear transformations in the LSTM with graph convolution operations. This deformation of the LSTM is naturally able to handle time series composed of graph data. The other is the EvolveGCN [170]. The motivation of this work is based on the argumentation that the parameters of GCNs which process time series should also evolve (differently) over time. They used RNN to control and update the parameters of the GCN at different time steps. Therein, the parameters of the GCN can be regarded as the hidden states or outputs of the RNN. Based on this idea, they proposed two variants to control the parameters of GCN at each step using the hidden state (EGCU-H) and the output (EGCU-O) of RNN, respectively. Some other works [171, 172] utilize an autoencoder, a variational graph autoencoder, or a GAN to model discrete dynamic graphs.

3.5.2 Continuous Dynamic Graph Neural Network

For continuous representation, the current work also focuses on two aspects: one is the RNN-based framework, where the representational information of the nodes is delivered and maintained by RNNs; the other is the temporal point process (TTP) based framework, where the fitting of the data is done by parameterizing the TTP with neural networks (NNs).

- RNN-based framework: The motivation of this kind of work is that nodes involved in the transformation need to be updated as long as a change (or event) occurs, making the node embeddings continuously updated. There are two typical works adopting this framework. Ma et al. [173] proposed a streaming graph based method, which accomplishes the update of node embeddings for an event through three components: (i) interact unit, (ii) update/propagate unit, and (iii) merge unit, among which the core component is the time-aware LSTM-based update/propagate unit; Leskovec et al. [174] proposed JODIE, which considers the bipartite

graph of customers and goods, captures temporal information of the customer and goods nodes with two RNNs, where node features of the corresponding goods and users are updated immediately after a new transaction occurs.

- TTP-based framework: Temporal point process (TTP) is a traditional model for modeling asynchronous time sequences. And continuous dynamic networks are typical asynchronous sequences on a continuous time domain. DyRep [175] uses NNs to parameterize TTP so that it can be optimized to capture the dynamics of the evolution of the graph structure and the dynamics of the nodes. By modeling these two co-evolutionary processes simultaneously, DyRep is able to achieve richer graph characterizations. DyRep is also the only framework that can predict when events occur (others can only predict when events occur). Recent work has also attempted to better model the TTP process using neural relational inference and self-attention. Moreover, other temporal models have been tried to model dynamic networks, e.g., GHN [176]which uses the Hawkes model.

In general, continuous DGNNs can only handle a few specific types of continuous dynamic graphs so far, such as interactive graphs and streaming graphs, and more general DGNNs that can solve more problems are yet to be developed.

CHAPTER 4

Temporal Knowledge Graph Embeddings for Knowledge Graph completion

In Chapter 1, we mention that queries in TKGs are often related to time information. For example, the query (*?*, *president of*, *USA*) can have different answers at different timestamps. When the attached timestamp is *[2017-2021]*, the predicted answer is expected to be *Donald Trump*. Static KGC (SKGC) models such as TransE learn only from time-unknown facts. Therefore, they cannot distinguish relations with similar semantic meanings. For instance, they often confuse entities such as *Donald Trump* and *George Bush* when predicting *(?, president of, USA)*.

To tackle this problem, some TKGE models [129, 132, 133, 177] encode time information in their embeddings for TKGC. In this chapter, we scope the problem of TKGC only. As mentioned in Challenge 1, although TKGC models have been proven to outperform SKGC models, previous works on TKGC models still have limitations in different aspects,

1) Temporal Uncertainty: The evolution of entity embeddings has randomness because the features of an entity at a certain time are not completely determined by the past information;

2) Temporal Interpretability: Interpretable time data models can help with the understanding of the temporal evolution of entity/relation semantics;

3) Time Representation: Timestamps in TKGs can be represented in different forms, e.g., time points, time intervals, beginning time or end time;

4) Time Distribution: Time data in TKGs can have different distributions, e.g., uniform distribution, or long-tailed distribution;

5) Time granularity: The distributions of time data in TKGs can be uniform or long-tailed, and the method of splitting time can have an impact on model performance;

6) Model Expressiveness: Most of the previous TKGC models are temporal extensions of TransE and DistMult, and thus have issues of learning some specific relation patterns;

7) Temporal Regularization: Time embedding technique lacks consideration for the physical characteristics of time data and thus need specific regularization methods.

In this chapter, we are dedicated to addressing the first research question (RQ1). The main contributions of this chapter are stated in the following paragraphs.

Research Question 1 (RQ1)

How can we effectively incorporate time information into KGE models to enhance temporal knowledge graph completion?

In this chapter, we first formally define the problem of TKGC and the evaluation metrics used for TKGC models. Then, we introduce the most common benchmarks used for TKGC. Importantly, we present three novel TKGC models which address the RQ1 and overcome the limitations of the existing methods mentioned above.

In Section 4.3, we show a TKGC model, **ATiSE** [32] which incorporates time information into entity/relation representations by using **A**dditive **Ti**me **SE**ries decomposition. Additive time series decomposition provides an interpretable modeling way by taking the evolution processes of entity embeddings and relation embeddings as multi-dimensional time series and considers the temporal uncertainty during the evolution process.

In Section 4.4, we present our proposed second TKGC model, **TeRo** [178] which defines the temporal evolution of an entity embedding as a rotation from the initial time to the current time in the complex vector space. TeRo overcomes the limitations of early TKGC models and has the capability of capturing various relation patterns. Moreover, it adapts well to TKGs where time information is represented in different forms. We also study the effect of time granularity on TeRo's performance in this section.

In Section 4.5, we introduce the third TKGC model, **TGeomE** [179] which performs 4th-order tensor factorization of a TKG using multivector embeddings from a multi-dimensional geometric algebra and considers a new linear temporal regularization for retaining the ordering and distance information between different timestamps. We theoretically prove that TGeomE is fully expressive and subsumes several existing TKGC models.

As shown in Figure 4.1, three new TKGC models presented in this chapter address a part of the limitations of the existing TKGC models, respectively.

This chapter is based on the following publications [32, 34, 178, 179]:

1. Chengjin Xu, Mojtaba Nayyeri, Fouad Alkhoury, Hamed Yazdi and Jens Lehmann. "Temporal knowledge graph completion based on time series gaussian embedding", International Semantic Web Conference, Springer, 2020;

2. Chengjin Xu, Mojtaba Nayyeri, Fouad Alkhoury, Hamed Yazdi and Jens Lehmann. "TeRo: A Time-aware Knowledge Graph Embedding via Temporal Rotation", Proceedings of the 28th International Conference on Computational Linguistics, Barcelona, Spain (Online), 2020;

3. Chengjin Xu, Yung-Yu Chen, Mojtaba Nayyeri and Jens Lehmann. "Temporal Knowledge Graph Completion using a Linear Temporal Regularizer and Multivector Embeddings", Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, 2021;

4. Chengjin Xu, Yung-Yu Chen, Mojtaba Nayyeri and Jens Lehmann. "Geometric Algebra based Embeddings for Static and Temporal Knowledge Graph Completion", IEEE Transactions on Knowledge and Data Engineering, 2022.

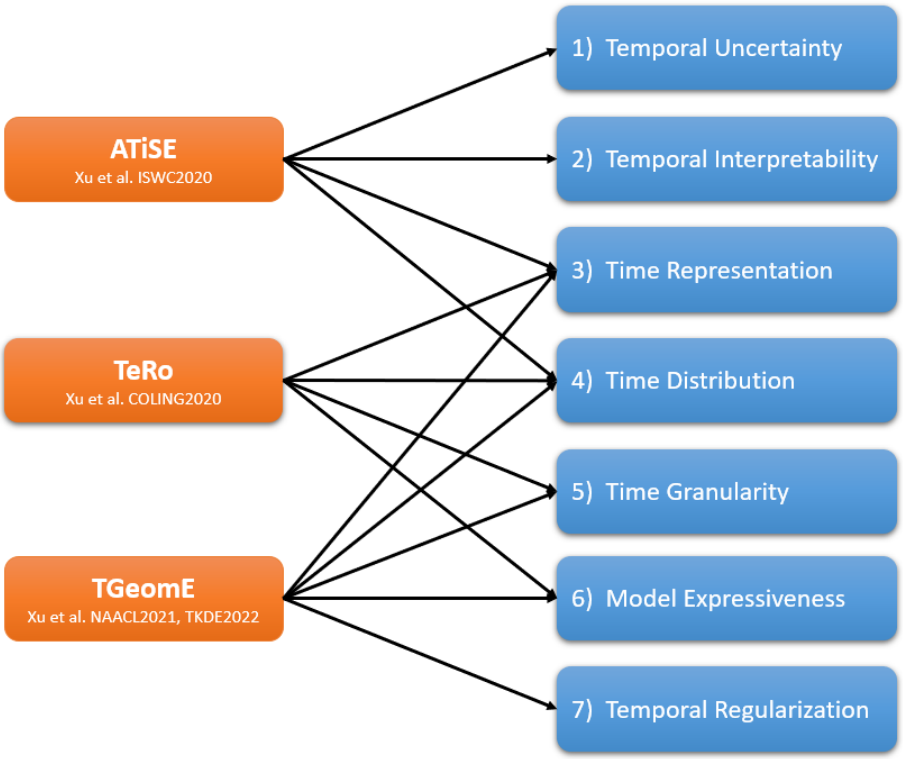


Figure 4.1: Contributions of our proposed TKGC models to RQ1.

In the above papers, the design and implementation of all presented TKGC models were done by the Ph.D. candidate, who also conducted most evaluation experiments and handled the writing of these papers.

4.1 Problem Definition and Evaluation Metrics

Problem Definition of Temporal Knowledge Graph Completion

Let a Temporal Knowledge Graph be defined as $\mathcal{G} = (\mathcal{E}, \mathcal{R}, \mathcal{T}, \mathcal{Q})$ where $\mathcal{E}$, $\mathcal{R}$, $\mathcal{T}$ and $\mathcal{Q}$ denote the set of entities, relations, timestamps and observed fact quadruples in $\mathcal{G}$. For a quadruple $(e_s, r, e_o, t) \in \mathcal{Q}_{\text{test}}$ and its related query $(e_s, r, ?, t)$ where $\mathcal{Q}_{\text{test}}$ denote the testing quadruples, the goal of TKGC is ranking e_o as high as possible. The goal of answering a subject query $(?, r, e_o, t)$ is similarly defined.

The task of TKGC is to predict the missing entity in an incomplete quadruple. Formally, given the query $(e_s, r, ?, t)$ or $(?, r, e_o, t)$, a TKGC model is expected to predict the correct entity among the set of entities. Generally, a TKGC model computes the score of a quadruple with a well-designed scoring function and the learned TKGEs. TKGEs are learned by maximizing the scores of observed

quadruples.

For each test quadruple (e_s, r, e_o, t), we first generate candidate quadruples $C_o = \{(e_s, r, e'_o, t) : e'_o \in \mathcal{E}\}$ and $C_s = \{(e'_s, r, e_o, t) : e'_s \in \mathcal{E}\}$ by replacing e_s or e_o with all possible entities. Different from the time-unwise filtered setting [12] which filters the quadruples appearing either in the training, validation or test set from the candidate list, we only filter the candidate quadruples $\xi \in Q$ existing in the dataset. This ensures that the facts which do not appear at time t are still considered as candidates for evaluating the given test quadruple. We obtain the final rank of the test quadruple (e_s, r, e_o, t) among filtered candidate quadruples $\overline{C}_o = \{(e_s, r, e'_o, t) : (e_s, r, e'_o, t) \in C_s, (e_s, r, e'_o, t) \notin Q\}$ and $\overline{C}_s = \{(e'_s, r, e_o, t) : \{(e'_s, r, e_o, t) \in C_s, \{(e'_s, r, e_o, t) \notin Q\}$ by sorting their scores. Then we determine the rank of (e_s, r, e_o, t) relative to all $(e_s, r, e'_o, t) \in \overline{C}_o$ using the scores, which is denoted as Rank$(e_s$—$r, e_o, t)$. A similar definition Rank$(e_o$—$e_s, r, t)$ applies to the second query $(e_s, r, ?, t)$.

Two evaluation metrics are used here, i.e., Mean Reciprocal Rank (MRR) and Hits@k (generally k = 1, 3, 10). The Mean Reciprocal Rank (MRR) is the mean of the reciprocal values of all computed ranks, i.e.,

$$\text{MRR} = \frac{1}{2|Q_{\text{test}}|} \sum_{(e_s, r, e_o, t) \in Q_{\text{test}}} \left(\frac{1}{\text{Rank}(e_s|r, e_o, t)} + \frac{1}{\text{Rank}(e_o|e_s, r, t)} \right). \tag{4.1}$$

And the fraction of test quadruples ranking in the top k is called Hits@k, i.e.,

$$\text{Hits@k} = \frac{1}{2|Q_{\text{test}}|} \sum_{(e_s, r, e_o, t) \in Q_{\text{test}}} \Big(I\big(\text{Rank}(e_s|r, e_o, t) \leq k\big) + I\big(\text{Rank}(e_o|e_s, r, t) \leq k\big) \Big), \tag{4.2}$$

where I denotes the indicator function.

4.2 Temporal Knowledge Graph Completion Datasets

Currently, there are seven commonly used benchmark datasets in the field of TKGC research, which are constructed on four TKG databases, namely Wikidata [9], YAGO [8], GDELT [11] and the integrated crisis early warning system (ICEWS) [10]. The seven datasets are GDELT-500 [133], ICEWS14, ICEWS05-15, YAGO15k, Wikidata11k [132], YAGO11k, and Wikidata12k [129]. Most timestamps in YAGO and Wikidata are time intervals, while all the timestamps in GDELT and ICEWS are time points.

GDELT[1]: The GDELT database records news reported in print, broadcast, and web media in approximately 100 languages in every country from 1969 to the present, and is updated every 15 min. GDELT consists of two main databases, namely an event database and a global knowledge graph. GDELT-500 is a subset of GDELT that is commonly used for the TKGC study.

ICEWS[2]: The ICEWS database covers more than 100 data sources and 250 national and regional political events, and is updated daily. Two subsets of ICEWS are used for the study of TKGC, namely ICEWS14 and ICEWS05-15.

Wikidata[3]: Wikidata is a free collaborative multilingual knowledge base hosted by the Wikimedia

[1] https://www.gdeltproject.org/

[2] https://dataverse.harvard.edu/dataverse/icews

[3] https://www.wikidata.org/wiki/Wikidata:Main_Page

Foundation and designed to support Wikipedia, Wikimedia Commons, and other Wikimedia projects. Two subsets of Wikidata are commonly used for TKGC research, namely Wikidata11k and Wikidata12k.

YAGO[4]: YAGO is a linked database developed by the Max Planck Institute in Germany. The database mainly integrates data from 3 sources, Wikipedia, WordNet, and GeoNames.YAGO integrates WordNet's vocabulary definitions with Wikipedia's classification system, making YAGO have a richer entity classification system. YAGO also considers temporal and spatial knowledge, by adding attribute descriptions of temporal and spatial dimensions for many knowledge entries. YAGO11k and YAGO15k are two subsets of YAGO that are used for the study of TKGC.

The statistical results of the above seven datasets are shown in Table 4.1. In this thesis, we choose ICEWS14, ICEWS05-15, YAGO11k and Wikidta12k as datasets for the following reasons: 1. ICEWS14 and ICEWS05-15 are two well-established event-based datasets that are commonly used in previous literature [32, 132, 133], these two datasets are subsets of ICEWS corresponding to facts in 2014 and facts between 2005 and 2015, where all time annotations are time points; 2. YAGO15k, Wikidata11k, YAGO11k and Wikidata12k are subsets of YAGO3 and Wikidata where a part of time annotations are time intervals. In YAGO15k and Wikidata11k, each time interval only contains either beginning dates or end dates, shaped like '*occurSince 2003*' or '*occurUntill 2005*' and a part of facts in YAGO15k exclude time information. Thus we prefer to use YAGO11k and Wikidata12k where each fact includes time information and time annotations are represented in various forms, i.e., time points like [*2003-01-01*, *2003-01-01*], beginning or end time like [*2003*, *##*], and time intervals like [*2003*, *2005*].

Dataset	$\lvert\mathcal{E}\rvert$	$\lvert\mathcal{R}\rvert$	Time Span(year)	$\lvert\mathcal{Q}_{\text{train}}\rvert$	$\lvert\mathcal{Q}_{valid}\rvert$	$\lvert\mathcal{Q}_{\text{test}}\rvert$
ICEWS14	6,869	230	2014	72,826	8,941	8,963
ICEWS05-15	10,094	251	2005-2015	368,962	46,275	46,092
YAGO11k	10,623	10	-431-2844	16,406	2,050	2,051
Wikidata12k	12,554	24	19-2020	32,497	4,062	4,062
GDELT-500	500	20	2015-2016	2,735,685	341,961	341,961
YAGO15k	15,403	34	1513-2017	110,441	13,815	13,800
Wikidata11k	11,134	95	25-2020	121,422	14,374	14,283

Table 4.1: Statistics of TKGC datasets.

Time data in TKGs can have different types of distributions, e.g., uniform distribution and long-tailed distribution. As shown in Figure 4.2, the numbers of ICEWS facts occurring at different timestamps have a uniform distribution since ICEWS collects data from breaking news in recent years and is updated every day. Meanwhile, YAGO11k and Wikidata12k have long time spans of more than 2000 years but most of the recorded facts occurred between 1800 and 2020. It is essential to use different time data preprocessing methods for TKG datasets with different time distributions.

[4] https://www.mpi-inf.mpg.de/departments/databases-and-information-systems/research/yago-naga/yago

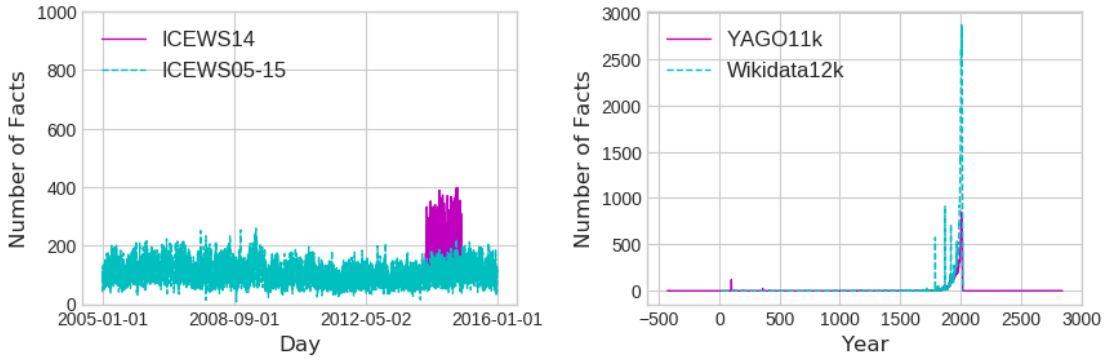


Figure 4.2: Time distribution of numbers of facts in different TKGs.

4.3 A TKGC Model Based on Additive Time Series Decomposition

The content of the this section is based on our work in the paper titled "Temporal knowledge graph completion based on time series gaussian embedding" (Xu et al., ISWC 2020) [32].

4.3.1 Introduction

Previous TKGC models embed time information into a latent space, e.g. representing time as a vector. These models cannot capture some properties of time information such as the length of time interval as well as the order of two time points. Moreover, these models ignore the uncertainty during the temporal evolution. We argue that the evolution of an entity representation has randomness because the features of an entity at a certain time are not completely determined by past information. For example, (*Steve Jobs, diedIn, California*) happened on 2011-10-05. The semantic characteristics of this entity should have a sudden change at this time point. However, due to the incompleteness of knowledge in KGs, this change can not be predicted only according to its past evolutionary trend. Therefore, the representation of *Steve Jobs* is supposed to include some random components to handle this uncertainty, e.g. a Gaussian noise component.

In order to address the above problems, in this section, we introduce a TKGC model, ATiSE[5], which uses additive time series decomposition to fit the evolution process of KG representations. ATiSE fits the evolution process of an entity or relation as a multi-dimensional additive time series which consists of a trend component, a seasonal component and a random component. At each time step, each entity and relation is represented as a multi-dimensional Gaussian distribution which introduces randomness. The mean of an entity/relation representation at a certain time step indicates its current expected position, which is obtained from its initial representation, its linear change term, and its seasonality term. The covariance which describes the temporal uncertainty during its evolution, is denoted as a constant diagonal matrix for computing efficiency. As shown in Figure 4.3, the labels indicate the positions of entities and relations. In the representations, we might infer that *Bill Clinton* was *presidentOf USA* in 1998 and *Barack Obama* was *presidentOf USA* in 2010. Our contributions are as follows.

[5] The code is avaible at https://github.com/soledad921/ATISE

- Learning the representations for temporal KGs is a relatively unexplored problem because most existing KG embedding models only learn from time-unknown facts. We present ATiSE, a TKGC model to incorporate time information into the KG representations.

- We specially consider the temporal uncertainty during the evolution process of KG representations. Thus, we model each entity/relation as a Gaussian distribution at each time step. The mean vectors of multi-dimensional Gaussian distributions of entities and relations indicate their position which changes over time and the covariance matrices indicate the corresponding temporal uncertainty. A symmetric KL-divergence between two Gaussian distributions is designed to compute the scores of facts for optimization.

- Different from the previous TKGC models which use time embedding to incorporate time information, ATiSE fits the evolution process of KG representations as a multi-dimensional additive time series. Our work establishes a previously unexplored connection between relational processes and time series analysis with the potential to open a new direction of research on reasoning over time.

- ATiSE can adapt well to various TKGs where timestamps have different representation forms and the numbers of timestamps have different distributions. Our experimental results show that ATiSE significantly outperforms previous TKGC models and several state-of-the-art static KGC on four TKGC benchmarks.

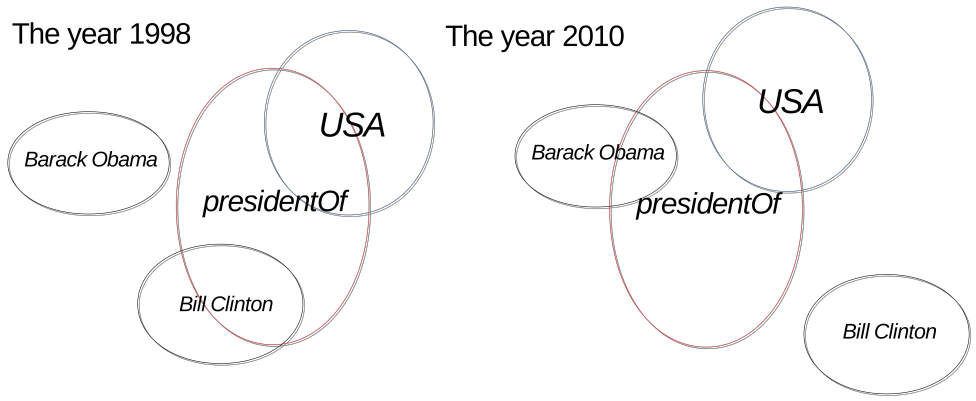


Figure 4.3: Illustration of the means and (diagonal) variances of entities and relations in a temporal Gaussian Embedding Space.

4.3.2 Methodology

In this section, we present a detailed description of our proposed method, ATiSE, which not only uses relational properties between entities in triples but also incorporates the associated temporal meta-data by using additive time series decomposition.

Additive Time Series Embedding Model

As mentioned in Section 2.2.1, a time series is a series of time-oriented data. Time series analysis is widely used in many fields, ranging from economics and finance to managing production operations, to the analysis of political and social policy sessions [99]. An important technique for time series analysis is additive time series decomposition, which has been introduced in Section 2.2.1. This technique decomposes a time series Y_t into three components as follows,

$$Y_t = T_t + S_t + R_t. \tag{4.3}$$

where T_t, S_t and R_t denote the trend component, the seasonal component and the random component (i.e. "noise"), respectively.

In ATiSE, we regard the evolution of an entity/relation representation as an additive time series. For each entity/relation, we use a linear function and a Sine function to fit the trend component and the seasonal component respectively due to their simplicity. Considering the efficiency of model training, we model the irregular term by using a Gaussian noise instead of a moving average model (MA model) [180], since training an MA model requires a global optimization algorithm which will lead to more computation consumption.

To incorporate temporal information into traditional KGs, a new temporal dimension is added to fact triples, denoted as a quadruple (e_s, r, e_o, t). It represents the creation of relation edge r between subject entity e_s, and object entity e_o at time step t. The score term $\phi(e_s, r, e_o, t) = f_t(\mathbf{e}_s, \mathbf{r}, \mathbf{e}_o)$ can represent the conditional probability or the confidence value of this quadruple, where $\mathbf{e}_s, \mathbf{e}_o \in \mathcal{R}^k$, $\mathbf{r} \in \mathcal{R}^k$ denote k-dimensional embeddings of e_s, e_o and r. In term of a long-term fact $(e_s, r, e_o, [t_s, t_e])$, we consider it to be a positive triple for each time step between t_s and t_e. t_s and t_e denote the start and end time, during which the triple (e_s, r, e_o) is valid.

At each time step, the time-specific representation of an entity e_i or a relation r should be updated as $\mathbf{e}_{i,t}$ or $\mathbf{r}_t$. Thus, the score of a quadruple (e_s, r, e_o, t) can be represented as $f(\mathbf{e}_{s,t}, \mathbf{r}_t, \mathbf{e}_{o,t})$. We utilize additive time series decomposition to fit the evolution process of each entity/relation representation as:

$$\begin{aligned} \mathbf{e}_{i,t} &= \mathbf{e}_i + \alpha_{e,i}\mathbf{w}_{e,i}t + \boldsymbol{\beta}_{e,i}\sin(2\pi\boldsymbol{\omega}_{e,i}t) + \mathcal{N}(0, \boldsymbol{\Sigma}_{e,i}) \\ \mathbf{r}_{j,t} &= \mathbf{r}_j + \alpha_{r,j}\mathbf{w}_{r,j}t + \boldsymbol{\beta}_{r,j}\sin(2\pi\boldsymbol{\omega}_{r,j}t) + \mathcal{N}(0, \boldsymbol{\Sigma}_{r,j}) \end{aligned} \tag{4.4}$$

where the $\mathbf{e}_i \in \mathbb{R}^k$ and $\mathbf{r}_j \in \mathbb{R}^k$ are the time-independent latent representations of the ith entity which is subjected to $||\mathbf{e}_i||_2 = 1$ and the jth relation which is subjected to $||\mathbf{r}_j||_2 = 1$. $\mathbf{e}_i + \alpha_{e,i}\mathbf{w}_{e,i}t$ and $\mathbf{r}_j + \alpha_{r,j}\mathbf{w}_{r,j}t$ are the trend components where the coefficients $\alpha_{e,i}$ and $\alpha_{r,j}$ denote the evolutionary rates of $\mathbf{e}_{i,t}$ and $\mathbf{r}_{j,t}$, the vectors $\mathbf{w}_{e,i} \in \mathbb{R}^k$ and $\mathbf{w}_{r,j} \in \mathbb{R}^k$ represent the corresponding evolutionary directions which are restricted to $||\mathbf{w}_{e,i}||_2 = ||\mathbf{w}_{r,j}||_2 = 1$. $\boldsymbol{\beta}_{e,i}\sin(2\pi\boldsymbol{\omega}_{e,i}t)$ and $\boldsymbol{\beta}_{r,j}\sin(2\pi\boldsymbol{\omega}_{r,j}t)$ are the corresponding seasonal components where $\boldsymbol{\beta}_{e,i} \in \mathbb{R}^k$ and $\boldsymbol{\beta}_{r,j} \in \mathbb{R}^k$ denote the amplitude vectors, $\boldsymbol{\omega}_{e,i} \in \mathbb{R}^k$ and $\boldsymbol{\omega}_{r,j} \in \mathbb{R}^k$ denote the frequency vectors. The Gaussian noise terms $\mathcal{N}(0, \boldsymbol{\Sigma}_{e,i})$ and $\mathcal{N}(0, \boldsymbol{\Sigma}_{r,j})$ are the random components, where $\boldsymbol{\Sigma}_{e,i} \in \mathbb{R}^k$ and $\boldsymbol{\Sigma}_{r,j} \in \mathbb{R}^k$ denote the corresponding diagonal covariance matrices.

In other words, for a fact (e_s, r, e_o, t), entity embeddings $\mathbf{e}_{s,t}$ and $\mathbf{e}_{o,t}$ obey Gaussian probability distributions: $\mathbf{P}_{s,t} \sim \mathcal{N}(\bar{\mathbf{e}}_{s,t}, \boldsymbol{\Sigma}_s)$ and $\mathbf{P}_{o,t} \sim \mathcal{N}(\bar{\mathbf{e}}_{o,t}, \boldsymbol{\Sigma}_o)$, where $\bar{\mathbf{e}}_{s,t}$ and $\bar{\mathbf{e}}_{o,t}$ are the mean vectors of $\mathbf{e}_{s,t}$ and $\mathbf{e}_{o,t}$, which do not include the random components. Similarly, the relation is represented as $\mathbf{P}_{r,t} \sim \mathcal{N}(\bar{\mathbf{r}}_t, \boldsymbol{\Sigma}_r)$.

Similar to translation-based KGC models, we consider the transformation result of ATiSE from the subject to the object to be akin to the relation in a positive fact. We use the following formula to express this transformation: $\mathbf{P}_{s,t} - \mathbf{P}_{o,t}$, which corresponds to the probability distribution $\mathbf{P}_{e,t} \sim \mathcal{N}\left(\boldsymbol{\mu}_{e,t}, \boldsymbol{\Sigma}_e\right)$. Here, $\boldsymbol{\mu}_{e,t} = \bar{\mathbf{e}}_{s,t} - \bar{\mathbf{e}}_{o,t}$ and $\boldsymbol{\Sigma}_e = \boldsymbol{\Sigma}_s + \boldsymbol{\Sigma}_o$. Combined with the probability of relation $\mathbf{P}_{r,t} \sim \mathcal{N}\left(\bar{\mathbf{r}}_t, \boldsymbol{\Sigma}_r\right)$, we measure the similarity between $\mathbf{P}_{e,t}$ and $\mathbf{P}_r$ to score the fact. Given a triple $\left(e_s, r, e_o\right)$ is valid at time t but invalid at $t+1$, it is expected that $\mathbf{P}_{s,t} - \mathbf{P}_{o,t} \approx \mathbf{P}_{r,t}$, i.e., the distributions $\mathbf{P}_{e,t}$ and $\mathbf{P}_{r,t}$ overlap as much as possible, while $\mathbf{P}_{e,t+1}$ is distant from $\mathbf{P}_{r,t+1}$ as shown in Figure 4.4.

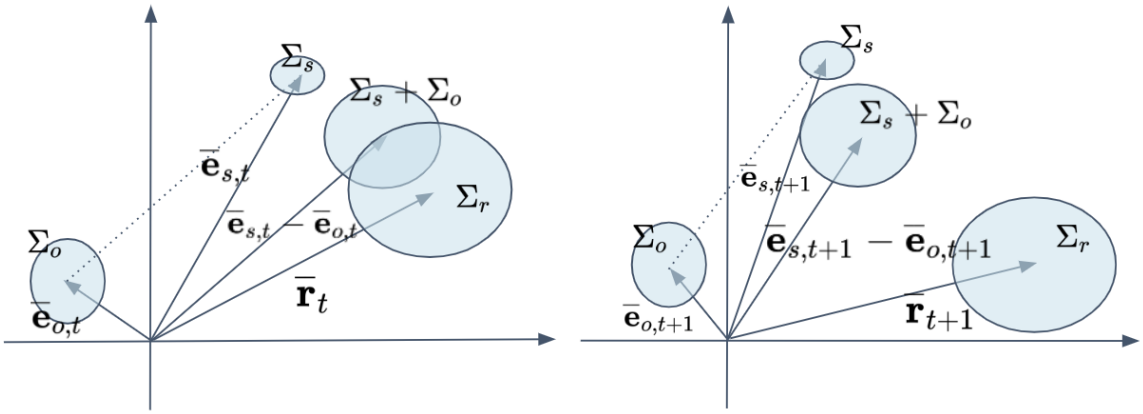


Figure 4.4: Illustration of the assumption of ATiSE.

KL divergence is a straightforward method of measuring the similarity of two probability distributions. We optimize the following score function based on the KL divergence between the entity-transformed distribution and relation distribution [181].

$$\begin{aligned}\phi(e_s, r, e_o, t) &= \mathcal{D}_{\mathcal{KL}}(\mathbf{P}_{r,t}, \mathbf{P}_{e,t}) = \int_{x \in \mathcal{R}^k} \mathcal{N}(x; \mathbf{r}_t, \boldsymbol{\Sigma}_r) \log \frac{\mathcal{N}(x; \boldsymbol{\mu}_{e,t}, \boldsymbol{\Sigma}_e)}{\mathcal{N}(x; \mathbf{r}_t, \boldsymbol{\Sigma}_r)} \mathrm{d}x \\ &= \frac{1}{2}\left\{tr(\boldsymbol{\Sigma}_r^{-1}\boldsymbol{\Sigma}_e) + (\bar{\mathbf{r}}_t - \boldsymbol{\mu}_{e,t})^\top \boldsymbol{\Sigma}_r^{-1}(\bar{\mathbf{r}}_t - \boldsymbol{\mu}_{e,t}) - \log \frac{det(\boldsymbol{\Sigma}_e)}{det(\boldsymbol{\Sigma}_r)} - k\right\}\end{aligned} \tag{4.5}$$

where, $tr(\boldsymbol{\Sigma})$ and $\boldsymbol{\Sigma}^{-1}$ indicate the trace and inverse of the diagonal covariance matrix, respectively.

Since the computation of the determinants of the covariance matrices in Equation 4.5 is time consuming, we define a symmetric similarity measure based on KL divergence to simplify the computation of the score function.

$$\phi(e_s, r, e_o, t) = \frac{1}{2}(\mathcal{D}_{\mathcal{KL}}(\mathbf{P}_{r,t}, \mathbf{P}_{e,t}) + \mathcal{D}_{\mathcal{KL}}(\mathbf{P}_{e,t}, \mathbf{P}_{r,t})) \tag{4.6}$$

Considering the simplified diagonal covariance, we can compute the trace and inverse of the matrix simply and effectively for ATiSE. The gradient of log determinant is $\frac{\partial \log detA}{\partial A} = A^{-1}$, the gradient $\frac{\partial x^T A^{-1} y}{\partial A} = -A^{-T} x y^T A^{-T}$, and the gradient $\frac{\partial tr(X^T A^{-1} Y)}{\partial A} = -(A^{-1} Y X^T A^{-1})^T$ [182]. Noteworthily, for ATiSE, the positive facts are expected to have low scores since $\mathbf{P}_{e,t} \approx \mathbf{P}_{r,t}$ when (e_s, r, e_o, t) is valid.

Complexity

In Table 4.2, we summarize the scoring functions of several existing (T)KGC approaches and ATiSE and compare their space complexities. n_{token} denote the number of temporal tokens used in [132]; k is the dimension of embeddings. $\langle \mathbf{x}, \mathbf{y}, \mathbf{z} \rangle = \sum_i x_i y_i z_i$ denotes the tri-linear dot product; $\mathbf{Re}(\cdot)$ denotes the real part of a complex embedding [14]; $\otimes$ denotes the Hamilton product between quaternion embeddings; $\triangleleft$ denotes the normalization of a quaternion embedding. $\mathbf{P}_t$ denotes the temporal projection for embeddings [129]; $\mathbf{LSTM}(\cdot)$ denotes an LSTM neural network; $[\mathbf{r}; \mathbf{t}_{seq}]$ denotes the concatenation of the relation embedding and the sequence of temporal tokens [132]; $\overrightarrow{\mathbf{e}}$ and $\overleftarrow{\mathbf{e}}$ denote the temporal part and untemporal part of a time-specific diachronic entity embedding $\mathbf{e}^t$ [133]; $\mathbf{r}^{-1}$ denotes the embeddings of inverse relation of r, i.e., $(e_s, r, e_o, t) \leftrightarrow (e_o, r^{-1}, e_s, t)$.

As shown in Table 4.2, ATiSE has the same space complexity and time complexity as SKGC models and DE-SimplE. On the other hand, the space complexities of TTransE, HyTE , TA-TransE or TA-DistMult will be higher than ATiSE if $|\mathcal{T}|$ or n_{token} is much larger than $|\mathcal{E}|$ and $|\mathcal{R}|$.

Model	Scoring Function	Space Complexity	Time Complexity
TransE	$\|\|\mathbf{e}_s + \mathbf{r} - \mathbf{e}_o\|\|$	$O(\|\mathcal{E}\|k + \|\mathcal{R}\|k)$	$O(k)$
DistMult	$\langle \mathbf{e}_s, \mathbf{r}, \mathbf{e}_o \rangle$	$O(\|\mathcal{E}\|k + \|\mathcal{R}\|k)$	$O(k)$
ComplEx	$\mathbf{Re}(\langle \mathbf{e}_s, \mathbf{r}, \bar{\mathbf{e}}_o \rangle)$	$O(\|\mathcal{E}\|k + \|\mathcal{R}\|k)$	$O(k)$
RotatE	$\|\|\mathbf{e}_s \circ \mathbf{r} - \mathbf{e}_o\|\|$	$O(\|\mathcal{E}\|k + \|\mathcal{R}\|k)$	$O(k)$
QuatE	$\mathbf{e}_s \otimes \mathbf{r}^{\triangleleft} \cdot \mathbf{e}_o$	$O(\|\mathcal{E}\|k + \|\mathcal{R}\|k)$	$O(k)$
TTransE	$\|\|\mathbf{e}_s + \mathbf{r} + \mathbf{t} - \mathbf{e}_o\|\|$	$O(\|\mathcal{E}\|k + \|\mathcal{R}\|k + \|\mathcal{T}\|k)$	$O(k)$
HyTE	$\|\|\mathbf{P}_t(\mathbf{e}_s) + \mathbf{P}_t(\mathbf{r}) - \mathbf{P}_t(\mathbf{e}_o)\|\|$	$O(\|\mathcal{E}\|k + \|\mathcal{R}\|k + \|\mathcal{T}\|k)$	$O(k)$
TA-TransE	$\|\|\mathbf{e}_s + \mathbf{LSTM}([\mathbf{r}; \mathbf{t}_{seq}]) - \mathbf{e}_o\|\|$	$O(\|\mathcal{E}\|k + \|\mathcal{R}\|k + n_{token}k)$	$O(k)$
TA-DistMult	$\langle \mathbf{e}_s, \mathbf{LSTM}([\mathbf{r}; \mathbf{t}_{seq}]), \mathbf{e}_o \rangle$	$O(\|\mathcal{E}\|k + \|\mathcal{R}\|k + n_{token}k)$	$O(k)$
DE-SimplE	$\frac{1}{2}(\langle \overrightarrow{\mathbf{e}}^t_s, \mathbf{r}, \overleftarrow{\mathbf{e}}^t_o \rangle + \langle \overrightarrow{\mathbf{e}}^t_0, \mathbf{r}^{-1}, \overleftarrow{\mathbf{e}}^t_s \rangle)$	$O(\|\mathcal{E}\|k + \|\mathcal{R}\|k)$	$O(k)$
ATiSE	$\mathcal{D}_{\mathcal{KL}}(\mathbf{P}_{e,t}, \mathbf{P}_{r,t})$	$O(\|\mathcal{E}\|k + \|\mathcal{R}\|k + \|\mathcal{T}\|k)$	$O(k)$

Table 4.2: Comparison of ATiSE with several baseline models for space and time complexity.

Optimization

In this section, we use the margin-based log loss proposed in [110] for optimizing ATiSE. This loss function has been proved to be more effective than the margin rank loss function proposed in [12] on optimizing translation-based KGC models.

$$\mathcal{L} = \sum_{\xi \in \mathcal{Q}_{\text{train}}} \sum_{\xi' \in \mathcal{Q}'_{\text{train}}} -\log \sigma(\gamma - \phi(\xi)) - \log \sigma(\phi(\xi') - \gamma) \tag{4.7}$$

where $\mathcal{Q}_{\text{train}} = \{(e_s, r, e_o, t)\}$ is the set of training quadruples, and $\mathcal{Q}'_{\text{train}}$ is the set of negative samples corresponding to $\mathcal{Q}_{\text{train}}$. Negative samples $\xi' \in \mathcal{Q}'_{\text{train}}$ are generated by randomly corrupting subjects or objects of the positives $\xi \in \mathcal{Q}_{\text{train}}$ such as (e'_s, r, e_o, t) and (e_s, r, e'_o, t). To avoid overfitting, we add some regularizations while learning ATiSE. As described in Equation 4.4, the norms of the time-independent representations of entities and relations, as well as the norms of all evolutionary

Algorithm 1: The learning algorithm of **ATiSE**

input: The training set $\mathcal{Q}_{\text{train}} = \{(e_s, r, e_o, t)\}$, entity set $\mathcal{E}$, relation set $\mathcal{R}$, embedding dimension k, margin γ, batch size b, the ratio of negative samples over the positives η, learning rate lr, restriction values c_{min} and c_{max} for covariance, and a scoring function $f_t(\mathbf{e}_s, \mathbf{r}, \mathbf{e}_o)$ where $e_s, e_o \in \mathcal{E}$, $r \in \mathcal{R}$.

output: Time-independent embeddings for each entity e_i and relation r_j (the original representation vectors and the covariance matrices), the evolutionary rate and the evolutionary direction vector for each entity and relation.

initialize $\mathbf{e}_i, \mathbf{r}_j \leftarrow$ uniform $(-\frac{6}{\sqrt{d}}, \frac{6}{\sqrt{d}})$, $e_i \in \mathcal{E}, r_j \in \mathcal{R}$

$\mathbf{w}_{e,i}, \mathbf{w}_{r,j} \leftarrow$ uniform $(-\frac{6}{\sqrt{d}}, \frac{6}{\sqrt{d}})$, $e_i \in \mathcal{E}, r_j \in \mathcal{R}$

$\mathbf{\Sigma}_{e,i}, \mathbf{\Sigma}_{r,j} \leftarrow$ uniform (c_{min}, c_{max}), $e_i \in \mathcal{E}, r_j \in \mathcal{R}$

$\boldsymbol{\omega}_{e,i}, \boldsymbol{\omega}_{r,j} \leftarrow$ uniform (c_{min}, c_{max}), $e_i \in \mathcal{E}, r_j \in \mathcal{R}$

$\alpha_{e,i}, \alpha_{r,j} \leftarrow$ uniform $(0, 0)$, $e_i \in \mathcal{E}, r_j \in \mathcal{R}$

$\boldsymbol{\beta}_{e,i}, \boldsymbol{\beta}_{r,j} \leftarrow$ uniform $(0, 0)$, $e_i \in \mathcal{E}, r_j \in \mathcal{R}$

loop

$\mathbf{e}_i \leftarrow \mathbf{e}_i / ||\mathbf{e}_i||_2$, $e_i \in \mathcal{E}$

$\mathbf{r}_j \leftarrow \mathbf{r}_j / ||\mathbf{r}_j||_2$, $r_j \in \mathcal{R}$

$\mathbf{w}_{e,i} \leftarrow \mathbf{w}_{e,i} / ||\mathbf{w}_{e,i}||_2$, $e_i \in \mathcal{E}$

$\mathbf{w}_{r,j} \leftarrow \mathbf{w}_{r,j} / |\mathbf{w}_{r,j}||_2$, $r_j \in \mathcal{R}$

$\mathcal{Q}_b \leftarrow$ sample$(\mathcal{Q}_{\text{train}}, b)$ // sample a minibatch

for $(e_s, r, e_o, t) \in \mathcal{Q}_b$ **do**

$\mathcal{Q}'_b = \{(e'_{s,m}, r, e'_{o,m}, t)\}_{m=1\ldots\eta}$ // generate η negative samples

end for

Update $\mathbf{e}_i$, $\mathbf{w}_i$, α_i and $\mathbf{r}_j$ based on Equation 4.9 and 4.7 w.r.t.
$\mathcal{L} = \sum_{\xi \in \mathcal{Q}_b} \sum_{\xi' \in \mathcal{Q}'_b} -\log \sigma(\gamma - f(\xi)) - \log \sigma(f(\xi') - \gamma)$

regularize the covariances for each entity and relation based on Constraint 4.8,
$\mathbf{\Sigma}_{e,i} \leftarrow \max(cmin, \min(cmax, \mathbf{\Sigma}_{e,i}))$, $e_i \in \mathcal{E}$
$\mathbf{\Sigma}_{r,j} \leftarrow \max(cmin, \min(cmax, \mathbf{\Sigma}_{r,j}))$, $r_j \in \mathcal{R}$

end loop

direction vectors, are restricted by 1. Besides, the following constraint is used for guaranteeing that the covariance matrices are positive definite and of appropriate size when we minimize the loss:

$$\forall l \in \mathcal{E} \cup \mathcal{R}, c_{min}\mathbf{I} \leq \mathbf{\Sigma}_l \leq c_{max}\mathbf{I} \tag{4.8}$$

where $\mathbf{I}$ denotes the k-dimensional identity matrix, c_{min} and c_{max} are two tunable positive constants. We use $\mathbf{\Sigma}_l \leftarrow \max(c_{min}, \min(c_{max}, \mathbf{\Sigma}_l))$ to achieve this regularization for diagonal covariance matrices. This constraint 4.8 for the covariance is considered during both the initialization and training process.

4.3.3 Experiments

To show the capability of ATiSE, we compare it with several state-of-the-art SKGC models and

the existing TKGC models on four TKG datasets. Particularly, we also conduct an ablation study to analyze the effects of the dimension of entity/relation embeddings and various components of the additive time series decomposition.

Time Data Preprocessing

As described in Section 4.2, most facts in YAGO11k and Wikidata12k involve time intervals. To deal with a fact $(e_s, r, e_o, [t_s, t_e])$ involving a time interval, we discretize it into multiple quadruples which only involve single time steps, i.e., $\{(e_s, r, e_o, t_s), (e_s, r, e_o, t_{s+1}), \cdots, (e_s, r, e_o, t_e)\}$, where t_s and t_e denote the start time and the end time. During the evaluation process, we define the score of a time interval-based $(e_s, r, e_o, [t_s, t_e])$ as,

$$\phi(e_s, r, e_o, [t_b, t_e]) = \sum_{t=t_b}^{t_e} \phi(e_s, r, e_o, t). \tag{4.9}$$

For a time interval with the missing beginning date or end date, e.g., [2003-##-##, ####-##-##] representing 'since 2003', we use the first time step or the last time step to represent the missing beginning time or end time.

It is mentioned in Section 4.2 that the numbers of occurrences of different timestamps (dates) in ICEWS datasets have approximately uniform distributions while time data have long-tailed distributions in YAGO11k and Wikidata12k at year level. Thus, we fix the length between adjacent time steps as 1 day in ICEWS datasets. For YAGO11k and Wikidata12k, we only deal with year level granularity by dropping the month and date information since its temporal scope is very wide. To alleviate the negative effect of the long-tail property of time data in YAGO11k and Wikidata12k, we distribute the timestamps in the TKGs into different time steps uniformly, i.e., the less frequent continuous year mentions like years between -431 and 100 in YAGO11k are clubbed into the same time steps but years with high frequency form individual time steps, e.g., the years 2013 and 2014. We enforce the number of facts related to each time step to be no less than a minimum threshold of 300. By doing this, we obtain 70 and 81 discrete time steps in YAGO11k and Wikidata12k, respectively.

Baseline

We compare ATiSE with several state-of-the-art SKGC approaches and previous TKGC approaches, including TransE [12], DistMult [13], ComplEx-N3 [116], RotatE [110], QuatE^2 [117], TTransE [177], TA-TransE, TA-DistMult [132] and DE-SimplE [133]. ComplEx-N3 has been proven to have better performance than ComplEx [14] on FreeBase and WordNet datasets. And QuatE^2 has the best performance among all variants of QuatE as reported in [117].

DE-SimplE needs specific date information including year, month and day to score temporal facts, while most time annotations in YAGO and Wikidataset only contain year-level information. Thus, we cannot test these three models on YAGO11k and Wikidataset15k. Since the original source code of TA-TransE and TA-DistMult [132] is not released, we reimplement these models according to the implementation details reported in the original paper, in order to obtain their results on YAGO11k and Wikidata12k.

We do not take Know-Evolve [183] as baseline model due to its problematic formulation and implementation issues. Know-Evolve uses the temporal point process to model the temporal

evolution of each entity. The intensity function of Know-Evolve (Equation 3 in [183]) is defined as $\lambda_r^{s,o}(t|\bar{t}) = f(g_r^{s,o}(\bar{t}))(t - \bar{t})$, where $g(\cdot)$ is a score function, t is current time, and $\bar{t}$ is the most recent time point when either subject or object entity was involved in an event. This intensity function is used in inference to rank entity candidates. However, they don't consider concurrent event at the same timestamps, and thus $\bar{t}$ will become t after one event. For example, we have events $event_1 = (e_s, r, e_1, t_1)$, $event_2 = (e_s, r, e_2, t_1)$. After $event_1$, $\bar{t}$ will become t (subject e_s's most recent time point), and thus the value of intensity function for $event_2$ will be 0. This is problematic in inference since if $t = \bar{t}$, then the intensity function will always be 0 regardless of entity candidates. In their code, they give the highest ranks (first rank) for all entities including the ground truth object in this case, which we think is unfair since the scores of many entity candidates including the ground truth object might be 0 due to their formulation. It has been proven that the performances of Know-Evolve on ICEWS datasets drop down to almost zero after this issue fixed [184].

Experimental Setup

We use Adam optimizer [185] to train ATiSE and select the optimal hyperparameters by early validation stopping according to MRR on the validation set. The maximum epoch is restricted to 5000, and the mini-batch size b is fixed as 512. We tune the embedding dimension k in $\{100, 200, 300, 400, 500\}$, the ratio of negatives over positive training samples η in $\{1, 3, 5, 10\}$ and the learning rate lr in $\{0.00003, 0.0001, 0.0003, 0.001\}$. The margin γ is varied in the range $\{1, 2, 3, 5, 10, 20, \cdots, 120\}$. We select the pair of restriction values c_{min} and c_{max} for covariance among $\{(0.0001, 0.1), (0.003, 0.3), (0.005, 0.5), (0.01, 1)\}$. The default configuration for ATiSE is as follows: $lr = 0.00003$, $k = 500$, $\eta = 10$, $\gamma = 1$, $(c_{min}, c_{max}) = (0.005, 0.5)$. Below, we only list the non-default parameters: $\gamma = 120$, $(c_{min}, c_{max}) = (0.003, 0.3)$ on ICEWS14; $\gamma = 100$, $(c_{min}, c_{max}) = (0.003, 0.3)$ on ICEWS05-15.

Main Results

Table 4.3 and 4.4 show the results for knowledge graph completion task, in which the best results among all models are written bold. In Table 4.3, * indicates that the results are taken from [132], $^\diamond$ indicates that the results are taken from [133] and dashes indicates that the results are unobtainable. On ICEWS14 and ICEWS05-15, ATiSE outperforms all baseline models, considering MR, MRR, Hits@10 and Hits@1. Compared to DE-SimplE which is a very recent state-of-the-art TKGC model, ATiSE gets improvement of 5% on ICEWS14 regarding MRR, and improves Hits@10 by 3% and 6% on ICEWS14 and ICEWS05-15 respectively. On YAGO11k and Wikidata12k where timestamps in facts are time intervals, ATiSE surpasses baseline models regarding MRR, Hits@1, Hits@3. Regarding Hits@10, ATiSE achieves the state-of-the-art results on Wikidata12k and the second best results on YAGO11k. As mentioned in Section 4.3.3, the results of DE-SimplE on YAGO11k and Wikidata12k are unobtainable.

A part of results listed on Table 4.3 and 4.4 are obtained based on the implementations released in [110, 116, 129]. We list the implementation details of some baseline models as follows:

- We use the implementation released in [110] to test RotatE on all four datasets, and DistMult on YAGO11k and Wikidata12k. The source code is revised to adopt the time-wise filtered setting. To search the optimal configurations for RotatE and DistMult, we follow the experimental setups reported in [110] except setting the maximum dimension as 500 and the maximum negative sampling ratio as 10. The default optimal configuration for RotatE and DistMult is as follows:

	ICEWS14				ICEWS05-15			
Metrics	MRR	Hits@1	Hits@3	Hits@10	MRR	Hits@1	Hits@3	Hits@10
TransE*	.280	.094	-	.637	.294	.090	-	.663
DistMult*	.439	.323	-	.672	.456	.337	-	.691
ComplEx-N3	.467	.347	.527	.716	.481	.362	.535	.729
RotatE	.418	.291	.478	.690	.304	.164	.355	.595
QuatE^2	.471	.353	.530	.712	.482	.370	.529	.727
TTransE$^\diamond$	.255	.074	-	.601	.271	.084	-	.616
HyTE$^\diamond$	.297	.108	.416	.655	.316	.116	.445	.681
TA-TransE*	.275	.095	-	.625	.299	.096	-	.668
TA-DistMult*	.477	.363	-	.686	.474	.346	-	.728
DE-SimplE$^\diamond$	.526	.418	.592	.725	.513	**.392**	.578	.748
ATiSE	**.550**	**.436**	**.629**	**.750**	**.519**	.378	**.606**	**.794**

Table 4.3: Knowledge graph completion results of ATiSE on ICEWS14 and ICEWS05-15.

	YAGO11k				Wikidata12k			
Metrics	MRR	Hits@1	Hits@3	Hits@10	MRR	Hits@1	Hits@3	Hits@10
TransE	.100	.015	.138	.244	.178	.100	.192	.339
DistMult	.158	.107	.161	.268	.222	.119	.238	.460
ComplEx-N3	.167	.106	.154	.282	.233	.123	.253	.436
RotatE	.167	.103	.167	**.305**	.221	.116	.236	.461
QuatE^2	.164	.107	.148	.270	.230	.125	.243	.416
TTransE	.108	.020	.150	.251	.172	.096	.184	.329
HyTE	.105	.015	.143	.272	.180	.098	.197	.333
TA-TransE	.127	.027	.160	.326	.178	.030	.267	.429
TA-DistMult	.161	.103	.171	.292	.218	.122	.232	.447
ATiSE	**.170**	**.110**	**.171**	.288	**.280**	**.175**	**.317**	**.481**

Table 4.4: Knowledge graph completion results of ATiSE on YAGO11k and Wikidata12k.

$lr = 0.0001$, $b = 1024$, $d = 500$, $\eta = 10$. Below, we only list the non-default parameters: for RotatE, the optimal margins are $\gamma = 36$ on ICEWS14, $\gamma = 48$ on ICEWS05-15, $\gamma = 3$ on YAGO11k and $\gamma = 6$ on Wikidata12k; for DistMult, the optimal regularizer weights are $r = 0.00001$ on YAGO11k and Wikidata12k.

- We use the implementation released in [116] to test ComplEx-N3 and QuatE^2 on all four datasets. The source code is revised to adopt the time-wise filtered setting. To search the optimal configurations for ComplEx-N3 and QuatE^2, we follow the experimental setups reported in [116] except setting the maximum dimension as 500. The default optimal configuration for ComplEx-N3 and QuatE^2 is as follows: $lr = 0.1$, $d = 500$, $b = 1000$. Below, we list the optimal regularizer weights: for ComplEx-N3, $r = 0.01$ on ICEWS14 and ICEWS05-15, $r = 0.1$ on YAGO11k and Wikidata12k; for QuatE^2, $r = 0.01$ on ICEWS14 and YAGO11k, $r = 0.05$ on

ICEWS05-15, $r = 0.1$ on Wikidata.

- We use the implementation released in [129] to test TransE, TTransE and HyTE on YAGO11k and Wikidata12k for obtaining their performances regarding MRR, Hits@1 and Hits@3. We follow the optimal configurations reported in [129]. As shown in Table 4.4, Hits@10s of TransE and TTransE we get are better than those reported in [129].
- As shown in Table 4.3, other baseline results are taken from [132, 133].

Ablation Study

In this section, we analyze the effects of the dimension and various components of entity/relation embeddings on ATiSE's performances.

The embedding dimension is an important hyperparameter for each (T)KGC model. A high embedding dimension might be beneficial to boost the performance of a (T)KGC model. For instance, ComplEx-N3 and QuatE^2 achieve the state-of-the-art results on knowledge graph compeletion over SKGs with 2000-dimensional embeddings [116, 117]. On the other hand, a lower embedding dimension leads to less consumption on training time and memory space, which is quite important for the applications of (T)KGC models on large-scale datasets. Figure 4.5 shows the performances of ATiSE with different embedding dimensions on ICEWS14. With a same embedding dimension of 100 as DE-SimplE [133], ATiSE still achieves the state-of-the-art results on ICEWS14. An ATiSE model with an embedding dimension of 100 trained on ICEWS14 has a memeory size of 14.2Mb while a DE-SimplE model and a QuatE^2 model with the same embedding dimension have memory sizes of 13.3Mb and 12.4Mb. And the memory size of an ATiSE model increases linearly with its embedding dimension. Moreover, training an ATiSE model with an embedding dimension of 100 takes 2.8 seconds per epoch on a single GeForce RTX2080, and an ATiSE with 500-dimensional embeddings takes 3.7 seconds per epoch.

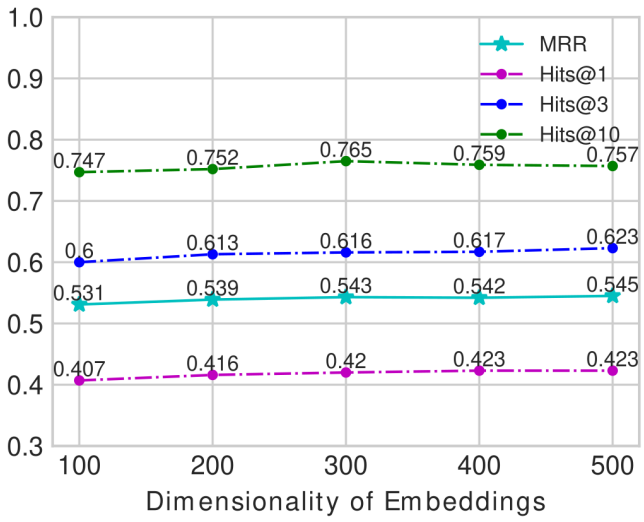


Figure 4.5: Results for ATiSE with different embedding dimensions on ICEWS14.

To analyze the effects of different components of entity/relation representation in ATiSE, we developed three comparison models, namely, ATiSE-SN, ATiSE-TN and ATiSE-TS, which exclude the trend component, seasonal component and the noise component, respectively. The entity representations of these three comparison models are as follows:

$$\begin{aligned}\mathbf{e}_{i,t}^{\mathrm{SN}} &= \mathbf{e}_i + \boldsymbol{\beta}_{e,i}\sin(2\pi\boldsymbol{\omega}_{e,i}t) + \mathcal{N}(0, \boldsymbol{\Sigma}_{e,i}),\\ \mathbf{e}_{i,t}^{\mathrm{TN}} &= \mathbf{e}_i + \alpha_{e,i}\mathbf{w}_{e,i}t + \mathcal{N}(0, \boldsymbol{\Sigma}_{e,i}),\\ \mathbf{e}_{i,t}^{\mathrm{TS}} &= \mathbf{e}_i + \alpha_{e,i}\mathbf{w}_{e,i}t + \boldsymbol{\beta}_{e,i}\sin(2\pi\boldsymbol{\omega}_{e,i}t).\end{aligned} \tag{4.10}$$

For ATiSE-TS consisting of the trend component and the seasonal component, we use the translation-based scoring function [12] to measure the plausibility of the fact (e_s, r, e_o, t).

$$f_t^{TS}(\mathbf{e}_s, \mathbf{r}, \mathbf{e}_o) = ||\mathbf{e}_{s,t}^{\mathrm{TS}} + \mathbf{r}_t^{\mathrm{TS}} - \mathbf{e}_{o,t}^{\mathrm{TS}}|| \tag{4.11}$$

We report the MRRs and Hits@10 of ATiSE-SN, ATiSE-TN and ATiSE-TS on knowledge graph completion over ICEWS14 and YAGO11k. As shown in Table 4.5, we find that the removals of the trend component and the noise component have remarkable negative effects on the performance of ATiSE on knowledge graph completion since the model could not address the temporal uncertainty of entity/relation representations without the noise component and the trend component contains the main time information. In ATiSE, different types of entities might have big difference in the trend component. For instance, we find that the embeddings of entities representing people, e.g., *Barack*

Datasets	ICEWS14				YAGO11k			
Metrics	MRR	Hits@1	Hits@3	Hits@10	MRR	Hits@1	Hits@3	Hits@10
ATiSE-SN	.405	.284	.488	.710	.139	.095	.143	.249
ATiSE-TN	.536	.407	.626	.771	.167	.105	.170	.282
ATiSE-TS	.323	.127	.429	.676	.115	.023	.145	.274
ATiSE	**.550**	**.436**	**.629**	**.750**	**.170**	**.110**	**.171**	**288**

Table 4.5: Knowledge graph completion results of ablation experiments.

Relations	$\overline{\mathrm{TS}}$	$\overline{\lvert\alpha_r\rvert}$	$\overline{\lvert\boldsymbol{\beta}_r\rvert}$	$\overline{\lvert\boldsymbol{\omega}_r\rvert}$
wasBornIn	1.0	0.142	0.000	1.032
worksAt	18.7	0.046	0.058	0.294
playsFor	4.7	0.071	0.046	0.766
hasWonPrize	28.6	0.010	0.107	0.041
isMarriedTo	16.5	0.049	0.076	0.090
owns	24.9	0.017	0.088	0.101
graduatedFrom	38.1	0.016	0.104	0.029
deadIn	1.0	0.249	0.006	0.897
isAffliatedTo	25.8	0.014	0.049	0.126
created	27.1	0.011	0.040	0.087

Table 4.6: Relations in YAGO11k and statistics of their representation parameters.

Obama, generally have higher evolution rates than those representing cities or nations, e.g., *USA*.

ATiSE-TN performs worse than ATiSE on YAGO11k where facts involve time intervals. Different from ICEWS14 dataset which is an event-based dataset where all relations or predicates are instantaneous, there exist both short-term relations and long-term relations in YAGO11k. Adding seasonal components into evolving entity/relation representations is helpful for distinguishing short-term patterns and long-term patterns in YAGO11k. In Table 4.6, we list different relations in YAGO11k and the mean step numbers of their duration time ($\overline{\text{TS}}$), as well as the corresponding parameters learned from ATiSE, including the evolutionary rate $|\alpha_r|$, the mean amplitude $\overline{|\boldsymbol{\beta}_r|}$ and the mean frequency $\overline{|\omega_r|}$ of the seasonal component for each relation. It can be seen from Table 4.6 that short-term relations learned by ATiSE, e.g., *wasBornIn*, generally have higher evolutionary rates, and their seasonal components have smaller amplitudes and higher frequencies than long-term relations, e.g., *isMarriedTo*.

4.3.4 Conclusion

We introduce ATiSE, a TKGC model that incorporates time information into KG representations by fitting the temporal evolution of entity/relation representations over time as additive time series. Considering the uncertainty during the temporal evolution of KG representations, ATiSE maps the representations of temporal KGs into the space of multi-dimensional Gaussian distributions. The covariance of an entity/relation representation represents its randomness component. Experimental results demonstrate that ATiSE significantly outperforms the state-of-the-art methods on knowledge graph completion over four TKG benchmarks. Our work establishes a previously unexplored connection between relational processes and time series analysis with the potential to open a new direction of research on reasoning over time.

Compared to the previous TKGC work, ATiSE has the following advantages:

- ATiSE considers the temporal uncertainty during the evolution of semantics of entities and relations;
- ATiSE provides an interpretable method to model the changes of entity and relation embeddings over time with an additive time serires embedding model;
- ATiSE adapts well to various TKGs where timestamps have different representation forms;
- ATiSE adapts well to various TKGs where time data have different distributions.

4.4 A TKGC Model Based on Temporal Complex Rotation

The content of the this section is based on our work in the paper titled "TeRo: A Time-aware Knowledge Graph Embedding Model via Temporal Rotation" (Xu et al., COLING 2020) [178].

4.4.1 Introduction

Previous TKGC models [32, 129, 132, 177, 186] including ATiSE are shown to have better performances on knowledge graph completion over TKGs than SKGC models. However, most of the previous TKGC models are the temporal extensions of TransE [12] and DistMult [13], and thus are not fully expressive for some relation patterns [110].

In this section, we present a novel approach for TKGC, TeRo, which defines the temporal evolution of an entity embedding as a rotation from the initial time to the current time in the complex vector

space. We show the limitations of the existing TKGC models and the advantage of our presented model in learning various relation patterns over time.

Specially, for facts involving time intervals, each relation is represented as a pair of dual complex embeddings which are used to handle the beginning and the end of the relation, respectively. In this way, TeRo can adapt well to datasets where time annotations are represented in various forms: time points, beginning or end time, and time intervals. A running example is provided to show how TeRo models time presentations of different forms in TKGs.

Most of the previous TKGC-related works as far as we know use specific time granularities for various TKGs. For example, the time granularity of the ICEWS datasets is fixed as 24 hours in [32, 129]. In this section, we adopt various time-split approaches for different TKG datasets and investigate the effect of time granularities on the performance of TeRo. For evaluation, we compare the performance of TeRo on knowledge graph completion over four different TKGs with the state-of-the-art SKGC models and the existing TKGC models. The experimental results demonstrate that TeRo outperforms other baseline models significantly by inferring various relation patterns and encoding time information.

4.4.2 Methodology

Although various SKGC models have been developed to learn multi-relational interactions between entities, all of them have problems with inferring temporary relations which are only valid for a certain time point or last for a certain time period. To illustrate this by example, assume we are given a quadruple (*Barack Obama, president of, USA, 2014*) as a training sample, where the relation *visits* is a temporary relation. If we query (*?, president of, USA, 2018*), a trained SKGC model probably returns the incorrect answer *Barack Obama* due to the validity of the triple (*Barack Obama, president of, USA*), while the correct answer is *Donald Trump* considering the given time constraint. On the other hand, most of the existing TKGC models, which are extended from TransE [12] and DistMult [13], incorporate time information in the embedding space, but have limitations on learning transitive relations or asymmetric relations.

Scoring Function

To overcome the limitations of these existing SKGC and TKGC models on learning and inferring over TKGs, we propose a new TKGC model, TeRo, which defines the temporal evolution of an entity embedding as a rotation in the complex vector space. We map each entity $e_i \in \mathcal{E}$ and relation $r_j \in \mathcal{R}$ to their complex embeddings, i.e., $\mathbf{e}_i, \mathbf{r}_j \in \mathbb{C}^k$; then we define the functional mapping induced by each time step t as an element-wise rotation from the time-independent entity embedding $\mathbf{e}_i$ to the time-specific entity embedding $\mathbf{e}_{i,t}$. The mapping function is defined as follows:

$$\mathbf{e}_{i,t} = \mathbf{e}_i \circ \mathbf{t}, \tag{4.12}$$

where $\circ$ denotes the Hermitian dot product between complex vectors. Here, we constrain the modulus of each element of $\mathbf{t} \in \mathbb{C}^k$, i.e., $t^j \in \mathbb{C}$, to be $|t^j| = 1$. By doing this, t^j is of the form $e^{\mathbf{i}\theta_t^j}$, which corresponds to a counter-clockwise rotation by θ_t^j radians around the origin of the complex plane, and only affects the phases of the entity embeddings in the complex vector space. This idea is motivated by Euler's identity $e^{\mathbf{i}\theta} = cos\theta + \mathbf{i}sin\theta$, which indicates that a unitary complex number can be regarded

as a rotation in the complex plane.

We regard the relation embedding $\mathbf{r}$ as translation from the time-specific subject embedding $\mathbf{e}_{s,t}$ to the conjugate of the time-specific object embedding $\bar{\mathbf{e}}_{o,t}$ for a single quadruple $(e_s, r, e_o, t) \in \mathcal{Q}$. The scoring function is defined as:

$$\phi(e_s, r, e_o, t) = ||\mathbf{e}_{s,t} + \mathbf{r} - \bar{\mathbf{e}}_{o,t}||. \tag{4.13}$$

Noteworthily, a positive quadruple are expected to have low scores for TeRo. Figure 4.6 provides the illustration of TeRo with only 1-dimensional embeddings.

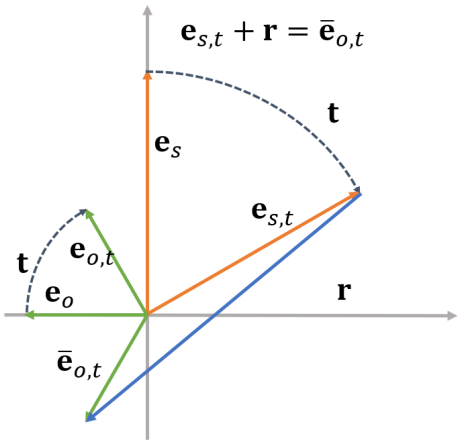


Figure 4.6: Illustration of TeRo with one-dimensional embeddings.

For a fact (e_s, r, e_o, t) occurring in a certain time interval, i.e., $t = [t_b, t_e]$ where t_b, t_e denote the beginning time and the end time of the fact, we separate this fact into two quadruples, namely, (e_s, r_b, e_o, t) and (e_s, r_e, e_o, t). Here, we extend the relation set $\mathcal{R}$ in a TKG which involves time intervals to a pair of dual relation sets, $\mathcal{R}_b$ and $\mathcal{R}_e$. A relation $r_b \in \mathcal{R}_b$ is used to handle the beginning of relation r, meanwhile a relation $r_e \in \mathcal{R}_e$ is used to handle the end of relation r. By doing this, we score a fact $(e_s, r, e_o, [t_b, t_e])$ as the mean value of scores of two quadruples, (e_s, r_b, e_o, t_b) and (e_s, r_e, e_o, t_e) which represent the beginning and the end of this fact respectively.

$$\phi(s, r, o, [t_b, t_e]) = \frac{1}{2}(||\mathbf{e}_{s,t_b} + \mathbf{r}_b - \bar{\mathbf{e}}_{o,t_b}|| + ||\mathbf{e}_{s,t_e} + \mathbf{r}_e - \bar{\mathbf{e}}_{o,t_e}||) \tag{4.14}$$

Specially, for a fact missing either the beginning time or the end time, e.g., $(e_s, r, e_o, [t_b, -])$ or $(e_s, r, e_o, [-, t_e])$, the score of this fact is equal to the score of the quadruple involving the known time, i.e., $\phi(e_s, r, e_o, [t_b, -]) = \phi(e_s, r_b, o, t_b)$, $\phi(e_s, r, e_o, [-, t_e]) = \phi(e_s, r_e, e_o, t_e)$.

Taking a YAGO11k fact (*Archuduchess Yolande, isMarriedTo, Archduke Carl Ludwig, [1950,−]*) as an example, we first decompose this fact into two quadruples involving time points as shown in Fig. 4.7. Since the time annotation *[1950,-]* misses the ending time, one of these two quadruples involves an unknown time point. We then map entities into complex vectors $\mathbf{e}_{2378}$ and $\mathbf{e}_{6165}$ according to their indices. As mentioned before, we create two embeddings for each relation to handle its beginning and end. In this case, we use relation embeddings $\mathbf{r}_4$ and $\mathbf{r}_{14}$ to represent *isMarriedTo+Since* and *isMarriedTo+Until*.

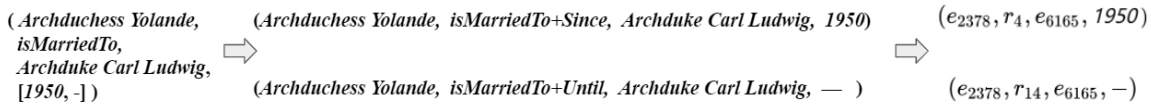


Figure 4.7: Illustration of the decomposition of a temporal fact time involving a time interval.

Loss Function

Same as ATiSE, we use the margin-based log loss for optimizing TeRo. Moreover, we introduce a self-adversarial learning strategy in the loss function to assign different weight to different negative samples. The loss for each quadruple is defined as,

$$\mathcal{L}(\xi) = -\log \sigma(\gamma - \phi(\xi)) - \sum_{i=1}^{\eta} p_i \log \sigma(\phi(\xi_i') - \gamma), \tag{4.15}$$

where $\xi \in \mathcal{Q}_{\text{train}}$ is a positive training quadruple, ξ_i' is the ith negative sample corresponding to ξ generated by randomly corrupting the subject or the objects of ξ such as (e_s', r, e_o, t) and (e_s, r, e_o', t), p_i denotes the weight the ξ_i', $\sigma(\cdot)$ denotes the sigmoid function, γ is a fixed margin, η is the ratio of negatives over positive training samples. The weight p_i' of the negative sample ξ_i is computed as,

$$p_i = \frac{\exp(-\phi(\xi_i'))}{\sum_{j=1}^{\eta} \exp(-\phi(\xi_j'))}. \tag{4.16}$$

The motivation of the self-adversarial learning strategy is to assign large weights to hard negative samples. In our case, the negative samples are expected to have high scores and thus negative samples with low scores are regarded as hard negative samples with large weights during optimization.

Model Expressiveness

Static KGE models and some existing TKGE models which are the temporal extensions of TransE or DistMult have limitations on capturing some key relation patterns which are defined as follows.

Definition of Relation Patterns

Definition 1. If $\exists\ e_s, e_o, t_1, t_2$ make $(e_s, r, e_o, t_1) \wedge \neg(e_s, r, e_o, t_2)$ hold True, then the relation r is a **temporary** relation.

Definition 2. If $\forall e_s, e_o, t, (e_s, r, e_o, t) \Rightarrow (e_o, r, e_s, t)$, then the relation r is a **symmetric** relation.

Definition 3. If $\exists\ e_s, e_o, t$ make $(e_s, r, e_o, t) \wedge \neg(e_o, r, e_s, t)$ hold true, then the relation r is an **asymmetric** relation.

As mentioned before, existing SKGC models and previous TKGC models have difficulty modeling all of the above relation patterns:

- SKGC models can not model temporary relations r_{temp}, e.g., '*dead in*' and '*born in*', since $\phi_{\text{SKGC}}(e_s, r_{temp}, e_o, t_1) \equiv \phi_{\text{SKGC}}(e_s, r_{temp}, e_o, t_2)$.

- Temporal extensions of TransE (denoted as T-TransE) including HyTE, TTransE, TA-TransE have difficulty modeling symmetric relations r_{sym}, e.g., '*friend of*' and '*married to*', since $||\mathbf{e}_{s,t} + \mathbf{r}_{sym,t} - \mathbf{e}_{o,t}|| = ||\mathbf{e}_{o,t} + \mathbf{r}_{sym,t} - \mathbf{e}_{s,t}|| = 0 \Rightarrow \mathbf{r}_{sym,t} = \mathbf{0}$.

- Due to the natural symmetric characteristic of DistMult, temporal extensions of DistMult (denoted as T-DistMult) including TDistMult, TA-DistMult and Know-Evolve can not model asymmetric relations r_{asym}, e.g., '*parent Of*', since $f_{\text{T-DistMult}}(e_s, r, e_o, t) = \langle \mathbf{e}_{s,t}, \mathbf{r}_t, \mathbf{e}_{o,t} \rangle = f_{\text{T-DistMult}}(e_o, r_{asym}, e_s, t)$, where $\mathbf{e}_{s,t}, \mathbf{e}_{o,t}, \mathbf{r}_{asym,t}$ are time-specific entity/relation embeddings corresponding to different T-DistMult models.

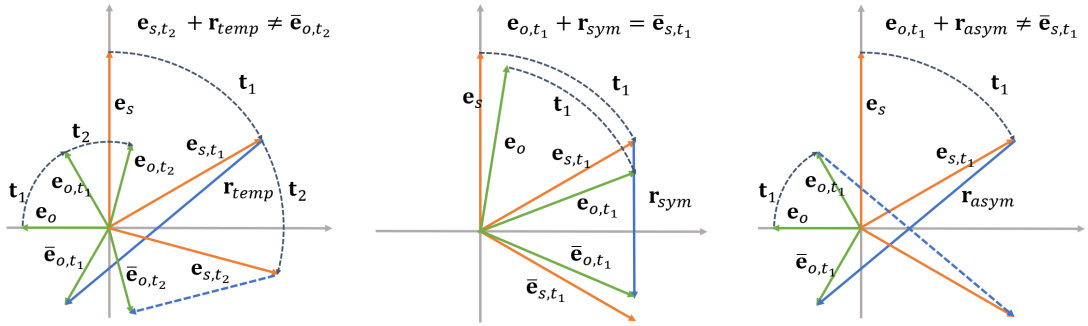


Figure 4.8: Illustrations of TeRo modeling different relation patterns.

By defining each time step as a rotation in the complex vector spaces, TeRo can capture all of the above three relation patterns. Given an observed fact (e_s, r, e_o, t) where $\mathbf{e}_{s,t_1} + \mathbf{r} = \bar{\mathbf{e}}_{o,t_1}$:

- as shown in Figure 4.8(a), if r_{temp} is a temporary relation, we can have $\mathbf{e}_{s,t_2} + \mathbf{r} \neq \bar{\mathbf{e}}_{o,t_2}$ for TeRo to make $(e_s, r, e_o, t_1) \wedge \neg(e_s, r, e_o, t_2)$ hold true.

- as shown in Figure 4.8(b), if r_{sym} is a symmetric relation, we can have $\mathbf{e}_{o,t_1} + \mathbf{r}_{asym} = \bar{\mathbf{e}}_{s,t_1}$ with $\mathbf{Re}(\mathbf{r}_{sym}) = \mathbf{0}$ for TeRo to make $(e_s, r, e_o, t) \Rightarrow (e_o, r, e_s, t)$ hold true. Thus, TeRo can represent multiple reflexive relations as different embeddings due to the conjugate operations of object embeddings.

- as shown in Figure 4.8(c), if r_{asym} is an asymmteric relation, we can have $\mathbf{e}_{o,t_1} + \mathbf{r}_{asym} \neq \bar{\mathbf{e}}_{s,t_1}$ with $\mathbf{Re}(\mathbf{r}_{asym}) \neq \mathbf{0}$ for TeRo to make $r(s, o, t_1) \wedge \neg r(o, s, t_1)$ hold true .

Complexity

As shown in Table 4.7, the space complexity of TeRo and TransE will be close if $|\mathcal{T}| < n_e$. In practice, we can achieve this condition by tuning the time granularity.

Model	Scoring Function	Space Complexity	Time Complexity
TransE	$\|\|\mathbf{e}_s + \mathbf{r} - \mathbf{e}_o\|\|$	$O(\|\mathcal{E}\|k + \|\mathcal{R}\|k)$	$O(k)$
TTransE	$\|\|\mathbf{e}_s + \mathbf{r} + \mathbf{t} - \mathbf{e}_o\|\|$	$O(\|\mathcal{E}\|k + \|\mathcal{R}\|k + \|\mathcal{T}\|k)$	$O(k)$
HyTE	$\|\|\mathbf{P}_t(\mathbf{e}_s) + \mathbf{P}_t(\mathbf{r}) - \mathbf{P}_t(\mathbf{e}_o)\|\|$	$O(\|\mathcal{E}\|k + \|\mathcal{R}\|k + \|\mathcal{T}\|k)$	$O(k)$
TA-TransE	$\|\|\mathbf{e}_s + \mathbf{LSTM}([\mathbf{r}; \mathbf{t}_{seq}]) - \mathbf{e}_o\|\|$	$O(\|\mathcal{E}\|k + \|\mathcal{R}\|k + n_{token}k)$	$O(k)$
TA-DistMult	$\langle \mathbf{e}_s, \mathbf{LSTM}([\mathbf{r}; \mathbf{t}_{seq}]), \mathbf{e}_o \rangle$	$O(\|\mathcal{E}\|k + \|\mathcal{R}\|k + n_{token}k)$	$O(k)$
DE-SimplE	$\frac{1}{2}(\langle \overrightarrow{\mathbf{e}}^t_s, \mathbf{r}, \overleftarrow{\mathbf{e}}^t_o \rangle + \langle \overrightarrow{\mathbf{e}}^t_0, \mathbf{r}^{-1}, \overleftarrow{\mathbf{e}}^t_s \rangle)$	$O(\|\mathcal{E}\|k + \|\mathcal{R}\|k)$	$O(k)$
ATiSE	$\mathcal{D}_{\mathcal{KL}}(\mathbf{P}_{e,t}, \mathbf{P}_{r,t})$	$O(\|\mathcal{E}\|k + \|\mathcal{R}\|k)$	$O(k)$
TeRo	$\|\|\mathbf{e}_{s,t} + \mathbf{r} - \bar{\mathbf{e}}_{o,t}\|\|$	$O(\|\mathcal{E}\|k + \|\mathcal{R}\|k + \|\mathcal{T}\|k)$	$O(k)$

Table 4.7: Comparison of TeRo with TransE and previous TKGC models for space and time complexity.

4.4.3 Experiments

To show the capability of TeRo, we compare it with the state-of-the-art SKGC models and the existing TKGC models including ATiSE on four TKG datasets. Particularly, we conduct an ablation study to analyze the effects of the time granularity and embedding dimension on TeRo's performance and an efficiency study of dual relation embeddings used in TeRo.

Time Data Preprocessing

In the case of ATiSE, we use different time-split methods for different TKGs. Specifically, for ICEWS datasets in which time data have a uniform distribution, the time unit is fixed as 1 day while the lengths of different time steps are different for the balance of numbers of triples in different time steps in YAGO11k and Wikidata12k. However, it has not been investigated whether the lengths of time steps affect the performances of TKGC models. In this section, we follow ATiSE to use two different

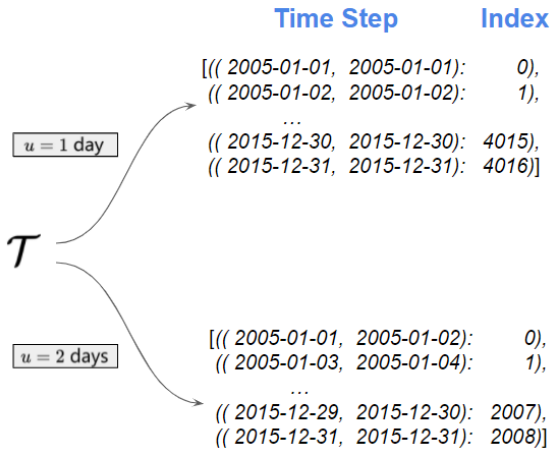


Figure 4.9: Illustrations of the construction of the time set $\mathcal{T}$ with different time granularity parameters u for ICEWS05-15.

time-split methods for uniformly distributed data and long-tailed distributed data.

Uniform processing. we test TeRo with different time units, denoted as u, in a range of $\{1, 2, 3, 7, 14, 30, 90$ and $365\}$ days for ICEWS datasets. The change of time unit will reconstruct the set of time steps $\mathcal{T}$. As shown in Figure 4.9, for ICEWS05-15, when the time unit u is 1 day, we have totally 4017 time steps and the date *2005-01-02* is represented by the second time step, i.e., t_1. If the time unit is changed as 2 days, the total number of time steps will be 2009 and the date *2005-01-02* will be denoted as t_0.

Clubbing processing. The main motive behind clubbing processing is to distribute the time annotations in the TKG into discrete time steps uniformly. Specifically, years with high frequency form individual time steps but less frequent year mentions are clubbed into the same time step in order to alleviate the effect of the long-tail property of time data. In ATiSE, we apply a minimum threshold of 300 triples per time step during construction of $\mathcal{T}$ for YAGO11k and Wikidata12k. In this section, we try different minimum thresholds tr amongst $\{1, 3, 10, 30, 100, 300, 1000, 3000, 10000, 30000\}$ for clubbing years. For YAGO11k, $\mathcal{T}$ include 388 time steps with $thre = 1$ as shown in Figure 4.10. Years like *-453*, and *1950* are all taken as independent time steps. When $thre$ for YAGO11k rises to 100, the number of time steps drops to 127 and years between *-431* and *1865* are clubbed into a same time step.

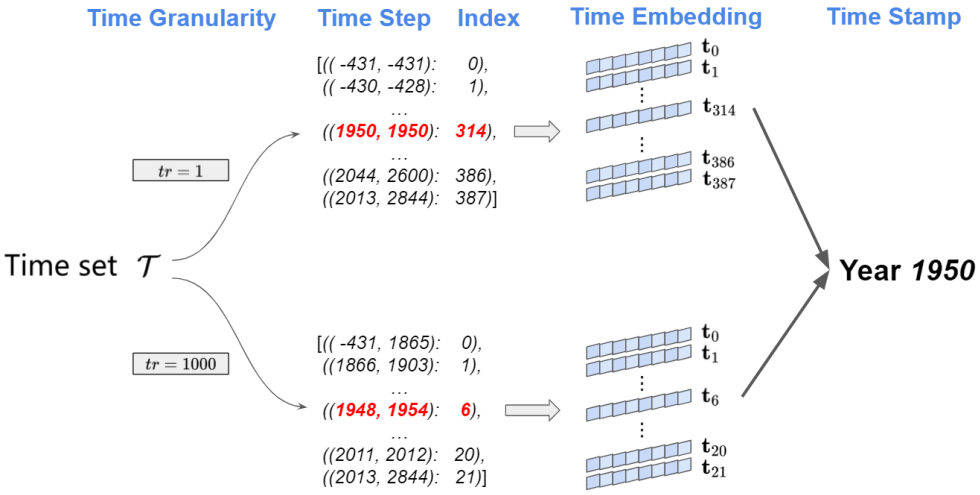


Figure 4.10: Illustrations of the construction of the time set $\mathcal{T}$ with different time granularity parameters $thre$ for YAGO11k.

Baseline

We compare TeRo with several state-of-the-art SKGC approaches and previous TKGC approaches, including TransE [12], DistMult [13], ComplEx-N3 [116], RotatE [110], QuatE [117], TTransE [177], TA-TransE, TA-DistMult [132], DE-SimplE [133] and ATiSE [32]. The results of baselines are taken from Table 4.3 and 4.4. As mentioned in Section 4.3.3, DE-SimplE which mainly focuses on event-based datasets, cannot model time intervals or time annotations missing moth and day information which are common in YAGO and Wikidata. Thus its result on YAGO11k and Wikidata12k are unobtainable.

Experimental Setup

We implement our proposed model in PyTorch. The code is available at https://github.com/soledad921/ATISE.

We select the optimal hyperparameters by early validation stopping according to MRR on the validation set. We restrict the maximum number of iterations to 5000. Following the setup used in [32], the batch size $b = 512$ is kept for all datasets, the embedding dimension k is tuned in $\{100, 200, 300, 400, 500\}$, the ratio of negative over positive training samples η is tuned in $\{1, 3, 5, 10\}$ and the margin γ is tuned in $\{1, 2, 3, 5, 10, 20, \cdots, 120\}$. Regarding optimizer, we choose Adagrad [187] for TeRo and tune the learning rate r in a range of $\{1, 0.3, 0.1, 0.03, 0.01\}$. Specially, the time granularity parameters u and $thre$ are also regarded as hyperparameters for TeRo as mentioned before.

The default configuration for TeRo is as follows: $d = 500$, $\eta = 10$. Below, we only list the non-default parameters: $lr = 0.1$, $\gamma = 110$, $u = 1$ on ICEWS14; $lr = 0.1$, $\gamma = 120$, $u = 2$ on ICEWS05-15; $lr = 0.1$, $\gamma = 50$, $thre = 100$ on YAGO11k; $lr = 0.3$, $\gamma = 20$, $thre = 300$ on Wikidata12k.

Main Results

Table 4.8 and 4.9 list all knowledge graph completion results of our proposed model and baseline models on four datasets. TeRo surpasses all baseline embedding models regarding all metrics on all datasets except that ATiSE gets the better Hits@3 and Hits@10 than TeRo on ICEWS14. Compared to ATiSE, TeRo achieves the improvement of 1.2 MRR points, 6.7 MRR points, 1.7 MRR points and 1.9 MRR points on ICEWS14, ICEWS05-15, YAGO11k and Wikidata12k respectively.

Datasets	ICEWS14				ICEWS05-15			
Metrics	MRR	Hits@1	Hits@3	Hits@10	MRR	Hits@1	Hits@3	Hits@10
TransE	.280	.094	-	.637	.294	.090	-	.663
DistMult	.439	.323	-	.672	.456	.337	-	.691
ComplEx-N3	.467	.347	.527	.716	.481	.362	.535	.729
RotatE	.418	.291	.478	.690	.304	.164	.355	.595
QuatE	.471	.353	.530	.712	.482	.370	.529	.727
TTransE	.255	.074	-	.601	.271	.084	-	.616
HyTE	.297	.108	.416	.655	.316	.116	.445	.681
TA-TransE	.275	.095	-	.625	.299	.096	-	.668
TA-DistMult	.477	.363	-	.686	.474	.346	-	.728
DE-SimplE	.526	.418	.592	.725	.513	.392	.578	.748
ATiSE	.550	.436	**.629**	**.750**	.519	.378	.606	.794
TeRo	**.562**	**.468**	.621	.732	**.586**	**.469**	**.668**	**.795**

Table 4.8: Knowledge graph completion results of TeRo on ICEWS14 and ICEWS05-15.

Datasets	YAGO11k				Wikidata12k			
Metrics	MRR	Hits@1	Hits@3	Hits@10	MRR	Hits@1	Hits@3	Hits@10
TransE	.100	.015	.138	.244	.178	.100	.192	.339
DistMult	.158	.107	.161	.268	.222	.119	.238	.460
ComplEx-N3	.167	.106	.154	.282	.233	.123	.253	.436
RotatE	.167	.103	.167	.305	.221	.116	.236	.461
QuatE^2	.164	.107	.148	.270	.230	.125	.243	.416
TTransE	.108	.020	.150	.251	.172	.096	.184	.329
HyTE	.105	.015	.143	.272	.180	.098	.197	.333
TA-TransE	.127	.027	.160	.326	.178	.030	.267	.429
TA-DistMult	.161	.103	.171	.292	.218	.122	.232	.447
ATiSE	.170	.110	.171	.288	.280	.175	.317	.481
TeRo	**.187**	**.121**	**.197**	**.319**	**.299**	**.198**	**.329**	**.507**

Table 4.9: Knowledge graph completion results of TeRo on YAGO11k and Wikidata12k.

Ablation Study

In this section, we analyze the effect of the change of time granularity on the performance of TeRo. As mentioned before, we adopt two different time-split approaches for event-based datasets, i.e., ICEWS datasets, and time-wise KGs involving time intervals, i.e., YAGO11k as well as Wikidata12k. For ICEWS14 and ICEWS05-15, we use time steps with fixed length since the distribution of numbers of facts in ICEWS datasets over time is relatively uniform as shown in Figure 4.2. The time granularities of ICEWS datasets are equal to the lengths of time units u . On the other hand, the distributions of numbers of facts in YAGO15k and Wikidata12k are long-tailed. Thus we divide the time steps in YAGO15k and Wikidata12k by setting a mini threshold for the numbers of facts in each time step. Time granularities of these two datasets can be changed by setting different thresholds $thre$.

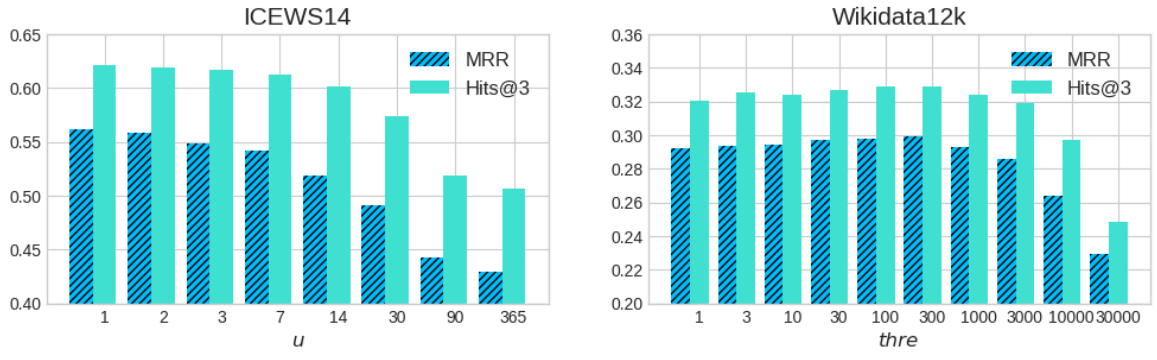


Figure 4.11: Results of TeRo with different time granularities on ICEWS14 and Wikidata12k.

In ICEWS14, time distribution is relatively uniform and thus representing time with a fine time granularity can provide more abundant time information. As shown in Figure 4.11, TeRo with fine

time granularities, e.g., 1 day, 2 days and 3 days, has better performance on ICEWS14 compared to TeRo with coarser time granularities regarding MRR and Hits@3. Likewise, the optimal time unit for TeRo on ICEWS05-15 has been proven by our experiments to be 2 days. For Wikidata12k, using a very fine time granularity is non-optimal due to the long-tail property of time data. On the other hand, using an overly coarse time granularity results in the invalid incorporation of time information. Figure 4.11 demonstrates the low performances of TeRo with coarse time granularities. More concretely, when time unit u is 1 year, all timestamps in ICEWS14 are represented by the same time embedding, which means this time embedding is temporally unmeaningful.

Knowledge Graph Completion Results	TeRo with u= 1 day	TeRo with u= 365 days
Colombia, Host a visit, ?, 2014-06-04	**Kyung-wha Kang**	John F. Kelly
Head of Government (China), visits, ?, 2014-07-04	**South Korea**	Serbia
UN Security Council, Criticize or denounce, ?, 2014-08-10	**North Korea**	Armed Band (South Sudan)
South Korea, Host a visit, ?, 2014-06-20	**Kim Jong-Un**	National Security Advisor (Japan)
Police (Australia), Accuse, ?, 2014-10-22	**Criminal (Australia)**	Citizen (Australia)

Table 4.10: Examples of knowledge graph completion results of TeRo with different time units on ICEWS14..

Table 4.10 demonstrates a few examples of knowledge graph completion results of TeRo models with time units u of two days and one year on ICEWS14. The correct prediction results are written bold. As shown in Table 4.10, in many cases, TeRo with $u = 1$ predicates correctly, meanwhile TeRo with $u = 365$ gives the wrong results. We notice that these predictions of TeRo with $u = 365$ in Table 4.10 would be valid if we disregarded the time constraints. For instance, (*Colombia*, *Host a visit*, *John F. Kelly*) happened on 2014-03-27, (*UN Security Council*, *Criticize or denounce*, *Armed Band (South Sudan)*) was true on 2014-08-07. According to the definitions in Section 4.4.2, *Host a visit* and *Criticize or denounce* are temporary relations. The above results prove that using a reasonable time granularity is helpful for TeRo effectively incorporating time information. And the inclusion of time information enables TeRo to capture temporary relations and improve its performance on knowledge graph completion over TKGs.

Efficiency Study

TeRo has the same space complextiy as TTransE [177] and HyTE [129]. Since we constrain the numbers of time steps of the four TKG datasets by tuning time granularities (183 time steps in ICEWS14, 1339 time steps in ICEWS05-15, 127 time steps in YAGO11k and 82 time steps in Wikidata12k), the numbers of time steps are much less than the numbers of entities in these datasets, which means that the space complexity of TeRo is close to the space complexity of TransE [12] as mentioned in Section 4.4.2. Regarding the concrete memory consumption, the recent state-of-the-art TKGC models, ATiSE [32] and DE-SimplE [133] have 1.8 times and 2.2 times as large memory size as TeRo on ICEWS14 with the same embedding dimension. The training processes of TeRo with 500-dimensional embeddings on ICEWS14, ICEWS05-15, YAGO11k and Wikidata12k take 4.3 seconds, 25.9 seconds, 1.9 seconds and 4.1 seconds per epoch on a single GeForce RTX 2080 device, respectivly.

It is also noteworthy that representing each relation as a pair of dual complex embeddings is helpful for saving training time on TKGs involving time intervals. Given a fact $(e_s, r, e_o, [t_b, t_e])$, some TKGE models, e.g., HyTE and ATiSE, discretize this fact into several quadruples involving continuous

time points, i.e., $[(e_s, r, e_o, t_b), (e_s, r, e_o, t_b + 1), \cdots, (e_s, r, e_o, t_e)]$. When $thre = 300$, each fact lasts for averagely around 15 and 8 time steps in YAGO11k and Wikidata12k. In other words, such method that discretizes facts involving time intervals expands the sizes of both datasets by 15 and 8 times. In this section, we propose a more efficient method to handle time intervals by using two different quadruples, (e_s, r_b, e_o, t_b) and (e_s, r_e, e_o, t_b) to represent the beginning and the end of each fact. In this way, we only expand the sizes of datasets as no more than twice as their original sizes.

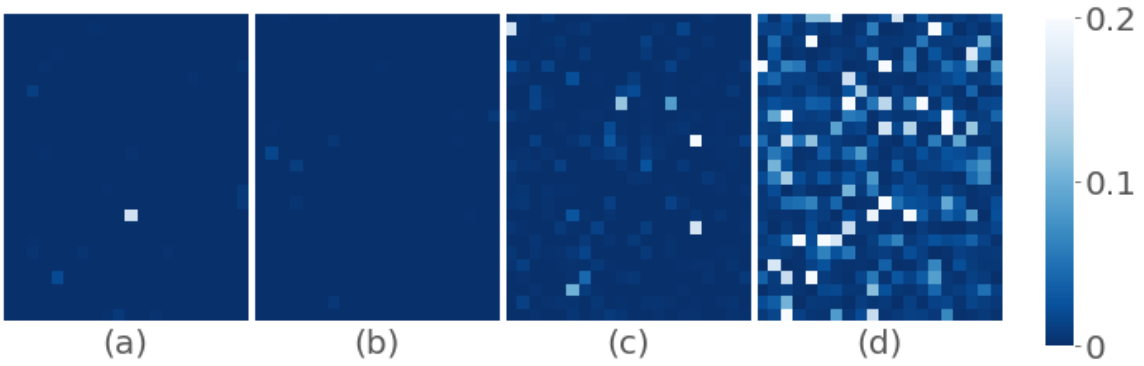


Figure 4.12: Visualization of the absolute difference vectors between $\mathbf{r}_b$ and $\mathbf{r}_e$

For relations r in YAGO11k, we analyze the similarities between the embeddings $\mathbf{r}_b$ and $\mathbf{r}_e$. Figure 4.12 illustrates the visualization of the absolute difference vectors between $\mathbf{r}_b$ and $\mathbf{r}_e$ for relations *deadIn* and *isMarriedTo* (reshaped into 25×20 matrices): (a) $|\mathbf{Re}(\mathbf{r}_b - \mathbf{r}_e)|/|\mathbf{Re}(\mathbf{r}_b)|$ for relation *deadIn*; (b) $|\mathbf{Im}(\mathbf{r}_b - \mathbf{r}_e)|/|\mathbf{Im}(\mathbf{r}_b)|$ for relation *deadIn*; (c) $|\mathbf{Re}(\mathbf{r}_b - \mathbf{r}_e)|/|\mathbf{Re}(\mathbf{r}_b)|$ for relation *isMarriedTo*; (d) $|\mathbf{Im}(\mathbf{r}_b - \mathbf{r}_e)|/|\mathbf{Im}(\mathbf{r}_b)|$ for relation *isMarriedTo*. As shown in Figure 4.12, for short-term relations, e.g., *deadIn*, the real parts of $\mathbf{r}_b$ and $\mathbf{r}_e$, as well as their imaginary parts, have high similarities since r_b and r_e always happen at the same time and have the same semantics. By contrast, for long-term relations, e.g., *isMarriedTo*, the real parts of $\mathbf{r}_b$ and $\mathbf{r}_e$ show their semantic similarities and the imaginary parts capture their temporal dissimilarities.

4.4.4 Conclusion

In this subsection, we introduce TeRo, a new TKGC model which represents entities or relations as single or dual complex embeddings and temporal changes as rotations of entity embeddings in the complex vector space. Our model is advantageous with its capability in modelling several key relation patterns and handling time annotations in various forms. Experimental results show that TeRo remarkably outperforms several state-of-the-art SKGC models and previous TKGC models on knowledge graph completion over four well-established TKG datasets.

Compared to previous TKGC work, TeRo has the following advantages:

- TeRo adapts well to TKG datasets where timestamps have different forms with dual relation embeddings;
- TeRo adopts two different time-split approaches for various TKGC datasets with different time data distribution;
- TeRo has capability in modeling several key relation patterns;
- Choosing the optimal time granularity for TeRo can further improve its performance.

4.5 A TKGC Model Based on Multivector Embeddings and Linear Temporal Regularizer

The content of the this section is based on our work in the paper titled "Geometric Algebra based Embeddings for Static and Temporal Knowledge Graph Completion" (Xu et al., TKDE 2022).

4.5.1 Introduction

Learning KG embeddings in the complex or hypercomplex space $\mathbb{C}$ has drawn a lot of research attentions owing to their abilities to model various relations (e.g., symmetry, anti-symmetry, inversion and composition). In comparison to models based on real-vector space embeddings, ComplEx [14], QuatE [117], RotatE [110] and TeRo [178] have demonstrated their strengths to achieve state-of-the-art results on knowledge graph completion by representing the components of entity/relation embeddings with complex numbers or quaternions, which can be further generalized within Clifford multivectors [188]. From these results, it is notable that models using complex/hypercomplex embeddings achieves better results than models based on real-number embeddings. From this point of view, the geometric algebra by Clifford [189] provides an elegant and efficient rotation representation in terms of multivector which is more general than Hamilton's [190] unit quaternion and thus might give more flexibility in terms of representing complex relation patterns. This shows potential for leveraging the generalization ability of multivectors for solving knowledge graph completion problems.

On another level, although complex-valued distance-based KGC models like RotatE and TeRo have the ability of capturing various relation patterns, complex-valued tensor decomposition models like ComplEx and QuatE have been theoretically proven to be **fully expressive**. The definition of a fully expressive tensor decomposition KGC models can be defined as,

Definition of A Fully Expressive Tensor Decomposition Model

A tensor factorization model is fully expressive if given any ground truth (full assignment of truth values to all facts), there exists an assignment of values to the embeddings of the entities and relations (as well as timestamps for a TKG) that accurately separates the correct facts from incorrect ones [114].

In this section, we present a TKGC approach TGeomE. We move beyond the complex-valued representations and introduce more expressive multivector embeddings to model entities, relations, and timestamps for TKGC, in which each component is a multivector from a n-grade geomeric algebra $\mathbb{G}^n$ ($n = 2, 3$) with scalars, vectors and bivectors, as well as trivectors (for $n = 3$). At a high level, our approach performs 4th-order tensor factorization of a temporal KG, using the asymmetric geometric product. The geometric product provides a greater extent of expressiveness compared to the complex Hermitian operator. We show that TGeomE can naturally model temporal relations and subsumes multiple TKGE models, e.g., TDistMult [186], TComplEx [33] and TIME-PLEX [131], which have been proven to be fully expressive.

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

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

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

4.5.2 Geometric Algebra

In addition to the well-known scalar and vector elements, there are bivectors, trivectors, n-vectors, and multivectors, which are generalizations of the well-known vectors in geometric algebras. The exterior product of two different unit vectors results in a unit bivector representing the oriented area element. Trivectors and n-vectors are 3-dimensional and n-dimensional oriented volumes. The elements introduced above: scalar A_0, vector A_1, bivector A_2, trivector A_3..., n-vector A_n can be combined to form a new kind of entity called a multivector M, i.e., $M = A_0 + A_1 + A_2 + ... + A_n$. Each element of a multivector has an associated grade. The grade indicates the number of vector factors of the exterior product. The scalar elements are 0-grade, the vectors 1-grade, the bivectors 2-grade, the trivectors 3-grade, etc. In this section, we take the 2-grade multivector space $\mathbb{G}^2$ and 3-grade multivector space $\mathbb{G}^3$ as examples to introduce basic operators in geometric algebras.

2-Grade and 3-Grade Multivectors

Considering an orthonormal basis $\{v_1, v_2, v_3, ..., v_n\}$ in an Euclidean vector space $\mathbb{R}^n$, the $\mathbb{G}^n$ multivector space is constructed by all of the 2^n subspaces. The algebra $\mathbb{G}^n$ is based on two rules:

$$(a)\ v_i v_i = 1, ; \quad (b)\ v_i v_j = -v_j v_i \text{ , where } i \neq j.$$

Taking 3-grade multivector space as an example (Fig. 4.13), a $\mathbb{G}^3$ vector space can be represented with 8-dimensional basis elements:

$$\begin{aligned} 1 &\text{ spans 0-grade vectors, scalars,} \\ \{v_1, v_2, v_3\} &\text{ spans 1-grade vectors, vectors,} \\ \{v_1v_2, v_2v_3, v_1v_3\} &\text{ spans 2-grade vectors, bivectors, and} \\ \{v_1v_2v_3\} &\text{ spans 3-grade vectors, trivectors,} \end{aligned}$$

A 3-grade multivector $M \in \mathbb{G}^3$ can then be written as:

$$M = a_0 + (a_1v_1 + a_2v_2 + a_3v_3) + (a_{12}v_1v_2 + a_{23}v_2v_3 + a_{13}v_1v_3) + a_{123}v_1v_2v_3,$$

where $a_0, a_1, a_2, a_3, a_{12}, a_{23}, a_{13}, a_{123}$ are the real coefficients of the basis elements. Likewise, a 2-grade multivector consists of one scalar, two vectors and one bivector from the vector space $\mathbb{R}^2$, i.e.,

$$M = a_0 + a_1v_1 + a_2v_2 + a_{12}v_1v_2.$$

The norm of a multivector $M \in \mathbb{G}^n$ is equal to the root of the square sum of real values of its all

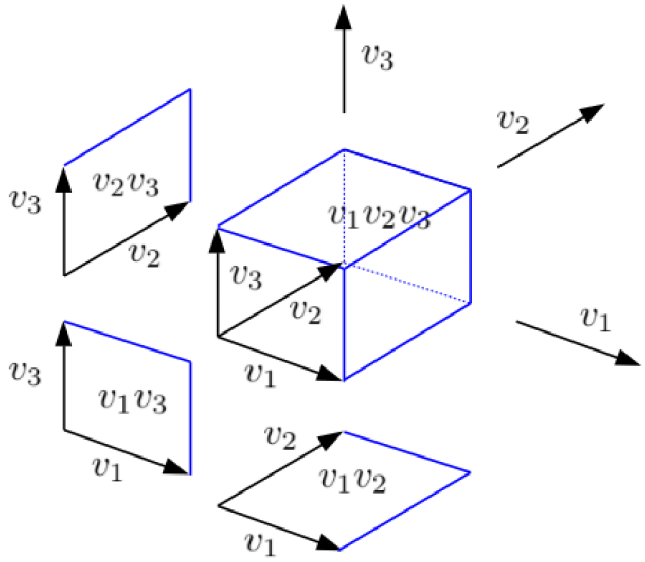


Figure 4.13: A visualization of a 3-grade multivector space $\mathbb{G}^3$.

elements. Taking a 2-grade multivector as an example, its norm is defined as:

$$||M|| = \sqrt{a_0^2 + a_1^2 + a_2^2 + a_{12}^2}. \tag{4.17}$$

Multivectors vs Complex Numbers & Quaternions

A complex number can be represented in the form of $a + b\mathbf{i}$, where $a, b \in \mathbb{R}$ and $\mathbf{i}$ is the imaginary unit, satisfying $\mathbf{i}^2 = -1$. The quaternion system extends the complex numbers with three imaginary units. Quaternions are generally represented in the form: $a + b\mathbf{i} + c\mathbf{j} + d\mathbf{k}$, where $\mathbf{i}^2 = \mathbf{j}^2 = \mathbf{k}^2 = \mathbf{ijk} = -1$, and $a, b, c, d \in \mathbb{R}$.

Noteworthly, any unit bivector from $\mathbb{G}^n$ has similar algebraic properties as the imaginary unit of complex numbers, i.e., $(v_i v_j)^2 = -v_i v_i v_j v_j = -1 = \mathbf{i}^2$. And the basic elements $\mathbf{i}$, $\mathbf{i}$, $\mathbf{k}$ of quaternions can be identified with the unit bivectors in a 3-grade multivector space $\mathbb{G}^3$, i.e.,

$$\begin{aligned} &v_1v_2 = \mathbf{i},\ v_2v_3 = \mathbf{j},\ v_1v_3 = \mathbf{k} \\ &v_1v_2v_2v_3v_1v_3 = v_1v_3v_1v_3 = -1 = \mathbf{ijk} \end{aligned} \tag{4.18}$$

Thus, complex numbers can be embedded into a subalgebra of $\mathbb{G}^2$ which are formed with scalars and bivectors. A quaternion is identified as a multivector comprising a scalar and three bivectors from $\mathbb{G}^3$.

Geometric Product and Clifford Conjugation

Geometric product is the main innovation of geometric algebra. In this following, we will go through the geometric product operations, using 2-grade multivectors as an example. And then the Clifford

conjugation follows.

Geometric Product: The geometric product is an operator on multivectors, denoted with the symbol $\times_n$. It is performed by multiplying every pair of components between the two multivectors. Let's directly see an example of geometric product on two multivectors in the 2-grade multivector space $\mathbb{G}^2$, $M_a = a_0 + a_1v_1 + a_2v_2 + a_{12}v_1v_2$ and $M_b = b_0 + b_1v_1 + b_2v_2 + b_{12}v_1v_2$,

$$\begin{aligned} M_a \times_2 M_b = {} & a_0b_0 + a_1b_1 + a_2b_2 - a_{12}b_{12} + (a_0b_1 + a_1b_0 - a_2b_{12} + a_{12}b_2)v_1 + \\ & (a_0b_2 + a_1b_{12} + a_2b_0 - a_{12}b_1)v_2 + (a_0b_{12} + a_1b_2 - a_2b_1 + a_{12}b_0)v_1v_2. \end{aligned} \tag{4.19}$$

The product of two 3-grade multivectors $M_a = a_0 + a_1v_1 + a_2v_2 + a_3v_3 + a_{12}v_1v_2 + a_{23}v_2v_3 + a_{13}v_1v_3 + a_{123}v_1v_2v_3$ and $M_b = b_0 + b_1v_1 + b_2v_2 + b_3v_3 + b_{12}v_1v_2 + b_{23}v_2v_3 + b_{13}v_1v_3 + b_{123}v_1v_2v_3$ from $\mathbb{G}^3$ is represented as follows.

$$\begin{aligned} M_a \times_3 M_b = {} & a_0b_0 + a_1b_1 + a_2b_2 + a_3b_3 - a_{12}b_{12} - a_{23}b_{23} - a_{13}b_{13} - a_{123}b_{123} \\ & + (a_0b_1 + a_1b_0 - a_2b_{12} + a_{12}b_2 - a_3b_{13} + a_{13}b_3 - a_{23}b_{123} - a_{123}b_{23})v_1 \\ & + (a_0b_2 + a_2b_0 + a_1b_{12} - a_{12}b_1 - a_3b_{23} + a_{23}b_3 + a_{13}b_{123} + a_{123}b_{13})v_2 \\ & + (a_0b_3 + a_3b_0 + a_1b_{13} - a_{13}b_1 + a_2b_{23} - a_{23}b_2 - a_{12}b_{123} - a_{123}b_{12})v_3 \\ & + (a_0b_{12} + a_{12}b_0 + a_1b_2 - a_2b_1 - a_{13}b_{23} + a_{23}b_{13} + a_3b_{123} + a_{123}b_3)v_1v_2 \\ & + (a_0b_{23} + a_{23}b_0 + a_1b_{123} + a_{123}b_1 + a_2b_3 - a_3b_2 - a_{12}b_{13} + a_{13}b_{12})v_2v_3 \\ & + (a_0b_{13} + a_{13}b_0 + a_1b_3 - a_3b_1 - a_2b_{123} - a_{123}b_2 + a_{12}b_{23} - a_{23}b_{12})v_1v_3 \\ & + (a_0b_{123} + a_{123}b_0 + a_1b_{23} + a_{23}b_1 - a_2b_{13} - a_{13}b_2 + a_3b_{12} + a_{12}b_3)v_1v_2v_3. \end{aligned} \tag{4.20}$$

Clifford Conjugation: The Clifford conjugation, denoted by $\overline{M}$, is a subsequence of **space inversion** and **reversion** on a multivector M. The **space inversion** or **grade involution** M^* is to assign a negative sign on the odd-grade basis elements, e.g. v_i to $-v_i$, and the **reversion** $M^\dagger$ is to reverse the product order of all basis elements, e.g. v_1v_2 to v_2v_1. From these, the Clifford conjugation of a multivector is as $\overline{M} = M^{\dagger *}$. For an M in $\mathbb{G}^2$, presetned as $M = A_0 + A_1 + A_2$, where $A_0 = a_0$, $A_1 = a_1v_1 + a_2v_2$, $A_2 = a_{12}v_1v_2$, we obtain its conjugation as $\overline{M} = A_0 - A_1 - A_2$. With the above properties, a geometric product on M in $\mathbb{G}^2$ and its conjugate $\overline{M}$ will be as follows,

$$M \times_2 \overline{M} = a_0^2 - a_1^2 - a_2^2 + a_{12}^2, \tag{4.21}$$

with only the scalar part left.

For more detailed introductions, concepts and functions of geometric algebras, one can refer to the relevant open access literature [191, 192].

4.5.3 Methodology

Let $\mathcal{E}$ denote the set of entities, $\mathcal{R}$ denote the set of relations. A triple is formed as (e_s, r, e_o) where $e_s, e_o \in \mathcal{E}$ are the subject and object entities, $r \in \mathcal{R}$ are the relation. A SKG can be represented as a 3-way tensor $\Omega_s \in \mathcal{E} \times \mathcal{R} \times \mathcal{E}$. For a TKG, we construct a set of time steps $\mathcal{T}$ in accordance with the time granularity of the dataset. For any timestamp appearing in the TKG, we can find a time step $t \in \mathcal{T}$ to represent it. A TKG can be expressed as a 4-way tensor $\Omega_t \in \mathcal{E} \times \mathcal{R} \times \mathcal{E} \times \mathcal{T}$. As shown in Fig. 4.14, each component in the tensor, X_{ijm} or X_{ijml}, denotes the truth value of its corresponding

triple (e_i, r_j, e_m) or quadruple (e_i, r_j, e_m, t_l) where $i, j, m, l \in \mathbb{R}$. Thus, to measure the truth value of any given triple (e_s, r, e_o) or quadruple (e_s, r, e_o, t), we could model the SKG or TKG by defining a tensor factorization-based scoring function $\phi(e_s, r, e_o)$ or $\phi(e_s, r, e_o, t)$.

Our presented approach TGeomE performs 4th-order tensor factorization for a TKG and embodies entities, relations and timestamps in a TKG as multi-dimensional multivectors in geometric algebras. Specially, TGeomE can be derived into several variants depending on the grade N of the target geometric algebras $\mathbb{G}^n$. In this paper, we mainly focus on two TGeomE models, i.e., TGeomE2 and TGeomE3, based on multivectors from $\mathbb{G}^2$ and $\mathbb{G}^3$. We also introduce a linear temporal regularizer for time embeddings to retain the order and distance information between timestamps.

Scoring Function

Following TeRo, we extend the relation set $\mathcal{R}$ of a TKG to a pair of dual relation sets, $\mathcal{R}_b$ and $\mathcal{R}_e$. A relation $r_b \in \mathcal{R}_b$ is used to handle the begin of relation r, meanwhile a relation $r_e \in \mathcal{R}_e$ is used to handle the end of relation r. By doing this, we score a fact $(e_s, r, e_o, [t_b, t_e])$ as the mean value of scores of two quadruples, (e_s, r_b, e_o, t_b) and (e_s, r_e, e_o, t_e) which represent the begin and the end of this fact respectively, i.e., $\phi(e_s, r, e_o, [t_b, t_e]) = \frac{1}{2}(\phi(e_s, r_b, e_o, t_b) + \phi(e_s, r_e, e_o, t_e))$. For a fact missing the begin time or the end time, e.g., $(e_s, r, e_o, [t_b, -])$ or $(e_s, r, e_o, [-, t_e])$, the score of this fact is equal to the score of the quadruple involving the known time, i.e., $\phi(e_s, r, e_o, [t_b, -]) = \phi(e_s, r_b, e_o, t_b)$, $\phi(e_s, r, e_o, [-, t_e]) = \phi(e_s, r_e, e_o, t_e)$. A time point t can be denoted as a special time interval $[t_b, t_e]$ where $t = t_b = t_e$.

TGeomE2 represents each entity/relation/timestamp as a k dimensional embedding $\mathbf{M} \in \mathbb{G}^{2\times k}$ in which each element is a 2-grade multivector, i.e., $\mathbf{M} = [M_1, \ldots, M_k]^\top$, $M_i \in \mathbb{G}^2, i = 1, \ldots, k$. Given a quadruple (e_s, r, e_o, t), embeddings of e_s, r, e_o and t are represented as $\mathbf{M}_s = [M_{s_1}, \ldots, M_{s_k}]^\top$, $\mathbf{M}_r = [M_{r_1}, \ldots, M_{r_k}]^\top$, $\mathbf{M}_o = [M_{o_1}, \ldots, M_{o_k}]^\top$, and $\mathbf{M}_t = [M_{t_1}, \ldots, M_{t_k}]^\top$, respectively. Note that each element of $\mathbf{M}$ is a 2-grade multivector, e.g., $M_{s_1} = \{s_0^1 + s_1^1 v_1 + s_2^1 v_2 + s_{12}^1 v_1 v_2,\ s_0^1, s_1^1, s_2^1, s_{12}^1 \in \mathbb{R}\}$.

Likewise, **TGeomE3** embeds a quadruple (e_s, r, e_o, t) into k-dimensional embeddings $\mathbf{M}_s, \mathbf{M}_r, \mathbf{M}_o$, $\mathbf{M}_t$ using 3-grade multivector, where each component in $\mathbf{M}_s, \mathbf{M}_r, \mathbf{M}_o$, and $\mathbf{M}_t$ is a 3-grade multivector $M_{s_i}, M_{r_i}, M_{o_i} \in \mathbb{G}^3$ for $i = 1, ..., k$. Take subject entity s as an example, an arbitrary component M_{s_i} in subject entities embedding $\mathbf{M}_s$ can be written as $M_{s_i} = s_0^i + s_1^i v_1 + s_2^i v_2 + s_{12}^i v_1 v_2 + s_{23}^i v_2 v_3 +$

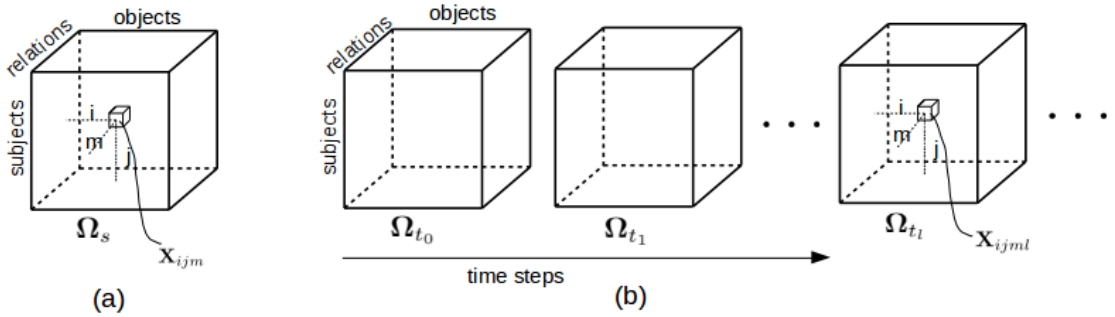


Figure 4.14: Illustrations of tensor decomposition models for SKGC and TKGC.

$s^i_{13}v_1v_3 + s^i_{123}v_1v_2v_3$. With the definition of the model, we then formulate our scoring function as,

$$\phi(e_s, r, e_o, t) = \langle \mathbf{Sc}(\mathbf{M}_s \otimes_n \mathbf{M}_{rt} \otimes_n \overline{\mathbf{M}}_o), \mathbf{1} \rangle, \tag{4.22}$$

where $n = 2$ for TGeomE2 and $n = 3$ for TGeomE3, $\otimes_n$ denotes element-wise **Geometric Product** between two k dimensional n-grade multivectors (e.g. $\mathbf{M}_s \otimes_n \mathbf{M}_r = [M_{s_1} \otimes_n M_{r_1}, \cdots, M_{s_k} \otimes_n M_{r_k}]^\top$), $\mathbf{M}_{rt} = \mathbf{M}_r \otimes_n \mathbf{M}_t$, **Sc** denotes the scalar component of a multivector, $\mathbf{1}$ denotes a $k \times 1$ vector having all k elements equal to one, $\overline{\mathbf{M}}$ denotes the element-wise conjugation of multivectors i.e. $\overline{\mathbf{M}} = [\overline{M}_1, \ldots, \overline{M}_k]^\top$, and $\langle a, b \rangle := \sum_k a_k b_k$ is the element-wise dot product.

Complexity

The time and space complexities of TGeomE models depend on the grade of multivectors n and the embedding dimension k. The space complexity of a TGeomE model is $O(k2^n)$, where $n = 2, 3$ in this work. Since the scoring function involves three geometric product operations between k-dimensional multivectors of n grades, the time complexity is $O(k4^n)$. Thus, when the grade of the multivector embedding n is fixed, both the time complexity and space complexity of our models are $O(k)$, which are the same as TeRo.

Expressiveness

In this section, we show that TGeomE is fully expressive and subsumes various state-of-the-art TKGC models, e.g., TDistMult [186], TComplEx [33] and TIME-PLEX [131].

We mentioned previously that a multivector could generalize number systems like complex numbers or quaternions due to the identical properties in the imaginary units. To represent a complex number using geometric algebra, a 2-grade multivector comes into play with a scalar and a bivector units. On the other hand, a quaternion is isomorphic to a 3-grade multivector with the scalar and three bivectors parts. In the followings, we explain that our models substantially subsume state-of-the-art models:

Definition of Subsumption Relationship between KGC Models

A model KGC1 subsumes KGC2 when any scoring over triples/quadruples of a KG measured by model KGC2 can also be obtained by KGC1 [193].

Subsumption of TDistMult and TComplEx: TDistMult and TComplEx extend DistMult and ComplEx to TKGE by temporalizing relation embeddings, i.e., doing Hermitian product with time

Model	Scoring Function	Space Complexity	Time Complexity
ATiSE	$\mathcal{D}_{\mathcal{KL}}(\mathbf{P}_{e,t}, \mathbf{P}_{r,t})$	$O(\lvert\mathcal{E}\rvert k + \lvert\mathcal{R}\rvert k)$	$O(k)$
TeRo	$\lvert\lvert\mathbf{e}_{s,t} + \mathbf{r} - \bar{\mathbf{e}}_{o,t}\rvert\rvert$	$O(\lvert\mathcal{E}\rvert k + \lvert\mathcal{R}\rvert k)$	$O(k)$
TGeomEn (n=2,3)	$\langle \mathbf{Sc}(\mathbf{M}_s \otimes_n \mathbf{M}_r \otimes_n \mathbf{M}_t \otimes_n \overline{\mathbf{M}}_o), \mathbf{1} \rangle$	$O(\lvert\mathcal{E}\rvert k2^n + \lvert\mathcal{R}\rvert k2^n + \lvert\mathcal{T}\rvert k2^n)$	$O(k4^n)$

Table 4.11: Comparison of TGeomE models with ATiSE and TeRo for space and time complexity.

embeddings $\boldsymbol{t}$. The score function of TDistMult and TComplEx is defined as follows,

$$\phi(e_s, r, e_o, t) = < \mathbf{e}_s, \mathbf{r}, \mathbf{e}_o, \mathbf{t} >, \text{ where } \mathbf{t} \in \mathbb{R}^k, \tag{4.23}$$

$$\phi(e_s, r, e_o, t) = \mathbf{Re}(< \mathbf{e}_s, \mathbf{r}, \bar{\mathbf{e}}_o, \mathbf{t} >), \text{ where } \mathbf{t} \in \mathbb{C}^k. \tag{4.24}$$

A 1-grade multivector consists of a scalar and a vector. And 2-grade multivetors without vector parts can perform as complex numbers. Thus, by setting the coefficients of the vector parts of embedding multivectors to zero for TGeomE1 and TGeomE2, the TGeomE1 and TGeomE2 scoring functions recover Equation 4.23 and 4.24, respectively. Considering that TGeomE2 subsumes TGeomE1, we can conclude that TGeomE2 subsumes both TDistMult and TComplEx. ComplEx has been proven to be fully expressive for SKGC with embeddings of length at most $|\mathcal{E}| \times |\mathcal{R}|$ [194]. Likewise, TComplEx is fully expressive for TKGC with embeddings of length at most $|\mathcal{E}| \times |\mathcal{R}| \times |\mathcal{T}|$. TGeomE2 subsumes TComplEx and thus inherits its full expressiveness.

Corollary 1

The TGeomEn ($n \geq 2$) models are fully expressive.

Subsumption of TIME-PLEX: The scoring function of TIME-PLEX comprises of a ComplEx scoring function and three temporal terms which are also defined in the style of ComplEx scoring function, i.e.,

$$\begin{aligned}\phi(e_s, r, e_o, t) = \mathbf{Re}(< \mathbf{e}_s, \mathbf{r}^{\mathbf{SO}}, \bar{\mathbf{e}}_o >) + \alpha \mathbf{Re}(< \mathbf{e}_s, \mathbf{r}^{\mathbf{ST}}, \bar{t} >) \\ + \beta \mathbf{Re}(< \mathbf{e}_o, \mathbf{r}^{\mathbf{OT}}, \bar{t} >) + \gamma \mathbf{Re}(< \mathbf{e}_s, \mathbf{e}_o, \bar{t} >).\end{aligned} \tag{4.25}$$

where α, β, γ are term weights, $\mathbf{r}^{\mathbf{SO}}$, $\mathbf{r}^{\mathbf{ST}}$ and $\mathbf{r}^{\mathbf{OT}}$ are vectors representing relation r. $\mathbf{r}^{\mathbf{ST}}$ represents a relation which is true for entity e_s at time t (similarly for $\mathbf{r}^{\mathbf{SO}}$ and $\mathbf{r}^{\mathbf{OT}}$). The non-temporal term is the same as the ComplEx scoring function. TComplEx can recover the scoring function of ComplEx by fixing each time embedding $\mathbf{t}$ as $\mathbf{1} + \mathbf{0i}$, i.e., the real part of each component of $\mathbf{t}$ is 0 and the imaginary part is 0. Thus, TComplEx subsumes ComplEx and TGeomE2 inheritedly subsumes ComplEx. It can be easily proven that $\mathbf{Re}(< \mathbf{e}_s, \mathbf{r}, \mathbf{e}_o, \bar{\mathbf{t}} >)$ subsumes the first temporal term of the TIME-PLEX scoring function $\mathbf{Re}(< \mathbf{e}_s, \mathbf{r}, \bar{\mathbf{t}} >)$ when $\mathbf{e}_o \equiv \mathbf{1} + \mathbf{0i}$. Likewise, $\mathbf{Re}(< \mathbf{e}_s, \mathbf{r}, \mathbf{e}_o, \bar{\mathbf{t}} >)$ also subsumes $\mathbf{Re}(< \mathbf{e}_o, \mathbf{r}, \bar{\mathbf{t}} >)$ and $\mathbf{Re}(< \mathbf{e}_s, \mathbf{e}_o, \bar{\mathbf{t}} >)$. By setting the coefficients of all bivector elements $v_1 v_2$ and half of the vector elements v_2 of $\mathbf{M}_s$, $\mathbf{M}_r$, $\mathbf{M}_o$, $\mathbf{M}_t$ for TGeomE2, the TGeomE2 scoring function equals to,

$$\begin{aligned}\phi(e_s, r, e_o, t) = (\mathbf{s}_0 \circ \mathbf{r}_0 \circ \mathbf{t}_0 + \mathbf{s}_0 \circ \mathbf{r}_1 \circ \mathbf{t}_1 - \mathbf{s}_1 \circ \mathbf{r}_1 \circ \mathbf{t}_0 + \mathbf{s}_1 \circ \mathbf{r}_0 \circ \mathbf{t}_1) \circ \mathbf{o}_0 \\ + (\mathbf{s}_0 \circ \mathbf{r}_0 \circ \mathbf{t}_1 - \mathbf{s}_0 \circ \mathbf{r}_1 \circ \mathbf{t}_0 - \mathbf{s}_1 \circ \mathbf{r}_0 \circ \mathbf{t}_0 - \mathbf{s}_1 \circ \mathbf{r}_1 \circ \mathbf{t}_1) \circ \mathbf{o}_1,\end{aligned} \tag{4.26}$$

where $\mathbf{s}_0 = [s_0^1, \ldots, s_0^k]^\top$, $\mathbf{r}_0 = [r_0^1, \ldots, r_0^k]^\top$, $\mathbf{o}_0 = [o_0^1, \ldots, o_0^k]^\top$, $\mathbf{t}_0 = [t_0^1, \ldots, t_0^k]^\top \in \mathbb{R}^k$ are the scalar components of multivector embeddings $\mathbf{M}_s, \mathbf{M}_r, \mathbf{M}_o$ and $\mathbf{s}_1, \mathbf{r}_1, \mathbf{o}_1, \mathbf{t}_1 \in \mathbb{R}^k$ denote the coefficients of the vector components. The above scoring function is also equal to $\mathbf{Re}(< \mathbf{e}_s, \mathbf{r}, \mathbf{e}_o, \bar{\mathbf{t}} >)$ when $\mathbf{s}_0$, $\mathbf{o}_0$, $\mathbf{r}_0$, $\mathbf{t}_0$ represent the real parts of complex embeddings $\mathbf{e}_s$, $\mathbf{e}_o$, $\mathbf{r}$, $\mathbf{t}$, and $\mathbf{s}_1$, $\mathbf{o}_1$, $\mathbf{r}_1$, $\mathbf{t}_1$ represent the imaginary parts of $\mathbf{e}_s$, $\mathbf{e}_o$, $\mathbf{r}$, $\mathbf{t}$. Thus, TGeomE2 subsumes $\mathbf{Re}(< \mathbf{e}_s, \mathbf{r}, \mathbf{e}_o, \bar{\mathbf{t}} >)$ and

consequently subsumes the three temporal terms of TIME-PLEX scoring function. That concludes that the ensemble of four independent TGeomE2 models subsumes TIME-PLEX.

Loss Function

Using full multiclass log-softmax loss function and N3 regularization has been proven to be helpful in boosting the performances of tensor decomposition-based KGC models [33, 116, 131, 195]. In this work, we follow such setting for TGeomE and utilize the reciprocal learning for simplifying the training process.

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

$$\begin{aligned}\mathcal{L}_\xi = &-\log\left(\frac{\exp(\phi(e_s, r, e_o, t))}{\sum_{e_i \in \mathcal{E}} \exp(\phi(e_i, r, e_o, t))}\right) - \log\left(\frac{\exp(\phi(e_o, r^{-1}, e_s, t))}{\sum_{e_i \in \mathcal{E}} \exp(\phi(e_i, r^{-1}, e_s, t))}\right) \\ &+ \lambda_\xi \sum_{i=1}^{k} (||M_{s_i}||_3^3 + ||M_{rt_i}||_3^3 + ||M_{o_i}||_3^3),\end{aligned} \tag{4.27}$$

where λ_ξ denotes the N3 regularization weight.

Temporal Regularization

For the goal to perform knowledge graph completion while considering the temporal aspect, several TKGE models like RTGE [196] and TComplEx [33] incorporate a temporal smoothing concept to inflict constraint on the time embeddings. RTGE and TComplEx propose to minimize the Euclidean distance between hyperplanes and time embeddings of adjacent time steps, respectively. With this, a **smoothing** temporal regularizer is defined as follows,

$$\mathcal{L}_\mathcal{T} = \frac{1}{|\mathcal{T}| - 1} \sum_{i=1}^{|\mathcal{T}|-1} ||\mathbf{M}_{t_{i+1}} - \mathbf{M}_{t_i}||_p^p. \tag{4.28}$$

where $\mathbf{M}_{t_i}$ is the embedding of the ith time step and $p = 3$ for N3 regularization.

Secondly, an orthogonal transformation is leveraged to align the neighboring embeddings to ensure that the cosine similarities of the embeddings across different time points are consistent [197]. In other words, this temporal regularization aims at addressing that the embeddings between neighboring time steps remain a rotational relationship. Such **rotational** temporal regularizer can be defined as,

$$\mathcal{L}_\mathcal{T} = \frac{1}{|\mathcal{T}| - 1} \sum_{i=1}^{|\mathcal{T}|-1} ||\mathbf{M}_{t_{i+1}} - \mathbf{M}_w \otimes_n \mathbf{M}_{t_i}||_p^p, \tag{4.29}$$

where $\mathbf{M}_w$ is the rotation embedding.

The time-varying process can be represented as an autoregressive model. On this basis, Yu et al. [198] assume the change of temporal embeddings over time fits an AR model. Such **autoregressive**

temporal regularizer can be defined as,

$$\mathcal{L}_{\mathcal{T}} = \frac{1}{|\mathcal{T}| - m} \sum_{i=1}^{|\mathcal{T}|-m} ||\mathbf{M}_{t_{i+m}} - \sum_{j=0}^{m-1} \mathbf{M}_j \otimes_n \mathbf{M}_{t_{i+j}}||_p^p, \tag{4.30}$$

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

In this work, we suggest to add a bias component between neighboring temporal embeddings so that difference between pairs has a slightly linear increase. We define it as,

$$\mathcal{L}_{\mathcal{T}} = \frac{1}{|\mathcal{T}| - 1} \sum_{i=1}^{|\mathcal{T}|-1} ||\mathbf{M}_{t_{i+1}} - \mathbf{M}_{t_i} - \mathbf{M}_b||_p^p, \tag{4.31}$$

where $\mathbf{M}_b$ denotes the bias embedding. This **linear** temporal regularizer prompts that embeddings of two distant time steps are relatively more different than those of two adjacent time steps, i.e., $||\mathbf{M}_{t_{i+m}} - \mathbf{M}_{t_i}|| > ||\mathbf{M}_{t_{i+1}} - \mathbf{M}_{t_i}||$ when $m \gg 1$.

By adding the regularization terms to the quadruple loss, the batch loss $\mathcal{L}_b$ is defined as

$$\mathcal{L}_b = \frac{1}{b} \sum_{\xi \in Q_b} \mathcal{L}_\xi + \lambda_{\mathcal{T}} \mathcal{L}_{\mathcal{T}}. \tag{4.32}$$

where b denotes the batch size, Q_b denotes the training batch, $\lambda_{\mathcal{T}}$ denotes the temporal regularization weight and $\mathcal{L}_\xi$ is obtained from Equation 4.27 with $\mathbf{M}_r$ being replaced by $\mathbf{M}_{rt}$. In this work, we compare the effects of the four above-mentioned temporal regularizers on TGeomE models.

4.5.4 Experiments

To show the capability of TGeomE, we compare it with state-of-the-art TKGC models including ATiSE and TeRo on four TKG datasets. Particularly, we conduct an ablation study to analyze the effect of the time granularity, embedding dimension and temporal regularizer on TGeomE's performance.

Baseline

Previous works including ATiSE [32] and TeRo [178] have shown that TKGC models outperform SKGC models on TKG benchmarks with the incorporation of time information. Thus we only compare the performances of TGeomE models against TKGC baselines. We use several state-of-the-art TKGC approaches as baselines. including TTransE [177], HyTE [129], TA-TransE, TA-DistMult [132], DE-SimplE [133], ATiSE [32], TeRo [178], TIME-PLEX(base) [131] and TComplEx [33]. We do not use the complete TIME-PLEX model and the TNTComplEx model as baselines since the former incorporates additional temporal constraints for some specific relations and the latter is designed for modelling a KG where some facts involve time information and others do not. Among the existing TKGC approaches, TComplEx achieves state-of-the-art results on TKGC.

In addition, we test two ensemble models TGeomE+ and TGeomE++ by exploiting two different KGE ensemble methods. Similar to RGCN+ [124], a TGeomE+ model linearly combines a TGeomE2 model and a TGeomE3 model which are trained separately at the model level, i.e.,

$\phi_{TGeomE+}(e_s, r, e_o) = \phi_{TGeomE2}(e_s, r, e_o) + \phi_{TGeomE3}(e_s, r, e_o)$. Different from TGeomE+, we use a relation-level ensemble method [193] for TGeomE++ to combine TGeomE2 and TGeomE3. Concretely, we construct for each relation a dataset that contains all of its training quadruples as well as an equal amount of corrupted quadruples. A logistic regression model is used, in which rescaled scores of individual TGeomE models are features and the class labels of quadruples in constructed datasets are variables. Scores of individual TGeomE models are linearly combined with the rescaling weights at the relation level during the evaluation process.

Experimental Setup

We implement our proposed model TGeomE in PyTorch. The code is available at https://github.com/soledad921/TeLM.

We use the Adagrad [187] optimizer with a learning rate of 0.1 to train TGeomE models. The batch size b is fixed as 1000. The regularization weights λ_ξ and $\lambda_\mathcal{T}$ are tuned in a range of {0, 0.001, 0.0025, 0.005, 0.0075, 0.01,. . . , 0.1}. To avoid too much memory consumption, we follow the setting in [33] to make the maximum embedding dimension no more than 2000. The above experimental setup is also used for evaluating TComplEx on YAGO11k and Wikidata12k. We follow ATiSE and TeRo to use two time-split methods for different TKG datasets and study the effects of the time granularity parameters used in Section 4.4.3, i.e., uniform processing and clubbing processing. Herein, the time granularity parameters u and $thre$ are also regraded as hyperparameters for TGeomE as mentioned in the previous section. The optimal hyperparameters for TGeomE2 are as follows: $\lambda_\xi = 0.0075$, $\lambda_\mathcal{T} = 0.01$, $u = 1$ on ICEWS14; $\lambda_\xi = 0.0025$, $\lambda_\mathcal{T} = 0.1$, $u = 1$ on ICEWS05-15; $\lambda_\xi = 0.025$, $\lambda_\mathcal{T} = 0.001$, $tr = 100$ on YAGO11k; $\lambda_\xi = 0.025$, $\lambda_\mathcal{T} = 0.0025$, $tr = 1$ on Wikidata12k. For TGeomE3, we only list the hyperparameters which are different from TGeomE2: $\lambda_\xi = 0.0025$, $\lambda_\mathcal{T} = 0.025$ on ICEWS14; $\lambda_\xi = 0.001$ on ICEWS05-15; $\lambda_\xi = 0.075$ on YAGO11k; $\lambda_\xi = 0.05$, $\lambda_\mathcal{T} = 0.01$, $tr = 10$ on Wikidata12k. The optimal embedding dimension is $k = 2000$.

	ICEWS14				ICEWS05-15			
	MRR	Hits@1	Hits@3	Hits@10	MRR	Hits@1	Hits@3	Hits@10
TTransE	.255	.047	-	.601	.271	.084	-	.616
HyTE	.297	.108	.416	.655	.316	.116	.445	.681
TA-TransE	.275	.095	-	.625	.299	.096	-	.668
TA-DistMult	.477	.363	-	.686	.474	.346	-	.728
DE-SimplE	.526	.418	.592	.725	.513	.392	.578	.748
ATiSE	.545	.423	.632	.757	.533	.394	.623	.803
TeRo	.562	.468	.621	.732	.586	.469	.668	.795
TIME-PLEX(base)	.589	.499	-	.761	.632	.542	-	.802
TComplEx	.61	.53	.66	.77	.66	.59	.71	.80
TGeomE2	.625	.545	.673	.774	.678	.599	.728	.823
TGeomE3	.621	.538	.671	.773	.673	.593	.723	.820
TGeomE+	.628	.545	.678	**.780**	.684	.603	.735	.831
TGeomE++	**.629**	**.546**	**.680**	**.780**	**.686**	**.605**	**.736**	**.833**

Table 4.12: Knowledge graph completion results on ICEWS14 and ICEWS05-15.

	YAGO11k				Wikidata12k			
	MRR	Hits@1	Hits@3	Hits@10	MRR	Hits@1	Hits@3	Hits@10
TTransE	.108	.020	.150	.251	.172	.096	.184	.329
HyTE	.136	.033	-	.298	.253	.147	-	.483
TA-DistMult	.155	.098	-	.267	.230	.130	-	.461
ATiSE	.185	.126	.189	.301	.252	.148	.288	.462
TeRo	.187	.121	.197	.319	.299	.198	.329	.507
TIME-PLEX(base)	.184	.110	-	.319	.324	.220	-	.528
TComplEx	.185	.127	.183	.307	.331	**.233**	.357	.539
TGeomE2	.191	.129	.194	.321	.332	.231	.360	.542
TGeomE3	.193	.129	.196	.324	.332	.232	.360	.545
TGeomE+	**.195**	**.130**	**.198**	**.327**	**.333**	.232	.361	**.546**
TGeomE++	**.195**	**.130**	.196	.326	**.333**	.232	**.362**	**.546**

Table 4.13: Knowledge graph completion results on YAGO11k and Wikidata12k.

Main Results

Temporal knowledge graph completion results on four TKG datasets are listed in Table 4.12 and 4.13. As shown in Table 4.3, TGeomE2 and TGeomE3 outperform all TKGE baseline models on ICEWS datasets across all metrics. Besides, we evaluate two combined models TGeomE+ and TGeomE++ combined with a trained TGeomE2 model and a separately trained TGeomE3. TGeomE+ and TGeomE++ gets more significant improvements on ICEWS datasets. Compared to TComplEx which is the current state-of-the-art and uses complex embeddings, TGeomE+ and TGeomE++ improves MRR by 1.8% and 1.9% on ICEWS14, 2.4% and 2.6% on ICEWS05-15.

On YAGO11k and Wikidata12k, TGeomE models outperform other TKGE models regarding all metrics except that the Hits@1 of TComplEx is 0.1% higher than TGeomE models on Wikidata12k.

Ablation Study

Fig. 4.15 (a) and (b) show the temporal knowledge graph completion results of TGeomE2 on ICEWS14 and YAGO11k. It can be seen that the performances of TGeomE2 rise with the embedding dimension k increasing in a range of $\{20, 50, 100, 200, 500, 1000, 2000\}$. TGeomE2 with $k = 500$ obtains a higher MRR than TComplEx with $k = 1740$ reported in[33] (0.612 vs 0.61) and has fewer free parameters.

As mentioned in Section 4.4.3, two different time granularity selection techniques are adopted, i.e., uniforming processing for ICEWS datasets and clubbing processing for YAGO11k and Wikidata12k. When time data approximately follow a uniform distribution, fine-grained time representations could provide more abundant information. Thus, the performance of TGeomE2 on ICEWS14 declines with the time unit u becoming larger as shown in Fig. 4.15 (c). Differently, for YAGO11k where time data has a long tail, the tail classes of time data might be overly sparse with a fine time granularity. Fig. 4.15 (d) shows that the optimal minimum threshold $thre$ used for the clubbing process is 100. Compared to the finest time granularity $thre = 1$, using $thre = 100$ could alleviate the long-tail problem of time data of YAGO11k and reduce the memory consumption by narrowing the time set $\mathcal{T}$. By contrast,

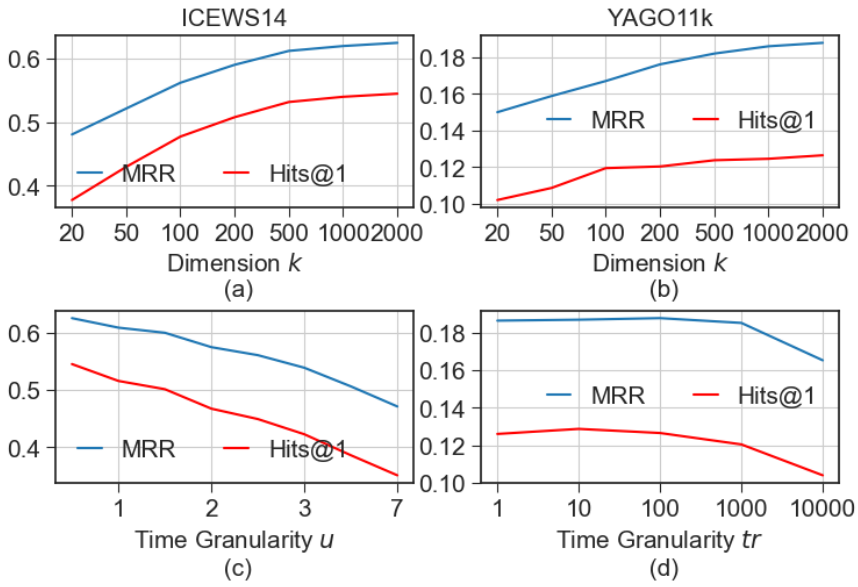


Figure 4.15: Knowledge graph completion results of TGeomE2 with different time granularities and embedding dimensions.

coarse-grained time steps could not fully express time information and lead to low performances.

In this work, we study the effects of various temporal regularizers on the performances of TGeomE2 models. Four temporal regularizers are compared as mentioned in Section 4.5.3, i.e., a novel linear temporal regularizer, a smoothing temporal regularizer, a rotational temporal regularizer, and an autoregressive temporal regularizer. Fig. 4.16 shows the knowledge graph completion results of TGeomE2 on ICEWS14 trained with various temporal regularizers. It can be seen that the linear temporal regularizer is not only superior to other temporal regularizers, but also has a more gradual slope curve within $\lambda_{\mathcal{T}} \in \{0.001, ..., 0.1\}$. We conclude that the linear temporal regularizer is more robust since its learnable bias component could automatically adapt to different $\lambda_{\mathcal{T}}$.

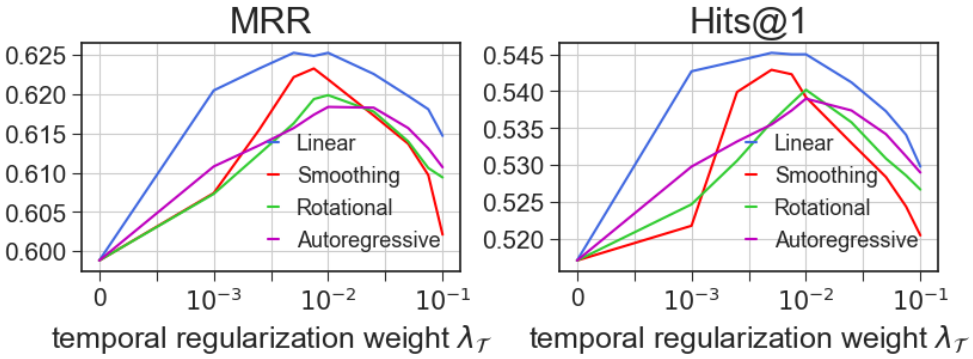


Figure 4.16: Knowledge graph completion results of TGeomE2 trained with various temporal regularizers on ICEWS14.

Fig. 4.17 illustrates the normalized 2-d PCA projections of the 2000-dimensional multivector

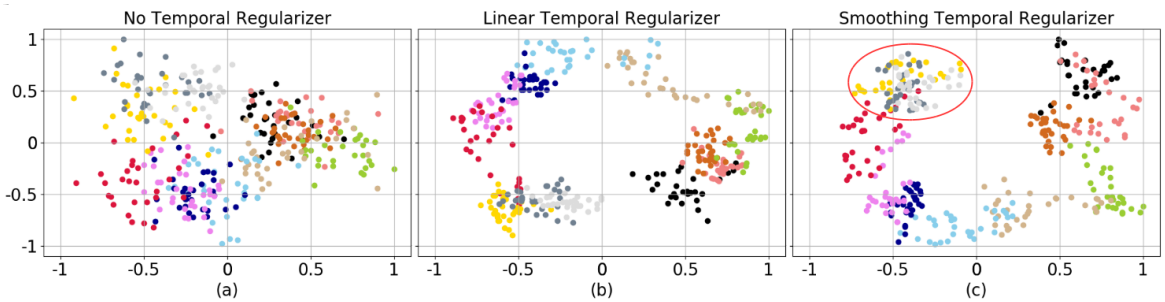


Figure 4.17: Normalized 2-d PCA projection of the 2000 dimensional time embeddings obtained by training TGeomE2 models on ICEWS14 with various temporal regularizers.

embeddings of time points in ICEWS14, where dots having the same color indicate embeddings of time points in the same month. These embeddings are learned from three TGeomE2 models which are trained without any temporal regularization, with a linear temporal regularizer and with a smoothing temporal regularizer, respectively. As shown in Fig. 4.17(a), time embeddings of TGeomE2 trained without a temporal regularizer gather together disorderly. Fig. 4.17(b) and (c) show that time embeddings representing time points in October 2014, November 2014 and December 2014 (located in the red circle), which are learned with the smoothing temporal regularizer, are not well divided while time embeddings trained with the linear temporal regularizer form natural clusters and ordering. In conclusion, the linear temporal regularizer enhances TGeomE models' performances and provides better geometric meanings for time embeddings.

4.5.5 Conclusion

In this section, we introduce a new TKGC model, TGeomE, which performs 4th-order tensor factorization of a TKG using multivector embeddings for knowledge graph representation and a linear temporal regularizer for learning time embeddings. Compared to real-valued and complex-valued embeddings, multivector embeddings provide better generalization capacity and richer expressiveness with a higher degree of freedom. Moreover, the linear temporal regularizer provides better geometric meanings for time embeddings and improves the performances of TGeomE compared to the temporal smoothness. TGeomE models trained with the linear temporal regularizer achieve state-of-the-art results on TKGC over four well-known datasets.

Compared to previous TKGC work, TGeomE has the following advantages:

- TGeomE adapts well to TKG datasets involving various forms of timestamps, e.g., time points, begin or end time, and time intervals;
- TGeomE adopts two different time-split approaches for various TKGC datasets with different time data distributions;
- TGeomE is fully expressive for TKGC and subsumes several existing TKGC models;
- Choosing the optimal time granularity for TGeomE can further improve its performance.
- The linear temporal regularizer effectively helps time embeddings with retaining geometric meanings of timestamps.

4.6 Conclusion

In this chapter, we first introduce the application of temporal knowledge graph embedding models in the task of TKGC and point out the limitations of existing TKGC methods in different aspects, i.e., 1) temporal uncertainty; 2) temporal interpretability; 3) time representation; 4) time distribution; 5) time granularity; 6) model expressiveness; 7) temporal regularization. In Section 4.1, we give the problem definition of the TKGC task and introduce the related evaluation metrics, i.e., MRR and Hits@k. In Section 4.2, we introduce common temporal knowledge graphs including GDELT, ICEWS, Wikidata and YAGO, as well as TKGC benchmark datasets which are extracted from these TKGs. Among all TKG benchmark datasets, we select four well-built datasets for evaluating our proposed TKGC models considering the diversity of characteristics of selected TKG datasets. After that, we present three new TKGC models to address the above-mentioned limitations of the existing TKGC methods. In Section 4.3, we introduce the first TKGC model, ATiSE, which incorporates time information into entity/relation representations by using additive time series decomposition. ATiSE considers the uncertainty during the temporal evolution of entity/relation representations and adapts well to various TKGs. Overall, ATiSE overcomes the limitations 1-4) of previous TKGC models and achieves state-of-the-art TKGC results on four TKG benchmark datasets. In Section 4.4, we introduce the second TKGC model, TeRo, which defines the evolution of an entity embedding as a rotation from the initial time to the current time in the complex vector space. Different from ATiSE, TeRo uses dual relation embeddings to handle the beginning and end of a fact, which is a very efficient way to model facts involving time intervals. We also show that TeRo has a better expressiveness than previous TKGC models and can balance the parameter redundancy and information richness of time embeddings by using suitable time granularities. TeRo overcomes the limitations 3-6) of previous TKGC models and outperforms ATiSE on the task of TKGC. In Section 4.5, we introduce the third TKGC model, TGeomE, which performs 4th-order tensor factorization of a TKG using multivector embeddings from a multi-dimensional geometric algebra and considers a new linear temporal regularization for retaining the ordering and distance information between different timestamps. As a tensor decomposition model, TGeomE has a better model expressiveness than distance-based models like ATiSE and TeRo as well as other state-of-the-art tensor decomposition models, and has been proven to be fully expressive for TKGC. TGeomE overcomes the limitations 3-7) of previous TKGC models and outperforms ATiSE, TeRo and other existing TKGC models on four TKG datasets.

CHAPTER 5

Multi-hop Logical Reasoning over Temporal Knowledge Graphs

Multi-hop logical reasoning over knowledge graphs (KGs) is a fundamental issue in artificial intelligence. In general, multi-hop reasoning answers a complex first-order logic (FOL) queries involving logical and relational operators. Current query embedding (QE) methods learn embeddings for queries and entities by different models such as geometric objects [17, 23, 139], probability distribution[16, 19], and logics[21, 140] to find the answerable entity set for the query. However, existing works only focus on static KGs. These methods can neither handle an entity query involving temporal information and operators nor answer the timestamp set for a temporal query.

As mentioned in Challenge 2, complex logical queries over TKGs often contain temporal logical operations. However, recent TKG-relevant researches focus on the task of TKGC, which is simply single-hop [33, 131, 132, 177, 199]. And temporal query embeddings have not been explored yet.

To fill the gap, we investigate how to perform multi-hop temporal logical reasoning over TKGs, namely, the task of multi-hop TKG reasoning (MTKGR). MTKGR aims at answering temporal complex queries, which have two main distinctions from existing queries on SKGs:

1) The answer set for a query on TKGs is either an entity set or a timestamp set, while that for prior queries on static KGs can only be entity sets;
2) As temporal information is included in the query, temporal operators such as **After**, **Before** should be considered apart from FOL operators.

In this chapter, we investigate the second research question proposed in this thesis.

Research Question 2 (RQ2)

How can we model temporal logical operations in KGE models to perform multi-hop reasoning over temporal knowledge graphs?

In order to study the MTKGR task, we first give the formal definition of the MTKGR task and the evaluation metrics used for TEA models in Section 5.1. In Section 5.2, we present three new multi-hop temporal query datasets which are extracted from three TKGC benchmark datasets, respectively. To address the RQ2, we present a temporal query embedding (TQE) framework based on vector logic, namely Temporal Feature-Logic Embedding framework (TFLEX) in Section 5.3. To our best

knowledge, TFLEX is the first QE framework which supports all FOL operations and extra temporal operations (**After**, **Before** and **Between**). In Section 5.4, we give a summary of this chapter. Overall, this chapter is based on the following paper [200]:

1. Xueyuan Lin, **Chengjin Xu**, Haihong E, Fenglong Su, Gengxian Zhou, Tianyi Hu, Li Ningyuan, Mingzhi Sun and Haoran Luo. "TFLEX: Temporal Feature-Logic Embedding Framework for Complex Reasoning over Temporal Knowledge Graph", The Thirty-Sixth Annual Conference on Neural Information Processing Systems. 2022. (Under Review)

The first author of the above paper is a master student supervised by the Ph.D. candidate during his internship at International Digital Economy Academy. The formulation, design, and implementation of the TFLEX framework and the paper writing were done mostly by the first author and the Ph.D. candidate, who also participated in dataset construction and experimental evaluation. Other co-authors mainly contributed to the dataset construction, experiments, and proofreading of this paper.

5.1 Problem Definition and Evaluation Metrics

Given a TKG $\mathcal{G} = (\mathcal{E}, \mathcal{R}, \mathcal{T}, \mathcal{Q})$, we formally define the temporal logical query over TKGs as follows.

Definition of Temporal Logical Queries

Let a temporal knowledge graph be $\mathcal{G} = (\mathcal{E}, \mathcal{R}, \mathcal{T}, \mathcal{Q})$. An temporal logical query q consists of a non-variable anchor entity set $E_q \subseteq \mathcal{E}$, a non-variable anchor timestamp set $T_q \subseteq \mathcal{T}$, existentially quantified bound variables $E_1, E_2, \cdots, E_k$ and $T_1, T_2, \cdots, T_k$, and a single target variable $E_?$ or $T_?$ (query answer), relations $r \in \mathcal{R}$ in the KG and logical operations existential quantification $\exists$, conjunction $\cap$, disjunction $\cup$, negation $\neg$ and extra operations **After**, **Before** and **Between** on any timestamp set T. The disjunctive normal form (DNF) of an entity query q is defined as follows:

$$q[E_?] = E_?, \exists E_1, \cdots, E_k, T_1, \cdots, T_k : c_1 \vee c_2 \vee \cdots \vee c_n$$

where c_i represents conjunctions of one or more literals a, i.e., $c_i = a_{i1} \wedge a_{i2} \wedge \cdots \wedge a_{im}$ and a literal a represents an entity atomic formula or its negation, i.e., $a_{ij} = r(e_q, E, t_q)$ or $\neg r(e_q, E, t_q)$ or $r(E', E, t_q)$ or $\neg r(E', E, t_q)$ or $r(e_q, E, T)$ or $\neg r(e_q, E, T)$ or $r(E', E, T)$ or $\neg r(E', E, T)$, where $e_q \in E_q, E \in \{E_1, E_2, \cdots, E_k\}, E' \in \{E_1, E_2, \cdots, E_k\}, t_q \in T_q, T \in \{T_1, T_2, \cdots, T_k\}$ and $E \neq E'$. The DNF of a timestamp query q:

$$q[T_?] = T_?, \exists E_1, \cdots, E_k, T_1, \cdots, T_k : d_1 \vee d_2 \vee \cdots \vee d_n$$

where d_i represents conjunctions of one or more literals t, i.e., $d_i = b_{i1} \wedge b_{i2} \wedge \cdots \wedge b_{im}$ and a literal b represents an timestamp atomic formula or its negation, aftertime or beforetime, i.e., $b_{ij} = f \odot r(e_{q1}, e_{q2}, T)$ or $f \odot r(E, E', T)$ or $f \odot r(e_{q1}, E, T)$ or $f \odot r(E, e_{q2}, T)$, where $e_{q1} \in E_q, e_{q2} \in E_q, E \in \{E_1, E_2, \cdots, E_k\}, T \in \{T_?, T_1, T_2, \cdots, T_k\}, f = f_1 \odot f_2 \odot \ldots \odot f_n, f_{1,2,\cdots,n} \in \{1, \neg, \textbf{After}, \textbf{Before}\}$, 1 is identity operation, $\neg$ is logical not, **After** and **Before** are temporal operations.

A temporal query computation graph is a directed acyclic graph (DAG) whose nodes represent entity/timestamp sets $S \subseteq V_a \cup V \cup T_a \cup T$ in the query structure, while directed edges represent logical or relational operations acting on these sets. A query computation graph specifies how the reasoning of the query is proceeded on the TKG. Starting from anchor sets, we obtain the answer set after applying operations iteratively on non-answer sets according to the directed edges in the query computation graph. The operation types on the query computation graph are defined as follows:

Definitions of Logical Operation Types in Temporal Logical Queries

- Relational Projection P. Given an entity set $S_1 \subseteq \mathcal{E}$, a timestamp set $S_2 \subseteq \mathcal{T}$ (or an entity set $S_2 \subseteq \mathcal{E}$ for entity projection) and a relation $r \in \mathcal{R}$, projection operation maps S_1 and S_2 to another set: $S' = \begin{cases} \cup_{(e \in S_1, t \in S_2)} \{e' | (e, r, e', t) \in \mathcal{Q}\}, & P \text{ is entity projection} \\ \cup_{(e \in S_1, e' \in S_2)} \{t | (e, r, e', t) \in \mathcal{Q}\}, & P \text{ is timestamp projection} \end{cases}$
- Intersection I. Given a set of entity sets or timestamp sets $\{S_1, \cdots, S_n\}$, the intersection operation computes logical intersection of these sets $\cap_{i=1}^{n} S_i$.
- Union U. Given a set of entity sets or timestamp sets, the union operation computes logical union of these sets $\cup_{i=1}^{n} S_i$.
- Complement/Negation C. The complement set of a given set S is $\bar{S} = \begin{cases} \mathcal{E} - S, & S \subseteq \mathcal{E} \\ \mathcal{T} - S, & S \subseteq \mathcal{T} \end{cases}$
- Extended temporal operators f. Given a timestamp set S, extended operators compute a certain set of timestamps S': $S' = \begin{cases} \{t' | \text{for some } t' \in \mathcal{T}, t' > \max(S)\}, & f \text{ is After} \\ \{t' | \text{for some } t' \in \mathcal{T}, t' < \min(S)\}, & f \text{ is Before} \end{cases}$

With the above notations and definitions, the problem of MTKGR is formally defined as follows,

Problem Definitions of Multi-hop Temporal Knowledge Graph Reasoning

Given a TKG $\mathcal{G} = (\mathcal{E}, \mathcal{R}, \mathcal{T}, \mathcal{Q})$ and a temporal logical query q, the task of MTKGR aims at traversing the edges of the TKG to answer the query by taking anchor sets of q as start points and iteratively executing FOL and temporal logical operators in the computation graph of q.

A query embedding $\mathbf{V}_q$ of a temporal logical query q is a continuous embedding. Starting from anchor entity sets and timestamp sets, we obtain the final embedding $\mathbf{V}_q$ for query q after applying operations according to the query computation graph. The problem of MTKGR boils down to answering the query q by finding the entity e (or timestamp t) whose embedding $\mathbf{e}$ (or $\mathbf{t}$) has the smallest distance $dist(\mathbf{e}, \mathbf{V}_q)$ (or $dist(\mathbf{t}, \mathbf{V}_q)$) to the embedding of query q.

In this case, the goal of the evaluation metric is to evaluate whether the trained model is able to discover the missing answers of a query on an incomplete TKG. Following the standard evaluation metrics [201], we adopt mean reciprocal rank (MRR) with the filtered setting as our metric. Given the

training, validation, test splits of the quadruples $\mathcal{Q}_{\text{train}}, \mathcal{Q}_{\text{valid}}, \mathcal{Q}_{\text{test}}$ in a TKG, we create three subgraphs $\mathcal{G}_{\text{train}} = (\mathcal{E}, \mathcal{R}, \mathcal{T}, \mathcal{Q}_{\text{train}})$, $\mathcal{G}_{\text{valid}} = (\mathcal{E}, \mathcal{R}, \mathcal{T}, \mathcal{Q}_{\text{train}} \cup \mathcal{Q}_{\text{valid}})$, $\mathcal{G}_{\text{test}} = (\mathcal{E}, \mathcal{R}, \mathcal{T}, \mathcal{Q}_{\text{train}} \cup \mathcal{Q}_{\text{valid}} \cup \mathcal{Q}_{\text{test}})$. We sample queries from $\mathcal{G}_{\text{train}}$ to train our models and store the training answers $\mathcal{A}_{\text{train}}$. During validation, we are given a set of evaluation queries. For each query we traverse the validation TKG, $\mathcal{G}_{\text{valid}}$, to obtain the answers $\mathcal{A}_{\text{valid}}$. The evaluation setup is to evaluate the performance of the model on predicting all missing answers $e \in \mathcal{A}_{\text{valid}} \setminus \mathcal{A}_{\text{train}}$ or $t \in \mathcal{A}_{\text{valid}} \setminus \mathcal{A}_{\text{train}}$. During the test phase, we have the same setup, the only difference is that we evaluate whether the model can discover all missing answers $\mathcal{A}_{\text{test}} \setminus \mathcal{A}_{\text{valid}}$. Given a test query q, we rank each missing answer $e \in \mathcal{A}_{\text{test}} \setminus \mathcal{A}_{\text{valid}}$ or $t \in \mathcal{A}_{\text{test}} \setminus \mathcal{A}_{\text{valid}}$ against the negative answer set $\mathcal{E} \setminus \mathcal{A}_{\text{test}}$ or $\mathcal{T} \setminus \mathcal{A}_{\text{test}}$. Denoting the rank of the answer v (an entity e or a timestamp t) as $\text{Rank}(v|q)$ the MRR of a query q can be calculated as follows,

$$\text{MRR}(q) = \frac{1}{|\mathcal{A}_{\text{test}} \setminus \mathcal{A}_{\text{valid}}|} \sum_{v \in \mathcal{A}_{\text{test}} \setminus \mathcal{A}_{\text{valid}}} \frac{1}{\text{Rank}(v|q)}. \tag{5.1}$$

For a test set of queries, we average MRRs for all test queries as the final performance of the model.

5.2 Multi-hop Temporal Query Datasets

We generate multi-hop temporal query datasets from three common TKGC benchmarks, i.e., ICEWS14, ICEWS05-15, and GDELT-500. To generate temporal logical queries, we firstly define the temporal query structures as functions in python, denoted as Complex Query Function (CQF). We consider 34 kinds of diverse query structures. We use 28 query structures for training and then evaluate models on all the 34 query structures. The definitions of query structures are shown in Table 5.1 with Figure 5.1 and Figure 5.2. Then, we customize a python interpreter to parse the function to Abstract Syntax Tree (AST) and dynamically execute based on the following basic functions: **And**, **Or**, **Not**, **EntityProjection (Pe)**, **TimeProjection (Pt)**, **TimeAnd**, **TimeOr**, **TimeNot**, **Before**, and **After**. In this way, we are able to reuse the definition table of CQFs both in the dataset-sampling process and model-training process. In the dataset-sampling process, the output of CQF is an entity set or a timestamp set, corresponding to the definition of CQF. However, in the model training process, the output of CQF is an embedding vector because we replace the basic functions with neural logical operators. The basic logical set functions in the dataset-sampling process are defined in Table 5.2, where Se is the entity set and St is the timestamp set.

Given the standard split of quadruples into training ($\mathcal{Q}_{\text{train}}$), validation ($\mathcal{Q}_{\text{valid}}$) and test ($\mathcal{Q}_{\text{test}}$) sets, we append inverse relations and double the number of quadruples in the TKG. Then we create three subgraphs: $\mathcal{G}_{\text{train}} = (\mathcal{E}, \mathcal{R}, \mathcal{T}, \mathcal{Q}_{\text{train}})$, $\mathcal{G}_{\text{valid}} = (\mathcal{E}, \mathcal{R}, \mathcal{T}, \mathcal{Q}_{\text{train}} \cup \mathcal{Q}_{\text{valid}})$, $\mathcal{G}_{\text{test}} = (\mathcal{E}, \mathcal{R}, \mathcal{T}, \mathcal{Q}_{\text{train}} \cup \mathcal{Q}_{\text{valid}} \cup \mathcal{Q}_{\text{test}})$. Given a query q, let $\mathcal{A}_{\text{train}}$, $\mathcal{A}_{\text{valid}}$, and $\mathcal{A}_{\text{test}}$ denote a set of answers (entities or timestamps) obtained by running subgraph matching of q on $\mathcal{G}_{\text{train}}$, $\mathcal{G}_{\text{valid}}$ and $\mathcal{G}_{\text{test}}$. Starting from the anchor set, finally, subgraph matching completes the dataset-sampling process.

The statistics of the dataset are shown in Table 5.3. We report the average answers count of each query structure in Table 5.5. Table 5.4 show the count of query structures in the split of training, validation and test. The number of queries for each dataset is shown in Table 5.6, where **Pe** represents query answering $(e_s, r, ?, t)$, **Pt** represents query answering $(e_s, r, e_o, ?)$, **QoE** represents the query of entities (except **Pe**), **QoT** represents query of timestamps (except **Pt**), and **n1p** represents a query that is not **Pe** or **Pt**.

Type	Query Name	Query Structure Definition
entity multi-hop	**Pe2**	**Pe**(**Pe**(e_1, r_1, t_1), r_2, t_2)
	Pe3	**Pe**(**Pe**(**Pe**(e_1, r_1, t_1), r_2, t_2), r_3, t_3)
	Pe_Pt	**Pe**(e_1, r_1, **Pt**(e_2, r_2, e_3))
	e2i	**And**(**Pe**(e_1, r_1, t_1), **Pe**(e_2, r_2, t_2))
	e3i	**And**(**Pe**(e_1, r_1, t_1), **Pe**(e_2, r_2, t_2), **Pe**(e_3, r_3, t_3))
	e2i_Pe	**And**(**Pe**(**Pe**(e_1, r_1, t_1), r_2, t_2), **Pe**(e_2, r_3, t_3))
	Pe_e2i	**Pe**(**e2i**(e_1, r_1, t_1, e_2, r_2, t_2), r_3, t_3)
	Pe_t2i	**Pe**(e_1, r_1, **t2i**(e_2, r_2, e_3, e_4, r_3, e_5))
entity not	**e2i_NPe**	**And**(**Not**(**Pe**(**Pe**(e_1, r_1, t_1), r_2, t_2)), **Pe**(e_2, r_3, t_3))
	e2i_PeN	**And**(**Pe**(**Pe**(e_1, r_1, t_1), r_2, t_2), **Not**(**Pe**(e_2, r_3, t_3)))
	Pe_e2i_Pe_NPe	**Pe**(**And**(**Pe**(e_1, r_1, t_1), **Not**(**Pe**(e_2, r_2, t_2))), r_3, t_3)
	e2i_N	**And**(**Pe**(e_1, r_1, t_1), **Not**(**Pe**(e_2, r_2, t_2)))
	e3i_N	**And**(**Pe**(e_1, r_1, t_1), **Pe**(e_2, r_2, t_2), **Not**(**Pe**(e_3, r_3, t_3)))
entity union	**e2u**	**Or**(**Pe**(e_1, r_1, t_1), **Pe**(e_2, r_2, t_2))
	Pe_e2u	**Pe**(**Or**(**Pe**(e_1, r_1, t_1), **Pe**(e_2, r_2, t_2)), r_3, t_3)
time multi-hop	**Pt_lPe**	**Pt**(**Pe**(e_1, r_1, t_1), r_2, e_2)
	Pt_rPe	**Pt**(e_1, r_1, **Pe**(e_2, r_2, t_1))
	t2i	**TimeAnd**(**Pt**(e_1, r_1, e_2), **Pt**(e_3, r_2, e_4))
	t3i	**TimeAnd**(**Pt**(e_1, r_1, e_2), **Pt**(e_3, r_2, e_4), **Pt**(e_5, r_3, e_6))
	t2i_Pe	**TimeAnd**(**Pt**(**Pe**(e_1, r_1, t_1), r_2, e_2), **Pt**(e_3, r_3, e_4))
	Pt_le2i	**Pt**(**e2i**(e_1, r_1, t_1, e_2, r_2, t_2), r_3, e_3)
	Pt_re2i	**Pt**(e_1, r_1, **e2i**(e_2, r_2, t_1, e_3, r_3, t_2))
time not	**t2i_NPt**	**TimeAnd**(**TimeNot**(**Pt**(**Pe**(e_1, r_1, t_1), r_2, e_2)), **Pt**(e_3, r_3, e_4))
	t2i_PtN	**TimeAnd**(**Pt**(**Pe**(e_1, r_1, t_1), r_2, e_2), **TimeNot**(**Pt**(e_3, r_3, e_4)))
	Pe_t2i_PtPe_NPt	**Pe**(e_1, r_1, **TimeAnd**(**Pt**(**Pe**(e_2, r_2, t_1), r_3, e_3), **TimeNot**(**Pt**(e_4, r_4, e_5))))
	t2i_N	**TimeAnd**(**Pt**(e_1, r_1, e_2), **TimeNot**(**Pt**(e_3, r_2, e_4)))
	t3i_N	**TimeAnd**(**Pt**(e_1, r_1, e_2), **Pt**(e_3, r_2, e_4), **TimeNot**(**Pt**(e_5, r_3, e_6)))
time union	**t2u**	**TimeOr**(**Pt**(e_1, r_1, e_2), **Pt**(e_3, r_2, e_4))
	Pe_t2u	**Pe**(e_1, r_1, **TimeOr**(**Pt**(e_2, r_2, e_3), **Pt**(e_4, r_3, e_5)))
before, after	**Pe_aPt**	**Pe**(e_1, r_1, **After**(**Pt**(e_2, r_2, e_3)))
	Pe_bPt	**Pe**(e_1, r_1, **Before**(**Pt**(e_2, r_2, e_3)))
	Pe_at2i	**Pe**(e_1, r_1, **After**(**t2i**(e_2, r_2, e_3, e_4, r_3, e_5)))
	Pe_bt2i	**Pe**(e_1, r_1, **Before**(**t2i**(e_2, r_2, e_3, e_4, r_3, e_5)))
	between	**TimeAnd**(**After**(**Pt**(e_1, r_1, e_2)), **Before**(**Pt**(e_3, r_2, e_4)))

Table 5.1: Definitions of temporal query structures.

Name	Input	Output
And	Se_1, Se_2	$Se_1 \cap Se_2$
Or	Se_1, Se_2	$Se_1 \cup Se_2$
Not	Se	$\mathcal{E} \setminus Se$
EntityProjection (Pe)	Se, r, St	$\{e_o \mid e_s \in Se, t \in St, (e_s, r, e_o, t) \in \mathcal{Q}\}$
TimeProjection (Pt)	Qe_1, r, Qe_2	$\{t \mid e_s \in Se_1, e_o \in Se_2, (e_s, r, e_o, t) \in \mathcal{Q}\}$
TimeAnd	St_1, St_2	$St_1 \cap St_2$
TimeOr	St_1, St_2	$St_1 \cup St_2$
TimeNot	St	$\mathcal{T} \setminus St$
Before	St	$\{t \mid t < \min(St)\}$
After	St	$\{t \mid t > \max(St)\}$

Table 5.2: Basic logical set functions.

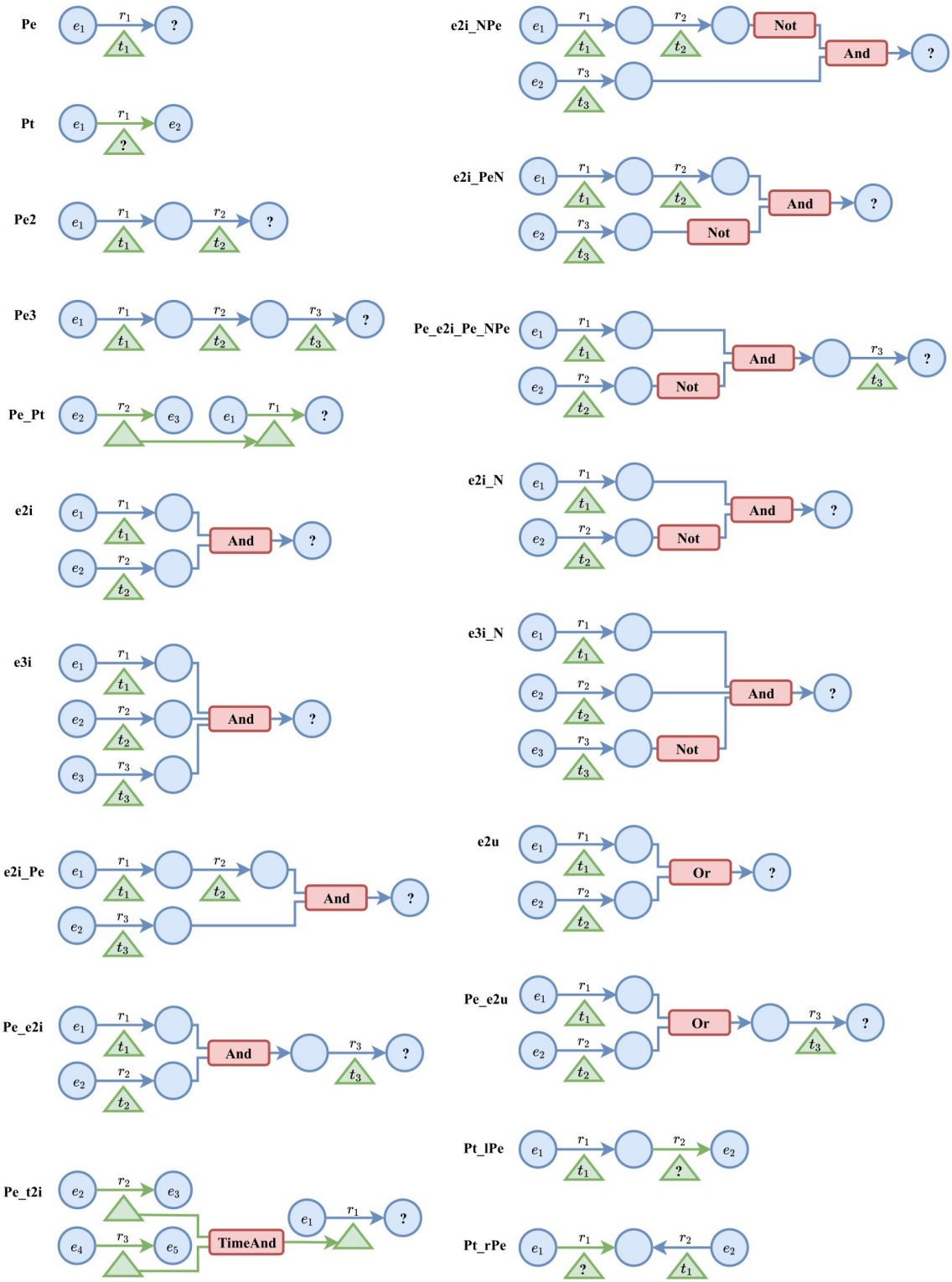


Figure 5.1: Illustrations of Query structures.

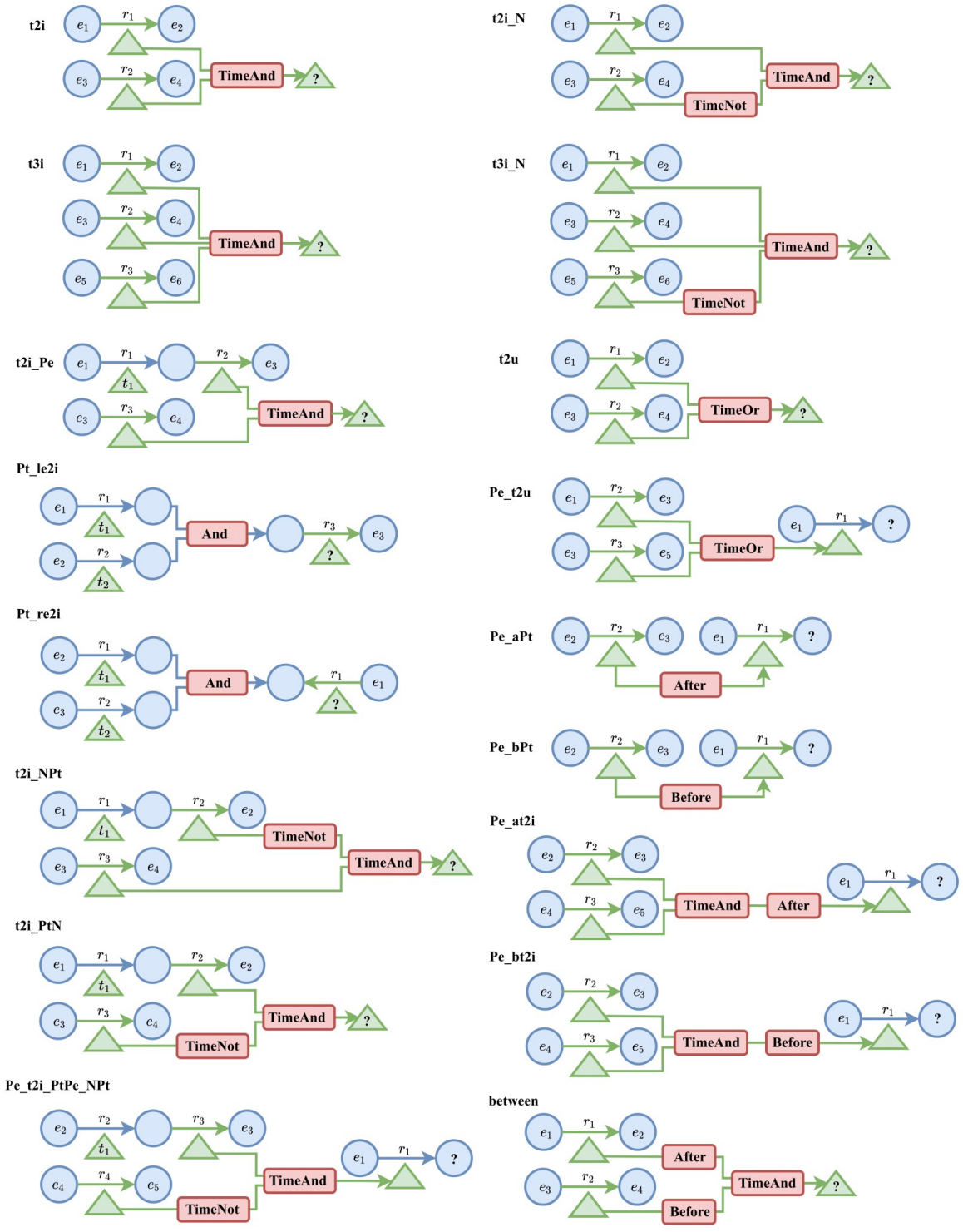


Figure 5.2: Illustrations of Query structures.

Dataset	Entities	Relations	Timestamps	|Training|	|Validation|	|Test|	Total Edges
ICEWS14	7,128	230	365	72,826	8,941	8,963	90,730
ICEWS05-15	10,488	251	4,017	386,962	46,275	46,092	479,329
GDELT-500	500	20	366	2,735,685	341,961	341,961	3,419,607

Table 5.3: Statistics on ICEWS14, ICEWS05-15, and GDELT-500.

	ICEWS14			ICEWS05-15			GDELT-500		
Query Name	**Train**	**Validate**	**Test**	**Train**	**Validate**	**Test**	**Train**	**Validate**	**Test**
Pe2	72826	3482	4037	368962	10000	10000	2215309	10000	10000
Pe3	72826	3492	4083	368962	10000	10000	2215309	10000	10000
Pe_Pt	7282	3385	3638	36896	10000	10000	221530	10000	10000
e2i	72826	3305	3655	368962	10000	10000	2215309	10000	10000
e3i	72826	2966	3023	368962	10000	10000	2215309	10000	10000
e2i_Pe	-	2913	2913	-	10000	10000	-	10000	10000
Pe_e2i	-	2913	2913	-	10000	10000	-	10000	10000
Pe_t2i	-	2913	2913	-	10000	10000	-	10000	10000
e2i_NPe	7282	3061	3192	36896	10000	10000	221530	10000	10000
e2i_PeN	7282	2971	3031	36896	10000	10000	221530	10000	10000
Pe_e2i_Pe_NPe	7282	2968	3012	36896	10000	10000	221530	10000	10000
e2i_N	7282	2949	2975	36896	10000	10000	221530	10000	10000
e3i_N	7282	2913	2914	36896	10000	10000	221530	10000	10000
e2u	-	2913	2913	-	10000	10000	-	10000	10000
Pe_e2u	-	2913	2913	-	10000	10000	-	10000	10000
Pt_lPe	7282	4976	5608	36896	10000	10000	221530	10000	10000
Pt_rPe	7282	3321	3621	36896	10000	10000	221530	10000	10000
t2i	72826	5112	6631	368962	10000	10000	2215309	10000	10000
t3i	72826	3094	3296	368962	10000	10000	2215309	10000	10000
t2i_Pe	-	2913	2913	-	10000	10000	-	10000	10000
Pt_le2i	7282	3226	3466	36896	10000	10000	221530	10000	10000
Pt_re2i	7282	3236	3485	36896	10000	10000	221530	10000	10000
t2i_NPt	7282	4873	5464	36896	10000	10000	221530	10000	10000
t2i_PtN	7282	3300	3609	36896	10000	10000	221530	10000	10000
Pe_t2i_PtPe_NPt	7282	3031	3127	36896	10000	10000	221530	10000	10000
t2i_N	7282	3135	3328	36896	10000	10000	221530	10000	10000
t3i_N	7282	2924	2944	36896	10000	10000	221530	10000	10000
t2u	-	2913	2913	-	10000	10000	-	10000	10000
Pe_t2u	-	2913	2913	-	10000	10000	-	10000	10000
Pe_aPt	7282	4134	4733	68262	10000	10000	221530	10000	10000
Pe_bPt	7282	3970	4565	36896	10000	10000	221530	10000	10000
Pe_at2i	7282	4607	5338	36896	10000	10000	221530	10000	10000
Pe_bt2i	7282	4583	5386	36896	10000	10000	221530	10000	10000
between	7282	2913	2913	36896	10000	10000	221530	10000	10000

Table 5.4: Query count for each dataset.

	ICEWS14			ICEWS05-15			GDELT-500		
Query Name	**Train**	**Validate**	**Test**	**Train**	**Validate**	**Test**	**Train**	**Validate**	**Test**
Pe2	1.03	2.19	2.23	1.02	2.15	2.19	2.61	6.51	6.13
Pe3	1.04	2.25	2.29	1.02	2.18	2.21	5.11	10.86	10.70
Pe_Pt	1.58	7.90	8.62	2.84	18.11	20.63	26.56	42.54	41.33
e2i	1.02	2.76	2.84	1.01	2.36	2.52	1.05	2.30	2.32
e3i	1.00	1.57	1.59	1.00	1.26	1.26	1.00	1.20	1.35
e2i_Pe	-	1.00	1.00	-	1.00	1.00	-	1.07	1.10
Pe_e2i	-	2.18	2.24	-	1.32	1.33	-	5.08	5.49
Pe_t2i	-	1.14	1.16	-	1.07	1.08	-	2.01	2.20
e2i_NPe	1.18	3.03	3.11	1.12	2.87	2.99	4.00	8.15	7.81
e2i_PeN	1.04	2.22	2.26	1.02	2.17	2.21	3.67	8.66	8.36
Pe_e2i_Pe_NPe	1.04	2.21	2.25	1.02	2.16	2.19	3.67	8.54	8.12
e2i_N	1.02	2.10	2.14	1.01	2.05	2.08	2.04	4.66	4.58
e3i_N	1.00	1.00	1.00	1.00	1.00	1.00	1.02	1.19	1.37
e2u	-	3.12	3.17	-	2.38	2.40	-	5.04	5.41
Pe_e2u	-	2.38	2.44	-	1.24	1.25	-	9.39	10.78
Pt_lPe	8.65	28.86	29.22	71.51	162.36	155.46	27.55	45.83	43.73
Pt_rPe	1.41	5.23	5.46	1.68	8.36	8.21	3.84	11.31	10.06
t2i	1.19	6.29	6.38	3.07	29.45	25.61	1.97	8.98	7.76
t3i	1.01	2.88	3.14	1.08	10.03	10.22	1.06	3.79	3.52
t2i_Pe	-	1.03	1.03	-	1.01	1.02	-	1.34	1.44
Pt_le2i	1.31	5.72	6.19	1.37	9.00	9.30	2.76	8.72	7.66
Pt_re2i	1.32	6.51	7.00	1.44	10.49	10.89	2.55	8.17	7.27
t2i_NPt	8.14	25.96	26.23	66.99	154.01	147.34	17.58	35.60	32.22
t2i_PtN	1.41	5.22	5.47	1.70	8.10	8.11	4.56	12.56	11.32
Pe_t2i_PtPe_NPt	1.08	2.59	2.70	1.08	2.47	2.62	4.10	12.02	11.37
t2i_N	1.15	3.31	3.44	1.21	4.06	4.20	2.91	8.78	7.56
t3i_N	1.00	1.02	1.03	1.01	1.02	1.02	1.15	3.19	3.20
t2u	-	4.35	4.53	-	5.57	5.92	-	9.70	10.51
Pe_t2u	-	2.72	2.83	-	1.24	1.28	-	9.90	11.27
Pe_aPt	4.67	16.73	16.50	18.68	43.80	46.23	49.31	66.21	68.88
Pe_bPt	4.53	17.07	16.80	18.70	45.81	48.23	67.67	84.79	83.00
Pe_at2i	7.26	22.63	21.98	30.40	60.03	53.18	88.77	101.60	101.88
Pe_bt2i	7.27	21.92	21.23	30.31	61.59	64.98	88.80	100.64	100.67
between	122.61	120.94	120.27	1407.87	1410.39	1404.76	214.16	210.99	207.85

Table 5.5: Average answers count for each dataset. All numbers are rounded to two decimal places.

	Training				Validation			Test		
Dataset	**Pe**	**Pt**	**QoE**	**QoT**	**Pe**	**Pt**	**n1p**	**Pe**	**Pt**	**n1p**
ICEWS14	273,710	27,371	59,078	8,000	66,990	66,990	10,000	66,990	66,990	10,000
ICEWS05-15	149,689	14,968	20,094	5,000	66,990	22,804	10,000	66,990	66,990	10,000
GDELT-500	107,982	10,798	16,910	4,000	66,990	17,021	10,000	66,990	66,990	10,000

Table 5.6: Numbers of various types of queries.

5.3 A Temporal QE Framework for Multi-hop TKG Reasoning

Existing works on multi-hop logical reasoning only focus on SKGs and FOL operations. However, complex queries over TKGs may expect timestamps as the answers and involve temporal logic. For example, one try to query which countries Xi Jinping visited but Obama id not visit during François Hollande was president of France over a TKG. This can be done by executing a temporal query and its computation graph as shown in Figure 5.3. The query sentence is parsed to temporal query structure. In the computation graph corresponding to the query, there are entity set (blue circle), timestamp set (green triangle), time projection (green arrow), entity projection (blue arrow) and logical operators (red rectangle).

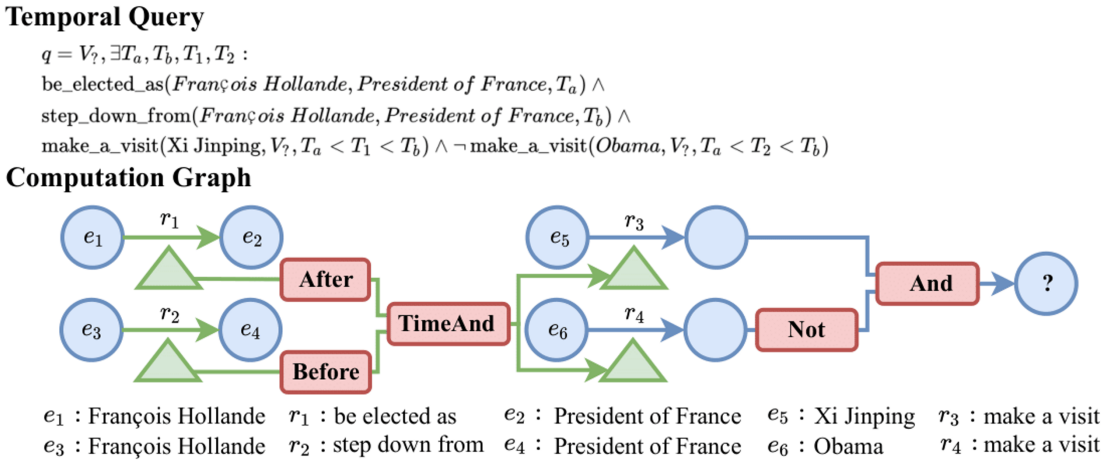


Figure 5.3: A typical multi-hop temporal logical query and its computation graph.

In this section, we present the first temporal logical query embedding framework, Temporal Feature-Logic Embeddings (TFLEX), to answer temporal logical queries over TKGs. In our framework, embeddings of objects (entity, query, timestamp) are divided into two parts, the entity part, and the timestamp part. Each part is further divided into feature components and logic components. On the one hand, the computation of the logic components follows vector logic, which enables our framework to handle all FOL operations. On the other hand, feature components are mingled and transformed under the guidance of logic components, thereby integrating logical information into the feature. Moreover, we extend vector logic to support extra temporal operations (**After**, **Before** and **Between**) to handle temporal operations in the queries.

5.3.1 Vector Logic

Vector logic is an elementary logical model based on matrix algebra. In vector logic, true values are mapped to the vector, and logical operators are executed by matrix computation. The datasets are avaible at `https://anonymous.4open.science/r/TFLEX-NIPS`.

Truth Value Vector Space A two-valued vector logic uses two d-dimensional ($d \geq 2$) column vectors $\vec{s}$ and $\vec{n}$ to represent true and false in the classic binary logic. The two vectors $\vec{s}$ and $\vec{n}$ are real-valued, normally orthogonal to each other, and normalized vectors, i.e., $\|\vec{s}\| = 1, \|\vec{n}\| = 1$. Truth

value vector space is generated by $V_2 = \{\vec{s}, \vec{n}\}$, and operations on vectors in truth value space are based on scalar product.

Operators The basic logical operators are associated with their own matrices by vectors in truth-value vector space. Two common types of operators are monadic and dyadic.

(1) **Monadic Operators** are functions: $V_2 \to V_2$. Two examples are Identity $I = \vec{s}\vec{s}^T + \vec{n}\vec{n}^T$ and Negation $N = \vec{n}\vec{s}^T + \vec{s}\vec{n}^T$ such that $I\vec{s} = \vec{s}, I\vec{n} = \vec{n}, N\vec{n} = \vec{s}, N\vec{s} = \vec{n}$.

(2) **Dyadic operators** are functions: $V_2 \otimes V_2 \to V_2$, where $\otimes$ denotes Kronecker product. Dyadic operators include conjunction C, disjunction D, implication IMPL, equivalence E, exclusive or XOR, etc. For example, the conjunction between two logical propositions $(p \wedge q)$ is performed by $C(\vec{u} \otimes \vec{v})$, where $C = \vec{s}(\vec{s} \otimes \vec{s})^T + \vec{n}(\vec{s} \otimes \vec{n})^T + \vec{n}(\vec{n} \otimes \vec{s})^T + \vec{n}(\vec{n} \otimes \vec{n})^T$. It can be verified that $C(\vec{s} \otimes \vec{s}) = \vec{s}, C(\vec{s} \otimes \vec{n}) = C(\vec{n} \otimes \vec{s}) = C(\vec{n} \otimes \vec{n}) = \vec{n}$. Dyadic operators which correspond to logical operations in classic binary logic are defined by their formulations to perform logical operations on truth value vectors. Their associated matrices have d^2 rows and d columns.

Many-valued Two-dimensional Logic Many-valued logic is introduced to include uncertainties in the logic vectors. Weighting $\vec{s}$ and $\vec{n}$ by probabilities, uncertainties are introduced: $\vec{f} = \epsilon\vec{s} + \delta\vec{n}$, where $\epsilon, \delta \in [0, 1],\ \epsilon + \delta = 1$. Besides, operations on vectors can be simplified to computation on the scalar of these vectors. For example, given two vectors $\vec{u} = \alpha\vec{s} + \beta\vec{n}, \vec{v} = \alpha'\vec{s} + \beta'\vec{n}$, we have:

$$\begin{aligned}
\mathbf{NOT}(\alpha) &= \vec{s}^T N\vec{u} = 1 - \alpha \\
\mathbf{OR}(\alpha, \alpha') &= \vec{s}^T D(\vec{u} \otimes \vec{v}) = \alpha + \alpha' - \alpha\alpha' \\
\mathbf{AND}(\alpha, \alpha') &= \vec{s}^T C(\vec{u} \otimes \vec{v}) = \alpha\alpha' \\
\mathbf{IMPL}(\alpha, \alpha') &= \vec{s}^T L(\vec{u} \otimes \vec{v}) = 1 - \alpha(1 - \alpha') \\
\mathbf{XOR}(\alpha, \alpha') &= \vec{s}^T X(\vec{u} \otimes \vec{v}) = \alpha + \alpha' - 2\alpha\alpha'
\end{aligned} \tag{5.2}$$

5.3.2 Methodology

In this section, we present a temporal feature-logic embedding framework (TFLEX) for the task of MTKGR.

Temporal Feature-Logic Embeddings for Queries and Entities

In this section, we design temporal embeddings for queries, entities, and timestamps. In general, the answers to queries may be entities or timestamps. Therefore, we propose to consider a part of an embedding as an entity part, while the rest is the timestamp part. In our framework, the embedding of a query set S_q is $\mathbf{V}_q = (\mathbf{q}_f^e, \mathbf{q}_l^e, \mathbf{q}_f^t, \mathbf{q}_l^t)$ where $\mathbf{q}_f^e \in \mathbb{R}^k$ is entity feature, $\mathbf{q}_l^e \in [0, 1]^k$ is entity logic, $\mathbf{q}_f^t \in \mathbb{R}^k$ is time feature, $\mathbf{q}_l^t \in [0, 1]^k$ is time logic respectively, k is the embedding dimension. The parameter $\mathbf{q}_l$ is the uncertainty feature of $\mathbf{q}_l\vec{s} + (1 - \mathbf{q}_l)\vec{n}$ in vector logic.

An entity $e \in \mathcal{E}$ is a special query without uncertainty. We propose to represent an entity as the query with logic part $\mathbf{0}$, which indicates that the entity's uncertainty is 0. Formally, the embedding of entity e is $\mathbf{e} = (\mathbf{e}_f, \mathbf{0}, \mathbf{0}, \mathbf{0})$, where $\mathbf{e}_f \in \mathbb{R}^k$ is the entity feature part and $\mathbf{0}$ is a k-dimensional vector with all elements being 0. Similarly, the embedding of timestamp t is $\mathbf{t} = (\mathbf{0}, \mathbf{0}, \mathbf{t}_f, \mathbf{0})$ with entity part and time logic being $\mathbf{0}$.

Logical Operators for Temporal Feature-Logic Embeddings

In this section, we introduce the designed logical operators, including projection, intersection, complement, union, and all other dyadic operators.

Projection Operator P_e and P_t. The goal of operator P_e is to map an entity set to another entity set under a given relation and a given timestamp, while operator P_t outputting a timestamp set given relation and two entity queries. We define a function $P_e : \mathbf{V}_q, \mathbf{r}, \mathbf{V}_t \mapsto \mathbf{V}'_q$ in the embedding space to represent **EntityProjection**, and $P_t : \mathbf{V}_{q_1}, \mathbf{r}, \mathbf{V}_{q_2} \mapsto \mathbf{V}'_q$ for **TimeProjection**, respectively. To implement P_e and P_t, we first represent relations as translations on query embeddings and assign each relation with relational embedding $\mathbf{r} = (\mathbf{r}^e_f, \mathbf{r}^e_l, \mathbf{r}^t_f, \mathbf{r}^t_l)$. Then we define P_e and P_t as:

$$\begin{aligned} P_e(\mathbf{V}_q, \mathbf{r}, \mathbf{V}_t) &= g(\mathbf{MLP}(\mathbf{V}_q + \mathbf{r} + \mathbf{V}_t)) \\ P_t(\mathbf{V}_{q_1}, \mathbf{r}, \mathbf{V}_{q_2}) &= g(\mathbf{MLP}(\mathbf{V}_{q_1} + \mathbf{r} + \mathbf{V}_{q_2})) \end{aligned} \tag{5.3}$$

where $\mathbf{MLP} : \mathbb{R}^{4k} \to \mathbb{R}^{4k}$ is a multi-layer perception network (MLP), + is element-wise addition and g is an activate function to generate $\mathbf{q}^e_l \in [0,1]^k, \mathbf{q}^t_l \in [0,1]^k$. P_e and P_t do not share parameters so the MLPs are different. We define g as:

$$[g(\mathbf{x})]_i = \begin{cases} [\mathbf{q}^e_f]'_i = x_i, & \text{if } 0 < i \le k, \\ [\mathbf{q}^e_l]'_{i-k} = \sigma(x_i), & \text{if } k < i \le 2k, \\ [\mathbf{q}^t_f]'_{i-2k} = x_i, & \text{if } 2k < i \le 3k, \\ [\mathbf{q}^t_l]'_{i-3k} = \sigma(x_i), & \text{if } 3k < i \le 4k, \end{cases} \tag{5.4}$$

where $[g(\mathbf{x})]_i$ is the i-th element of $g(\mathbf{x})$ and $\sigma(\cdot)$ is Sigmoid function.

Dyadic Operators. There are two types of dyadic operators for our framework to model. One for entity set and the other for timestamp set. Each type includes intersection (**AND**), union (**OR**), complement (**NOT**), etc. Based on the vector logic, the operators **AND**, **OR** and **NOT** for logic parts in query embeddings are defined as follows:

$$\begin{aligned} \underset{i=1}{\overset{n}{\mathbf{AND}}}\{\mathbf{q}_{i,l}\} &= \prod_{i=1}^{n} \mathbf{q}_{i,l} \\ \underset{i=1}{\overset{n}{\mathbf{OR}}}\{\mathbf{q}_{i,l}\} &= \sum_{i=1}^{n} \mathbf{q}_{i,l} - \sum_{1 \leqslant i < j \leqslant n} \mathbf{q}_{i,l}\mathbf{q}_{j,l} + \sum_{1 \leqslant i < j < k \leqslant n} \mathbf{q}_{i,l}\mathbf{q}_{j,l}\mathbf{q}_{k,l} + \ldots + (-1)^{n-1} \prod_{i=1}^{n} \mathbf{q}_{i,l} \\ \mathbf{NOT}\{\mathbf{q}_l\} &= \mathbf{1} - \mathbf{q}_l \end{aligned} \tag{5.5}$$

With the help of vector logic, our framework can model all dyadic operators directly. Below we take a unified way to build these operators.

We start from **intersection** operators I_e (on entity set) and I_t (on timestamp set). The goal of intersection operator I_e (I_t) is to represent $S_q = \cap_{i=1}^n S_{q_i}$ based on their entity parts (timestamp parts). Suppose that $\mathbf{V}_{q_i} = (\mathbf{q}^e_{i,f}, \mathbf{q}^e_{i,l}, \mathbf{q}^t_{i,f}, \mathbf{q}^t_{i,l})$ is temporal feature-logic embedding for S_q. We notice that there exists **Alignment Rule** in the process of reasoning. When performing entity set intersection I_e, we should also perform intersection on timestamp parts in order to align the entities into the same time set.

The same also holds for timestamp set intersection I_t and all other dyadic operators. Therefore, we firstly define the intersection operators as follows:

$$\begin{aligned} I_e(\mathbf{V}_{q_1}, ..., \mathbf{V}_{q_n}) &= (\sum_{i=1}^{n} \boldsymbol{\alpha}_i \mathbf{q}^e_{i,f}, \overset{n}{\underset{i=1}{\mathbf{AND}}}\{\mathbf{q}^e_{i,l}\}, \sum_{i=1}^{n} \boldsymbol{\beta}_i \mathbf{q}^t_{i,f}, \overset{n}{\underset{i=1}{\mathbf{AND}}}\{\mathbf{q}^t_{i,l}\}) \\ I_t(\mathbf{V}_{q_1}, ..., \mathbf{V}_{q_n}) &= (\sum_{i=1}^{n} \boldsymbol{\alpha}_i \mathbf{q}^e_{i,f}, \overset{n}{\underset{i=1}{\mathbf{AND}}}\{\mathbf{q}^e_{i,l}\}, \sum_{i=1}^{n} \boldsymbol{\beta}_i \mathbf{q}^t_{i,f}, \overset{n}{\underset{i=1}{\mathbf{AND}}}\{\mathbf{q}^t_{i,l}\}) \end{aligned} \tag{5.6}$$

where **AND** is the AND operator in vector logic, $\boldsymbol{\alpha}_i$ and $\boldsymbol{\beta}_i$ are attention weights. To notice the changes of logic, we compute $\boldsymbol{\alpha}_i$ and $\boldsymbol{\beta}_i$ via the following attention mechanism:

$$\begin{aligned} \boldsymbol{\alpha}_i &= \frac{\exp(\mathbf{MLP}([\mathbf{q}^e_{i,f}; \mathbf{q}^e_{i,l}]))}{\sum_{j=1}^{n} \exp(\mathbf{MLP}([\mathbf{q}^e_{j,f}; \mathbf{q}^e_{j,l}]))}, \mathbf{MLP} : \mathbb{R}^{2k} \to \mathbb{R}^k \\ \boldsymbol{\beta}_i &= \frac{\exp(\mathbf{MLP}([\mathbf{q}^t_{i,f}; \mathbf{q}^t_{i,l}]))}{\sum_{j=1}^{n} \exp(\mathbf{MLP}([\mathbf{q}^t_{j,f}; \mathbf{q}^t_{j,l}]))}, \mathbf{MLP} : \mathbb{R}^{2k} \to \mathbb{R}^k \end{aligned} \tag{5.7}$$

where **MLP** is a MLP network, $[\cdot;\cdot]$ is concatenation. The first self-attention neural network will learn the hidden information from entity logic and leverage to entity feature, while the second one gathers logical information from time logic to time feature.

Note that the computation of entity logic, and time logic obeys the law of vector logic, without any extra learnable parameters. In this way, all dyadic operators are modeled in our framework. Below we take **union** operator for example. We define entity union operator U_e and time union operator U_t according to **Alignment Rule** 5.3.2 as follows:

$$\begin{aligned} U_e(\mathbf{V}_{q_1}, ..., \mathbf{V}_{q_n}) &= (\sum_{i=1}^{n} \boldsymbol{\alpha}_i \mathbf{q}^e_{i,f}, \overset{n}{\underset{i=1}{\mathbf{OR}}}\{\mathbf{q}^e_{i,l}\}, \sum_{i=1}^{n} \boldsymbol{\beta}_i \mathbf{q}^t_{i,f}, \overset{n}{\underset{i=1}{\mathbf{AND}}}\{\mathbf{q}^t_{i,l}\}) \\ U_t(\mathbf{V}_{q_1}, ..., \mathbf{V}_{q_n}) &= (\sum_{i=1}^{n} \boldsymbol{\alpha}_i \mathbf{q}^e_{i,f}, \overset{n}{\underset{i=1}{\mathbf{AND}}}\{\mathbf{q}^e_{i,l}\}, \sum_{i=1}^{n} \boldsymbol{\beta}_i \mathbf{q}^t_{i,f}, \overset{n}{\underset{i=1}{\mathbf{OR}}}\{\mathbf{q}^t_{i,l}\}) \end{aligned} \tag{5.8}$$

where **AND**, **OR** are the AND, OR operators in vector logic respectively, α_i and β_i are attention weights, as designed in intersection operators. In the time part of entity union operator U_e and the entity part of time union operator U_t, we follows **Alignment Rule** 5.3.2 to perform intersection. Besides, please be aware that each operator owns its MLPs and parameters. These operators do not share parameters with each other.

Complement Operators: C_e and C_t The aim of C_e is to identify the complement of query set S_q such that $\neg S_q = \mathcal{E} \setminus S_q$, while C_t aims at calculating the complement $\neg S_q = \mathcal{T} \setminus S_q$ by the time parts. Suppose that $\mathbf{V}_q = (\mathbf{q}^e_f, \mathbf{q}^e_l, \mathbf{q}^t_f, \mathbf{q}^t_l)$, we define the complement operator C_e and C_t as:

$$\begin{aligned} C_e(\mathbf{V}_q) &= (f^e_{\text{not}}(\mathbf{q}^e_f), \mathbf{NOT}(\mathbf{q}^e_l), \mathbf{q}^t_f, \mathbf{q}^t_l) \\ C_t(\mathbf{V}_q) &= (\mathbf{q}^e_f, \mathbf{q}^e_l, f^t_{\text{not}}(\mathbf{q}^t_f), \mathbf{NOT}(\mathbf{q}^t_l)) \end{aligned} \tag{5.9}$$

where $f^e_{\text{not}}(\mathbf{q}_f) = \tanh(\mathbf{MLP}([\mathbf{q}^e_f; \mathbf{q}^e_l]))$, $f^t_{\text{not}}(\mathbf{q}^t_f) = \tanh(\mathbf{MLP}([\mathbf{q}^t_f; \mathbf{q}^t_l]))$ are feature negation functions, two $\mathbf{MLP} : \mathbb{R}^{2k} \to \mathbb{R}^k$ are MLP networks, **NOT** is NOT operation in Equation 5.2.

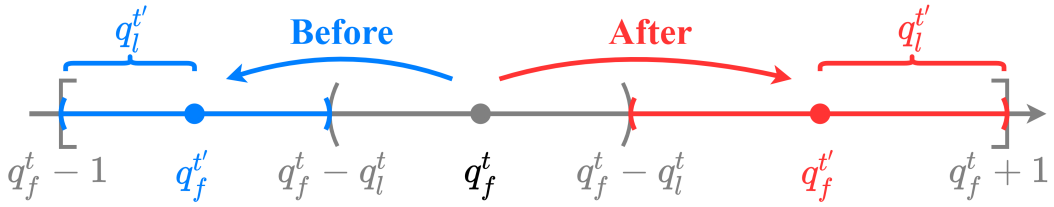


Figure 5.4: The computation of time part in temporal operators Before and After.

Temporal Operators: After A_t, Before B_t and Between D_t The operator After $A_t : \mathbf{V}_q \mapsto \mathbf{V}'_q$ (Before $B_t : \mathbf{V}_q \mapsto \mathbf{V}'_q$) aims at deducing the timestamps after(before) a given fuzzy time set S_q. Let $\mathbf{V}_q = (\mathbf{q}_f^e, \mathbf{q}_l^e, \mathbf{q}_f^t, \mathbf{q}_l^t)$, we define A_t and B_t as:

$$\begin{aligned} A_t(\mathbf{V}_q) &= (\mathbf{q}_f^e, \mathbf{q}_l^e, \mathbf{q}_f^t + \frac{1+\mathbf{q}_l^t}{2}, \frac{1-\mathbf{q}_l^t}{2}) \\ B_t(\mathbf{V}_q) &= (\mathbf{q}_f^e, \mathbf{q}_l^e, \mathbf{q}_f^t - \frac{1+\mathbf{q}_l^t}{2}, \frac{1-\mathbf{q}_l^t}{2}) \end{aligned} \tag{5.10}$$

The entity part does not change after computation because temporal operator only affects the time part (time feature $\mathbf{q}_f^t$ and time logic $\mathbf{q}_l^t$). The motivation of computation can be illustrated in Figure 5.4. Since $\mathbf{q}_l^t$ is the uncertainty of time feature $\mathbf{q}_f^t$, the time part can be viewed as an interval $[\mathbf{q}_f^t - \mathbf{q}_l^t, \mathbf{q}_f^t + \mathbf{q}_l^t]$ whose center is $\mathbf{q}_f^t$ and half-length is $\mathbf{q}_l^t$. The interval is covered by $[\mathbf{q}_f^t - 1, \mathbf{q}_f^t + 1]$ because the probability $\mathbf{q}_l^t < 1$. Then, after interval $[\mathbf{q}_f^t - \mathbf{q}_l^t, \mathbf{q}_f^t + \mathbf{q}_l^t]$ is the interval $[\mathbf{q}_f^t + \mathbf{q}_l^t, \mathbf{q}_f^t + 1]$ whose center is $\mathbf{q}_f^t + \frac{1+\mathbf{q}_l^t}{2}$ and half-length is $\frac{1-\mathbf{q}_l^t}{2}$, which gives the time part of embedding $\mathcal{A}_t(\mathbf{V}_q)$. Similarly, the time part of embedding $\mathcal{B}_t(\mathbf{V}_q)$ is $\mathbf{q}_f^t - \frac{1+\mathbf{q}_l^t}{2}$ (time feature) and $\frac{1-\mathbf{q}_l^t}{2}$ (time logic), which are generated from $[\mathbf{q}_f^t - 1, \mathbf{q}_f^t - \mathbf{q}_l^t]$ before $[\mathbf{q}_f^t - \mathbf{q}_l^t, \mathbf{q}_f^t + \mathbf{q}_l^t]$.

Loss Function Given a training query, we optimize a margin-based log loss

$$L = -\log \sigma(\gamma - d(\mathbf{v}; \mathbf{V}_q)) - \frac{1}{k}\sum_{i=1}^{k} \log \sigma(d(\mathbf{v}'_i; \mathbf{V}_q) - \gamma) \tag{5.11}$$

where $\gamma > 0$ is a fixed margin, k is the number of negative entities, and $\sigma(\cdot)$ is the sigmoid function. When query q is answering entities (timestamps), $v \in S_q$ is a positive entity (or timestamp), $v'_i \notin S_q$ is the i-th negative entity (timestamp).

Theoretical Analysis

To understand why the feature-logic framework works, we have the following propositions and provide the respective proof to show that our designed intersection and union operators obey commutative law and idempotence law of real logical operations.

Proposition 1

Commutativity: Given Temporal Feature-Logic embedding $\mathbf{V}_{q_a}, \mathbf{V}_{q_b}$, we have $I_e(\mathbf{V}_{q_a}, \mathbf{V}_{q_b}) = I_e(\mathbf{V}_{q_b}, \mathbf{V}_{q_a})$ and $U_e(\mathbf{V}_{q_a}, \mathbf{V}_{q_b}) = U_e(\mathbf{V}_{q_b}, \mathbf{V}_{q_a})$, $I_t(\mathbf{V}_{q_a}, \mathbf{V}_{q_b}) = I_t(\mathbf{V}_{q_b}, \mathbf{V}_{q_a})$ and $U_t(\mathbf{V}_{q_a}, \mathbf{V}_{q_b}) = U_t(\mathbf{V}_{q_b}, \mathbf{V}_{q_a})$.

Proof. For the intersection operations, as the calculations of I_e and I_t are identical, here, we only prove that I_e complies with commutative law. The entity feature part and the time feature part of the result are computed as a weighted summation of each query's corresponding parts. Since addition is commutative and the attention weights do not concern the order of calculations, both feature parts' calculations are commutative.
Then, we discuss the logic parts. The logic parts only include the **AND** in vector logic which, essentially, is just the multiplication of each element by the definition provided above. Because multiplication is surely commutative, the calculation of either entity logic part or time logic part is commutative. Thus the intersection operation $I_e(I_t)$ is commutative.
As for U_e and U_t, their feature parts have the same form of weighted summation as the intersection operations do. Thus, the feature parts of both U_e and U_t comply with commutative law. Also, the *time logic part* of U_e and the *entity logic part* of U_t solely concern **AND** operator which has been proved commutative before. The OR operator, by definition, gives the aggregation of different accumulative parts each of which is commutative itself. Also, the multiplications in each of the summations are commutative. Hence, **OR** operation is invariant to the order of calculations, which finally gives the calculations of *entity logic part* of U_e and the *time logic part* of U_t commutativity. Then we can naturally affirm that U_e and U_t are commutative as well.

Proposition 2

Idempotence: Given Temporal Feature-Logic embedding $\mathbf{V}_q$, we have $I_e(\mathbf{V}_q, \mathbf{V}_q) = \mathbf{V}_q$, $I_t(\mathbf{V}_q, \mathbf{V}_q) = \mathbf{V}_q$, $U_e(\mathbf{V}_q, \mathbf{V}_q) = \mathbf{V}_q$ and $U_t(\mathbf{V}_q, \mathbf{V}_q) = \mathbf{V}_q$

Proof. Suppose that $\mathbf{V}_q = (\mathbf{q}^e_f, \mathbf{q}^e_l, \mathbf{q}^t_f, \mathbf{q}^t_l)$. To calculate $I_e(\mathbf{V}_q, \mathbf{V}_q)$, the attention weights α_i and β_i in Equation 5.7 have the same value, because each attention weight corresponds to the same query embedding $\mathbf{V}_q$ and the attention weights are invariant to permutations. Therefore, the feature part remains: $\sum_{i=1}^n \alpha_i \mathbf{q}^e_{i,f} = (\sum_{i=1}^n \alpha_i)\mathbf{q}^e_{i,f} = \mathbf{q}^e_{i,f}$, $\sum_{i=1}^n \beta_i \mathbf{q}^e_{i,f} = \mathbf{q}^e_{i,f}$. Since the calculations of I_e, I_t, U_e and U_t for the feature part of query embeddings have identical forms, the feature parts of $I_e(\mathbf{V}_q, \mathbf{V}_q)$, $I_t(\mathbf{V}_q, \mathbf{V}_q)$, $U_e(\mathbf{V}_q, \mathbf{V}_q)$ and $U_t(\mathbf{V}_q, \mathbf{V}_q)$ are the same as $\mathbf{V}_q$.
For the logic part of $\mathbf{V}_q$, obviously, we have $\mathbf{AND}(\mathbf{q}^e_{i,l}, \mathbf{q}^e_{i,l}) = \mathbf{q}^e_{i,l}$, $\mathbf{AND}(\mathbf{q}^t_{i,l}, \mathbf{q}^t_{i,l}) = \mathbf{q}^t_{i,l}$, $\mathbf{OR}(\mathbf{q}^e_{i,l}, \mathbf{q}^e_{i,l}) = \mathbf{q}^e_{i,l}$, $\mathbf{OR}(\mathbf{q}^t_{i,l}, \mathbf{q}^t_{i,l}) = \mathbf{q}^t_{i,l}$, according to Equation 5.5.
To conclude, the operators I_e, I_t, U_e and U_t all have idempotence.

The motivation of temporal operators is to simulate the temporal semantics of corresponding temporal operations applied on the feature-logic embeddings. To begin with, the temporal feature-logic

embedding t of the initial time set is given. Next, we take **After** operator for example. The **After** operator is to predict the time fuzzy set after the given time set. Since the initial time set is represented by a temporal feature-logic embedding, which can be viewed as interval, the interval representation of after time set should appear after the initial interval without intersection. The computation of the after interval is presented in the paper. This interval interpretation of **After**, **Before** and **Between** operators also obeys the following law: t=**After**(**Before**(t)), t=**Before**(**After**(t)) and t=**Between**(**Before**(t), **After**(t)). These properties show that our temporal operators obey the rules of real temporal logic.

Additionally, we prove that the cost of **OR** operator's implementation is not as expensive as it seems to be. The process of computation can be described as follows:

Algorithm 2: Calculation of **OR** operator

input: Input query sets $\{S_{q_1}...S_{q_n}\}$
output: Result
1. **initialize:** Result = **0**
2. **loop**
3. **for** $S_q \in \{S_{q_1}...S_{q_n}\}$ **do**
4. Result = Result + $\mathbf{V}_q$ − Result · $\mathbf{V}_q$
5. **end for**

As we implement **OR** step by step on a series of n queries, the loop goes $n-1$ times in total. Assuming the embedding dimension of a query is k, we can have the cost of **OR** is $O(nk)$. Moreover, the time complexities of operators **AND** and **Not** are $O(nk)$ and $O(k)$, respectively. The computational complexities of different logical operators for temporal feature-logic embeddings are listed as below: **Projection Operators**: $O(k^2)$; **Intersection Operator**: $O(nk^2)$; **Union Operator**: $O(nk^2)$; **Complement Operator**: $O(k^2)$; **Before, After, Between**: $O(k)$.

5.3.3 Experiments

In this section, we evaluate the ability of TFLEX to reason over temporal knowledge graphs. Specifically, we aim to answer the following research sub-questions (RSQ): **RSQ2.1**: Can TFLEX perform multi-hop logical reasoning over temporal knowledge graphs? **RSQ2.2**: Is it necessary to divide the temporal feature-logic embeddings on objects into entity and timestamp parts? **RSQ2.3**: Do hyper-parameters strongly affect our model? **RSQ2.4**: Is the performance stable? **RSQ2.5**: Does TFLEX perform well on one-hop entity prediction? **RSQ2.6**: Do the temporal operator **Before** and **After** work? We first introduce experimental settings and then present the experimental results. The resource code of experiments is avaible at https://anonymous.4open.science/r/TFLEX-NIPS.

Experimental Setup

We implement our model with PyTorch and use Adam [185] as a gradient optimizer. For each experiment, we use a single GeForce GTX 1080Ti GPU. In order to find the best hyperparameters, we use grid search based on the performance on the validation datasets. We fix the learning rate $lr = 0.0001$, the negative sampling rate $\eta = 128$ and the maximum training step $ts = 300,000$, and tune the embedding dimension k in the range of $\{300, 400, 500, 600, 700, 800\}$ and the margin γ in the

range of $\{5, 10, 15, 20, 25, 30, 35, 40\}$. The optimal non-default configuration for TFLEX is as follows: $k = 800$, $\gamma = 15$ on ICEWS14; $k = 800$, $\gamma = 30$ on ICEWS05-15; $k = 800$, $\gamma = 30$ on GDELT-500.

Dataset	Pe	Pe2	Pe3	e2i	e3i	e2i_Pe	Pe_e2i	e2u	Pe_e2u	AVG
ICEWS14	.485	.395	.343	.722	.958	.377	.375	.414	.309	.486
ICEWS05-15	.464	.387	.355	.477	.935	.376	.367	.581	.554	.500
GDELT-500	.158	.062	.048	.220	.366	.168	.056	.089	.042	.134

Table 5.7: MRR results for queries answering entities. **AVG** denotes average performance.

Dataset	Pt	Pt_lPe	Pt_rPe	Pt_le2i	Pt_re2i	t2i	t3i	t2i_Pe	t2u	Pe_t2u	AVG
ICEWS14	.209	.084	.132	.159	.148	.309	.544	.969	.212	.339	.311
ICEWS05-15	.145	.013	.044	.055	.053	.070	.134	.937	.151	.423	.203
GDELT-500	.027	.028	.026	.026	.026	.027	.027	.029	.026	.091	.033

Table 5.8: MRR results for queries answering timestamps. **AVG** denotes average performance.

Dataset	e2i_NPe	e2i_PeN	Pe_e2i_Pe_NPe	e2i_N	e3i_N	AVG
ICEWS14	.394	.995	.344	.475	.997	.641
ICEWS05-15	.239	.955	.564	.456	.975	.638
GDELT-500	.099	.168	.064	.119	.268	.144

Table 5.9: MRR results for queries with negation answering entities. **AVG** denotes average performance.

Dataset	t2i_NPt	t2i_PtN	Pe_t2i_PtPe_NPt	t2i_N	t3i_N	AVG
ICEWS14	.132	.119	.386	.243	.987	.373
ICEWS05-15	.025	.043	.402	.182	.986	.327
GDELT-500	.028	.023	.091	.025	.028	.039

Table 5.10: MRR results for queries with negation answering timestamps. **AVG** denotes average performance.

Dataset	Pe_Pt	Pe_aPt	Pe_bPt	Pe_t2i	Pe_at2i	Pe_bt2i	between	AVG
ICEWS14	.188	.154	.157	.722	.166	.165	.039	.227
ICEWS05-15	.100	.073	.071	.362	.068	.060	.050	.106
GDELT-500	.061	.057	.056	.154	.055	.054	.018	.065

Table 5.11: MRR results for queries containing After, Before and Between. **AVG** denotes average performance.

Main Results

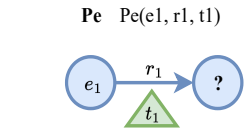
To answer **RSQ1**, we test TFLEX on ICEWS14, ICEWS05-15 and GDELT-500, and report the MRR results on all query structures. Table 5.18 shows the results on queries answering entities. Table 5.19 shows the results on queries answering timestamps. The results show that TFLEX can handle queries with negation answering entities (Table 5.20), queries with negation answering timestamps (Table 5.21) and queries containing After, Before and Between (Table 5.22). Note that we do not compare to other

baselines because existing methods (including QEs and TKGEs) are not able to handle temporal logic on timestamps set, especially the operator After, Before and Between.

Overall, GDELT has lower scores than ICEWS14. Because GDELT is too dense and each query has too many answers, which make the reasoning harder. In addition, we observe that MRR on each dataset decreases from the task **Pe** to **Pe2** to **Pe3**. The performance on **e2i** is worse than **e3i** since the answers of **e3i** are less than **e2i**. Therefore, we conclude that the performance of the framework decreases with the difficulty of the task increasing.

We also provide visualizations of the experimental results of some query instances of different query structures in Tables 5.12-5.17. Taking the query in Table 5.13 as an example, the **Pe2** query $q[V_?] = V_?, \exists V_a, r_1(e_1, V_a, t_1) \wedge r_2(V_a, V_?, t_2)$ where e_1 is "Head of Government (Latvia)", r_1 is "Make an appeal or request", t_1 is "2014-08-01", r_2 is "consult^{-1}", and t_2 is "2014-04-04", is semantically equal to a natural language question "On 2014-04-04, who consulted the man who was appealed to or requested by the Head of Government (Latvia) on 2014-08-01". We use TFLEX to execute the query and get answers. We classify the answer into easy, hard, and wrong. The easy answer is the correct answer that appears in the training set, and the hard answer is the correct answer that exists in the testing set instead of training set. Table 5.13 lists the top 5 answers of this query on ICEWS14 given by TFLEX.

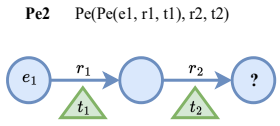
The MRR results on three datasets and the visualizations below demonstrate that TFLEX can perform reasoning over temporal knowledge graphs, which answers **RSQ2.1**.



e_1 : Philippines
r_1 : Criticize or denounce
t_1 : 2014-04-01

Query Sentence	Philippines denounced or criticized who on 2014-04-01?		
Query DNF	$q[V_?] = V_?, r_1(e_1, V_?, t_1)$		
Rank	**Query Answers**	**Correctness**	**Answer Type**
1	China	✔	Hard
2	Japan	✘	-
3	Malaysia	✘	-
4	Philippines	✘	-
5	Iran	✘	-

Table 5.12: Top 5 answers of a **Pe** query on ICEWS14

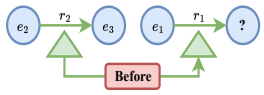


e_1 : Head of Government (Latvia)
r_1 : Make an appeal or request
t_1 : 2014-08-01
r_2 : $consult^{-1}$
t_2 : 2014-04-04

Query Sentence	On 2014-04-04, who consulted the man who was appealed to or requested by the Head of Government (Latvia) on 2014-08-01?		
Temporal Query	$q[V_?] = V_?, \exists V_a, r_1(e_1, V_a, t_1) \wedge r_2(V_a, V_?, t_2)$		
Rank	**Query Answers**	**Correctness**	**Answer Type**
1	François Hollande	✔	Easy
2	Taavi Rõivas	✔	Easy
3	Jyrki Katainen	✔	Hard
4	Angela Merkel	✘	-
5	Head of Government (Latvia)	✘	-

Table 5.13: Top 5 answers of a **Pe2** query on ICEWS14

Pe_bPt Pe(e1, r1, before(Pt(e2, r2, e3)))

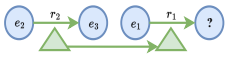


e_1 : South Korea
r_1 : Criticize or denounce
e_2 : Sadako Ogata
r_2 : *Express intent to meet or negotiate*$^{-1}$
e_3 : Head of Government(Bangladesh)

Query Sentence	Who expressed intent to meet or negotiate with the Head of Government (Bangladesh) after South Korea Criticized or denounced Sadako Ogata?		
Temporal Query	$q[V_?] = V_?, \exists T_a, T_b, r_1(e_1, V_?, Ta) \wedge T_b before T_a \wedge r_2(e_2, e_3, T_b)$		
Rank	**Query Answers**	**Correctness**	**Answer Type**
1	Japan	✔	Easy
2	North Korea	✔	Easy
3	China	✔	Easy
4	South Korea	✘	-
5	Kim Jong-Un	✘	-

Table 5.14: Top 5 answers of a **Pe_bPt** query on ICEWS14

Pe_Pt Pe(e1, r1, before(Pt(e2, r2, e3)))

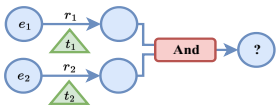


e_1 : Citizen (Thailand)
r_1 : *Arrest, detain, or charge with legal action*$^{-1}$
e_2 : Police (Greece)
r_2 : Arrest, detain, or charge with legal action
e_3 : Illegal Immigrant (Greece)

Query Sentence	Who arrested, detained, or charged the citizen (Thailand) with legal action after Police (Greece) arrested, detained, or charged the illegal immigrant (Greece) with legal action?		
Temporal Query	$q[V_?] = V_?, \exists T_a, T_b, r_1(e_1, V_?, Ta) \wedge T_b before T_a \wedge r_2(e_2, e_3, T_b)$		
Rank	**Query Answers**	**Correctness**	**Answer Type**
1	Military (Thailand)	✔	Easy
2	Municipal Court (Thailand)	✔	Easy
3	Thailand	✔	Hard
4	National Council for Peace and Order of Thailand	✘	-
5	Police (Cambodia)	✘	-

Table 5.15: Top 5 answers of a **Pe_Pt** query on ICEWS14

e2i And(Pe(e1, r1, t1), Pe(e2, r2, t2))



e_1 : Mohammad Javad Zarif
r_1 : Consult
t_1 : 2014-04-07
e_2 : Mohammad Javad Zarif
r_2 : $Consult^{-1}$
t_2 : 2014-04-07

Query Sentence	Who was consulted by Mohammad Javad Zarif on 2014-04-07 and consulted Mohammad Javad Zarif on 2014-04-07?		
Temporal Query	$q[V_?] = V_?, \exists V_a, V_b, r_1(e_1, V_a, t_1) \wedge r_2(e_2, V_b, t_2))$		
Rank	**Query Answers**	**Correctness**	**Answer Type**
1	Mohammad Javad Zarif	✔	Easy
2	Catherine Ashton	✔	Hard
3	Sebastian Kurz	✔	Easy
4	China	✔	Easy
5	Iran	✘	-

Table 5.16: Top 5 answers of a **e2i** query on ICEWS14

Pt_rPe And(Pe(e1, r1, t1), Pe(e2, r2, t2))

e_1 : France
r_1 : Meet at a 'third' location
t_1 : 2014-07-31
e_2 : Peter Humphrey
r_2 : $Makestatement^{-1}$

Query Sentence	At what time did France meet the person who made a statement to Peter Humphrey at a 'third' location?		
Temporal Query	$q[T_?] = T_?, \exists V_a, r_1(e_1, V_a, T_?) \wedge r_2(V_a, e_2, t_1))$		
Rank	**Query Answers**	**Correctness**	**Answer Type**
1	2014-11-21	✔	Hard
2	2014-11-25	✔	Easy
3	2014-11-20	✔	Easy
4	2014-09-23	✔	Easy
d 5	2014-09-21	✔	Easy

Table 5.17: Top 5 answers of a **Pt_rPe** query on ICEWS14

Quality Study

In this section, we conduct extra experiments to analyze TFLEX on the ICEWS14 dataset to answer **RSQ2.2** - **2.6**.

Necessity of dividing embeddings into entity parts and timestamp parts Since some queries answer entity sets and the others answer timestamp sets, it is naturally to divide the embedding into two parts, one for entity logic over entity sets, the other for temporal logic over timestamp sets. To validate this intuition, we conduct of an ablation study by developing a variant TFLEX-1F which combines the entity part and the timestamp part of an embedding into one feature part. We compare the performances of TFLEX and TFLEX-1F on ICEWS14 as shown in Table. It can be observed that the results of variant TFLEX-1F are worse than TFLEX on all tasks. Intuitively, one feature part mixes the semantics of entity and timestamp, thus different logical operators conflicting, causing the performance decrease.

Model	Pe	Pe2	Pe3	e2i	e3i	e2i_Pe	Pe_e2i	e2u	Pe_e2u	AVG
TFLEX	.485	.395	.343	.722	.958	.377	.375	.414	.309	.486
TFLEX-1F	.458	.326	.281	.578	.848	.284	.326	.345	.252	.411

Table 5.18: MRR comparison between TFLEX and its variant TFLEX-1F for queries answering entities.

Dataset	Pt	Pt_lPe	Pt_rPe	Pt_le2i	Pt_re2i	t2i	t3i	t2i_Pe	t2u	Pe_t2u	AVG
TFLEX	.209	.084	.132	.159	.148	.309	.544	.969	.212	.339	.311
TFLEX-1F	.229	.053	.101	.137	.133	.262	.491	.960	.205	.312	.288

Table 5.19: MRR comparison between TFLEX and its variant TFLEX-1F for queries answering timestamps.

Dataset	e2i_NPe	e2i_PeN	Pe_e2i_Pe_NPe	e2i_N	e3i_N	AVG
TFLEX	.394	.995	.344	.475	.997	.641
TFLEX-1F	.313	.956	.302	.380	.975	.585

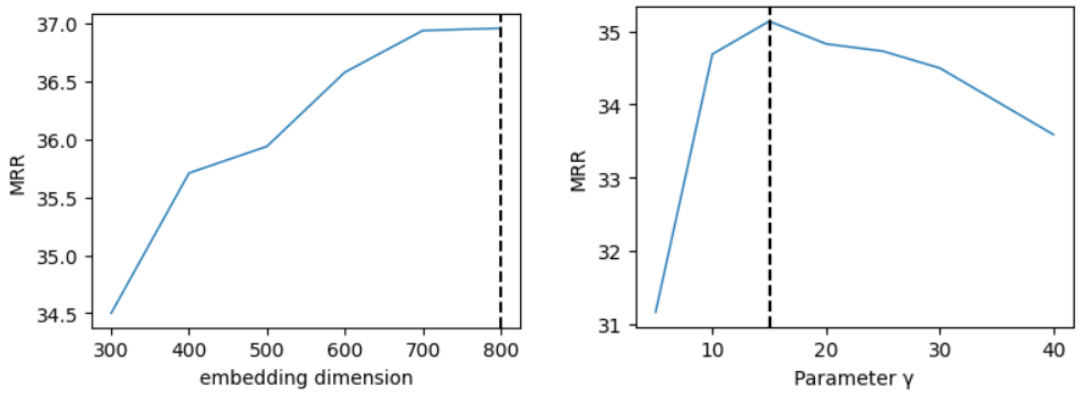
Table 5.20: MRR comparison between TFLEX and its variant TFLEX-1F for queries with negation answering entities.

Dataset	t2i_NPt	t2i_PtN	Pe_t2i_PtPe_NPt	t2i_N	t3i_N	AVG
TFLEX	.132	.119	.386	.243	.987	.373
TFLEX-1F	.025	.043	.402	.182	.986	.327

Table 5.21: MRR comparison between TFLEX and its variant TFLEX-1F for queries with negation answering timestamps.

Dataset	Pe_Pt	Pe_aPt	Pe_bPt	Pe_t2i	Pe_at2i	Pe_bt2i	between	AVG
TFLEX	.188	.154	.157	.722	.166	.165	.039	.227
TFLEX-1F	.100	.073	.071	.362	.068	.060	.050	.106

Table 5.22: MRR comparison between TFLEX and its variant TFLEX-1F for queries containing After, Before and Between.



(a) (b)

Figure 5.5: Impact of (a) embedding dimension k and (b) margin γ.

Impacts of Embedding Dimension Our experiments indicate that the selection of the embedding dimension has a substantial influence on the effectiveness of TFLEX. We train TFLEX with different embedding dimensions $k \in \{300, 400, 500, 600, 700, 800\}$ and plot results based on the validation set, as shown in Figure 5.5(a). With the increase of k, the model performance (indicated by MRR) increases rapidly and reaches its top at $k = 800$. Due to hardware limitations, We cannot try $k > 800$. But we can conclude that $k = 800$ is close to the optimal since performance increases slowly during $k = 700$ and $k = 800$. Therefore, we assign 800 as the best setting.

Impacts of Margin γ We train TFLEX with different margins $\gamma \in \{5, 10, 15, 20, 25, 30, 35, 40\}$ and plot MRR results in Figure 5.5(b). Too small and too large γ both get bad results, while $\gamma = 15$ in the middle is the best. Therefore, we choose $\gamma = 15$.

Stability of Performance In order to answer **RSQ2.4**, we evaluate the stability of the performance of TFLEX by running five times with random seeds $\{1, 10, 100, 1000, 10000\}$ and report the error bars of these results. Table 5.23 shows the error bar of TFLEX's MRR results on queries answering entities on ICEWS14. Table 5.24 shows the error bar of TFLEX's MRR results on queries answering timestamps on ICEWS14. Overall, the standard variances are small, which demonstrates that the performance of TFLEX is stable.

Pe	Pe2	Pe3	e2i	e3i	e2i_Pe	Pe_e2i	e2u	Pe_e2u	AVG
.485	.395	.343	.722	.958	.377	.375	.414	.309	.486
±0.00034	±0.00099	±0.00045	±0.00017	±0.00070	±0.00030	±0.00027	±0.00076	±0.00076	±0.00042

Table 5.23: The mean values and standard variances of TFLEX's MRR results for queries answering entities on ICEWS14.

Necessity of training on complex queries We compare our model with distance-based P_e operators (TTransE [177], HyTE [129], TA-TransE [132], DE-TransE [133], ATiSE [32], TeRo [178]) using only one-hop **Pe** training set in Table 5.25 where * denotes that results are taken from [133]. Hereby we use distance-based TKGC models as baselines because we notice that all existing QE

Pt	Pt_lPe	Pt_rPe	t2i	t3i	t2i_Pe	t2u	AVG
.539	.155	.111	.402	.524	.275	.194	.314
±0.00023	±0.00097	±0.00042	±0.00063	±0.00077	±0.00033	±0.00046	±0.00057

Table 5.24: The mean values and standard variances of TFLEX's MRR results for queries answering timestamps on ICEWS14.

	ICEWS14	ICEWS05-15	GDELT-500
TTransE*	.255	.271	.115
HyTE*	.297	.316	.118
TA-TransE	.275	.299	-
DE-TransE*	.326	.314	.126
ATiSE	.550	.519	.167
TeRo	.562	.586	.181
TFLEX	.485	.464	.158
TFLEX-Pe	.525	.536	.184

Table 5.25: MRR of **Pe** on ICEWS14, ICEWS05-15, and GDELT-500.

models are extended from distance-based KGC models due to their better generalizability on multi-hop reasoning. We can see that TFLEX outperforms most distance-based TKGC models, except ATiSE and TeRo. By training TFLEX with only **Pe** queries, we obtain a new variant TFLEX-Pe, which is comparable and even slightly better than ATiSE and TeRo on some TKGC datasets. However, TFLEX-**Pe** and other TKGC models can not answer other types of queries over TKGs since they do not model any logical operations. Note that in the case of single-hop entity query, we do not expect a performance gain by using vector logic, as single-hop entity query does not involve logical reasoning nor nor handle a large set of answer entities. Considering that the optimization objectives of TFLEX involve different types of complex temporal queries, it is acceptable that TFLEX still outperforms most existing distance-based TKGC models on one-hop entity prediction, which answers **RSQ2.5**.

Effectiveness of temporal operators To explore the temporal effects of A_t and B_t, we show how operators A_t and B_t change the semantic of predicted timestamp embedding. Let triple (e_1, r, e_2) represent a specific event. Then, the temporal operator **Pt** predicts the date when this event (e_1, r, e_2) occurs. And query **Pt_Before** (=**Before(Pt)**) predicts the date before this event happens, while **Pt_After** (=**After(Pt)**) predicts the time after this event. Let **t** be the output of **Pt**(e_1, r, e_2), where **t** is a time embedding, representing the timestamps when the event is the most likely to happens. Because **t** is fuzzy, we score it to all possible timestamps, and visualize it as similarity score distribution over all days in a year, from 1 to 365 in dataset ICEWS14. The similarity score is produced by normalized results of the score function, greater than 0. The higher the score, the closer the predicted date is to the day.

We plot three score distributions along time for a random triple with temporal operators A_t and B_t and projection operator P_t in Figure 5.6, respectively. In Figure 5.6, the horizontal axis Time indicates all possible days (365 days in a year), and the vertical Score indicates the similarity score on each

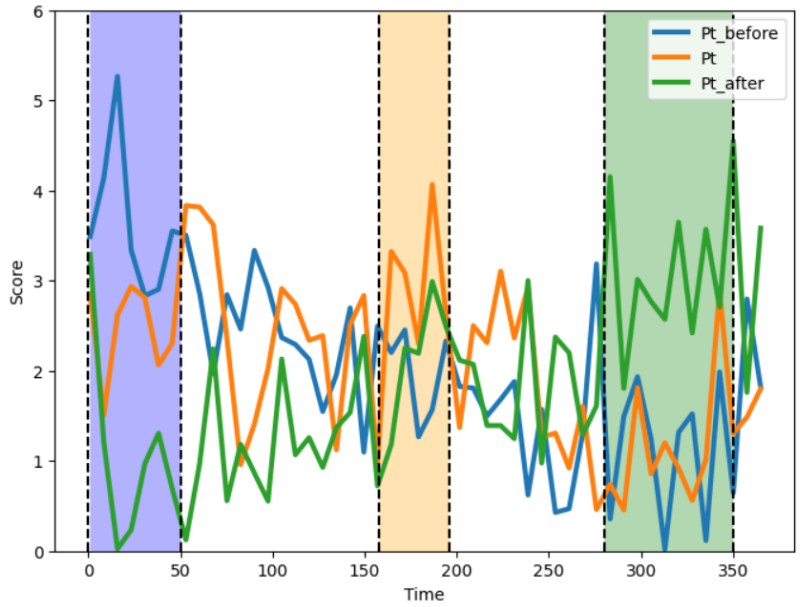


Figure 5.6: Scores distribution along time.

day. For each distribution, we highlight the periods of the highest scores with a colored background. These colored blocks represent the most likely happening time interval of the event. The periods where a score is highest, represented as a colored background, match the logical meaning of these operators. The order of periods is **Pt_Before**, **Pt**, **Pt_After**. This shows that our operators perform the time transformation correctly, which answers **RSQ2.6**.

5.4 Conclusion

In this chapter, we first specify the limitations of the existing QE methods and TKGC models. In section 5.1, we give the definitions of temporal logical queries and the relevant logical operations on TKGs, including relation projection, intersection, union, complement and extended temporal operators (before and after), and formally define the task of multi-hop TKG reasoning (MTKGR). Then, we introduce the evaluation metric of MTKGR. In section 5.2, we present three temporal query datasets generated from ICEWS14, ICEWS05-15, GDELT, respectively. Each dataset has 34 types of query structures, composed of 10 basic logical set functions. For all query structures, we show their illustrations and statistics. In section 5.3, we present the Temporal Feature-Logic embedding framework, TFLEX, to handle temporal logical queries in datasets. Vector logic is used to guide all FOL transformations on the logic part of embeddings. We also further extended vector logic to implement extra temporal operators (**Before**, **After** and **Between**). To the best of our knowledge, TFLEX is the first framework to support multi-hop logical reasoning on TKGs. Experiments on benchmark datasets demonstrate the efficacy of the proposed framework in handling different operations in temporal logical queries.

CHAPTER 6

Temporal Knowledge Graph Embeddings for Entity Alignment

Most KGs are independently extracted from separate data sources, focusing on different domains or languages. Therefore, using a single KG is oftentimes insufficient for the need for downstream applications and it is essential to fuse heterogeneous knowledge among different KGs where the same entities exist in different surface forms. Entity alignment (EA) in different KGs plays an important role in KG fusion. To address this problem, a lot of embedding-based EA methods [25, 27, 29, 153, 202] have been proposed to embed entities into low-dimensional vector spaces, and obtain the pairs of equivalent entities by computing the pair-wise distance between their vector representations.

In Challenge 3, we point out that time information could be helpful for entity alignment between TKGs. As in the case of Figure 6.1 in which the left and right subgraphs are extracted from two separate TKG respectively, the entities *Boris Johnson* and *David Cameron* can be easily distinguished by considering the timestamps of links within their neighborhood, in spite of the similarity of their neighboring nodes and connected relations.

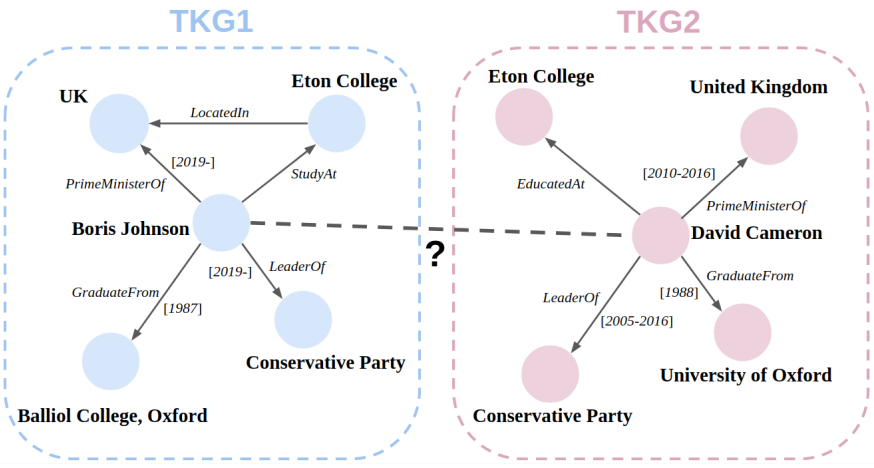


Figure 6.1: An illustration of entity alignment between TKGs.

In this chapter, we focus on the third research question proposed in this thesis, i.e., the task of temporal entity alignment (TEA) between TKGs.

Research Question 3 (RQ3)

Can the incorporation of time information be helpful for the performances of KGE models on the task of entity alignment between temporal knowledge graphs?

To perform the task of TEA, we need to overcome the following limitations of the existing embedding-based static entity alignment (SEA) methods.

1) Temporal Unawareness: The existing SEA approaches disregard time information in TKGs, leaving much room for improvement. Taking the case in Figure 6.1 as an example, given two entities, Boris Johnson and David Cameron, existing in two TKGs respectively, SEA approaches are likely to ignore time information and wrongly recognize these two entities as the same person in the real world due to the homogeneity of their neighborhood information.

2) Reliance on Alignment Seeds: Most SEA approaches rely on pre-aligned seed entity pairs to connect two KGs and use the unified KG structural embeddings to align entities. In practice, such labeled data is very costly to obtain. Noteworthily, timestamps in most TKGs are presented in Arabic numerals and have similar formats. Thus, timestamps representing the same dates across multiple TKGs can be easily aligned by manually uniforming their formats and used as prior knowledge.

3) Training Efficiency: As mentioned in Section 3.4, GNN-based methods have achieved big success in entity alignment between SKGs. However, the training processes of these methods are time-costly. To incorporate time information into GNN models, most of the existing temporal GNN models [35, 36, 203–205] generally discretize a temporal graph into multiple static snapshots in a timeline and use combinations of GNN models and recurrent models whereby the former handle graph information and the latter capture dynamism. Such combination architectures inevitably suffer from long training time.

4) Inductive Learning Inability: Open-world KGs are dynamic with new emerging entities and timestamps. The existing EA methods can not inductively model new emerging entities, which is important for learning and reasoning on an open-world knowledge graph.

In order to study the task of TEA in a systematic way, we first formally define the problem of TEA and the evaluation metrics used for TEA models in Section 6.1. Then, we present three new well-established TKG datasets extracted from ICEWS, YAGO and Wikidata as the references for evaluating SEA and TEA methods in Section 6.2. Importantly, we introduce two novel TEA models which address the RQ3 and overcome the limitations of the existing EA methods mentioned above.

In Section 6.3, we present our first GNN-based TEA approach, **TEA-GNN**, in which entities, relations and timestamps are embedded into a vector space and a self-attention mechanism is used for GNN to specify different weights to different neighboring nodes of each entity with the corresponding link features, i.e., embeddings of the relevant relations and timestamps.

In Section 6.4, we present the second temporal relational EA model, **TREA**. Herein, the initial feature of each entity is represented by fusing the embeddings of its connected relations and timestamps as well as its neighboring entities. Then, a GNN is employed to capture intra-graph information and a time-aware self-attention mechanism is used to assign different weights to different nodes

with orthogonal transformation matrices computed from embeddings of the relevant relations and timestamps in a neighborhood. Finally, a margin-based full multi-class log-loss is used for efficient training and a linear time regularizer is used to model unobserved timestamps.

In Section 6.5, we give a summary of this chapter. Overall, this chapter is based on the following publications [206, 207]:

1. Chengjin Xu, Fenglong Su and Jens Lehmann. "Time-aware Graph Neural Network for Entity Alignment between Temporal Knowledge Graphs", Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, 2021;

2. Chengjin Xu, Fenglong Su, Bo Xiong and Jens Lehmann. "Time-Aware Entity Alignment Using Temporal Relational Attention", Proceedings of the ACM Web Conference 2022, 2022.

In the above papers, the design, implementation, and evaluation of all presented TEA models were done by the Ph.D. candidate alone, who also handled the construction of two new TEA datasets and the writing of both papers. The resource codes and datasets are avaible at https://github.com/soledad921/TEA-GNN.

6.1 Problem Definition and Evaluation Metrics

Problem Definition of Temporal Entity Alignment

A TKG is a directed relational temporal graph $\mathcal{G} = (\mathcal{E}, \mathcal{R}, \mathcal{T}, \mathcal{Q})$ comprising four sets: entities $\mathcal{E}$, relations $\mathcal{R}$, timestamps $\mathcal{T}$, and quadruples $\mathcal{Q} \subseteq \mathcal{E} \times \mathcal{R} \times \mathcal{E} \times \mathcal{T}$. A TKG stores the real-world information in the form of quadruples (e_s, r, e_o, t), where $e_s, e_o \in \mathcal{E}$. Given two TKGs $\mathcal{G}_1 = (\mathcal{E}_1, \mathcal{R}_1, \mathcal{T}_1, \mathcal{Q}_1)$, $\mathcal{G}_2 = (\mathcal{E}_2, \mathcal{R}_2, \mathcal{T}_2, \mathcal{Q}_2)$, and a pre-aligned entity pair set $\mathcal{S} = \{(e_i, e_j)|e_i \in \mathcal{E}_1, e_j \in \mathcal{E}_2, e_i \equiv e_j\}$ as alignment seeds where $\equiv$ denotes equivalence, a unified time set $\mathcal{T}^* = \mathcal{T}_1 \cup \mathcal{T}_2$ can be easily obtained by uniforming the time formats of $\mathcal{T}_1$ and $\mathcal{T}_2$ and these two TKGs can be renewed as $\mathcal{G}_1 = (\mathcal{E}_1, \mathcal{R}_1, \mathcal{T}^*, \mathcal{Q}_1)$ and $\mathcal{G}_2 = (\mathcal{E}_2, \mathcal{R}_2, \mathcal{T}^*, \mathcal{Q}_2)$. The task of EA between TKGs aims at obtaining more potential equivalent entity pairs $\mathcal{P} = \{(e_i, e_j)|e_i \in \mathcal{E}_1, e_j \in \mathcal{E}_2, e_i \equiv e_j\}$ based on the prior information of $\mathcal{G}_1$, $\mathcal{G}_2$, $\mathcal{T}^*$and $\mathcal{S}$.

The task of TEA aims at finding equivalent entity pairs between TKGs. Formally, given an entity $e_i \in \mathcal{E}_1$, a TEA model is expected to predict the correct entity e_j among the set of entities $\mathcal{E}_2$ by computing the similarities between e_i and e_j with the learned TKGEs. TKGEs are learned by maximizing the similarity of two entities of each alignment seed.

For each test entity pairs (e_i, e_j), we first generate candidate entity pairs $C_1 = \{(e_i, e'_j) : e'_j \in \mathcal{E}_2\}$ and $C_2 = \{(e'_i, e_j) : e'_i \in \mathcal{E}_1\}$ by replacing e_i or e_j with all possible entities of the respective TKGs. Then we determine the rank of (e_i, e_j) relative to all $(e_i, e'_j) \in \overline{C}_1$ using the similarities, which is denoted as Rank(e_i—e_j). A similar definition Rank(e_j—e_i) applies to the second query $(?, e_j)$.

Two evaluation metrics are used here, i.e., Mean Reciprocal Rank (MRR) and Hits@k (generally k = 1, 3, 10). The Mean Reciprocal Rank (MRR) is the mean of the reciprocal values of all computed

ranks, i.e.,

$$\text{MRR} = \frac{1}{2|\mathcal{P}|} \sum_{(e_i,e_j)\in\mathcal{P}} \Big(\frac{1}{\text{Rank}(e_i|e_j)} + \frac{1}{\text{Rank}(e_j|e_i)} \Big). \tag{6.1}$$

And the fraction of test quadruples ranking in the top k is called Hits@k, i.e.,

$$\text{Hits@k} = \frac{1}{2|\mathcal{P}|} \sum_{(e_i,e_j)\in\mathcal{P}} \Big(I\big(\text{Rank}(e_i|e_j) \le k\big) + I\big(\text{Rank}(e_j|e_i) \le k\big) \Big), \tag{6.2}$$

where I denotes the indicator function.

6.2 Temporal Entity Alignment Datasets

In this paper, we use three TKG datasets extracted from ICEWS [10], YAGO [8], Wikidata [9] as new references for evaluating temporal and non-temporal EA methods, i.e., **DICEWS**, **YAGO-WIKI50K** and **YAGO-WIKI20K**. The statics of the three temporal EA datasets are listed in Table 6.1.

Dataset	$\lvert\mathcal{E}_1\rvert$	$\lvert\mathcal{E}_2\rvert$	$\lvert\mathcal{R}_1\rvert$	$\lvert\mathcal{R}_2\rvert$	$\lvert\mathcal{T}^*\rvert$	$\lvert\mathcal{Q}_1\rvert$	$\lvert\mathcal{Q}_2\rvert$	$\lvert\mathcal{P}\rvert+\lvert\mathcal{S}\rvert$	$\lvert\mathcal{S}\rvert$
DICEWS-1K/200	9,517	9,537	247	246	4,017	307,552	307,553	8,566	1,000/200
YAGO-WIKI50K-5K/1K	49,629	49,222	11	30	245	221,050	317,814	49,172	5,000/1,000
YAGO-WIKI20K	19,493	19,929	32	130	405	83,583	142,568	19,462	400

Table 6.1: Statistics of TEA datasets.

Integrated Crisis Early Warning System (ICEWS) is a publicly available large-scale event-based database that contains political events with specific time annotations extracted from millions of real-world news stories. It is noteworthy that time annotations in ICEWS are all time points, e.g., (*Barack Obama*, *Visit*, *Ukraine*, *2014-07-08*). ICEW05-15 [132] is a subset of ICEWS which contains 10,094 entities, 251 relations, 4,017 time steps and 461,329 temporal facts occurring during 2005 and 2015, and is commonly used as a TKG benchmark dataset in the community. **DICEWS** is built based on ICEWS05-15 in the similar way to the establishment of DFB datasets [151]. We randomly divide ICEWS05-15 quadruples into three subsets of the same size. And $\mathcal{Q}_1$ consists of the first and second subsets, and the second and third subsets make up $\mathcal{Q}_2$. In this way, $\mathcal{Q}_1$ and $\mathcal{Q}_2$ have the same size, and the overlap ratio of the amount of shared quadruples between $\mathcal{Q}_1$ and $\mathcal{Q}_2$ to all quadruples equals to 50%. The only difference between DICEWS-1K and DICEWS-200 is the number of alignment seed $\mathcal{S}$. In DICEWS-1K and DICEWS-200, i.e., 1,000 and 200 of entity pairs between TKGs are pre-known. The time unit of ICEWS datasets is 1 day, which means that each day is an individual time step. These two TKGs have the same set of time steps $\mathcal{T}^*$, i.e., the sequence of dates in the year 2005.

Wikidata is a free and open knowledge base that store structured data from Wikipedia. YAGO is also an open source knowledge base and is extracted from Wikipedia and other slacks sources. In these two knowledge bases, there are a large number of identical entities represented in different surface forms and a part of the facts are attached with timestamps of various forms, e.g., time points, start/end time and time intervals. Lacroix et al. [33] built a large-scale TKG dataset[1] from Wikidata, which contains 43,2715 entities, 407 relations and 1,724 time steps (only year information was kept)

[1] https://dl.fbaipublicfiles.com/tkbc/data.tar.gz

by filtering out high-frequency entities and relations. The whole dataset has over 7 millions of triples in total and about 10% of them are attached to specific timestamps. To build **YAGO-WIKI50K** from YAGO and Wikidata, the top 50,000 entities are first selected according to their frequencies in this dataset and linked to their equivalent YAGO entities[2] according to their QIDs and the mappings[3] of YAGO entities to Wikidata QIDs. Two TKGs are generated by filtering out facts only involving the selected entities from the above-mentioned subset of Wikidata and all YAGO facts and then attaching complementary time metadata[4] to the corresponding YAGO facts. In this step, a small part of entities are removed. At last, non-temporal facts are removed from these two filtered TKGs to make sure that YAGO-WIKI50K is **fully temporal** and only keep the year information of timestamps in YAGO facts and uniform the time formats of both TKGs to generate the shared time set $\mathcal{T}^*$. **YAGO-WIKI20K** is constructed in the same way except that the amount of selected Wikidata entities is reduced to 20,000 and keep the non-temporal facts in two TKGs of this **temporally hybrid** dataset.

6.3 A Temporal EA Model Using Temporal Graph Neural Network

The content of the this section is based on our work in the paper titled "Time-aware Graph Neural Networks for Entity Alignment between Temporal Knowledge Graphs" (Xu et al., EMNLP 2021) [206].

6.3.1 Introduction

As mentioned before, the existing EA methods disregard time information and TKGE models designed for the task of TKGC are not directly compatible with the EA setting. To cope with EA between TKGs, in this paper, we propose a novel Time-aware Entity Alignment approach based on Graph Neural Networks (TEA-GNN) for entity alignment between TKGs. Different from some temporal GNN models which discretize temporal graphs into multiple snapshots, we treat timestamps as properties of links between entities. We first map all entities, relations and timestamps in TKGs into an embedding space. To incorporate relation and time information into the GNN structure, we utilize a time-aware self-attention mechanism that assigns different importance weights to different nodes within a neighborhood, which are computed with the embeddings of the corresponding relations and timestamps. To further integrate time information into the final entity representations, we concatenate output features of entities with the summation of their neighboring time embeddings to get multi-view entity representations.

Specifically, we create an inverse relation r^{-1} for each relation r to integrate direction information. And a time-aware fact involving a time interval $(e_s, r, e_o, [t_b, t_e])$, where t_b and t_e denote the beginning and end time, is separated into two quadruples (e_s, r, e_o, t_b) and (e_o, r^{-1}, e_s, t_e), which represent the beginning and the end of the relation, respectively. In this way, TEA-GNN can adapt well to datasets where timestamps are represented in various forms: time points, beginning or end time, time intervals.

To verify our proposed approach, we evaluate TEA-GNN and its time-agnostic variant as well as several state-of-the-art SEA approaches on real-world datasets extracted from ICEWS, YAGO3 and Wikidata. Experimental results show that TEA-GNN significantly outperforms all baseline models

[2] http://resources.mpi-inf.mpg.de/yago-naga/yago3.1/yagoFacts.ttl.7z

[3] http://resources.mpi-inf.mpg.de/yago-naga/yago3.1/yagoWikidataInstances.ttl.7z

[4] http://resources.mpi-inf.mpg.de/yago-naga/yago3.1/yagoMetaFacts.ttl.7z

with the inclusion of time information. To the best of our knowledge, this work is the first attempt to perform entity alignment between TKGs using a time-aware embedding-based approach.

6.3.2 Methodology

To exploit both relation and time information for entity alignment, we first create an inverse link for each link so that each pair of inverse links between entities can represent relation directions and handle the beginning and end of the relation. A time-aware self-attention mechanism is employed in each GNN layer to assign different weights to entities according to relation and time information between them. Finally, entity pairs are predicted by applying a distance function to multi-view representations of entities.

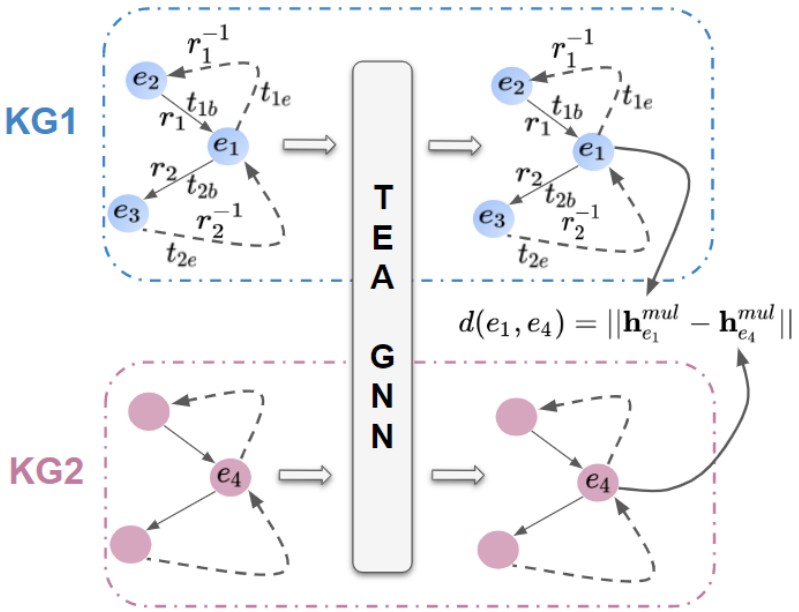


Figure 6.2: The Framework of TEA-GNN.

Inverse Link Generation

Time information t in a temporal fact (e_s, r, e_o, t) can be represented in various forms, e.g., time points, beginning or end time and time intervals. A time interval is shaped like $[t_b, t_e]$ where t_b and t_e denote the actual beginning time and end time of the fact, respectively. A time point can be represented as $[t_b, t_e]$ where $t_b = t_e$. Noteworthily, we represent a beginning or end time as $[t_b, t_0]$ or $[t_0, t_e]$ where $t_0 \in \mathcal{T}^*$ is the first time step in the time set denoting the unknown time information. A fact without known time information can be denoted as $(e_s, r, e_o, [t_0, t_0])$ to deal with heterogeneous temporal knowledge bases where a significant amount of relations might be non-temporal.

In order to integrate relation direction, we create an inverse relation r^{-1} for each relation r and extend the relation set $\mathcal{R} = \{r_0, r_1, \cdots, r_{|\mathcal{R}|-1}\} \rightarrow \{r_0, r_0^{-1}, \cdots, r_{|\mathcal{R}|-1}, r_{|\mathcal{R}|-1}^{-1}\}$. And each fact $(e_s, r, e_o, [t_b, t_e])$ is decomposed into two quadruples (e_s, r, e_o, t_b) and (e_o, r^{-1}, e_s, t_e) to handle

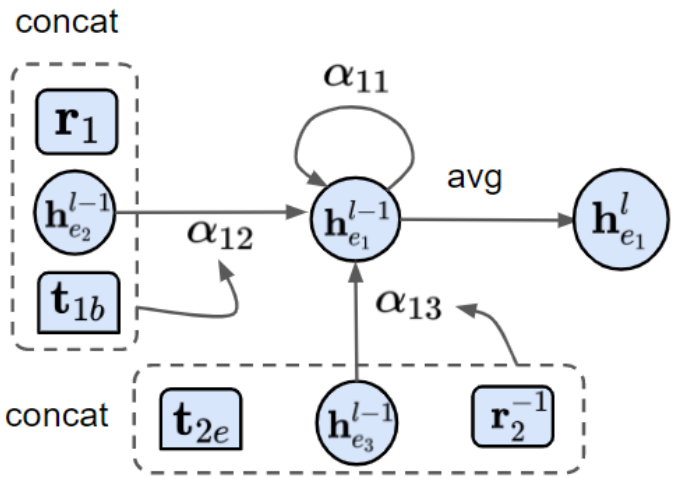


Figure 6.3: An illustration of the time-aware self-attention mechanism by the node e_1.

the beginning and the end of the relation, respectively. As shown in Figure 6.2, the dashed arrows represent the created inverse links.

Time-Aware Self-Attention

Graph Attention Network (GAT) [97] extends the vanilla Graph Convolutional Network (GCN) [95] by employing a self-attention mechanism to calculate the hidden representation of each node by attending over its neighbors. A GAT layer can be defined as follow,

$$\mathbf{h}_{e_i}^{l} = \sigma\Big(\sum_{e_j \in \mathcal{N}_{e_i}^{e} \cup e_i} \alpha_{i,j}^{l} \mathbf{W} \mathbf{h}_{e_j}^{l-1} \Big), \tag{6.3}$$

where $\mathbf{h}_{e_i}^{l} \in \mathbb{R}^k$ denotes the k-dimensional output feature vector of the entity e_i in the l-th hidden layer, $\sigma(\cdot)$ denotes the nonlinear activation function, $\mathcal{N}_{e_i}^{e}$ denotes the set of the neighboring entities of e_i, $\mathbf{W}$ denotes the shared transformation matrix, and $\alpha_{i,j}^{l}$ denotes the attention coefficient of e_j to e_i.

Time-aware self-attention aims at integrating both time and relation information into the generated entity representations by assigning different weights to different neighboring nodes according to the time and relation features of inward links between nodes. We define the weighted importance $\beta_{i,j}^{l}$ of neighboring entity e_j to e_i in the l-th hidden layer as follows,

$$\beta_{i,j}^{l} = \boldsymbol{\omega}^{\top} \left[\mathbf{h}_{e_i}^{l-1} \Big\| \mathbf{h}_{e_j}^{l-1} \Big\| \sum_{r_m \in \mathcal{L}_{ij}^{r}} \frac{\mathbf{r}_m}{|\mathcal{L}_{ij}^{r}|} \Big\| \sum_{t_n \in \mathcal{L}_{ij}^{\tau}} \frac{\mathbf{t}_n}{|\mathcal{L}_{ij}^{t}|} \right], \tag{6.4}$$

where $||$ denotes the concatenation operator, $\boldsymbol{\omega} \in \mathbb{R}^{4k}$ is a shared attention weight vector, $\mathcal{L}_{ij}^{r}$ and $\mathcal{L}_{ij}^{t}$ denote the sets of relations and time steps in the inward links from e_j to e_i respectively, $\mathbf{r}_m \in \mathbb{R}^k$ and $\mathbf{t}_n \in \mathbb{R}^k$ denote embeddings of relation $r_m \in \mathcal{L}_{ij}^{r}$ and timestamp $t_n \in \mathcal{L}_{ij}^{t}$. In the case of Figure 6.2 and 6.3, the inward links in the neighborhood of the entity e_1 include (e_2, r_1, e_1, t_{1b}) and

$(e_3, r_2^{-1}, e_1, t_{2e})$ in which e_1 performs as the object entity.

Following GAT, we define the normalized element $\alpha_{i,j}$ representing the connectivity from entity e_i to e_j with a LeakyReLU activation function (in which the negative input slope $\alpha = 0.2$) as follows,

$$\alpha_{i,j}^{l} = \frac{\exp(\text{LeakyReLU}(\beta_{i,j}^{l}))}{\sum_{e_m \in \mathcal{N}_i^e \cup \{e_i\}} \exp(\text{LeakyReLU}(\beta_{i,m}^{l}))}, \tag{6.5}$$

We take the original time-aware entity representations as the input features of entities in the first hidden layer. And the output features $h_{e_i}^{l}$ are obtained with a linear combination of the input features of neighboring entities and a nonlinear ReLU activation function $\sigma(\cdot)$, i.e.,

$$\mathbf{h}_{e_i}^{l} = \sigma\left(\frac{1}{M}\sum_{m=1}^{M}\left[\sum_{e_j \in \mathcal{N}_i^e \cup \{e_i\}} \alpha_{i,j}^{l,m}\mathbf{h}_{e_j}^{l-1}\right]\right). \tag{6.6}$$

Same as GAT, we utilize the averaging multi-head attention to stabilize the learning process of self-attention. M denotes the number of attention heads and $\alpha_{i,j}^{l,m}$ are normalized attention coefficients computed by the m-th attention mechanism in the l-th hidden layer.

Noteworthily, we do not follow GAT to use a linear transformation matrix in Equation 6.18, since we did not find that the linear transformation matrix is helpful in improving the performance of TEA-GNN. This also complies with the implementation in some other recent GNN-based SEA works [27, 29]. It is also worth noting that a TEA-GNN attention head has a lower time complexity compared to a single GAT attention head. In Section 2.2 of the original GAT paper [97], the authors proposed "The time complexity of a single GAT attention head computing F' features may be expressed as $O(|V|FF' + |E|F')$, where F is the number of input features, and $|V|$ and $|E|$ are the numbers of nodes and edges in the graph, respectively". Noteworthily, a weight matrix is used for each single attention head and the time complexity of the multiplications of and the feature vectors of $|V|$ nodes is $O(|V|FF')$. Since we use a weight vector instead of the weight matrix used in the vanilla GAT, the time complexity of the multiplications of and the feature vectors of $|V|$ entities is $O(|V|F)$. In our work, the number of nodes (entities), edges (quadruples) and input features are denoted as $\mathcal{E}$, Q and k. Thus, the time complexity of a single attention head in TEA-GNN is denoted as $O = |\mathcal{E}|k + |Q|k$, and the complexity of a single attention head in vanilla GAT would be $O = |\mathcal{E}|k^2 + |Q|k$ in our case. It can be seen that the computation of a TEA-GNN attention head is more efficient than a vanilla GAT attention head.

Multi-view Entity Representation

An entity align model aims at embedding two KGs into a unified vector space by pushing the seed alignments of entities together. In this work, the entity align model consists of multiple TEA-GNN layers and a distance function which measures the similarities between final representations of entities.

Let $\mathbf{h}_{e_i}^{0}$ denote the input feature vector of entity e_i in the first hidden layer. Noteworthily, $\mathbf{h}_{e_i}^{0} = \mathbf{e}_i$ is also the k-dimensional embedding vector of e_i in our case. A cross-layer representation is employed to capture multi-hop neighboring information in previous work [29] by concatenating output features of different layers. In the same way, we define the global output features $\mathbf{h}_{e_i}^{out}$ of e_i as

$$\mathbf{h}_{e_i}^{out} = [\mathbf{h}_{e_i}^{0} || \mathbf{h}_{e_i}^{1} || \cdots || \mathbf{h}_{e_i}^{L}], \tag{6.7}$$

where L is the number of layers.

We further concatenate the average embeddings of connected timestamps with output features of entities to get multi-view embeddings as final entity representations, i.e.,

$$\mathbf{h}_{e_i}^{mul} = [\mathbf{h}_{e_i}^{out} || \frac{1}{|\mathcal{N}_{e_i}^{\tau}|} \sum_{t_m \in \mathcal{N}_{e_i}^{\tau}} \mathbf{t}_m], \tag{6.8}$$

where $\mathcal{N}_{e_i}^{\tau}$ represents the set of timestamps around entity e_i.

Optimization

Entity alignments are predicted based on the distances between the final output features of entities from two KGs. For two entities $e_i \in \mathcal{E}_1$ and $e_j \in \mathcal{E}_2$ from different sources, we use L1 distance to measure the distance between them as follows,

$$d(e_i, e_j) = ||\mathbf{h}_{e_i}^{mul} - \mathbf{h}_{e_j}^{mul}||_1. \tag{6.9}$$

A margin rank loss is used as the optimization objective of the entity align model, i.e.,

$$\begin{aligned} \mathcal{L} = & \sum_{(e_i,e_j)\in\mathcal{S}} \sum_{(e_i,e_j')\in C_1} [d(e_i, e_j) + \gamma - d(e_i, e_j')]_+ \\ & + \sum_{(e_i,e_j)\in\mathcal{S}} \sum_{(e_i',e_j)\in C_2} [d(e_i, e_j) + \gamma - d(e_i', e_j)]_+, \end{aligned} \tag{6.10}$$

where $[x]_+$ represents the operation $Max(x, 0)$, γ denotes the margin, C_1 and C_2 are the sets of generated negative entity pairs, $e_i' \in \mathcal{E}_1$ and $e_j' \in \mathcal{E}_2$ are the negative entities of e_i and e_j. Negative entities are sampled randomly and an RMSprop optimizer [208] is used to minimize the loss function.

Following previous SEA works [29, 30], we adopt Cross-domain Similarity Local Scaling (CSLS) [209] as the distance metric during testing to measure similarities between entity embeddings.

Parameter Complexity

We compare the parameter complexity of TEA-GNN with several existing SEA models. As shown in Table 6.2, compared to parameter-efficient translational entity align models like MTransE, TEA-GNN uses additional parameters only for inverse relation embeddings, time embeddings and attention weight vectors, which are much fewer than parameters of entity embeddings in most cases.

6.3.3 Experiments

To show the capability of TEA-GNN, we compare it with several state-of-the-art SEA models and the TKGC models presented in this thesis on TEA datasets proposed in Section 6.2. Particularly, we conduct a case study to show the effect of incorporating time information and a sensitivity study to show the robustness of TEA-GNN against the size of alignment seeds. We also compare the training time of TEA-GNN and strong SEA baseline models.

EA Methods	Number of Trainable Parameters
MTransE	$(\lvert\mathcal{E}_1\rvert + \lvert\mathcal{E}_2\rvert + \lvert\mathcal{R}_1\rvert + \lvert\mathcal{R}_2\rvert)k$
JAPE	$(\lvert\mathcal{E}_1\rvert + \lvert\mathcal{E}_2\rvert + \lvert\mathcal{R}_1\rvert + \lvert\mathcal{R}_2\rvert)k$
BootEA	$(\lvert\mathcal{E}_1\rvert + \lvert\mathcal{E}_2\rvert + \lvert\mathcal{R}_1\rvert + \lvert\mathcal{R}_2\rvert)k$
GCN-Align	$(\lvert\mathcal{E}_1\rvert + \lvert\mathcal{E}_2\rvert)k + 2k^2$
MuGNN	$(\lvert\mathcal{E}_1\rvert + \lvert\mathcal{E}_2\rvert + \lvert\mathcal{R}_1\rvert + \lvert\mathcal{R}_2\rvert)k + 2k + k^2$
MRAEA	$(\lvert\mathcal{E}_1\rvert + \lvert\mathcal{E}_2\rvert + 2\lvert\mathcal{R}_1\rvert + 2\lvert\mathcal{R}_2\rvert)k + 3MLk$
HyperKA	$(\lvert\mathcal{E}_1\rvert + \lvert\mathcal{E}_2\rvert + \lvert\mathcal{R}_1\rvert + \lvert\mathcal{R}_2\rvert)k + Lk^2$
RREA	$(\lvert\mathcal{E}_1\rvert + \lvert\mathcal{E}_2\rvert + 2\lvert\mathcal{R}_1\rvert + 2\lvert\mathcal{R}_2\rvert)k + 3Lk$
KE-GCN	$(\lvert\mathcal{E}_1\rvert + \lvert\mathcal{E}_2\rvert + \lvert\mathcal{R}_1\rvert + \lvert\mathcal{R}_2\rvert)k + (\lvert\mathcal{R}_1\rvert + \lvert\mathcal{R}_2\rvert + 2)Lk^2$
TEA-GNN	$(\lvert\mathcal{E}_1\rvert + \lvert\mathcal{E}_2\rvert + 2\lvert\mathcal{R}_1\rvert + 2\lvert\mathcal{R}_2\rvert + \lvert\mathcal{T}^*\rvert)k + 4MLk$

Table 6.2: Comparison of numbers of trainable parameters between TEA-GNN and popular SEA methods.

Baselines

We compare TEA-GNN with three strong translational SEA models and six state-of-the-art GCN-based SEA models, including MTransE [25], JAPE [26], AlignE [153], GCN-Align [27], MuGNN [28], MRAEA [29], HyperKA [160], RREA [30] and KE-GCN [161]. We choose AlignE instead of BootEA since we do not use iterative learning for other models including our proposed models. Due to the deficiency of attribute information, we do not select attribute-aware entity alignment models, e.g., AttrE [155] or AttrGCN [210], and use the SE (Structural Embedding) variants of JAPE and GCN-Align as baseline models. Since we do not use entity names as enhancement information for our proposed models, entity alignment models using textual information like HGCN and RDGCN [157, 159] are also excluded from the baseline.

Besides, we also modify ATiSE, TeRo and TGeomE to make them compatible with EA setting. We merge the two TKGs in a dataset as one by letting the two entities in each seed entity alignment have the same embedding and use these modified models to learn embeddings. The optimization objectives of their training processes are still to maximize the scores of quadruples while their tasks at the evaluation phase are ranking entity pairs.

To verify the effectiveness of the integration of time information, we implement a time-unaware variant of TEA-GNN which takes all time steps $t_i \in \mathcal{T}^*$ as unknown time information t_0.

Experimental Setup

We use the source codes respective to baseline models for evaluation, except that we evaluate MTransE based on the implementation of the OpenEA framework [5].

For all baseline models, we mostly follow their default optimal configurations regarding learning rates lr, batch sizes b, negative sampling rates η, dropout rates dr, numbers of GNN layers L and mainly focus on the grid research of embedding dimensions k and margins γ (negative weights α for JAPE). We also follow the respective original papers to fix the numbers of multi-head attention mechanisms as 2 for MRAEA and set the balance weight $\beta = 0.9$ for GCN-Align. For all baseline

[5] https://github.com/nju-websoft/OpenEA/

models and our proposed models, we tune k in the range of $(25, 50, 75, 100)$, and γ or α in the range of $(0, 0.5, 1, 2, 3, 5, 7, 10, 15, 20)$. Specially, we use the same margin hyperparameters as the original paper for AlignE. To make a fair comparison, we use the same setup for our proposed model as MRAEA and RREA to fix $M = 2$ $L = 2$, $dr = 0.3$, $b = |\mathcal{E}_1| + |\mathcal{E}_2|$ and $\eta = b//|\mathcal{S}| + 1$ where $//$ denotes the round-down after division, and also conduct the same grid research of embedding dimensions k and margins γ for TEA-GNN and TU-GNN as what we do for baseline models.

The default configuration of TEA-GNN is as follows: embedding dimension $k = 100$, learning rate $lr = 0.005$, number of TEA-GNN layers $L = 2$, margin $\gamma = 1$ and dropout rate is 0.3. Below we only list the non-default hyperparamters: $\gamma = 3$ for DICEWS-200 and YAGO-WIKI20K; $k = 25$ for YAGO-WIKI50K-5K and YAGO-WIKI50K-1K. The non-default hyperparameters of TU-GNN are as follows: $\gamma = 3$ for DICEWS-1K and YAGO-WIKI20K; $\gamma = 5$ for DICEWS-200; $k = 25$ for YAGO-WIKI50K datasets.

Except that the experiment of MTransE is implemented based on the OpenEA framework [211], all experiments of baseline models are implemented based on their resource codes. All target models including our proposed models are trained on a GeForce GTX 1080Ti GPU. For a fair comparison, we set the maximum embedding dimension as 100 for all target models.

Main Results

Models	DICEWS-1K			DICEWS-200		
	MRR	Hits@1	Hits@10	MRR	Hits@1	Hits@10
MTransE	.150	.101	.241	.104	.067	.175
JAPE	.198	.144	.298	.138	.098	.210
AlignE	.593	.508	.751	.303	.222	.457
GCN-Align	.291	.204	.466	.231	.165	.363
MuGNN	.617	.525	.794	.412	.367	.583
MRAEA	.745	.675	.870	.564	.476	.733
HyperKA	.669	.588	.842	.474	.383	.653
RREA	.780	.722	.883	.719	.659	.824
KE-GCN	.650	.549	.827	.451	.373	.625
ATiSE	.135	.078	.242	.097	.050	.185
TeRo	.147	.093	.255	.105	.062	.190
TGeomE	.098	.055	.181	.063	.041	.108
TU-GNN	.693	.610	.848	.610	.518	.788
TEA-GNN	**.911**	**.887**	**.947**	**.902**	**.876**	**.941**

Table 6.3: Entity alignment results on DICEWS datasets.

Table 6.3 and 6.4 show the EA results of TEA-GNN and all baselines on DICEWS and YAGO-WIKI50K datasets. It can be shown that TEA-GNN remarkably outperforms all baseline models on four TKG datasets across all metrics. Compared to RREA which achieves the best results among all baseline models, TEA-GNN obtains improvements of 22.9%, 32.9%, 6.2% and 3.9% regarding

Models	YAGO-WIKI50K-5K			YAGO-WIKI50K-1K		
	MRR	Hits@1	Hits@10	MRR	Hits@1	Hits@10
MTransE	.322	.242	.477	.033	.012	.067
JAPE	.345	.271	.488	.157	.101	.262
Align	.800	.756	.883	.618	.565	.714
GCN-Align	.363	.581	.512	.711	.279	.217
MuGNN	.808	.762	.890	.632	.589	.733
MRAEA	.848	.806	.913	.685	.623	.801
HyperKA	.829	.784	.900	.665	.610	.775
RREA	.868	.828	.938	.753	.696	.859
KE-GCN	.831	.780	.910	.654	.600	.761
ATiSE	.305	.216	.465	.041	.015	.081
TeRo	.324	.244	.481	.045	.018	.088
TGeomE	.168	.116	.270	.023	.008	.053
TU-GNN	.839	.795	.916	.712	.647	.834
TEA-GNN	**.909**	**.879**	**.961**	**.775**	**.723**	**.871**

Table 6.4: Entity alignment results on YAGO-WIKI50K datasets.

Hits@1 on four TKG datasets, respectively. Another observation is that ATiSE and TeRo underperform translational SEA models even with additional time embeddings, and the results of TGeomE are also not ideal. These results demonstrate that TKGC models are not directly suitable for TEA tasks. A previous study [212] also shows that tensor decomposition SKGC models have poor performances on the SEA task.

Case Study

Entities to Be Aligned	Predictions	Similar Facts (Links) Involving Aligned Entities Between $\mathcal{Q}_1$ and $\mathcal{Q}_2$
Daniel Scioli (in $\mathcal{E}_1$ of DICEWS)	TEA-GNN: Daniel Scioli	(Daniel Scioli, Accuse, Senate (Argentina), 2015-06-28), (President (Argentina), Make Statement, Daniel Scioli, 2015-03-07), ... (in $\mathcal{Q}_1$ of DICEWS)
	TU-GNN: Agustín Rossi	(Agustín Rossi, Accuse, Senate (Argentina), 2009-08-26), (President (Argentina), Make Statement, Agustín Rossi, 2015-04-18), ... (in $\mathcal{Q}_2$ of DICEWS)
Leon Benko (in $\mathcal{E}_2$ of YAGO-WIKI50K)	TEA-GNN: ¡Leon_Benko¿	(¡Olivier_Fontenette¿, ¡playsFor¿, ¡K.V._Kortrijk¿, [2008, -]), ... (in $\mathcal{Q}_1$ of YAGO-WIKI50K)
	TU-GNN: ¡Olivier_Fontenette¿	(Leon Benko, member of sports team, K.V. Kortrijk, [2009, 2010]), ... (in $\mathcal{Q}_2$ of YAGO-WIKI50K)

Table 6.5: Examples of different alignment predictions between TEA-GNN and TU-GNN.

To study the effect of the integration of time information on the entity alignment performances of TEA-GNN, we conduct a case study of TEA-GNN and its time-unaware variant TU-GNN. Table 6.11

lists several examples that TEA-GNN gives different predictions from TU-GNN with consideration of additional time information. In the first case, TU-GNN wrongly aligns two entities from $\mathcal{G}_1$ and $\mathcal{G}_2$ of DICEWS, i.e., Daniel Scioli and Agustín Rossi, because these two entities have very similar connected links in $\mathcal{G}_1$ and $\mathcal{G}_2$ regardless of time information. As shown in Table 6.11, some links respective to these two entities in $\mathcal{G}_1$ and $\mathcal{G}_2$ have the same linked entities and relation types, leading to the result that TU-GNN identifies them as an equivalent entity pair. On the other hand, TEA-GNN can correctly distinguish these two entities since the relevant links have different timestamps. Similarly, TU-GNN recognizes a Wikidata entity Leon Benko (Q1389599) and a YAGO entity ¡Olivier_Fontenette¿ as the same person since these two persons played for the same football club, while TEA-GNN can learn that they played for different periods and thus are not the same person in the real world. These cases demonstrate the effect of time information on the performances of TEA-GNN.

Sensitivity Study

In this study, we would like to answer the following research questions: 1. Does the temporal EA method have better robustness to the size of pre-aligned entity pairs than static ones? 2. In TEA-GNN, what effect does the inclusion of time information have on entities with different time sensitivities?

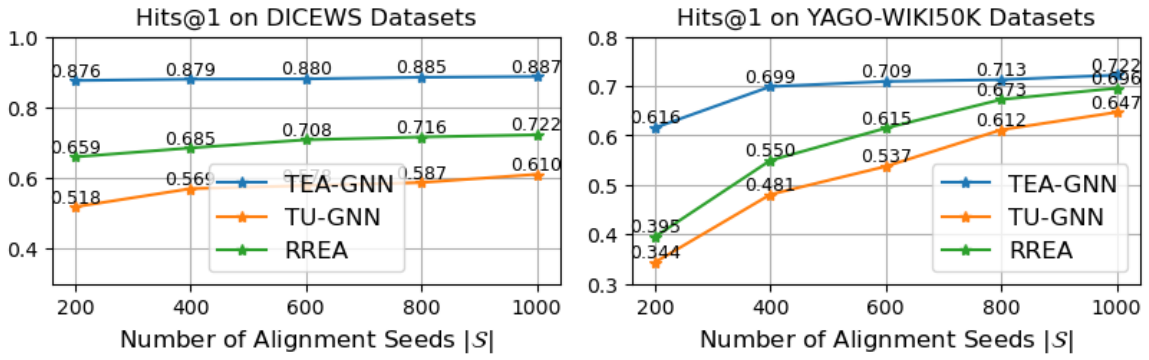


Figure 6.4: Hits@1 of TEA-GNN, TU-GNN and RREA, w.r.t. number of alignment seeds —$\mathcal{S}$—.

Numerically, compared to TU-GNN, TEA-GNN improves Hits@1 by 45.4% and 69.1% on DICEWS-1K and DICEWS-200, 10.6% and 11.6% on YAGO-WIKI50K-5K and YAGO-WIKI50K-1K. It can be shown that the improvements on datasets with fewer alignment seeds are more significant. To further verify this observation, we evaluate the performances of these two models and RREA on DICEWS and YAGO-WIKI50K datasets with different numbers of alignment seeds. As shown in Figure 6.4, the performance difference between TEA-GNN and two static models becomes greater with the decrease of the numbers $|\mathcal{S}|$ of alignment seeds from 1000 to 200. In practical applications, alignment seeds are difficult to obtain. Since TEA-GNN performs well with a small amount of pre-aligned entity pairs, it can more easily be applied in large-scale KGs compared to SEA methods.

We also conduct a study on the prediction accuracies of aligned entities which have different time sensitivities. As mentioned in Section 6.2, we generate a hybrid dataset YAGO-WIKI20K where 17.5% of YAGO facts and 36.6% of Wikidata facts are non-temporal. We divide all testing entity pairs in this dataset into two categories based on their sensitivity to time information, i.e., **highly time-sensitive** entity pairs and **lowly time-sensitive** entity pairs. Time sensitivity s_i of a single entity

e_i is defined as the ratio of the number of its time-aware connected links in which $\tau \neq \tau_0$ over the total number of all links $\mathcal{L}_i$ within its neighborhood, i.e.,

$$s_i = 1 - |\mathcal{L}_i^{\tau_0}|/|\mathcal{L}_i|, \tag{6.11}$$

where $\mathcal{L}_i^{\tau_0}$ denotes the set of time-unaware links connecting e_i. Given an entity pair (e_{i1}, e_{i2}) between $\mathcal{G}_1$ and $\mathcal{G}_2$, we call them as a highly time-sensitive entity pair if $s_{i1} \geqslant 0.5$ and $s_{i2} \geqslant 0.5$. Otherwise, they are lowly time-sensitive.

	Highly Time-Sensitive			**Lowly Time-Sensitive**			**In Total**		
	MRR	Hits@1	Hits@10	MRR	Hits@1	Hits@10	MRR	Hits@1	Hits@10
TEA-GNN	.888	853	.950	.364	.319	.449	.553	.512	.630
TU-GNN	.790	.737	.888	.366	.315	.463	.519	.468	.617

Table 6.6: Entity alignment results on different test sets of YAGO-WIKI20K.

Among 19,062 testing entity pairs of YAGO-WIKI20K, 6,898 of them are highly time-sensitive and others are lowly time-sensitive according to the above definitions. The entity alignment results of TEA-GNN and TU-GNN on the highly time-sensitive test set and the lowly time-sensitive test set are reported in Table 6.6. It can be shown that TEA-GNN and TU-GNN have close performance on entity alignment for lowly time-sensitive entity pairs while TEA-GNN remarkably outperforms TU-GNN on the highly time-sensitive test set. In other words, the effect of incorporation of time information is more significant when testing entity pairs are more time-sensitive.

Efficiency Study

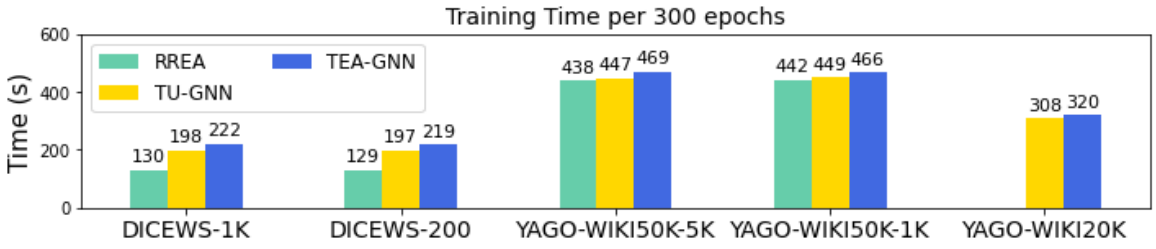


Figure 6.5: Training time per 300 epochs of EA models on different datasets.

As shown in Figure 6.5, the processing of the additional time information does not excessively increase the training time for TEA-GNN, compared to RREA and TU-GNN. Since we set the maximum number of epochs as 6000, the training processes of our proposed models on different datasets can be completed within a couple of hours on a single GeForce GTX 1080Ti GPU.

6.3.4 Conclusion

GNN-based methods have been successful for entity alignment between KGs, but lack consideration of time information. To address this challenge, we present a GNN-based method for entity alignment

between TKGs. The main contributions of this section are threefold:

- We propose a novel GNN-based approach, TEA-GNN, which can model TKGs with a time-aware self-attention mechanism to perform temporal entity alignment. To the best of our knowledge, this work is the first attempt to integrate time information into an embedding-based EA approach.
- Existing temporal GNN models typically discretize a temporal graph into multiple static snapshots and utilize a combination of GNNs and recurrent architectures. Differently, we treat timestamps as attentive properties of links between nodes. This method has been proven to be time-efficient in our case and could potentially be used for non-relational temporal graph representation learning.
- Experiments show that TEA-GNN remarkably outperforms the state-of-the-art SEA models on various well-built TKG datasets and has great robustness against the number of pre-aligned entity pairs.

6.4 An Inductive Temporal EA Model Using Temporal Relational Attention

The content of the this section is based on our work in the paper titled "Time-aware Entity Alignment using Temporal Relational Attentions" (Xu et al., WWW 2022) [207].

6.4.1 Introduction

TEA-GNN addresses some of the limitations of existing SEA methods to a certain degree, e.g., temporal unawareness, and reliance on alignment seeds. However, the training processes of TEA-GNN are still more time-costly than the existing SEA models. Moreover, TEA-GNN adopts the time embedding technique to represent time information, however, does not use time regularization to retain the physical characteristics of time data. Recently, inductive representation learning on large graphs has drawn increasing attentions. Inductive GNN models, like GraphSage [96], have the ability of inductively representing new emerging nodes in a dynamic graph. However, TEA-GNN and the existing GNN-based SEA methods lack inductive learning ability and are unable to represent new entities and timestamps emerging in an open-world KG (OKG).

To address the above issues, we introduce a novel Temporal Relational Entity Alignment method, TREA, for EA between TKGs and DKGs. TREA maps entities, relations and timestamps in TKGs into an embedding space, and the initial feature of each entity is represented by fusing the embeddings of its connected relations and timestamps as well as its neighboring entities so that a new emerging entity can also be inductively represented. We employ a graph neural network (GNN) to learn the semantics of entities and capture structural information, and a temporal relational attention mechanism is used to incorporate time and relation information of links into the GNN by assigning respective weights to different nodes within a neighborhood according to orthogonal transformation matrices computed with embeddings of the corresponding timestamps and relation. Then, a margin-based full multi-class log-loss is used for efficient training and a sequential time regularizer is used to model unobserved timestamps. At last, entities are aligned by computing the distances between their multi-view representations. The experimental results on three well-built TKG benchmarks show that

our method outperforms TEA-GNN and several state-of-the-art SEA models on tasks of both temporal EA and inductive EA (IEA).

6.4.2 Methodology

As mentioned in the introduction, a lot of recent knowledge bases involves temporal facts and real-world knowledge bases are often dynamic with new emerging entities and timestamps. However, most of the existing embedding-based EA methods disregard time information and are incapable of inductively modeling new emerging entities and timestamps. To address these defects, we propose a novel *Temporal Relational Entity Alignment* method, **TREA**. Figure 6.6 depicts that TREA consists of three major parts: (i) *Neighborhood Aggregation Representation* (NAR); (ii) *Temporal Relational Attention* (TRA); (iii) *Margin-based Multi-class Log-loss* (MML).

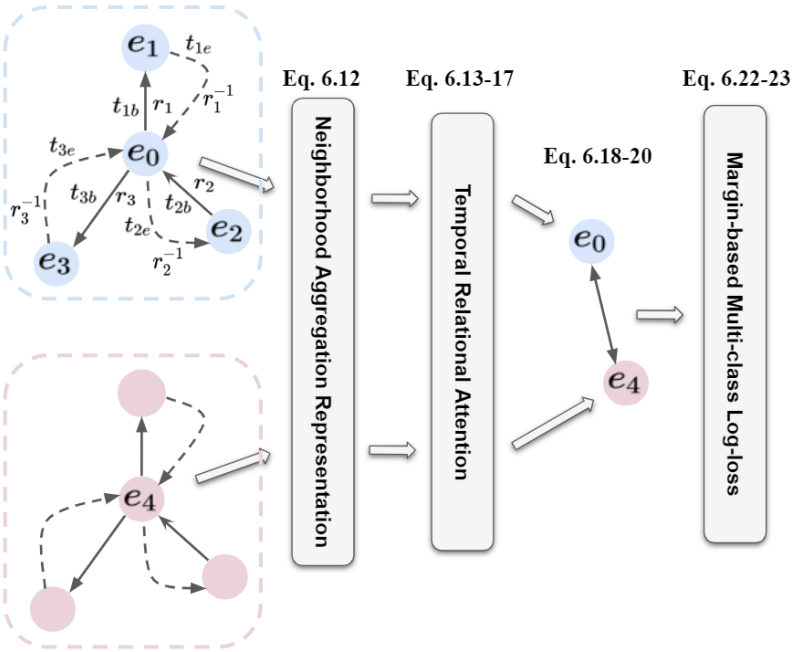


Figure 6.6: Framework of TREA.

Neighborhood Aggregation Representation

Timestamps in TKGs can be represented in different forms, i.e., time points, start/end time, and time intervals and some TKGs involve non-temporal facts. Following TEA-GNN, we use a time range ($[t_b, t_e]$) where t_b denotes the beginning time and t_e denotes the end time to represent each timestamp. Specifically, we have $t_b = t_e$ for a time point and $t_e = t_0$ or $t_b = t_0$ for a start time or end time where $t_0 \in \mathcal{T}$ denotes the first time step and indicates that the time data is unobtainable. We create an inverse link for each link to accumulate the direction information. Each inverse link involves a reciprocal relation r^{-1} corresponding to the relation r of the original link. Thus, the relation set $\mathcal{R}$ is extended to have its inverse. And the beginning time t_b and end time t_e of each timestamp are separately attached to the original link and the inverse link. Thus, each fact $(e_s, r, e_o, [t_b, t_e])$ can be decomposed into

two quadruples (e_s, r, e_o, t_b) and (e_s, r^{-1}, e_o, t_e). As shown in Figure 6.6, dashed arrows denote inverse links.

We map all entities, relations (including reverse relations) and time steps in TKGs into a vector space $\mathbb{R}^k$ where k denotes the dimension of the vector space. Embeddings of the entity e_i, relation r_m, time step t_n are denoted as $\mathbf{e}_i, \mathbf{r}_m, \mathbf{t}_n \in \mathbb{R}^k$, respectively. Let $\mathbf{h}^l_{e_i}$ denote the output feature vector of e_i in the l-th hidden layer and $\mathbf{h}^0_{e_i}$ be the input feature vector of e_i in the first hidden layer. To enforce neighborhood information into the entity representation, we average the embeddings of each entity and its neighboring entities and then concatenate the average entity embedding with the features of the inward links in the entity's neighborhood. The complete neighborhood aggregation representation $\mathbf{h}^0_{e_i}$ of an entity e_i can be formulated as,

$$\mathbf{h}^0_{e_i} = \Big[\frac{1}{|\mathcal{N}^e_i| + 1} \sum_{e_j \in \mathcal{N}^e_i \cup e_i} \mathbf{h}_{e_j} \Big\| \frac{1}{|\mathcal{N}^r_i|} \sum_{r_j \in \mathcal{N}^r_i} \mathbf{r}_j \Big\| \frac{1}{|\mathcal{N}^t_i|} \sum_{t_j \in \mathcal{N}^t_i} \mathbf{t}_j \Big] \tag{6.12}$$

where $\mathcal{N}^e_i$ is the set of neighboring entities of e_i, $\mathcal{N}^r_i$ and $\mathcal{N}^\tau_i$ are sets of relations and time steps which connect inwardly to e_i. $||$ denotes the concatenation operator. For a new emerging entity e_i without an entity embedding h_{e_i}, we only consider its neighboring entities $e_j \in \mathcal{N}^e_i$ which are observed in the original KG. By using neighborhood aggregation representation, we can represent both observed entities and new emerging entity in an inductive manner.

Temporal Relational Attention

To develop a temporal relational attention mechanism, we define new time-specific attention coefficient $\alpha_{i,j,n}$, relation-specific attention coefficient $\beta_{i,j,m}$, time-specific transformation matrix $\mathbf{W}_{t_n}$ and relation-specific transformation matrix $\mathbf{W}_{r_m}$ computed with the embeddings of the relation r_k and time step t_m of the corresponding link from e_j towards e_i. The attention coefficients are computed as,

$$\begin{aligned} \alpha_{i,j,n} &= \frac{\exp(\boldsymbol{\upsilon}_t^T \mathbf{t}_n)}{\sum_{e_j \in \mathcal{N}^e_{e_i}} \sum_{[t_{n'}, r_{m'}] \in \mathcal{L}_{ij}} \exp(\boldsymbol{\upsilon}_t^T \mathbf{t}_{n'})}, \\ \beta_{i,j,m} &= \frac{\exp(\boldsymbol{\upsilon}_r^T \mathbf{r}_m)}{\sum_{e_j \in \mathcal{N}^e_{e_i}} \sum_{[t_{n'}, r_{m'}] \in \mathcal{L}_{ij}} \exp(\boldsymbol{\upsilon}_r^T \mathbf{r}_{m'})}, \end{aligned} \tag{6.13}$$

where $\boldsymbol{\upsilon}_t, \boldsymbol{\upsilon}_r \in \mathbb{R}^k$ denote shared temporal and relational attention weight vectors, $\mathbf{h}_{t_n} \in \mathbb{R}^k$ denotes the embedding of the time step t_n, $\mathbf{h}_{r_m} \in \mathbb{R}^k$ denotes the embedding of the relation r_m, $\mathcal{L}_{ij}$ denotes the set of inward links from e_j to e_i and $[t_{n'}, r_{m'}] \in \mathcal{L}_{ij}$ indicates the presence of an observed quadruple $(e_j, r_{m'}, e_i, t_{n'})$.

In TEA-GNN, we find that using the standard linear transformation matrices fails to improve the performances of GNN-based EA models. Inspired by previous SEA works [30, 213], an ideal transformation operation in temporal entity alignment should satisfy two key criteria:

- **Temporal/Relational Differentiation**: According to the corresponding relation and time information, the embedding of an entity can be transformed into different temporal/relational spaces.

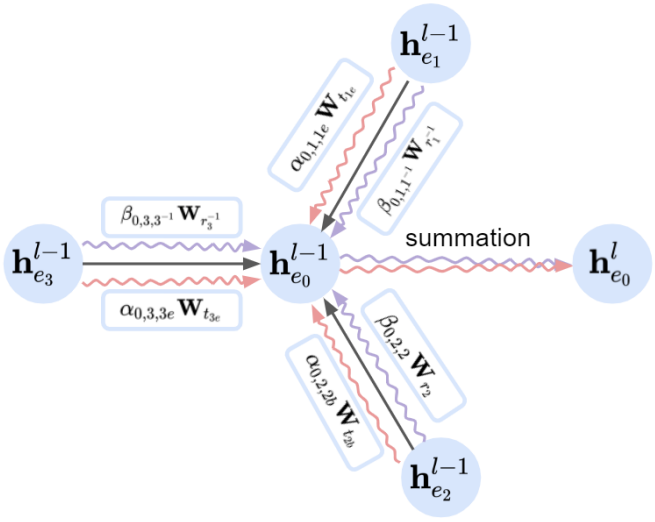


Figure 6.7: An illustration of temporal relation attention by entity e_0 on its neighborhood.

- **Dimensional Isometry**: When two entities in the same KG are transformed into the same temporal/relational space, their norms and relative distance should be retained.

To meet the key criteria, we design two new orthogonal transformation operations. For each relation embedding $\mathbf{r}_m$ and time embedding $\mathbf{t}_n$, we define the corresponding orthogonal transformation matrices $\mathbf{W}_{r_m}, \mathbf{W}_{t_n} \in \mathbb{R}^{k\times k}$ as follows,

$$\begin{aligned} \mathbf{W}_{r_m} &= \mathbf{I} - 2\widehat{\mathbf{r}}_m\widehat{\mathbf{r}}_m^\top, \\ \mathbf{W}_{t_n} &= \mathbf{I} - 2\widehat{\mathbf{t}}_n\widehat{\mathbf{t}}_n^\top, \end{aligned} \tag{6.14}$$

where $\widehat{\mathbf{r}}_m = \mathbf{r}_m/||\mathbf{r}_m||_2$ and $\widehat{\mathbf{t}}_n = \mathbf{t}_n/||\mathbf{t}_n||_2$ are the normalized time and relation embeddings. By doing this, we can easily prove that transformation matrices $\mathbf{W}_{r_m}$ and $\mathbf{W}_{t_n}$ are orthogonal. Taking $\mathbf{W}_{t_n}$ as an example, we can obtain

$$\mathbf{W}_{t_n}^\top\mathbf{W}_{t_n} = (\mathbf{I} - 2\mathbf{t}_n 2\mathbf{t}_n^\top)^\top(\mathbf{I} - 2\mathbf{t}_n 2\mathbf{t}_n^\top) = \mathbf{I} - 4\mathbf{t}_n\mathbf{t}_n^\top + 4\mathbf{t}_n\mathbf{t}_n^\top\mathbf{t}_n\mathbf{t}_n^\top = \mathbf{I}. \tag{6.15}$$

By using such orthogonal transformation matrices, the norms and the relative distances of entities can remain unchanged after transformation, i.e.,

$$\begin{aligned} &||\mathbf{h}_{e_i}\mathbf{W}_{t_n}|| = ||\mathbf{h}_{e_i}||, \\ &\mathbf{h}_{e_i}^\top\mathbf{h}_{e_i} = (\mathbf{h}_{e_i}\mathbf{W}_{t_n})^\top(\mathbf{h}_{e_i}\mathbf{W}_{t_n}). \end{aligned} \tag{6.16}$$

As shown in Figure 6.7, Red and purple arrows denote computations of time and relation attention, respectively. The entity feature update equation can be renewed by substituting Eq 6.13 and 6.14 into GAT update equation, i.e., Eq 6.3.2,

$$\mathbf{h}_{e_i}^l = \sigma\Big(\sum_{e_j\in\mathcal{N}_{e_i}^e}\sum_{[t_n,r_m]\in\mathcal{L}_{ij}}(\alpha_{i,j,n}\mathbf{W}_{t_n} + \beta_{i,j,m}\mathbf{W}_{r_m})\mathbf{h}_{e_j}^{l-1}\Big). \tag{6.17}$$

Here we employ $tanh(\cdot)$ as the activation function $\sigma(\cdot)$.

By using this temporal relational attention mechanism with the orthogonal transformation operations in GNN, TREA has fewer trainable parameters than TEA-GNN. As listed in Table 6.2, the number of trainable parameters of TEA-GNN is equal to $(|\mathcal{E}_1| + |\mathcal{E}_2| + 2|\mathcal{R}_1| + 2|\mathcal{R}_2| + |\mathcal{T}^*|)k + 4MLk$. Meanwhile, the number of trainable parameters of TREA is $(|\mathcal{E}_1| + |\mathcal{E}_2| + 2|\mathcal{R}_1| + 2|\mathcal{R}_2| + |\mathcal{T}^*|)k + 2Lk$. We further show the actual numbers of trainable parameters of TREA, TEA-GNN and several popular SEA models on different datasets in Section 6.4.3.

Margin-based Multi-class Log-loss

Following TEA-GNN, a cross-layer representation is employed to capture multi-hop neighboring information by stacking entity features from different layers. We define a global output features $\mathbf{h}_{e_i}^{out}$ of e_i as,

$$\mathbf{h}_{e_i}^{out} = [\mathbf{h}_{e_i}^0 || \mathbf{h}_{e_i}^1 || \cdots || \mathbf{h}_{e_i}^L], \tag{6.18}$$

where L denotes the number of attention layers.

To obtain multi-view representations for entities, we concatenate entity output features generated from GNN with averages of embeddings of their neighboring relations and timestamps. The multi-view entity representation of e_i is defined as follows,

$$\mathbf{h}_{e_i}^{mul} = \Big[\mathbf{h}_{e_i}^{out} || \frac{1}{|\mathcal{N}_i^r|} \sum_{r_m \in \mathcal{N}_i^r} \mathbf{r}_m || \frac{1}{|\mathcal{N}_i^t|} \sum_{t_n \in \mathcal{N}_i^t} \mathbf{t}_n \Big]. \tag{6.19}$$

The optimization objective of an embedding-based EA model is to enforce that entities of each alignment seed have close representations. During training, we use L2 distance as the metric to define the difference of representations of two entities e_i and e_j as follows,

$$d(e_i, e_j) = ||\mathbf{h}_{e_i}^{mul} - \mathbf{h}_{e_j}^{mul}||_2. \tag{6.20}$$

Most of the existing embedding-based EA methods including TEA-GNN employ a pair-wise margin ranking loss (MRL) function to minimize the distances between training entity pairs as Eq. 6.10. Since the negative samples are randomly selected and the margin ranking loss function treats each negative sample equally, the whole training process might be influenced by easy negative samples which are low-quality and uninformative, and thus suffer from slow convergence.

Lacroix et al. [116] employ a full negative sampling strategy instead of random negative sampling for training KGC model to achieve fast convergence. And a multi-class logistic loss function is used to ensure that the optimization objective mainly focuses on hard negative samples,

$$\mathcal{L} = \sum_{\xi \in \mathcal{F}_{\text{train}}} \Big[-\phi(\xi) + \log \sum_{\xi' \in \mathcal{F}'_\xi \cup \xi} \exp(\phi(\xi')) \Big], \tag{6.21}$$

where $\mathcal{F}_{\text{train}}$ denote the sets of training facts, $\mathcal{F}'_\xi$ denotes negative samples corresponding to the training fact ξ, $\phi(\xi)$ represents the score of the fact ξ.

In this work, we also adopt full negative sampling for fast convergence and a LogSumExp operation is used to find hard negative samples [202]. The margin-based logistic loss function for entity pairs

can be defined as follows,

$$\begin{aligned}\mathcal{L}_p = &\sum_{(e_i,e_j)\in\mathcal{S}} \log\Big[1+\sum_{e'_j\in\mathcal{E}_2}\exp(\gamma+d(e_i,e_j)-d(e_i,e'_j))\Big]\\ &+\sum_{(e_i,e_j)\in\mathcal{S}} \log\Big[1+\sum_{e'_i\in\mathcal{E}_1}\exp(\gamma+d(e_i,e_j)-d(e'_i,e_j))\Big].\end{aligned} \tag{6.22}$$

To retain the distance and sequence information between different timestamps, we follow TGeomE introduced in Section 4.5 to use a linear time regularizer (LTR) for time embeddings in addition to the loss function for entity pairs, based on the assumption that embeddings of two distant time steps are relatively more different than those of two adjacent time steps. The complete loss function used for our model is defined as,

$$\mathcal{L} = \mathcal{L}_p + \lambda_t \sum_{i=1}^{|\mathcal{T}*|-1} ||\mathbf{t}_{i+1} - \mathbf{t}_i - \mathbf{t}_b||_2, \tag{6.23}$$

where λ_t denotes the time regularization weight, $\mathbf{t}_b$ is the slop vector of time embeddings.

During testing, we follow TEA-GNN to adopt CSLS [209] as the distance metric to measure similarities between entity embeddings.

6.4.3 Experiments

To show the capability of TREA, we compare it with TEA-GNN and several state-of-the-art SEA models. Particularly, we conduct an ablation study of the effect of different components of TREA and a robustness study of the effect of the size of alignment seeds. We also compare the training time and numbers of trainable parameters between TREA, TEA-GNN and several popular SEA models. Importantly, we also test TREA under the inductive EA (IEA) setting to validate its ability of inductive learning.

Experimental Setting

Following TEA-GNN, we tune k in the range of $(25, 50, 75, 100)$, λ_t in the range of $(0, 0.001, 0.005, 0.01, 0.05, \ldots, 1)$ and γ in the range of $(0, 0.5, 1, 2, 3, 5, 7, 10, 15, 20)$. For a fair comparison, we use the same setup for TREA as TEA-GNN and RREA to fix $L = 2$ and $dr = 0.3$. Specially, we fix $b = 1024$, $ep = 100$ and adopt a RMSprop optimizer [208] with $lr = 0.005$ for TREA. The optimal configuration of TREA regarding d, λ_t and γ are listed as below: $k = 100$, $\lambda_t = 0.001$, $\gamma = 0$ for DICEWS-1K and DICEWS-200; $k = 50$, $\lambda_t = 0.01$, $\gamma = 0$ for YAGO-WIKI50K-5K; $k = 50$, $\lambda_t = 0.005$, $\gamma = 1$ for YAGO-WIKI50K-1K; $k = 100$, $\lambda_t = 0$, $\gamma = 0$ for YAGO-WIKI20K. We implement TREA using Tensorflow and Keras on a single GeForce GTX TITAN X GPU with 12GB RAM. We take the results of TEA-GNN and several popular SEA models reported in Section 6.3.3 for comparison.

To test TREA under the IEA setting, we resplit quadruples of DICEWS dataset according to timestamps of facts. Specifically, we use two quadruple sets $\mathcal{Q}'_1$, $\mathcal{Q}'_2$ which contain quadruples in $\mathcal{Q}_1$, $\mathcal{Q}_2$ occurring before *2014-01-01* for training. By doing this, 1,064 and 1,079 entities in $\mathcal{E}_1$ and $\mathcal{E}_2$ as well as 730 timestamps in $\mathcal{T}$ are not observed during the training process. We select 1,000 pre-aligned pairs as the training set, none of which involves unobserved entities. The unseen entities, timestamps

and quadruples only appear in the testing phase. We do not use YAGO-WIKI datasets since most facts in these datasets occur in the last few timestamps.

Main Results

Models	DICEWS-1K			DICEWS-200		
	MRR	Hits@1	Hits@10	MRR	Hits@1	Hits@10
MTransE	.150	.101	.241	.104	.067	.175
JAPE	.198	.144	.298	.138	.098	.210
AlignE	.593	.508	.751	.303	.222	.457
GCN-Align	.291	.204	.466	.231	.165	.363
MuGNN	.617	.525	.794	.412	.367	.583
MRAEA	.745	.675	.870	.564	.476	.733
HyperKA	.669	.588	.842	.474	.383	.653
RREA	.780	.722	.883	.719	.659	.824
KE-GCN	.650	.549	.827	.451	.373	.625
TEA-GNN	.911	.887	.947	.902	.876	.941
TREA	**.933**	**.914**	**.966**	**.927**	**.910**	**.960**

Table 6.7: Entity alignment results on DICEWS datasets.

Models	YAGO-WIKI50K-5K			YAGO-WIKI50K-1K		
	MRR	Hits@1	Hits@10	MRR	Hits@1	Hits@10
MTransE	.322	.242	.477	.033	.012	.067
JAPE	.345	.271	.488	.157	.101	.262
Align	.800	.756	.883	.618	.565	.714
GCN-Align	.363	.581	.512	.711	.279	.217
MuGNN	.808	.762	.890	.632	.589	.733
MRAEA	.848	.806	.913	.685	.623	.801
HyperKA	.829	.784	.900	.665	.610	.775
RREA	.868	.828	.938	.753	.696	.859
KE-GCN	.831	.780	.910	.654	.600	.761
TEA-GNN	.909	.879	.961	.775	.723	.871
TREA	**.958**	**.940**	**.989**	**.885**	**.840**	**.937**

Table 6.8: Entity alignment results on YAGO-WIKI50K datasets.

In Tables 6.7 and 6.8, we report the performances of TREA and all baseline methods on DICEWS and YAGO-WIKI50K datasets. Among all baseline models, TEA-GNN obtains the best performance

since TEA-GNN can also capture time information by utilizing a temporal relational graph neural network. Compared to TEA-GNN, TREA improves Hits@1 by 3.0%, 3.9%, 6.9% and 16.2% on four datasets respectively. Noteworthily, the performance difference between TEA-GNN and TREA on DICEWS datasets is smaller. One possible reason is that facts in two TKGs of each DICEW dataset are all from ICEWS05-15 and have a high overlap ratio of 50%, and thus both TEA-GNN and TREA are able to align entities well between these two homogeneous TKGs with their similar abilities of modeling time information. By contrast, two TKGs of each YAGO-WIKI50K dataset are extracted from different knowledge bases, and YAGO-WIKI50K datasets are much sparse than DICEWS datasets at the entity level. Thus, it is more important to force the input feature of each entity to reflect its neighborhood semantics, i.e., the embeddings of neighboring entities, relations and timestamps. Moreover, using Margin-based Multi-class Log-loss (MML) can be more helpful in finding hard negative samples among a larger amount of negative heterogeneous entity pairs. Compared to DICEWS datasets where two TKGs are homogeneous, YAGO-WIKI50K datasets are extracted from different resources and are sparser at entity level, which is closer to application scenarios in the real world. Therefore, TREA achieves more significant improvements on YAGO-WIKI50K datasets with Neighborhood Aggregation Represetation (NAR) and MML. We conduct an ablation study to empirically verify the above arguments in the later paragraphs.

Ablation Study

Although TREA and TEA-GNN use similar temporal relational attention mechanisms, TREA uses relation/time-specific orthogonal transformation operation (OTO) instead of linear transformation operation (LTO) used in GAT and achieves better performances with NAR, MML and linear time regularizer (LTR). To demonstrate the effectiveness of each design of TREA, we conduct an ablation experiment on DICEWS-200 and YAGO-WIKI50K-1K. We implement two variants of TREA excluding NAR and STR, respectively, i.e., TREA (-NAR) and TREA (-LTR). We also test a variant of TREA, TREA (w. LTO) which uses LTO instead of OTO in the temporal relational attention mechanism. Additionally, TREA trained with Margin Rank Loss (MRL), i.e., TREA (W. MRL) is considered.

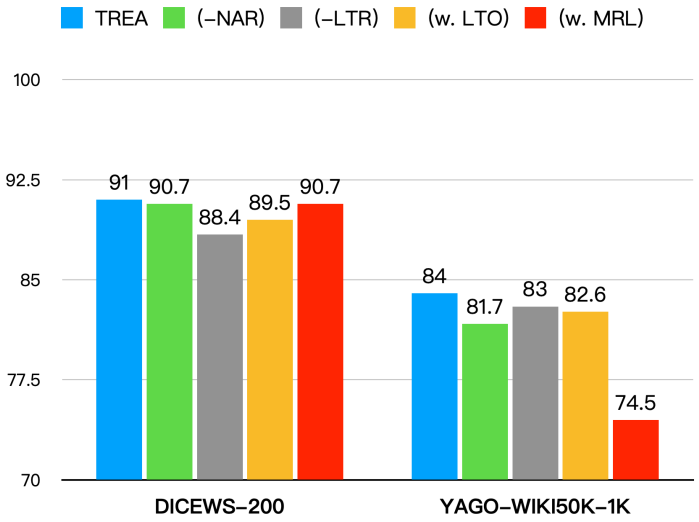


Figure 6.8: Hits@1 of TREA variants on DICEWS-200 and YAGO-WIKI50K-1K.

On YAGO-WIKI50K-1K, NAR improves TREA's performance and TREA trained with MML outperforms TREA trained with MRL as shown in Figure 6.8. However, these two technologies fail to have a positive effect on DICEWS-200. As mentioned in Section **??**, NAR is important to making the input information of entities as rich as possible when TKGs are sparse at the entity level. Thus, NAR is less necessary for dense TKGs like DICEWS. Moreover, the training process on DICEWS-200 might be mainly influenced by positive samples due to the lack of hard negative samples while hard negative samples play an important role in the training process on YAGO-WIKI50K-1K. Thus, MML scarcely changes the performance of TREA on DICEWS-200, but have significant effects on YAGO-WIKI50K-1K. Specifically, e_1 and e_2 can be regarded as hard negative samples of each other if there exist multiple observed similar-looking quadruple pairs, shaped like (e_s, r, e_1, t) and (e_s, r, e_2, t), which have the same subjects, relations and timestamps but involve these two entities respectively. For each observed quadruple (e_s, r, e_o, t) in YAGO-WIKI50K-1K, there are on average 5.22 similar-looking quadruples shaped like (e_s, r, e'_o, t). By contrast, this number drops to 0.15 in DICEWS-200. Thus, the average number of hard negative samples of each entity in the training data of DICEWS-200 is much lower than YAGO-WIKI50K-1K. Different from NAR and MML, STR helps improve the performance of TREA on DICEWS200 because the distribution of time data in DICEWS200 is dense and uniform. Meanwhile, some timestamps are missed in YAGO-WIKI50K and most timestamps are concentrated in the last few years with a long tail of other timestamps, which can not be well modeled by a sequential time model. There are obvious declines in the performances of TREA on both datasets after using LTO instead of OTO, which supports our intuition that the transformation operation in TEA should satisfy two key criteria, i.e., temporal/relational differentiation and dimensional isometry.

Robustness Study

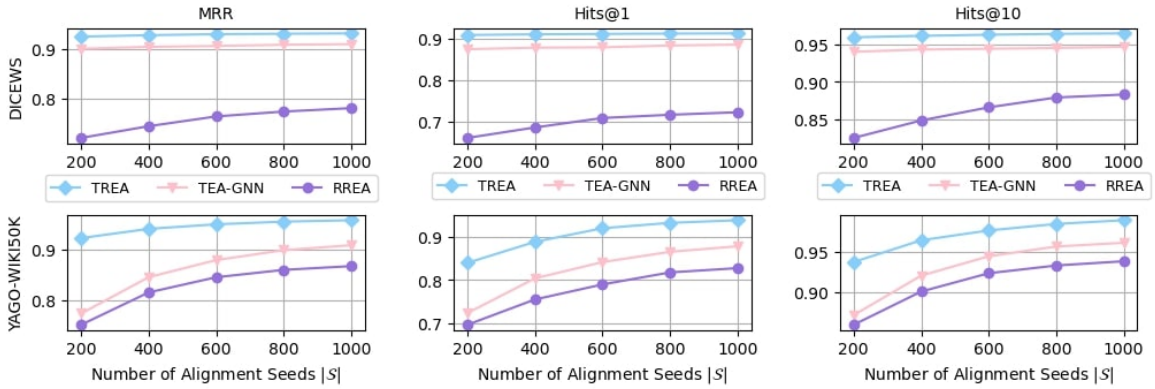


Figure 6.9: Entity alignment results w.r.t. different sizes of alignment seeds.

It is costly to annotate pre-aligned entity pairs manually, especially for the large-scale KGs. Thus, it is essential for an EA method to maintain an effective performance with a small proportion of pre-aligned entities. To verify the robustness of TREA, we test TREA and the two best performing baseline models, i.e., TEA-GNN and RREA, with $|\mathcal{S}|$ varying from 200 to 1,000 with step size of 200 and $|\mathcal{S}|$ varying from 1,000 to 5,000 with step size of 1,000 on DICEWS and YAGO-WIKI50K, respectively. As shown in Figure 6.9, TREA is not only superior to TEA-GNN and RREA in all seed

sizes, but also has a more gradual slope curve. This demonstrates that our model is less dependent on additional training data due to its robust model structure and learning effectiveness, and it is promising to have good capability of generalization.

Efficiency Study

Table 6.9 lists the numbers of trainable parameters of our method and all baseline models, and their overall time costs on DICEWS and YAGO-WIKI50K dataset, including data loading, pre-processing, training, and evaluating. As shown in Table 6.9, the training efficiency of TREA exceeds most baseline models. Only GCN-Align has less training time than TREA on both datasets, since it uses a small negative sampling rate and it is evaluated only once during the training process. While most GNN-based EA models trained with MRL need thousands of training epochs, TREA can converge within 100 epochs by adopting MML. With the inclusion of time embeddings, TREA does not excessively increase the number of trainable parameters on DICEWS datasets, compared to baseline models. On YAGO-WIKI50K datasets, TREA has even fewer free parameters than most baseline models since lower-dimensional embeddings ($k = 50$) are used. In general, the high efficiency of TREA makes the time-aware EA on large-scale KGs possible.

Models	**DICEWS**		**YAGO-WIKI50K**	
	Time Cost	Parameter Number	Time Cost	Parameter Number
MTransE	284	1.95M	1,624	9.89M
JAPE	953	1.95M	4,083	9.89M
AlignE	9,797	1.95M	5,384	9.89M
GCN-Align	39	1.92M	613	9.91M
MuGNN	2,173	1.96M	22,457	9.90M
MRAEA	2,647	2.01M	14,338	7.42M
HyperKA	6,389	1.97M	40,951	9.91M
RREA	1,538	2.00M	6,487	4.95M
KE-GCN	1,704	2.18M	9,510	11.61M
TEA-GNN	4,410	2.40M	9,350	4.95M
TREA	128	2.41M	2,655	4.96M
(w. MRL)	1,948	2.41M	6,327	4.96M

Table 6.9: Time costs (seconds) and numbers of trainable parameters of EA methods.

Inductive Entity Alignment

In the real world, most KGs are dynamic with new emerging entities and timestamps. However, the existing EA methods hold a closed-world assumption that KGs are fixed and are unable to model representations of new emerging entities and timestamps. Taking RREA and TEA-GNN as examples, neither of them can model input features of unseen entities since their embeddings are unobtainable or untrained. Moreover, RREA does not consider time information and TEA-GNN can not model unseen timestamps. By contrast, TREA can model the input features with its known neighborhood

information by using Eq. 6.12, i.e., NAR. In Eq. 6.23, we force the change of timestamp embeddings to satisfy a linear equation over time by using LTR. Thus, we can roughly estimate embeddings of future timestamps with embeddings of observed timestamps and the temporal slope.

We compare TREA with TEA-GNN and RREA on DICEWS under the IEA setting. To enable TEA-GNN and RREA to model unseen entities, we drop links flowing from unobserved entities towards observed entities and remove unseen entities' input features from their final representations in the testing phase, which causes the information loss inevitably. We use the last seen timestamps to represent future timestamps for TEA-GNN since the embedding of unknown time information t_0 is not updated during the training process. 1,064 and 1,079 entities as well as 730 timestamps are unseen before 2014-01-01 in DICEWS. We select 1,000 entity pairs only involving observed entities as training set and the rest are testing set. Among testing entity pairs, 1,027 entity pairs involve unseen entities, called **unobserved entity pairs**, and others are called **observed entity pairs**. Table 6.10 shows that TREA significantly outperforms TEA-GNN and TREA under the open-world setting, especially on unobserved entities. These experimental results support our argument that TREA can effectively perform EA tasks between inductive KGs (IKGs).

Models	**Unobserved Entity Pairs**			**Observed Entity Pairs**		
	MRR	Hits@1	Hits@10	MRR	Hits@1	Hits@10
RREA	.253	.075	.483	.407	.361	.580
TEA-GNN	.324	.155	.590	.513	.392	.748
TREA	.479	.342	.748	.643	.549	.825

Table 6.10: Inductive entity alignment results on DICEWS.

In Table 6.10, we show some examples that TEA-GNN gives wrong predictions and TREA predicts correctly for unobserved entity pairs under IEA setting. Without NAR and LTR, TEA-GNN more frequently gives wrong predictions for unobserved entities since it can only recognize straightforward neighborhood information, but can hardly learn the semantics of new emerging entities.

Unobserved Entity Alignment	TREA	TEA-GNN
Yoon Sang-jick $\in \mathcal{E}_1$	**Yoon Sang-jick** $\in \mathcal{E}_2$	Andrew Robb $\in \mathcal{E}_2$
Edwin Lacierda $\in \mathcal{E}_1$	**Edwin Lacierda** $\in \mathcal{E}_2$	Wen Jiabao $\in \mathcal{E}_2$
Dunya Maumoon $\in \mathcal{E}_1$	**Dunya Maumoon** $\in \mathcal{E}_2$	Maumoon Abdul Gayoom $\in \mathcal{E}_2$

Table 6.11: Examples of different alignment predictions between TREA and TEA-GNN under IEA setting.

6.4.4 Conclusion

Embedding models have been successful for entity alignment between KGs, but lack consideration of time information and the dynamism of open-world KGs. To address these challenges, we present an inductive GNN-based model for TEA, which uses an efficient attention mechanism to learn both time and relation information with relation and time-specific orthogonal transformation operations. In addition, a neighborhood aggregation representation is used to incorporate neighborhood information

into entitie's input features and is able to represent observed entities and new emerging entities. A margin-based multi-class log-loss is used for fast parameter optimization. A linear time regularizer helps to model unobserved time representations. Experimental results on two TKG datasets show that TREA achieves higher performances than TEA-GNN and the state-of-the-art EA methods on both tasks of TEA and IEA.

6.5 Conclusion

The availability of recent TKGs creates the need for performing entity alignment between TKGs. In this chapter, we first point out the limitations of existing KGE-based static EA (SEA) methods in different aspects, i.e., 1) temporal unawareness; 2) reliance on alignment seeds; 3) training efficiency; 4) inductive learning inability. To overcome these limitations of SEA methods, especially on temporal unawareness, an intuitive idea is to incorporate time information in TKGs into the existing SEA methods. In Section 6.1, we give the problem definition of the temporal entity alignment (TEA) task and introduce the relevant evaluation metrics. In Section 6.2, we present three new TKG datasets extracted from ICEWS, YAGO and Wikidata as references for evaluating SEA and TEA methods. Then, we present two novel TEA models to address the limitations of the existing SEA methods mentioned at the beginning of this chapter. In Section 6.3, we introduce the first TEA model, TEA-GNN, in which a self-attention mechanism is used for GNN to specify different weights to different neighboring nodes of each entity with the corresponding link features, i.e., embeddings of the relevant relations and timestamps. TEA-GNN considers time information for EA between TKGs and uses an efficient temporal graph self-attention mechanism by taking timestamps as attentive properties of links between entities and using the time embedding technique. Overall, TEA-GNN addresses the limitations 1-3) of previous SEA models and most temporal GNN models, and achieves state-of-the-art TEA results on three TKG benchmark datasets. In Section 6.4, we introduce the second TEA model, TREA, which is not only more effective for TEA but also has the ability of inductive graph representation, i.e., representing new emerging entities and timestamps. Besides, TREA uses a more effective temporal relational attention mechanism which is based on orthogonal transformation operations and a multi-class log-loss function is adopted for efficient training. Building on top of TEA-GNN, TREA improves the abilities to address the limitations 1-3) and showes its effectiveness on both tasks of transductive EA and inductive EA between TKGs.

CHAPTER 7

Conclusion

KGE has been proven to be an efficient tool for fast reasoning over KGs. KGE-based methods have achieved great success in different learning tasks over KGs, e.g., KG completion, multi-hop logical reasoning over KGs and entity alignment between KGs. The recent availability of TKGs has created a need for new KGE methods that can reason over time. However, most existing KGE works only focus on SKGs and ignore time information in TKGs, leaving much room for improvement. In this thesis, we follow the research objective of incorporating time information into KGE models for different learning tasks over TKGs. We defined the research problem in Chapter 1 and discussed the significant challenges to overcome in order to achieve the research objective. To achieve our research objective and overcome the research challenges, we have broken down the research problem into three research questions in regard with three different learning tasks over TKGs, i.e., temporal KG completion (TKGC), multi-hop temporal KG reasoning (MTKGR), and temporal entity alignment (TEA). In Chapter 2, we discussed all the fundamentals and necessary background concepts required for this thesis, including the preliminary concepts in the areas of Knowledge Graphs, Distributed Representation Learning and Temporal Learning. Chapter 3 covers state-of-the-art research works in various domains which are strongly related to the research objective and research questions. In the subsequent three chapters of the thesis, we presented different TKGE models and frameworks to tackle our research questions. In Chapter 4, we mainly discussed the limitations of the previous TKGE models on the TKGC task and presented three TKGE models to address the relevant issues. Chapter 5 defined a new learning task over TKGs, namely, MTKGR. In Chapter 5, the first temporal query embedding (TQE) framework was presented to specifically handle this new learning task and new TKG datasets were generated for evaluating the performances of our TQE method on MTKGR. In Chapter 6, we defined another new learning task for TKGE models, namely, TEA. We presented two novel GNN-based TEA models and three TKG datasets as new references for evaluating SEA and TEA models.

In this chapter, we provide a summary of our research questions and contributions towards them, elaborating on the main findings that validate our research questions. In the end, we list a few limitations of this research that have not been covered in the scope of the thesis and the future directions for the relevant research community.

7.1 Research Contributions

In this section, we summarize the contributions provided in the thesis from Chapter 4, 5, 6. Individually, we first state the research questions and then the contributions towards them.

Research Question 1 (RQ1)

How can we effectively encode time information into KGE models to enhance temporal knowledge graph completion?

In Chapter 4, we answered this question by exploring new TKGE models for the task of TKGC. Obviously, it is necessary to consider time information when answering a temporal query like (*?*, *president of*, *USA*, *2014*) for a TKGC task. Many studies have shown that TKGE models perform significantly better than the SKGE models on TKGC tasks due to the inclusion of time information. However, the previous TKGE models still underperform on TKGC tasks due to their own limitations. At the beginning of Chapter 4, we first summarized the limitations of previous TKGE models on TKGC tasks, i.e., 1) temporal uncertainty; 2) temporal interpretability; 3) time representation; 4) time distribution; 5) time granularity; 6) model expressiveness; 7) temporal regularization. To overcome the above limitations, we presented three new TKGE models specifically for TKGC tasks, i.e., ATiSE, TeRo and TGeomE, in Sections 4.3, 4.4 and 4.5, respectively. Previous TKGE models ignored the uncertainty during the temporal evolution of KG representations and lacked interpretability. ATiSE addressed limitations 1-2) by fitting the evolution process of each entity or relation as a multi-dimensional additive time series which composes of a trend component, a seasonal component and a random component. We also exploited time interval discretization and different time-splitting methods for processing time data in order to address limitations 3-4). We conducted evaluation experiments for ATiSE and showed that ATiSE outperformed prior TKGE models on TKGC and adapted well to various TKGs which have different time representation forms and time distributions, and the additive time series decomposition technique provided good interpretability for the parameter learning of ATiSE. To make time interval representation more efficient, TeRo uses dual relation embeddings to handle the beginning point and end point of a fact. Some subsequent works [33, 179] on TKGC followed this approach. TeRo also addressed limitations 5-6) by studying the effect of time granularity and using complex embeddings, respectively. We proved that TeRo could capture several key relation patterns which are not well modeled by prior TKGE models including ATiSE. Thanks to better model expressiveness, TeRo achieved better results than ATiSE. Although TeRo has a good model expressiveness, it is not fully expressive for TKGC. By contrast, TGeomE performs a 4-order tensor decomposition for TKGC using multi-vector embeddings from geometric algebra. We proved that TGeomE is fully expressive and subsumes several existing TKGE models. Moreover, a part of TKGE models like TeRo use the time embedding technique without the consideration of physical characteristics of time data. Thus, we introduced a linear temporal regularizer for TGeomE to address the limitation 7). Experimental results showed that TGeomE achieved new state-of-the-art on TKGC.

In conclusion, we presented three new TKGE models for the task of TKGC, which overcame the limitations of prior TKGE models and achieved the state-of-the-art at the time of their publication. We have made the implementations and resources of our presented models open and accessible to the relevant community. Currently, they have been widely used as strong baselines in the TKGE

community and inspired some follow-up works at different level.

Research Question 2 (RQ2)

How can we model temporal logical operations in KGE models to perform multi-hop reasoning over temporal knowledge graphs?

Chapter 5 addressed this research question by generating new temporal query datasets from three common TKGC benchmarks and proposing a temporal query embedding (TQE) framework. The existing QE methods only focus on multi-hop KG reasoning (MKGR) over SKGs and disregard time information, while TKGC models can only perform single-hop queries. To fill this gap, an intuitive idea is to design a temporal QE framework that can handle both FOL operations and temporal logical operations for multi-hop reasoning over TKGs. The key points of RQ2 are how to design such a TQE framework and how to verify its effectiveness. Therefore, we first formally define the temporal logical query and the task of multi-hop TKG reasoning (MTKGR), and introduced new temporal logical operations for the MTKGR task in addition to common FOL operations. Based on FOL operations and temporal logical operations, we defined 34 kinds of diverse query structures combined from 10 types of basic logic set functions. For each query structure, we instantiated numerous queries from TKG quadruples to form new query datasets. Importantly, we presented the first TQE framework, namely, Temporal Feature-Logic Embedding framework (TFLEX). In TFLEX, embeddings of objects (entity, query, timestamp) are divided into two parts, the entity part, and the timestamp part. Each part is further divided into feature components and logic components. On the one hand, the computation of the logic components follows vector logic, which enables our framework to handle all FOL operations. On the other hand, feature components are mingled and transformed under the guidance of logic components, thereby integrating logical information into the feature. Moreover, we extended vector logic to support extra temporal operations (**After**, **Before** and **Between**) to handle temporal operations in the queries.

The contributions of our work are summarized as follows:

- For the first time, the definition of the MTKGR task was given.
- We presented the first TQE framework, namely Temporal Feature-Logic Embedding framework (TFLEX), which supports all FOL operations and extra temporal operations, as well as entity queries and timestamp queries.
- We generated three new temporal query datasets for the MTKGR task. Experiments on three generated datasets demonstrate the efficacy of the presented TQE framework.

Research Question 3 (RQ3)

Can the incorporation of time information be helpful for the performances of KGE models on the task of entity alignment between temporal knowledge graphs?

This research question was investigated in Chapter 6. GNN-based KGE methods have achieved great success on the task of EA between SKGs. However, EA between TKGs has not been explored and existing TKGE models designed for TKGC are not well compatible with the EA setting. Intuitively,

the incorporation of time information could be helpful for TEA tasks. In Chapter 6, we confirmed this conjecture through adequate experiments.

To investigate this research question, we first formally defined the task of TEA, and generated three TKG datasets, namely, DICEWS, YAGO-WIKI50K, YAGO-WIKI20K from ICEWS, YAGO and Wikidata, as new references for evaluating SEA models and TEA models. The above three TEA datasets contain around 10,000, 50,000, 20,000 entity pairs and 600,000, 500,000, 200,000 temporal facts, respectively.

Then, we presented the first GNN-based TKGE model which is specifically dedicated to the task of TEA, namely, TEA-GNN. Most temporal GNN models decompose a temporal graph into a sequence of static snapshots and use combinations of GNN models and temporal dynamic models, e.g., LSTM, which suffer from the sparsity of snapshot graphs and excessively long training time. By contrast, TEA-GNN embeds entities, relations and timestamps into a vector space and treats timestamps as attentive properties of edges between entities with a temporal graph self-attention mechanism. This approach has been proven to be time-efficient in our case and could potentially be used for non-relational temporal graph representation learning. Due to the inclusion of time information, TEA-GNN significantly outperformed the existing state-of-the-art SEA models and TKGC models on all three TEA datasets and showed great robustness against the size of labeled data.

Taking this idea one step further, we presented the second TEA model using temporal relational attention, i.e., TREA. The prior EA models including TEA-GNN lack inductive learning ability, i.e., unable to efficiently generalize to unseen entities and timestamps. In contrast, TREA is an inductive EA model that leverages entity neighborhood information and linear temporal regularization to efficiently generate representations on previously unseen data. TREA also adopts a new loss function, which significantly increased the training efficiency of EA models. Experimental results showed that TREA outperformed TEA-GNN and other baseline models on both tasks of transductive EA and inductive EA between TKGs, and had better scalability.

To sum up, three new TKG datasets and two novel GNN-based TKGE models were presented in Chapter 6 to investigate RQ3. Through adequate experiments, we showed that the incorporation of time information remarkably improved the performances of KGE models on TEA tasks. We made all relevant datasets and source codes available online for the reproducibility and convenience of the possible follow-up works.

7.2 Limitations and Future Directions

Despite the overall achieved research objective, there are a few limitations of this research that have not been covered in the scope of the thesis. We list the following limitations:

- Although various TKGC methods have been presented in Chapter 4, there is still a lot of room for improvement in the prediction accuracy, especially on YAGO and Wikidata datasets. One possible reason is that our TKGC models learn TKGE only on quadruples observed in TKGs. A lot of research works on SKGC tried to capture extra information from triples, e.g., relation paths [152], graph contexts [214] and relational rules [215], and integrate such information into SKGE. Our TKGC models currently have not considered such extra information. Moreover, our TKGC models can only perform transductive prediction, i.e., unable to predict new emerging entities in a dynamic TKG.

- In Chapter 5, we defined three extra temporal operators (**Before**, **After** and **Between**) for multi-hop logical reasoning over TKGs. However, there exists more temporal operators in the real world. For example, Allen et al. [216] defined 13 type of temporal relations of two intervals, including before/after, during/contains, overlaps/overlapped-by, meets/met-by, starts/started-by, finishes/finished-by and equal. It is essential to promote these temporal operators to TKGs in real application scenarios.
- It is very common to exploit multi-modal knowledge to help with entity alignment between KGs, e.g., entity names, textual description, attribute information and entity images. Our TEA models presented in Chapter 6 do not integrate multi-modal knowledge. Besides, the TEA datasets we used in Chapter 6 only include tens of thousands of entities. However, the real-world TKGs might have millions of entities.
- All TKGE models presented in this thesis were trained and evaluated on reliable subsets of human-curated TKGs. The robustness of our TKGE models against noisy data (e.g., incorrect quadruples involving wrong entities or time stamps, and wrongly aligned entity pairs) was not explored.

Based on our findings, and the contributions made in this thesis, we now present some of the future directions for the research community:

- Discovering temporal rules in temporal databases is important, as it can provide a better understanding of data. However, how to mine temporal rules between relations over TKGs is still a big challenge that has not been solved. It would be quite interesting to investigate the possibility of mining temporal rules from TKGs and further inject temporal rules into TKGE models to improve the performances of the existing TKGC methods.
- The proposed TQE framework TFLEX considered two main temporal operators and classified semantic features of objects in TKGs (entities, relations, timestamps) into entity feature component and time feature component. Perhaps one could explore further in this direction and extend the approach to more temporal operators and more complex temporal queries. TFLEX could also possibly be improved by unifying the entity feature component and the time feature component in the embedding.
- To improve entity alignment performances of existing TEA models over TKGs, a future direction would be integrating other types of information beyond quadruples. And it would be necessary to establish larger-scale TKG datasets as new references for evaluating TEA and SEA models.
- In real-world application scenarios, TKGs are extracted from texts, which are often incomplete and contain errors. Therefore, it would also be interesting to study the robustness of the existing TKGE models against noisy data in TKGs and seek ways of improving their robustness.
- Most of the exsiting TKGE models focus on the fundamental learning tasks over TKGs. It would be significantly valuable to apply TKGE models to the relevant downstream tasks, e.g., temporal question answering and temporal recommendation system.

7.3 Closing Remarks

In the real world, knowledge evolves over time. Recent years, more and more KGs attach time information to the triple facts, which creates the need for new representation learning methods that can be used for reasoning over such temporal KGs.

During this thesis, we advanced the state-of-the-art in TKG representation learning on several fronts by setting up TKG benchmarks and developing TKGE models dedicated to different learning tasks over TKGs. More specifically, we proposed:

- three TKGE models specifically developed for TKGC tasks that overcome the limitations of prior TKGC methods,
- the first temporal QE framework that can handle both FOL operations and temporal logical operations in temporal queries over TKGs,
- three temporal query datasets generated from common TKGC benchmarks,
- two GNN-based TKGE models which utilize time information for entity alignment between TKGs,
- three TKG datasets as new references for evaluating SEA and TEA methods.

Future research work can build upon the datasets and models developed as contributions presented during this thesis. These contributions could provide a foundation for temporal knowledge graph representation learning and its relevant applications. Furthermore, contributions of this thesis have already made an impact on the community of KG, as all publications, resources and data related to this thesis have been available online and several other projects are working on the TKG datasets and models published within the thesis duration.

Bibliography

[1] S. Ji, S. Pan, E. Cambria, P. Marttinen and P. S. Yu,
A survey on knowledge graphs: Representation, acquisition and applications,
arXiv preprint arXiv:2002.00388 (2020) (cit. on p. 1).

[2] L. Ehrlinger and W. Wöß, *Towards a definition of knowledge graphs.*,
SEMANTiCS (Posters, Demos, SuCCESS) **48** (2016) 2 (cit. on p. 1).

[3] K. Bollacker, C. Evans, P. Paritosh, T. Sturge and J. Taylor,
"Freebase: a collaboratively created graph database for structuring human knowledge",
Proceedings of the 2008 ACM SIGMOD international conference on Management of data,
2008 1247 (cit. on pp. 2, 18).

[4] J. Frey, M. Hofer, D. Obraczka, J. Lehmann and S. Hellmann,
"DBpedia FlexiFusion the best of Wikipedia¿ Wikidata¿ your data",
International Semantic Web Conference, Springer, 2019 96 (cit. on pp. 2, 18).

[5] D. Liu et al., "News Graph: An Enhanced Knowledge Graph for News Recommendation.",
KaRS@ CIKM, 2019 1 (cit. on p. 2).

[6] Y. Bengio, R. Ducharme and P. Vincent, *A neural probabilistic language model*,
Advances in Neural Information Processing Systems **13** (2000) (cit. on pp. 3, 24).

[7] T. Mikolov, K. Chen, G. Corrado and J. Dean,
Efficient estimation of word representations in vector space,
arXiv preprint arXiv:1301.3781 (2013) (cit. on pp. 3, 24).

[8] F. M. Suchanek, G. Kasneci and G. Weikum, "Yago: a core of semantic knowledge",
Proceedings of the 16th international conference on World Wide Web, ACM, 2007 697
(cit. on pp. 3, 18, 50, 116).

[9] F. Erxleben, M. Günther, M. Krötzsch, J. Mendez and D. Vrandečić,
"Introducing Wikidata to the linked data web", *International Semantic Web Conference*,
Springer, 2014 50 (cit. on pp. 3, 50, 116).

[10] J. Lautenschlager, S. Shellman and M. Ward, *ICEWS Event Aggregations*, 2015,
URL: https://doi.org/10.7910/DVN/28117 (cit. on pp. 3, 18, 50, 116).

[11] K. Leetaru and P. A. Schrodt, "Gdelt: Global data on events, location, and tone, 1979–2012",
ISA annual convention, vol. 2, 4, Citeseer, 2013 1 (cit. on pp. 3, 18, 50).

[12] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston and O. Yakhnenko,
"Translating embeddings for modeling multi-relational data",
Advances in Neural Information Processing Systems, 2013 2787
(cit. on pp. 3, 36, 50, 56, 58, 62–64, 69, 72).

[13] B. Yang, W.-t. Yih, X. He, J. Gao and L. Deng,
"Embedding entities and relations for learning and inference in knowledge bases", *ICLR*, 2015
(cit. on pp. 3, 36, 58, 63, 64, 69).

[14] T. Trouillon, J. Welbl, S. Riedel, É. Gaussier and G. Bouchard,
Complex Embeddings for Simple Link Prediction, arXiv preprint arXiv:1606.06357 (2016)
(cit. on pp. 3, 36, 56, 58, 74).

[15] W. Hamilton, P. Bajaj, M. Zitnik, D. Jurafsky and J. Leskovec,
Embedding logical queries on knowledge graphs,
Advances in neural information processing systems **31** (2018) (cit. on pp. 4, 39, 41).

[16] N. Choudhary, N. Rao, S. Katariya, K. Subbian and C. K. Reddy,
Probabilistic Entity Representation Model for Chain Reasoning over Knowledge Graphs,
Advances in Neural Information Processing Systems (2021) (cit. on pp. 4, 41, 89).

[17] H. Ren, W. Hu and J. Leskovec,
Query2box: Reasoning over knowledge graphs in vector space using box embeddings,
International Conference on Learning Representation (2020) (cit. on pp. 4, 39–41, 89).

[18] B. Kotnis, C. Lawrence and M. Niepert,
Answering complex queries in knowledge graphs with bidirectional sequence encoders,
Thirty-Fifth AAAI Conference on Artificial Intelligence (2021) (cit. on p. 4).

[19] H. Ren and J. Leskovec,
Beta embeddings for multi-hop logical reasoning in knowledge graphs,
Advances in Neural Information Processing Systems **33** (2020) 19716
(cit. on pp. 4, 39–41, 89).

[20] X. Chen, Z. Hu and Y. Sun,
Fuzzy Logic based Logical Query Answering on Knowledge Graph,
International Conference on Machine Learning (2021) (cit. on p. 4).

[21] E. Arakelyan, D. Daza, P. Minervini and M. Cochez,
"Complex Query Answering with Neural Link Predictors",
International Conference on Learning Representations, 2021,
URL: https://openreview.net/forum?id=Mos9F9kDwkz (cit. on pp. 4, 41, 89).

[22] H. Sun, A. Arnold, T. Bedrax Weiss, F. Pereira and W. W. Cohen,
Faithful embeddings for knowledge base queries,
Advances in Neural Information Processing Systems **33** (2020) 22505 (cit. on pp. 4, 41).

[23] Z. Zhang, J. Wang, J. Chen, S. Ji and F. Wu,
Cone: Cone embeddings for multi-hop reasoning over knowledge graphs,
Advances in Neural Information Processing Systems **34** (2021) (cit. on pp. 4, 39, 41, 89).

[24] L. Liu, B. Du, H. Ji, C. Zhai and H. Tong,
"Neural-Answering Logical Queries on Knowledge Graphs",
Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining,
2021 1087 (cit. on pp. 4, 41).

[25] M. Chen, Y. Tian, M. Yang and C. Zaniolo,
"Multilingual knowledge graph embeddings for cross-lingual knowledge alignment", *IJCAI*, 2017 (cit. on pp. 4, 42, 113, 122).

[26] Z. Sun, W. Hu and C. Li,
"Cross-lingual entity alignment via joint attribute-preserving embedding",
International Semantic Web Conference, Springer, 2017 628 (cit. on pp. 4, 42, 122).

[27] Z. Wang, Q. Lv, X. Lan and Y. Zhang,
"Cross-lingual knowledge graph alignment via graph convolutional networks",
Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, 2018 349 (cit. on pp. 4, 7, 43, 113, 120, 122).

[28] Y. Cao et al., "Multi-Channel Graph Neural Network for Entity Alignment", *ACL*, 2019 (cit. on pp. 4, 7, 43, 122).

[29] X. Mao, W. Wang, H. Xu, M. Lan and Y. Wu, "MRAEA: an efficient and robust entity alignment approach for cross-lingual knowledge graph",
Proceedings of the 13th International Conference on Web Search and Data Mining, 2020 420 (cit. on pp. 4, 43, 113, 120–122).

[30] X. Mao, W. Wang, H. Xu, Y. Wu and M. Lan, "Relational Reflection Entity Alignment",
Proceedings of the 29th ACM International Conference on Information & Knowledge Management, 2020 1095 (cit. on pp. 4, 43, 121, 122, 129).

[31] R. Trivedi, H. Dai, Y. Wang and L. Song,
"Know-evolve: Deep temporal reasoning for dynamic knowledge graphs",
international conference on machine learning, PMLR, 2017 3462 (cit. on p. 4).

[32] C. Xu, M. Nayyeri, F. Alkhoury, H. Yazdi and J. Lehmann,
"Temporal knowledge graph completion based on time series gaussian embedding",
International Semantic Web Conference, Springer, 2020 654
(cit. on pp. 5, 48, 51, 52, 63, 64, 69, 70, 72, 82, 109).

[33] T. Lacroix, G. Obozinski and N. Usunier,
"Tensor Decompositions for temporal knowledge base completion", *ICLR*, 2020
(cit. on pp. 5, 38, 74, 79, 81–84, 89, 116, 140).

[34] C. Xu, Y.-Y. Chen, M. Nayyeri and J. Lehmann, "Temporal Knowledge Graph Completion using a Linear Temporal Regularizer and Multivector Embeddings",
Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, 2021 2569 (cit. on pp. 5, 48).

[35] F. Manessi, A. Rozza and M. Manzo, *Dynamic graph convolutional networks*,
Pattern Recognition **97** (2020) 107000 (cit. on pp. 7, 114).

[36] J. Chen, X. Xu, Y. Wu and H. Zheng,
Gc-lstm: Graph convolution embedded lstm for dynamic link prediction,
arXiv preprint arXiv:1812.04206 (2018) (cit. on pp. 7, 114).

[37] J. F. Allen, *Towards a general theory of action and time*, Artificial intelligence **23** (1984) 123 (cit. on p. 8).

[38] A. Sheth and K. Thirunarayan, *Semantics empowered web 3.0: managing enterprise, social, sensor, and cloud-based data and services for advanced applications*, Synthesis Lectures on Data Management **4** (2012) 1 (cit. on p. 15).

[39] P. Jackson, *Introduction to expert systems*, (1986) (cit. on p. 17).

[40] E. A. Feigenbaum, *Knowledge Engineering: The Applied Side of Artificial Intelligence.*, tech. rep., STANFORD UNIV CA DEPT OF COMPUTER SCIENCE, 1980 (cit. on p. 17).

[41] C. Fellbaum, "WordNet", *Theory and applications of ontology: computer applications*, Springer, 2010 231 (cit. on p. 17).

[42] D. B. Lenat, *CYC: A large-scale investment in knowledge infrastructure*, Communications of the ACM **38** (1995) 33 (cit. on p. 17).

[43] T. Berners-Lee, J. Hendler and O. Lassila, *The semantic web*, Scientific american **284** (2001) 34 (cit. on p. 17).

[44] X. Dong et al., "Knowledge vault: A web-scale approach to probabilistic knowledge fusion", *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2014 601 (cit. on p. 18).

[45] Z. C. Khan and T. Mashiane, "An analysis of facebook's graph search", *2014 Information Security for South Africa*, IEEE, 2014 1 (cit. on p. 18).

[46] Z. Wang, S. Yan, H. Wang and X. Huang, *An overview of microsoft deep qa system on stanford webquestions benchmark*, Microsoft Corporation. Microsoft Research Technical Report MSR-TR-2014-121. Verfügbar unter http://research. microsoft. com/pubs/228312/Microsoft% 20Deep% 20QA. pdf. Zugegriffen **18** (2014) 2014 (cit. on p. 18).

[47] O. Etzioni et al., "Web-scale information extraction in knowitall: (preliminary results)", *Proceedings of the 13th international conference on World Wide Web*, 2004 100 (cit. on p. 18).

[48] S. P. Ponzetto and M. Strube, "WikiTaxonomy: A large scale knowledge resource", *ECAI 2008*, IOS Press, 2008 751 (cit. on p. 18).

[49] R. Navigli and S. P. Ponzetto, "BabelNet: Building a very large multilingual semantic network", *Proceedings of the 48th annual meeting of the association for computational linguistics*, 2010 216 (cit. on p. 18).

[50] R. Speer, J. Chin and C. Havasi, "Conceptnet 5.5: An open multilingual graph of general knowledge", *Thirty-first AAAI conference on artificial intelligence*, 2017 (cit. on p. 18).

[51] F. Niu, C. Zhang, C. Ré and J. W. Shavlik, *DeepDive: Web-scale Knowledge-base Construction using Statistical Learning and Inference.*, VLDS **12** (2012) 25 (cit. on p. 18).

[52] A. Carlson, J. Betteridge, R. C. Wang, E. R. Hruschka Jr and T. M. Mitchell, "Coupled semi-supervised learning for information extraction", *Proceedings of the third ACM international conference on Web search and data mining*, 2010 101 (cit. on p. 18).

[53] W. Wu, H. Li, H. Wang and K. Q. Zhu,
"Probase: A probabilistic taxonomy for text understanding",
Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data,
2012 481 (cit. on p. 18).

[54] D. Vrandečić and M. Krötzsch, *Wikidata: a free collaborative knowledgebase*,
Communications of the ACM **57** (2014) 78 (cit. on p. 18).

[55] Z. Wang et al., "XLore: A Large-scale English-Chinese Bilingual Knowledge Graph.",
International semantic web conference (Posters & Demos), vol. 1035, 2013 121
(cit. on p. 18).

[56] X. Niu et al., "Zhishi. me-weaving chinese linking open data",
International Semantic Web Conference, Springer, 2011 205 (cit. on p. 18).

[57] B. Xu et al., "CN-DBpedia: A never-ending Chinese knowledge extraction system",
International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems, Springer, 2017 428 (cit. on p. 18).

[58] J. J. Miller, "Graph database applications and concepts with Neo4j", *Proceedings of the southern association for information systems conference, Atlanta, GA, USA*, vol. 2324, 36,
2013 (cit. on p. 18).

[59] R. Pointer, N. Kallen, E. Ceaser and J. Kalucki, *Introducing flockdb*, Twitter, Inc. May (2010)
(cit. on p. 18).

[60] J. Li, A. Sun, J. Han and C. Li, *A survey on deep learning for named entity recognition*,
IEEE Transactions on Knowledge and Data Engineering **34** (2020) 50 (cit. on p. 20).

[61] O. Etzioni, M. Banko, S. Soderland and D. S. Weld,
Open information extraction from the web, Communications of the ACM **51** (2008) 68
(cit. on p. 20).

[62] O. Etzioni, A. Fader, J. Christensen, S. Soderland et al.,
"Open information extraction: The second generation",
Twenty-Second International Joint Conference on Artificial Intelligence, 2011 (cit. on p. 20).

[63] M. Miwa and Y. Sasaki,
"Modeling joint entity and relation extraction with table representation", *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2014
1858 (cit. on p. 20).

[64] Q. Li and H. Ji, "Incremental Joint Extraction of Entity Mentions and Relations.", *ACL (1)*,
2014 402 (cit. on p. 20).

[65] R. Studer, V. R. Benjamins and D. Fensel, *Knowledge engineering: principles and methods*,
Data & knowledge engineering **25** (1998) 161 (cit. on p. 21).

[66] W. Wong, W. Liu and M. Bennamoun,
Ontology learning from text: A look back and into the future,
ACM Computing Surveys (CSUR) **44** (2012) 1 (cit. on p. 21).

[67] Y. Bengio, *Learning deep architectures for AI*, Now Publishers Inc, 2009 (cit. on p. 22).

[68] Y. Bengio, A. Courville and P. Vincent,
Representation learning: A review and new perspectives,
IEEE transactions on pattern analysis and machine intelligence **35** (2013) 1798 (cit. on p. 22).

[69] T. Mikolov, K. Chen, G. Corrado and J. Dean,
Efficient estimation of word representations in vector space,
arXiv preprint arXiv:1301.3781 (2013) (cit. on p. 23).

[70] J. Pennington, R. Socher and C. D. Manning, "Glove: Global vectors for word representation",
Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP), 2014 1532 (cit. on pp. 23, 24).

[71] P. Wang, Y. Qian, F. K. Soong, L. He and H. Zhao,
Part-of-speech tagging with bidirectional long short-term memory recurrent neural network,
arXiv preprint arXiv:1510.06168 (2015) (cit. on p. 23).

[72] D. Chen and C. D. Manning, "A fast and accurate dependency parser using neural networks",
Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP), 2014 740 (cit. on p. 23).

[73] G. Lample, M. Ballesteros, S. Subramanian, K. Kawakami and C. Dyer,
Neural architectures for named entity recognition, arXiv preprint arXiv:1603.01360 (2016)
(cit. on p. 23).

[74] W. Y. Zou, R. Socher, D. Cer and C. D. Manning,
"Bilingual word embeddings for phrase-based machine translation",
Proceedings of the 2013 conference on empirical methods in natural language processing,
2013 1393 (cit. on p. 23).

[75] M. E. Peters et al., "Deep Contextualized Word Representations",
Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers),
New Orleans, Louisiana: Association for Computational Linguistics, 2018 2227,
URL: https://aclanthology.org/N18-1202 (cit. on pp. 23, 24).

[76] B. McCann, J. Bradbury, C. Xiong and R. Socher,
Learned in translation: Contextualized word vectors,
Advances in neural information processing systems **30** (2017) (cit. on pp. 23, 24).

[77] J. Devlin, M.-W. Chang, K. Lee and K. Toutanova,
Bert: Pre-training of deep bidirectional transformers for language understanding,
arXiv preprint arXiv:1810.04805 (2018) (cit. on pp. 23, 25).

[78] A. Radford, K. Narasimhan, T. Salimans and I. Sutskever,
Improving language understanding by generative pre-training, (2018) (cit. on pp. 23, 25).

[79] R. Collobert et al., *Natural language processing (almost) from scratch*,
Journal of machine learning research **12** (2011) 2493 (cit. on p. 24).

[80] P. Bojanowski, E. Grave, A. Joulin and T. Mikolov,
Enriching word vectors with subword information,
Transactions of the association for computational linguistics **5** (2017) 135 (cit. on p. 24).

[81] A. Joulin, E. Grave, P. Bojanowski and T. Mikolov,
Bag of tricks for efficient text classification, arXiv preprint arXiv:1607.01759 (2016)
(cit. on p. 24).

[82] J. Reisinger and R. Mooney, "Multi-prototype vector-space models of word meaning",
Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics, 2010 109 (cit. on p. 24).

[83] E. H. Huang, R. Socher, C. D. Manning and A. Y. Ng,
"Improving word representations via global context and multiple word prototypes",
Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), 2012 873 (cit. on p. 24).

[84] J. Howard and S. Ruder, *Universal language model fine-tuning for text classification*,
arXiv preprint arXiv:1801.06146 (2018) (cit. on p. 24).

[85] A. Vaswani et al., *Attention is all you need*,
Advances in neural information processing systems **30** (2017) (cit. on p. 25).

[86] Z. Yang et al., *Xlnet: Generalized autoregressive pretraining for language understanding*,
Advances in neural information processing systems **32** (2019) (cit. on p. 25).

[87] Z. Dai et al., *Transformer-xl: Attentive language models beyond a fixed-length context*,
arXiv preprint arXiv:1901.02860 (2019) (cit. on p. 25).

[88] Y. Liu et al., *Roberta: A robustly optimized bert pretraining approach*,
arXiv preprint arXiv:1907.11692 (2019) (cit. on p. 25).

[89] K. Clark, M.-T. Luong, Q. V. Le and C. D. Manning,
Electra: Pre-training text encoders as discriminators rather than generators,
arXiv preprint arXiv:2003.10555 (2020) (cit. on p. 25).

[90] Y. Wang, Y. Hou, W. Che and T. Liu, *From static to dynamic word representations: a survey*,
International Journal of Machine Learning and Cybernetics **11** (2020) 1611 (cit. on p. 25).

[91] S. Qiao et al., *A fast parallel community discovery model on complex networks through approximate optimization*,
IEEE Transactions on Knowledge and Data Engineering **30** (2018) 1638 (cit. on p. 25).

[92] B. Perozzi, R. Al-Rfou and S. Skiena, "Deepwalk: Online learning of social representations",
Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining, 2014 701 (cit. on p. 26).

[93] F. Xia et al., *Graph learning: A survey*,
IEEE Transactions on Artificial Intelligence **2** (2021) 109 (cit. on p. 26).

[94] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner and G. Monfardini,
The graph neural network model, IEEE transactions on neural networks **20** (2008) 61
(cit. on p. 26).

[95] T. N. Kipf and M. Welling,
"Semi-supervised classification with graph convolutional networks", *ICLR*, 2017
(cit. on pp. 27, 119).

[96] W. Hamilton, Z. Ying and J. Leskovec, *Inductive representation learning on large graphs*, Advances in neural information processing systems **30** (2017) (cit. on pp. 27, 127).

[97] P. Veličković et al., "Graph attention networks", *ICLR*, 2018 (cit. on pp. 27, 28, 119, 120).

[98] J. Zhou et al., *Graph neural networks: A review of methods and applications*, AI Open **1** (2020) 57 (cit. on p. 28).

[99] D. C. Montgomery, C. L. Jennings and M. Kulahci, *Introduction to time series analysis and forecasting*, John Wiley & Sons, 2015 (cit. on pp. 29, 54).

[100] S. Bai, J. Z. Kolter and V. Koltun, *An empirical evaluation of generic convolutional and recurrent networks for sequence modeling*, arXiv preprint arXiv:1803.01271 (2018) (cit. on p. 32).

[101] S. Hochreiter and J. Schmidhuber, *Long short-term memory*, Neural computation **9** (1997) 1735 (cit. on p. 33).

[102] J. Chung, C. Gulcehre, K. Cho and Y. Bengio, *Empirical evaluation of gated recurrent neural networks on sequence modeling*, arXiv preprint arXiv:1412.3555 (2014) (cit. on p. 33).

[103] A. Vaswani et al., *Attention is all you need*, Advances in neural information processing systems **30** (2017) (cit. on p. 33).

[104] S. M. Kazemi et al., *Time2vec: Learning a vector representation of time*, arXiv preprint arXiv:1907.05321 (2019) (cit. on p. 34).

[105] Z. Wang, J. Zhang, J. Feng and Z. Chen, "Knowledge Graph Embedding by Translating on Hyperplanes.", *AAAI*, Citeseer, 2014 1112 (cit. on p. 36).

[106] Y. Lin, Z. Liu, M. Sun, Y. Liu and X. Zhu, "Learning entity and relation embeddings for knowledge graph completion", *Twenty-ninth AAAI conference on artificial intelligence*, 2015 (cit. on p. 36).

[107] G. Ji, S. He, L. Xu, K. Liu and J. Zhao, "Knowledge graph embedding via dynamic mapping matrix", *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, vol. 1, 2015 687 (cit. on p. 36).

[108] S. He, K. Liu, G. Ji and J. Zhao, "Learning to represent knowledge graphs with gaussian embedding", *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, ACM, 2015 623 (cit. on p. 36).

[109] H. Xiao, M. Huang, Y. Hao and X. Zhu, *Transg: A generative mixture model for knowledge graph embedding*, arXiv preprint arXiv:1509.05488 (2015) (cit. on p. 36).

[110] Z. Sun, Z.-H. Deng, J.-Y. Nie and J. Tang, *Rotate: Knowledge graph embedding by relational rotation in complex space*, arXiv preprint arXiv:1902.10197 (2019) (cit. on pp. 36, 56, 58, 59, 63, 69, 74).

[111] M. Nickel, V. Tresp and H.-P. Kriegel,
"A Iee-Way Model for Collective Learning on Multi-Relational Data.", *ICML*, vol. 11, 2011 809 (cit. on p. 36).

[112] M. Nickel, L. Rosasco and T. Poggio, "Holographic embeddings of knowledge graphs", *Thirtieth Aaai conference on artificial intelligence*, 2016 (cit. on p. 36).

[113] H. Liu, Y. Wu and Y. Yang, "Analogical inference for multi-relational embeddings", *International conference on machine learning*, PMLR, 2017 2168 (cit. on p. 36).

[114] S. M. Kazemi and D. Poole, "Simple embedding for link prediction in knowledge graphs", *Advances in neural information processing systems*, 2018 4284 (cit. on pp. 36, 74).

[115] I. Balažević, C. Allen and T. M. Hospedales,
Tucker: Tensor factorization for knowledge graph completion,
arXiv preprint arXiv:1901.09590 (2019) (cit. on p. 36).

[116] T. Lacroix, N. Usunier and G. Obozinski,
"Canonical tensor decomposition for knowledge base completion",
International Conference on Machine Learning (ICML), 2018
(cit. on pp. 36, 38, 58–61, 69, 81, 131).

[117] S. Zhang, Y. Tay, L. Yao and Q. Liu, "Quaternion knowledge graph embeddings", *Advances in Neural Information Processing Systems*, 2019 2731
(cit. on pp. 36, 58, 61, 69, 74).

[118] A. Bordes, X. Glorot, J. Weston and Y. Bengio,
A semantic matching energy function for learning with multi-relational data,
Machine Learning **94** (2014) 233 (cit. on p. 37).

[119] R. Socher, D. Chen, C. D. Manning and A. Ng,
Reasoning with neural tensor networks for knowledge base completion,
Advances in neural information processing systems **26** (2013) (cit. on p. 37).

[120] T. Dettmers, P. Minervini, P. Stenetorp and S. Riedel,
"Convolutional 2d knowledge graph embeddings",
Thirty-Second AAAI Conference on Artificial Intelligence, 2018 (cit. on p. 37).

[121] D. Q. Nguyen, D. Q. Nguyen, T. D. Nguyen and D. Phung,
A convolutional neural network-based model for knowledge base completion and its application to search personalization, Semantic Web **10** (2019) 947 (cit. on p. 37).

[122] T. Vu, T. D. Nguyen, D. Q. Nguyen, D. Phung et al., "A capsule network-based embedding model for knowledge graph completion and search personalization", *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, 2019 2180 (cit. on p. 37).

[123] L. Guo, Z. Sun and W. Hu,
"Learning to exploit long-term relational dependencies in knowledge graphs",
International Conference on Machine Learning, PMLR, 2019 2505 (cit. on p. 37).

[124] M. Schlichtkrull et al., "Modeling relational data with graph convolutional networks", *European semantic web conference*, Springer, 2018 593 (cit. on pp. 37, 82).

[125] S. Vashishth, S. Sanyal, V. Nitin and P. Talukdar,
"Composition-based Multi-Relational Graph Convolutional Networks",
International Conference on Learning Representations, 2020,
URL: https://openreview.net/forum?id=BylA_C4tPr (cit. on p. 37).

[126] L. Cai, B. Yan, G. Mai, K. Janowicz and R. Zhu, "TransGCN: Coupling transformation assumptions with graph convolutional networks for link prediction",
Proceedings of the 10th International Conference on Knowledge Capture, 2019 131 (cit. on p. 37).

[127] S. M. Kazemi et al., *Representation Learning for Dynamic Graphs: A Survey.*,
J. Mach. Learn. Res. **21** (2020) 1 (cit. on p. 37).

[128] J. Tingsong et al., *Encoding Temporal Information for Time-Aware Link Prediction*,
Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (2016) 2350 (cit. on p. 38).

[129] S. S. Dasgupta, S. N. Ray and P. Talukdar,
"HyTE: Hyperplane-based Temporally aware Knowledge Graph Embedding",
Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, 2018 2001 (cit. on pp. 38, 47, 50, 56, 59, 61, 63, 64, 72, 82, 109).

[130] Y. Ma, V. Tresp and E. A. Daxberger, *Embedding models for episodic knowledge graphs*,
Journal of Web Semantics **59** (2019) 100490 (cit. on p. 38).

[131] P. Jain, S. Rathi, S. Chakrabarti et al.,
Temporal Knowledge Base Completion: New Algorithms and Evaluation Protocols,
arXiv preprint arXiv:2005.05035 (2020) (cit. on pp. 38, 74, 79, 81, 82, 89).

[132] A. Garcıa-Durán, S. Dumančić and M. Niepert,
"Learning Sequence Encoders for Temporal Knowledge Graph Completion", *EMNLP*, 2018 (cit. on pp. 38, 39, 47, 50, 51, 56, 58, 59, 61, 63, 69, 82, 89, 109, 116).

[133] R. Goel, S. M. Kazemi, M. Brubaker and P. Poupart,
"Diachronic Embedding for Temporal Knowledge Graph Completion", *AAAI*, 2020 (cit. on pp. 39, 47, 50, 51, 56, 58, 59, 61, 69, 72, 82, 109).

[134] J. Jung, J. Jung and U. Kang,
"Learning to walk across time for interpretable temporal knowledge graph completion",
Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining, 2021 786 (cit. on p. 39).

[135] M. Richardson and P. Domingos, *Markov logic networks*, Machine learning **62** (2006) 107 (cit. on p. 39).

[136] P. Singla and P. Domingos, "Discriminative training of Markov logic networks", *AAAI*, vol. 5, 2005 868 (cit. on p. 39).

[137] L. V. Harsha Vardhan, G. Jia and S. Kok,
"Probabilistic logic graph attention networks for reasoning",
Companion Proceedings of the Web Conference 2020, 2020 669 (cit. on p. 39).

[138] J. Bai, Z. Wang, H. Zhang and Y. Song,
Query2Particles: Knowledge Graph Reasoning with Particle Embeddings,
arXiv preprint arXiv:2204.12847 (2022) (cit. on p. 40).

[139] N. Choudhary, N. Rao, S. Katariya, K. Subbian and C. K. Reddy,
"Self-supervised hyperboloid representations from logical queries over knowledge graphs",
Proceedings of the Web Conference 2021, 2021 1373 (cit. on pp. 41, 89).

[140] F. P. S. Luus et al., *Logic Embeddings for Complex Query Answering*,
ArXiv **abs/2103.00418** (2021) (cit. on pp. 41, 89).

[141] D. Garg et al., *Quantum embedding of knowledge for reasoning*,
Advances in Neural Information Processing Systems **32** (2019) 5594 (cit. on p. 41).

[142] S. K. Srivastava et al., *Inductive Quantum Embedding*,
Advances in Neural Information Processing Systems **33** (2020) (cit. on p. 41).

[143] F. Scharffe, Y. Liu and C. Zhou,
"Rdf-ai: an architecture for rdf datasets matching, fusion and interlink", *Proc. IJCAI 2009 workshop on Identity, reference, and knowledge representation (IR-KR), Pasadena (CA US)*,
2009 23 (cit. on p. 41).

[144] J. Volz, C. Bizer, M. Gaedke and G. Kobilarov,
"Discovering and maintaining links on the web of data",
International Semantic Web Conference, Springer, 2009 650 (cit. on p. 41).

[145] A.-C. N. Ngomo and S. Auer,
"LIMES—a time-efficient approach for large-scale link discovery on the web of data",
Twenty-Second International Joint Conference on Artificial Intelligence, 2011 (cit. on p. 41).

[146] J. Han, J. Pei and M. Kamber, *Data mining: concepts and techniques*, Elsevier, 2011
(cit. on p. 41).

[147] R. Tempo, G. Calafiore and F. Dabbene, "Statistical Learning Theory",
Randomized Algorithms for Analysis and Control of Uncertain Systems, Springer, 2013 123
(cit. on p. 41).

[148] I. Bhattacharya and L. Getoor, "A latent dirichlet model for unsupervised entity resolution",
Proceedings of the 2006 SIAM International Conference on Data Mining, SIAM, 2006 47
(cit. on p. 41).

[149] R. Hall, C. Sutton and A. McCallum,
"Unsupervised deduplication using cross-field dependencies", *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2008 310
(cit. on p. 41).

[150] D. M. Blei, A. Y. Ng and M. I. Jordan, *Latent dirichlet allocation*,
Journal of machine Learning research **3** (2003) 993 (cit. on p. 41).

[151] H. Zhu, R. Xie, Z. Liu and M. Sun,
"Iterative Entity Alignment via Joint Knowledge Embeddings.", *IJCAI*, vol. 17, 2017 4258
(cit. on pp. 42, 116).

[152] Y. Lin et al., *Modeling relation paths for representation learning of knowledge bases*,
arXiv preprint arXiv:1506.00379 (2015) (cit. on pp. 42, 142).

[153] Z. Sun, W. Hu, Q. Zhang and Y. Qu,
"Bootstrapping Entity Alignment with Knowledge Graph Embedding.", *IJCAI*, vol. 18, 2018 4396 (cit. on pp. 42, 113, 122).

[154] M. Chen, Y. Tian, K.-W. Chang, S. Skiena and C. Zaniolo, *Co-training embeddings of knowledge graphs and entity descriptions for cross-lingual entity alignment*,
arXiv preprint arXiv:1806.06478 (2018) (cit. on p. 42).

[155] B. D. Trisedya, J. Qi and R. Zhang,
"Entity alignment between knowledge graphs using attribute embeddings",
Proceedings of the AAAI Conference on Artificial Intelligence, vol. 33, 01, 2019 297 (cit. on pp. 42, 122).

[156] Q. Zhu, X. Zhou, J. Wu, J. Tan and L. Guo,
"Neighborhood-Aware Attentional Representation for Multilingual Knowledge Graphs.", *IJCAI*, 2019 1943 (cit. on p. 43).

[157] Y. Wu et al., "Relation-Aware Entity Alignment for Heterogeneous Knowledge Graphs", *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, 2019 5278 (cit. on pp. 43, 122).

[158] R. Ye, X. Li, Y. Fang, H. Zang and M. Wang, "A Vectorized Relational Graph Convolutional Network for Multi-Relational Network Alignment.", *IJCAI*, 2019 4135 (cit. on p. 43).

[159] Y. Wu, X. Liu, Y. Feng, Z. Wang and D. Zhao,
"Jointly Learning Entity and Relation Representations for Entity Alignment", *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*,
Association for Computational Linguistics, 2019 240,
URL: https://www.aclweb.org/anthology/D19-1023 (cit. on pp. 43, 122).

[160] Z. Sun et al., "Knowledge Association with Hyperbolic Knowledge Graph Embeddings", *EMNLP*, 2020 (cit. on pp. 43, 122).

[161] D. Yu, Y. Yang, R. Zhang and Y. Wu,
"Knowledge Embedding Based Graph Convolutional Network",
Proceedings of the Web Conference 2021, 2021 1619 (cit. on pp. 43, 122).

[162] K. Zeng, C. Li, L. Hou, J. Li and L. Feng,
A comprehensive survey of entity alignment for knowledge graphs, AI Open **2** (2021) 1 (cit. on p. 43).

[163] S. El Alaoui and B. Ramamurthy, *EAODR: A novel routing algorithm based on the modified temporal graph network model for DTN-based interplanetary networks*,
Computer Networks **129** (2017) 129 (cit. on p. 44).

[164] J. Skarding, B. Gabrys and K. Musial, *Foundations and Modeling of Dynamic Networks Using Dynamic Graph Neural Networks: A Survey*, IEEE Access **9** (2021) 79143 (cit. on p. 44).

[165] Y. Seo, M. Defferrard, P. Vandergheynst and X. Bresson,
"Structured sequence modeling with graph convolutional recurrent networks",
International Conference on Neural Information Processing, Springer, 2018 362
(cit. on p. 45).

[166] A. Narayan and P. H. Roe, *Learning graph dynamics using deep neural networks*,
IFAC-PapersOnLine **51** (2018) 433 (cit. on p. 45).

[167] M. Niepert, M. Ahmed and K. Kutzkov, "Learning convolutional neural networks for graphs",
International conference on machine learning, PMLR, 2016 2014 (cit. on p. 45).

[168] F. Manessi, A. Rozza and M. Manzo, *Dynamic graph convolutional networks*,
Pattern Recognition **97** (2020) 107000 (cit. on p. 45).

[169] A. Sankar, Y. Wu, L. Gou, W. Zhang and H. Yang,
"Dysat: Deep neural representation learning on dynamic graphs via self-attention networks",
Proceedings of the 13th International Conference on Web Search and Data Mining, 2020 519
(cit. on p. 45).

[170] A. Pareja et al., "Evolvegcn: Evolving graph convolutional networks for dynamic graphs",
Proceedings of the AAAI Conference on Artificial Intelligence, vol. 34, 04, 2020 5363
(cit. on p. 45).

[171] P. Goyal, S. R. Chhetri, N. Mehrabi, E. Ferrara and A. Canedo,
Dynamicgem: A library for dynamic graph embedding methods,
arXiv preprint arXiv:1811.10734 (2018) (cit. on p. 45).

[172] P. Goyal, S. R. Chhetri and A. Canedo,
dyngraph2vec: Capturing network dynamics using dynamic graph representation learning,
Knowledge-Based Systems **187** (2020) 104816 (cit. on p. 45).

[173] Y. Ma, Z. Guo, Z. Ren, J. Tang and D. Yin, "Streaming graph neural networks",
Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval, 2020 719 (cit. on p. 45).

[174] S. Kumar, X. Zhang and J. Leskovec,
"Predicting dynamic embedding trajectory in temporal interaction networks", *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*,
2019 1269 (cit. on p. 45).

[175] R. Trivedi, M. Farajtabar, P. Biswal and H. Zha,
"Dyrep: Learning representations over dynamic graphs",
International conference on learning representations, 2019 (cit. on p. 46).

[176] Z. Han, Y. Ma, Y. Wang, S. Günnemann and V. Tresp,
Graph hawkes neural network for forecasting on temporal knowledge graphs,
arXiv preprint arXiv:2003.13432 (2020) (cit. on p. 46).

[177] J. Leblay and M. W. Chekol, "Deriving validity time in knowledge graph",
Companion of the The Web Conference 2018 on The Web Conference 2018,
International World Wide Web Conferences Steering Committee, 2018 1771
(cit. on pp. 47, 58, 63, 69, 72, 82, 89, 109).

[178] C. Xu, M. Nayyeri, F. Alkhoury, H. Shariat Yazdi and J. Lehmann, "TeRo: A Time-aware Knowledge Graph Embedding via Temporal Rotation", *Proceedings of the 28th International Conference on Computational Linguistics*, Barcelona, Spain (Online): International Committee on Computational Linguistics, 2020 1583, URL: https://aclanthology.org/2020.coling-main.139 (cit. on pp. 48, 63, 74, 82, 109).

[179] C. Xu, M. Nayyeri, Y.-Y. Chen and J. Lehmann, *Geometric Algebra based Embeddings for Static and Temporal Knowledge Graph Completion*, IEEE Transactions on Knowledge and Data Engineering (2022) 1 (cit. on pp. 48, 140).

[180] S. Ho and M. Xie, *The use of ARIMA models for reliability forecasting and analysis*, Computers & industrial engineering **35** (1998) 213 (cit. on p. 54).

[181] D. Yu, K. Yao, H. Su, G. Li and F. Seide, "KL-divergence regularized deep neural network adaptation for improved large vocabulary speech recognition", *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, IEEE, 2013 7893 (cit. on p. 55).

[182] K. B. Petersen, M. S. Pedersen et al., *The matrix cookbook*, Technical University of Denmark **7** (2008) 510 (cit. on p. 55).

[183] R. Trivedi, H. Dai, Y. Wang and L. Song, *Know-Evolve: Deep Temporal Reasoning for Dynamic Knowledge Graphs*, (2017) (cit. on pp. 58, 59).

[184] W. Jin, C. Zhang, P. Szekely and X. Ren, *Recurrent Event Network for Reasoning over Temporal Knowledge Graphs*, arXiv preprint arXiv:1904.05530 (2019) (cit. on p. 59).

[185] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization", *Proceedings of the International Conference on Learning Representations (ICLR)*, 2015 (cit. on pp. 59, 104).

[186] Y. Ma, V. Tresp and E. A. Daxberger, *Embedding models for episodic knowledge graphs*, Journal of Web Semantics (2018) 100490 (cit. on pp. 63, 74, 79).

[187] J. Duchi, E. Hazan and Y. Singer, *Adaptive subgradient methods for online learning and stochastic optimization.*, Journal of machine learning research **12** (2011) (cit. on pp. 70, 83).

[188] J. M. Chappell, A. Iqbal, L. J. Gunn and D. Abbott, *Functions of multivector variables*, Plos one **10** (2015) e0116943 (cit. on p. 74).

[189] S. Franchini, G. Vassallo and F. Sorbello, *A brief introduction to Clifford algebra*, (2010) (cit. on p. 74).

[190] S. W. R. H. L. P. F. H. M. Ed. and Dub., *LXXVIII. On quaternions; or on a new system of imaginaries in Algebra*, The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science **25** (1844) 489, eprint: https://doi.org/10.1080/14786444408645047, URL: https://doi.org/10.1080/14786444408645047 (cit. on p. 74).

[191] E. Chisolm, *Geometric algebra*, arXiv preprint arXiv:1205.5935 (2012) (cit. on p. 77).

[192] J. M. Chappell, A. Iqbal, L. J. Gunn and D. Abbott, *Functions of multivector variables*, PloS one **10** (2015) e0116943 (cit. on p. 77).

[193] Y. Wang, R. Gemulla and H. Li, "On multi-relational link prediction with bilinear models", *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018 (cit. on pp. 79, 83).

[194] T. Trouillon et al., *Knowledge graph completion via complex tensor factorization*, arXiv preprint arXiv:1702.06879 (2017) (cit. on p. 80).

[195] C. Xu, M. Nayyeri, Y.-Y. Chen and J. Lehmann, "Knowledge Graph Embeddings in Geometric Algebras", *Proceedings of the 28th International Conference on Computational Linguistics*, Barcelona, Spain (Online): International Committee on Computational Linguistics, 2020 530, URL: https://aclanthology.org/2020.coling-main.46 (cit. on p. 81).

[196] Y. Xu et al., *Time-aware Graph Embedding: A temporal smoothness and task-oriented approach*, arXiv preprint arXiv:2007.11164 (2020) (cit. on p. 81).

[197] U. Singer, I. Guy and K. Radinsky, *Node Embedding over Temporal Graphs*, CoRR **abs/1903.08889** (2019), arXiv: 1903.08889, URL: http://arxiv.org/abs/1903.08889 (cit. on p. 81).

[198] H.-F. Yu, N. Rao and I. S. Dhillon, "Temporal regularized matrix factorization for high-dimensional time series prediction", *Advances in neural information processing systems*, 2016 847 (cit. on p. 81).

[199] J. Wu, M. Cao, J. C. K. Cheung and W. L. Hamilton, "TeMP: Temporal Message Passing for Temporal Knowledge Graph Completion", *EMNLP*, 2020 (cit. on p. 89).

[200] X. Lin et al., *TFLEX: Temporal Feature-Logic Embedding Framework for Complex Reasoning over Temporal Knowledge Graph*, Advances in Neural Information Processing Systems (2022) (cit. on p. 90).

[201] H. Ren et al., *SMORE: Knowledge Graph Completion and Multi-hop Reasoning in Massive Knowledge Graphs*, arXiv preprint arXiv:2110.14890 (2021) (cit. on p. 91).

[202] X. Mao, W. Wang, Y. Wu and M. Lan, "Boosting the speed of entity alignment 10×: Dual attention matching network with normalized hard sample mining", *Proceedings of the Web Conference 2021*, 2021 821 (cit. on pp. 113, 131).

[203] Y. Li, R. Yu, C. Shahabi and Y. Liu, *Diffusion convolutional recurrent neural network: Data-driven traffic forecasting*, arXiv preprint arXiv:1707.01926 (2017) (cit. on p. 114).

[204] S. Yan, Y. Xiong and D. Lin, "Spatial temporal graph convolutional networks for skeleton-based action recognition", *Proceedings of the AAAI conference on artificial intelligence*, vol. 32, 1, 2018 (cit. on p. 114).

[205] A. Pareja et al., "Evolvegcn: Evolving graph convolutional networks for dynamic graphs", *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, 2020 5363 (cit. on p. 114).

[206] C. Xu, F. Su and J. Lehmann, "Time-aware Graph Neural Network for Entity Alignment between Temporal Knowledge Graphs",
Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, Online and Punta Cana, Dominican Republic: Association for Computational Linguistics, 2021 8999, URL: https://aclanthology.org/2021.emnlp-main.709
(cit. on pp. 115, 117).

[207] C. Xu, F. Su, B. Xiong and J. Lehmann,
"Time-Aware Entity Alignment Using Temporal Relational Attention", WWW '22, Virtual Event, Lyon, France: Association for Computing Machinery, 2022 788, ISBN: 9781450390965, URL: https://doi.org/10.1145/3485447.3511922
(cit. on pp. 115, 127).

[208] T. Tieleman and. Hinton,
"rmsprop: Divide the gradient bya running average of its recent magnitude", *Lecture 6.5*, 2012
(cit. on pp. 121, 132).

[209] A. Conneau, G. Lample, M. Ranzato, L. Denoyer and H. Jégou,
"Word translation without parallel data", *ICLR 2018*, 2018 (cit. on pp. 121, 132).

[210] Z. Liu et al.,
"Exploring and Evaluating Attributes, Values, and Structures for Entity Alignment", *EMNLP*, 2020 (cit. on p. 122).

[211] Z. Sun et al.,
A Benchmarking Study of Embedding-based Entity Alignment for Knowledge Graphs, Proceedings of the VLDB Endowment **13** (2020) 2326,
URL: http://www.vldb.org/pvldb/vol13/p2326-sun.pdf (cit. on p. 123).

[212] K. Zeng, C. Li, L. Hou, J. Li and L. Feng,
A comprehensive survey of entity alignment for knowledge graphs, AI Open **2** (2021) 1
(cit. on p. 124).

[213] S. Pei, L. Yu and X. Zhang,
"Improving Cross-lingual Entity Alignment via Optimal Transport", *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, International Joint Conferences on Artificial Intelligence Organization, 2019 3231,
URL: https://doi.org/10.24963/ijcai.2019/448 (cit. on p. 129).

[214] M. Schlichtkrull et al., "Modeling relational data with graph convolutional networks", *European semantic web conference*, Springer, 2018 593 (cit. on p. 142).

[215] M. Nayyeri, C. Xu, M. M. Alam, J. Lehmann and H. S. Yazdi,
Logicenn: A neural based knowledge graphs embedding model with logical rules,
IEEE Transactions on Pattern Analysis and Machine Intelligence (2021) (cit. on p. 142).

[216] J. F. Allen, *Maintaining Knowledge about Temporal Intervals*, Commun. ACM **26** (1983) 832,
ISSN: 0001-0782, URL: https://doi.org/10.1145/182.358434 (cit. on p. 143).

APPENDIX A

List of Publications

- *Conference Papers* (*peer reviewed*)

  1. **Chengjin Xu**, Mojtaba Nayyeri, Fouad Alkhoury, Hamed Yazdi and Jens Lehmann, "Temporal knowledge graph completion based on time series gaussian embedding", International Semantic Web Conference, pp. 654-671. Springer, Cham, 2020. DOI: https://doi.org/10.1007/978-3-030-62419-4_37

  2. **Chengjin Xu**, Mojtaba Nayyeri, Fouad Alkhoury, Hamed Yazdi and Jens Lehmann. "TeRo: A Time-aware Knowledge Graph Embedding via Temporal Rotation", Proceedings of the 28th International Conference on Computational Linguistics, pp. 1583–1593. International Committee on Computational Linguistics, 2020. DOI: https://doi.org/10.18653/v1/2020.coling-main.139

  3. **Chengjin Xu**, Yung-Yu Chen, Mojtaba Nayyeri and Jens Lehmann. "Temporal Knowledge Graph Completion using a Linear Temporal Regularizer and Multivector Embeddings", Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pp. 2569–2578. Association for Computational Linguistics, 2021. DOI: https://doi.org/10.18653/v1/2021.naacl-main.202

  4. **Chengjin Xu**, Fenglong Su and Jens Lehmann. "Time-aware Graph Neural Network for Entity Alignment between Temporal Knowledge Graphs", Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, pp. 8999–9010. Association for Computational Linguistics, 2021. DOI: https://doi.org/10.18653/v1/2021.emnlp-main.709

  5. **Chengjin Xu**, Fenglong Su, Bo Xiong and Jens Lehmann. "Time-Aware Entity Alignment Using Temporal Relational Attention", Proceedings of the ACM Web Conference 2022, pp. 788-797. Association for Computing Machinery, 2022. DOI: https://doi.org/10.1145/3485447.3511922

- *Journal Papers* (*peer reviewed*)

  6. **Chengjin Xu**, Yung-Yu Chen, Mojtaba Nayyeri and Jens Lehmann. "Geometric Algebra based Embeddings for Static and Temporal Knowledge Graph Completion",

IEEE Transactions on Knowledge and Data Engineering. IEEE, 2022. DOI: https://doi.org/10.1109/TKDE.2022.3151435

- *Working Drafts*

  7. Xueyuan Lin, **Chengjin Xu**, Haihong E, Fenglong Su, Gengxian Zhou, Tianyi Hu, Li Ningyuan, Mingzhi Sun and Haoran Luo. "TFLEX: Temporal Feature-Logic Embedding Framework for Complex Reasoning over Temporal Knowledge Graph", The Thirty-Sixth Annual Conference on Neural Information Processing Systems. 2022. (Under Review) URL: https://arxiv.org/pdf/2205.14307.pdf

- *Miscellaneous Papers* (*peer reviewed*)
  Following publications originated during and are related to this thesis but are not part of the thesis itself,

  8. Mojtaba Nayyeri, **Chengjin Xu**, Sahar Vahdati, Nadezhda Vassilyeva, Emanuel Sallinger, Hamed Shariat Yazdi, and Jens Lehmann. "Fantastic knowledge graph embeddings and how to find the right space for them." In International Semantic Web Conference, pp. 438-455. Springer, Cham, 2020. DOI: https://doi.org/10.1007/978-3-030-62419-4_25
  9. Mojtaba Nayyeri, **Chengjin Xu**, Yadollah Yaghoobzadeh, Sahar Vahdati, Mirza Mohtashim Alam, Hamed Shariat Yazdi, and Jens Lehmann. "Loss-aware pattern inference: A correction on the wrongly claimed limitations of embedding models." In Pacific-Asia Conference on Knowledge Discovery and Data Mining, pp. 77-89. Springer, Cham, 2021. DOI: https://doi.org/10.1007/978-3-030-75768-7_7
  10. **Chengjin Xu**, Mojtaba Nayyeri, Sahar Vahdati, and Jens Lehmann. "Multiple Run Ensemble Learning with Low-Dimensional Knowledge Graph Embeddings." In 2021 International Joint Conference on Neural Networks. IEEE, 2021. URL: https://arxiv.org/pdf/2104.05003.pdf
  11. Mojtaba Nayyeri, **Chengjin Xu**, Mirza Mohtashim Alam, Jens Lehmann, and Hamed Shariat Yazdi. "Logicenn: A neural based knowledge graphs embedding model with logical rules." IEEE Transactions on Pattern Analysis and Machine Intelligence. IEEE, 2021. DOI: https://doi.org/10.1109/TPAMI.2021.3121646
  12. Mojtaba Nayyeri, **Chengjin Xu**, Franca Hoffmann, Mirza Mohtashim Alam, Jens Lehmann, and Sahar Vahdati. "Knowledge Graph Representation Learning using Ordinary Differential Equations." In Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, pp. 9529-9548. Association for Computational Linguistics, 2021. DOI: https://doi.org/10.18653/v1/2021.emnlp-main.750
  13. Bo Xiong, Shichao Zhu, Mojtaba Nayyeri, **Chengjin Xu**, Shirui Pan, Chuan Zhou, and Steffen Staab. "Ultrahyperbolic Knowledge Graph Embeddings." In Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery & Data Mining, 2022. URL: https://arxiv.org/pdf/2206.00449.pdf

APPENDIX B

Abbreviations and Acronyms

AR	Autoregressive
ARIMA	Autoregressive Integrated Moving Average
AST	Abstract Syntax Tree
AVP	Attribute Value Pair
CNN	Convolutional Neural Network
CSLS	Cross-domain Similarity Local Scaling
DAG	Directed Acyclic Graph
DGNN	Dynamic Graph Neural Network
DL	Deep Learning
DNF	Disjunctive Normal Form
EA	Entity Alignment
EPFO	Existential Positive First-Order
FOL	First-Order Logic
GAT	Graph Attention Network
GCN	Graph Convonlutional Network
GE	Graph Embedding
GNN	Graph Neural Network
GPU	Graphics Processing Unit
GRU	Gate Recurrent Unit
HTML	Hyper Text Markup Language
IEA	Inductive Entity Alignment
KG	Knowledge Graph
KGC	Knowledge Graph Completion
KGE	Knowledge Graph Embedding
LDA	Latent Dirichlet Allocation
LM	Language Model
LSTM	Long Short Term Memory (Networks)
MA	Moving Average
MKGR	Multi-hop Knowledge Graph Reasoning
ML	Machine Learning

MLP	Multi-Layer Perceptron
MRR	Mean Reciprocal Rank
MTKGR	Multi-hop Temporal Knowledge Graph Reasoning
NLP	Natural language Processing
NN	Neural Network
NSP	Next Sentence Prediction
OIE	Open Domain Extraction
OKG	Open-world Knowledge Graph
OWL	Web Ontology Language
POS	Part-Of-Speech
QE	Query Embedding
RDF	Resource Description Framework
RNN	Recurrent Neural Network
SEA	Static Entity Alignment
SKG	Static Knowledge Graph
SKGC	Static Knowledge Graph Completion
SKGE	Static Knowledge Graph Embedding
SVM	Support Vector Machine
TEA	Temporal Entity Alignment
TKG	Temporal Knowledge Graph
TKGC	Temporal Knowledge Graph Completion
TKGE	Temporal Knowledge Graph Embedding
TQE	Temporal Query Embedding
TTP	temporal point process
W3C	World Wide Web Consortium
XML	Extensible Markup Language

List of Figures

List of Tables