Master's Thesis

Learning One-shot Relations in Temporal Knowledge Graphs

Department of Statistics
Ludwig-Maximilians-Universität München

Author: Bailan He
Supervisor: Prof. Dr. Volker Tresp

Munich, October 24th, 2022

DECLARATION

This thesis is a presentation of my original research work. Wherever contributions of others are involved, every effort is made to indicate this clearly, with due reference to the literature, and acknowledgement of collaborative research and discussions. The work was done under the guidance of Prof. Dr. Volker Tresp and PhD candidate Zifeng Ding, at the Datenbanksysteme und Data Mining, Ludwig-Maximilians-Universität München.

Abstract

Knowledge graphs (KGs) provide a principled method of representing structural relations between entities, and are widely used in various artificial intelligence applications, such as recommendation systems, intelligent customer service, and question answering. By introducing time information into KGs, temporal knowledge graphs (TKGs) can describe the ever-changing facts of the world. Similar to KGs, TKGs are also known to be highly incomplete, which draws huge attention to developing TKG reasoning methods for link prediction on TKGs. To better model the knowledge graph representations, most of previous methods require a sufficient amount of data for each relations. However, it has been noticed that sparse relations (i.e., relations occur for very few times) are quite common in both KGs and TKGs. To solve this problem, few-shot learning method is used. Few-shot learning employs a meta-learning framework and the model needs to learn to predict the unseen links of one sparse relation with only few observed events. In this thesis, we aim to predict new facts under a challenging setting where only one training sample is available. We revisit the previous works related to few-shot relational learning in KGs and extend two existing TKG reasoning tasks, i.e., interpolated and extrapolated link prediction, to the one-shot setting and propose four new large-scale datasets for these two tasks. Our new datasets have a substantial number of associated TKG facts, which greatly alleviate instability in model training and evaluation. Furthermore, we propose a model learning **m**eta representations of **o**ne-**s**ho**t** relations for solving both tasks in TKGs (MOST). MOST employs a meta-information extractor for learning contextualized time-aware entity representations and a meta-representation learner to compute meta representations for sparse relations. Furthermore, MOST employs a metric function to calculate the plausibility scores of TKG quadruples. Additionally, we fix the unfair evaluation settings employed by previous KG few-shot learning methods. Finally, we evaluate MOST on all four newly-proposed datasets and compare MOST with a large group of baselines on both TKG link prediction tasks. The experiment results show that MOST achieves state-of-the-art performance on both interpolation and extrapolation tasks while keeping a low time cost.

Keywords: temporal knowledge graphs, few-shot learning, link prediction.

Contents

List of Figures

List of Tables

1 Introduction

1.1 Motivation

Integrating human knowledge into the computing system was an important idea. In the 1990s, the World-Wide Web [4] was designed to serve as a collection of human knowledge, which provides an opportunity for widespread knowledge sharing. On this basis, in order to include more significant content, the semantic web [3] has expanded the World-Wide Web with specific structures. In 2012, the idea of knowledge graphs (KGs) was proposed by Google. A Knowledge Graph is defined as a large-scale semantic network knowledge, that expresses and stores in the form of a directed graph, which has the advantages of semantic richness, friendly structure, and ease of understanding. Formally, KGs represent a collection of facts as a set of triplets in the form of (s, r, o), where s, o, r mean subject entity, object entity and relation between them, respectively. In recent years, due to the great characteristics in expressing human prior knowledge [28], knowledge graphs have drawn great attention and have been widely and successfully applied in many fields such as natural language processing, question-answering systems, and recommendation systems.

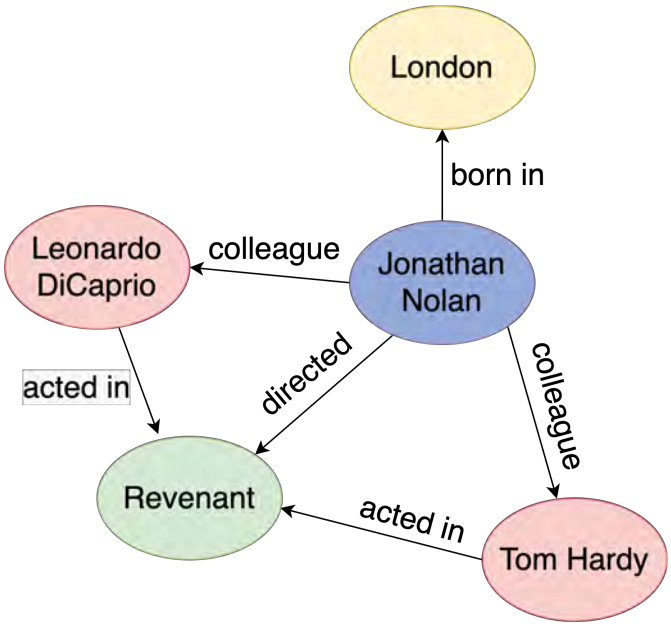

Figure. 1. An example of Knowledge Graph.

In knowledge graphs, we usually use the term "Entity" to express the nodes in a

graph and "Relation" to express the "edges" in a graph. Entities refer to things in the real world, such as people, places, concepts, drugs, companies, etc., and are normally represented by circles or ellipses. Relations are used to express the certain connection between different entities and are represented by directed arrows. The direction of the arrow is usually from subject to object. Figure. 1 shows an example of a KG. Five entities and five kinds of relations are specified in this KG. More specifically, in this KG, the fact that Jonathan Nolan directed the Revenant is represented by a triplet (*Jonathan Nolan*, *directed*, *Revenant*). *Jonathan Nolan* and *Revenant* are two entities, *directed* is the relation between them, denoted with a directed arrow and labeled with a character name.

Human knowledge is often time-sensitive. In the KG example mentioned above, the fact that Jonathan Nolan directed the Revenant does not hold forever. However, static KGs do not include temporal information. To capture the temporal aspect, time information t is added in KGs. Such KGs are called temporal knowledge graphs (TKGs). A fact of TKGs can be expressed in a quadruple in the form of (s, r, o, t).

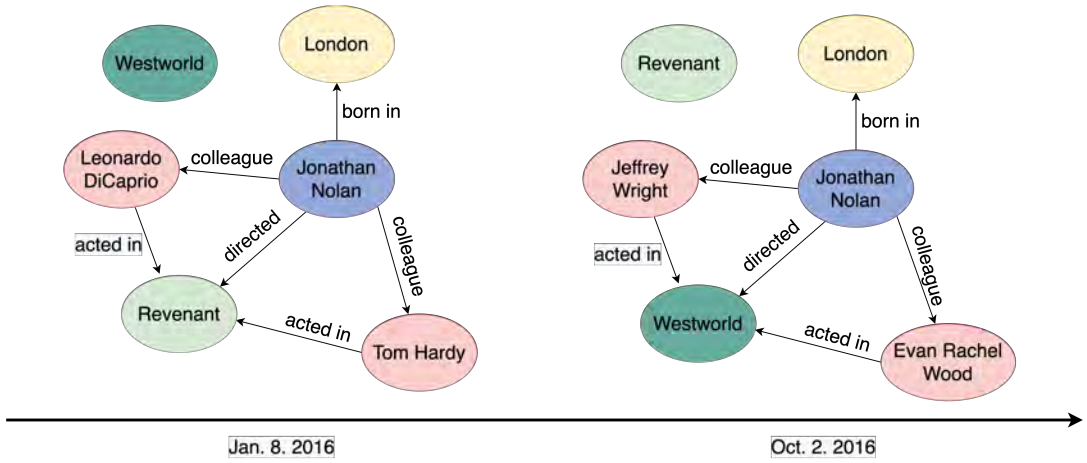

Figure. 2. An example of Temporal Knowledge Graph.

Figure. 2 shows an example of a TKG. In this TKG, Jonathon Nolan directed the Revenant in January 2016. However, in October 2016, the relation *directed*, between Jonathon Nolan and Revenant no longer existed because he had already finished directing Revenant and started directing Westworld. The facts about Jonathon Nolan are different at the different timestamps. In our example, (*Jonathan Nolan*, *directed*, *Revenant*) is transformed into (*Jonathan Nolan*, *directed*, *Revenant*, *Oct.2.2016*). TKGs can encode the ever-changing knowledge by incorporating time information.

Knowledge graph embedding (KGE), also known as knowledge representation learning (KRL), is a crucial research direction that paves the way for many knowledge acquisition tasks and downstream applications, e.g., link prediction and question answering. The key goal of KGE is to learn the low-dimensional embedding of entities and relations and find a suitable score function to measure the plausibility of KG facts.

As knowledge graphs are often incomplete, new triples must be continuously added to the knowledge graph, as well as missing entities or relations need to be inferred from existing graphs. This results in a task called knowledge graph completion (KGC). Preliminary researchers mainly focus on building embedding-based models to solve this task. However, these models often fail to capture multi-hop relations. Recently, researchers have incorporated logical rules or explored multi-hot relation paths to address this issue.

TKG representation learning can be seen as a branch of KRE. By extending the triple (s, r, o) to a quadruple (s, r, o, t), we can get the embedding containing the time information can be obtained, namely temporal information embedding. Similar to KGs, TKGs are also known to be highly incomplete. There have been several works aiming to propose TKG reasoning models to infer missing links ([20, 26, 21]). Most of these reasoning models require a huge amount of data to learn an expressive temporal information embedding. However, Xiong et al. [46] and Mirtaheri et al. [27] find a large part of KG and TKG relations are long-tail distributed (long-tail means some relations only occur for a few times), which leads to the degenerative performance of existing reasoning methods. A series of works [46, 8, 52, 34] employ the few-shot learning (FSL) methods to address this issue. Few-shot learning is a machine learning method in which the training dataset contains limited resources. Humans are very good at recognizing a new object from a tiny sample. For example, a child can recognize a 'zebra' or a 'rhinoceros' with just a few pictures in a book. By using few-shot learning, a model can learn from a few examples like humans. On top of this these methods, Mirtaheri et al. [27] develop a method to alleviate these problems for TKGs by considering temporal dependencies between facts. They formulate a one-shot extrapolated link prediction (LP) task for TKGs and propose new datasets based on benchmark TKG databases ICEWS [6], and GDELT [22]. In this work, we focus on both interpolated and extrapolated LP tasks under the one-shot setting.

1.2 Method Introduction

1.2.1 Temporal Knowledge Graph Link Prediction

Let $\mathcal{E}$, $\mathcal{R}$ and $\mathcal{T}$ represent a finite set of entities, relations, and timestamps. A temporal knowledge graph (TKG), $\mathcal{G} = \{(s, r, o, t)|s, o \in \mathcal{E}, r \in \mathcal{R}, t \in \mathcal{T}\} \subseteq \mathcal{E} \times \mathcal{R} \times \mathcal{E} \times \mathcal{T}$ is a relational graph that consists of a set finite of facts denoted with quadruples in the form of (s, r, o, t). A complete TKG $\mathcal{G}$ contains both the observed facts $\mathcal{G}_{obs}$ and the unobserved true facts $\mathcal{G}_{un}$. Two types of facts are independent, i.e., $\mathcal{G} = \mathcal{G}_{obs} \cup \mathcal{G}_{un}$, where $\mathcal{G}_{obs} \cap \mathcal{G}_{un} = \emptyset$. There are two types of LP tasks, i.e.

Given the observed facts $\mathcal{G}_{obs}$, TKG LP aims to predict the ground truth object (or subject) entities of LP queries $(s_q, r_q, ?, t_q)$ (or $(?, r_q, o_q, t_q)$), where $(s_q, r_q, o_q, t_q) \in \mathcal{G}_{un}$. In the interpolated LP, the prediction is based on all the observed facts $\{(s, r, o, t_i)|t_i \in \mathcal{T}\} \subseteq \mathcal{G}_{obs}$, while in the extrapolated LP, the prediction can only be based on the observed facts $\{(s, r, o, t_i)|t_i < t_q\} \subseteq \mathcal{G}_{obs}$ appearing before the query timestamp t_q. TKG interpolated LP is also named as TKG completion and TKG extrapolated LP is also named as TKG forecasting or TKG link forecasting.

1.2.2 Few-Shot Learning

As a machine learning problem, Few-shot learning (FSL) asks models to perform well on test examples for each class, with only a few (usually a small number, e.g., 1 or 3) labeled class-specific training examples. When the number of labeled examples equals 1, FSL problems become one-shot learning problems. Traditional machine learning methods fail to perform well in FSL problems since they require a large amount of training data concerning each class to achieve good performance. However, several approaches, e.g., metric learning, data augmentation, and meta-learning have shown success in addressing FSL problems. [42, 25]. In this paper, we use meta-learning framework to solve the FSL problem. Meta-learning approaches aim to quickly learn novel concepts by generalizing from previously encountered learning tasks, which fit well with solving the problems in the few-shot setting.

1.2.3 Episodic Training

Episodic training [40] is a meta-learning framework to solve FSL problems. Different from the traditional training process, models are trained over episodes in episodic training. Each episode can be considered as a mini-training process on a training task T, where a number of "training data" (denoted as the support set $\mathcal{S}$) and "test data" (denoted as the query set $\mathcal{Q}$) are sampled. Then, loss function l_θ is calculated over the query set conditioned on the support set. θ denotes the model parameters. A model is trained over a large dataset of training tasks with episodic training. For a given support set, the model needs to minimize a loss over a batch of examples in the query set. Assume there is a large set of training tasks $\mathbb{T} = \{T_i\}_{i=1}^{N}$, where $T_i = \{\mathcal{S}_i, \mathcal{Q}_i\}$ and N is the total number of the training tasks, the training objective of a model is given as $\theta = \arg\min_\theta \mathbb{E}_{T_i \sim \mathbb{T}} \left[\frac{1}{|\mathcal{Q}_i|} \sum_{q \in \mathcal{Q}_i} [l_\theta(q|\mathcal{S}_i)] \right]$. q denotes a data example in the query set $\mathcal{Q}_i$. Episodic training manages to simulate the few-shot situation when there are only a small number of data examples sampled to form the support set of each training task T. For example, to solve one-shot learning problems, only one data example is sampled to form the support set of each task T.

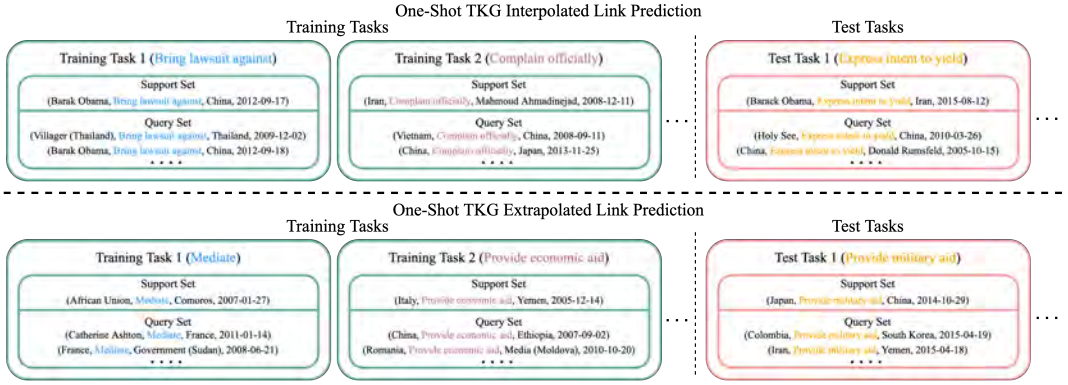

Figure. 3. Example of One-Shot TKG LP task.

Figure. 3 presents one example for each of the one-shot TKG LP tasks. Models are trained over a number of training tasks, then evaluated with validation and test tasks (we omit validation tasks in the figure for brevity). Each training (or validation, test) task T_r corresponds to a sparse relation r and consists of a support quadruple and a number of query quadruples that contain r. As shown in the examples, the timestamp of T_r's support quadruple is smaller than the minimum timestamp of T_r's query quadruples in the one-shot extrapolation setting, while there is no such constraint in the one-shot interpolation setting.

1.3 One-Shot Temporal Knowledge Graph Link Prediction Setup

Similar to the setting of few-shot KG LP [46], the one-shot TKG LP tasks (both interpolation and extrapolation) are taken as one-shot learning problems. Furthermore, we can formulate the one-shot TKG LP tasks into meta-learning problems and solve them with meta-learning approaches. First, we define the newly-proposed task, i.e., one-shot TKG interpolated LP. And we also redefine the one-shot TKG extrapolated LP task as it is flawed and unclear in [27].

For a TKG $\mathcal{G}$, we divide all its relations $\mathcal{R}$ into two groups, i.e., frequent relations $\mathcal{R}_{freq}$ and sparse relations $\mathcal{R}_{sp}$, where $\mathcal{R}_{freq} \cap \mathcal{R}_{sp} = \emptyset$ and $\mathcal{R} = \mathcal{R}_{freq} \cup \mathcal{R}_{sp}$. A background graph $\mathcal{G}' \subseteq \mathcal{E} \times \mathcal{R}_{freq} \times \mathcal{E} \times \mathcal{T}$ is constructed by including all the quadruples concerning frequent relations, where $\mathcal{G}' \subseteq \mathcal{G}$.

Definition 1 (One-Shot TKG Interpolated Link Prediction). Assume only one quadruple (s_0, r, o_0, t_0) corresponding to each sparse relation r is observed, where $r \in \mathcal{R}_{sp}$, $s_0, o_0 \in \mathcal{E}$ and $t_0 \in \mathcal{T}$. Given (s_0, r, o_0, t_0) and the whole background graph $\mathcal{G}'$, one-shot TKG interpolated LP aims to predict the missing entity of each LP query, i.e., $(s_q, r, ?, t_q)$ or $(?, r, o_q, t_q)$, derived from the unobserved quadruples containing r, where $s_q, o_q \in \mathcal{E}$ and $t_q \in \mathcal{T}$.

Definition 2 (One-Shot TKG Extrapolated Link Prediction). Assume only one quadruple (s_0, r, o_0, t_0) corresponding to each sparse relation r is observed, where $r \in \mathcal{R}_{sp}$, $s_0, o_0 \in \mathcal{E}$ and $t_0 \in \mathcal{T}$. Given (s_0, r, o_0, t_0), together with a set of observed TKG facts that appear before t_0 and belong to the background graph $\mathcal{G}'$, one-shot TKG extrapolated LP aims to predict the missing entity of each LP query, i.e., $(s_q, r, ?, t_q)$ or $(?, r, o_q, t_q)$, derived from the unobserved quadruples containing r, where $s_q, o_q \in \mathcal{E}$, $t_q \in \mathcal{T}$ and $t_q > t_0$.

Both subject and object entity prediction are of great importance in traditional KG and TKG LP, but previous work regarding few-shot link prediction on both KGs and TKGs [46, 8, 52, 34, 27] only consider object entity prediction. In this paper, we consider both subject and object entity prediction to make the task settings more reasonable and comprehensive.

We further define the one-shot TKG LP tasks with meta-learning framework. Following [46, 27], a background graph $\mathcal{G}' = \{(s, r, o, t) | s, o \in \mathcal{E}, r \in \mathcal{R}_{freq}, t \in \mathcal{T}\}$ is constructed by including all the quadruples concerning frequent relations, and it is also observable to the TKG reasoning model. We also assume that we have access to a set of training tasks for episodic training. Each training task T_r corresponds

to a sparse relation $r \in \mathcal{R}_{sp}^{train}$ ($\mathcal{R}_{sp}^{train} \subset \mathcal{R}_{sp}$). $T_r = \{\mathcal{S}_r, \mathcal{Q}_r\}$, where $\mathcal{S}_r$ is the support set of T_r containing only one support quadruple (s_0, r, o_0, t_0), and $\mathcal{Q}_r = \{(s_q, r, o_q, t_q)\}$ is the query set of T_r containing a number of r-related quadruples other than (s_0, r, o_0, t_0). The set of all training tasks is denoted as the meta-training set $\mathbb{T}_{meta-train}$. We use the loss function $l_\theta((s_q, r, o_q, t_q)|\mathcal{S}_r)$ to indicate how well the TKG reasoning model works on the query quadruple (s_q, r, o_q, t_q), given the support set $\mathcal{S}_r$. The training objective of the model is given as:

$$\theta = \arg\min_{\theta} \mathbb{E}_{T_r \sim \mathbb{T}_{meta-train}} \left[\frac{1}{|\mathcal{Q}_r|} \sum_{q \in \mathcal{Q}_r} [l_\theta(q|\mathcal{S}_r)] \right], \tag{1}$$

where θ denotes the model parameters, q represents a query quadruple (s_q, r, o_q, t_q). T_r is sampled from the meta-training set $\mathbb{T}_{meta-train}$, and $|\mathcal{Q}_r|$ denotes the number of the query quadruples regarding the sparse relation r. For every sparse relation r, we want our model to accurately predict the missing entities of all the LP queries $(s_q, r, ?, t_q)$ (or $(?, r, o_q, t_q)$) derived from query quadruples $(s_q, r, o_q, t_q) \in \mathcal{Q}_r$, with only one observed r-specific support quadruple (s_0, r, o_0, t_0) from $\mathcal{S}_r$.

After training, model will be evaluated on a meta-validation set $\mathbb{T}_{meta-valid}$ and a meta-test set $\mathbb{T}_{meta-test}$, where $\mathcal{R}_{sp}^{valid} \subset \mathcal{R}_{sp}$,$\mathcal{R}_{sp}^{test} \subset \mathcal{R}_{sp}$ and $\mathcal{R}_{sp}^{train} \cap \mathcal{R}_{sp}^{valid} = \phi, \mathcal{R}_{sp}^{train} \cap \mathcal{R}_{sp}^{test} = \phi, \mathcal{R}_{sp}^{valid} \cap \mathcal{R}_{sp}^{test} = \phi$. Similar to meta-training, for each sparse relation in meta-validation and meta-test, only one associated quadruple is added in the support set, and all the links in its query set are to be predicted.

In the interpolated LP, no constraint exists for the timestamp t_0 of its support quadruple (s_0, r, o_0, t_0) regarding sparse relation r and we assume that the whole background graph is always observable, while in the extrapolated LP, we imposed temporal constraint: $t_0 < \min(\{t_q|(s_q, r, o_q, t_q) \in \mathcal{Q}_r\})$ and only the background graph before the one-shot support quadruple is observable. Following the extrapolation setting in [27], we keep the non-overlapped time span (Figure 4) of meta-learning sets ($\mathbb{T}_{meta-train}$, $\mathbb{T}_{meta-valid}$, $\mathbb{T}_{meta-test}$). The maximum timestamp of the quadruples in $\mathbb{T}_{meta-train}$ is smaller than the minimum timestamp of the quadruples in $\mathbb{T}_{meta-valid}$, and the maximum timestamp of the quadruples in $\mathbb{T}_{meta-valid}$ is smaller than the minimum timestamp of the quadruples in $\mathbb{T}_{meta-test}$.

1.4 Contribution

Our work's main contribution is summarized as follows:

- We propose the one-shot TKG interpolated LP task. To the best of our

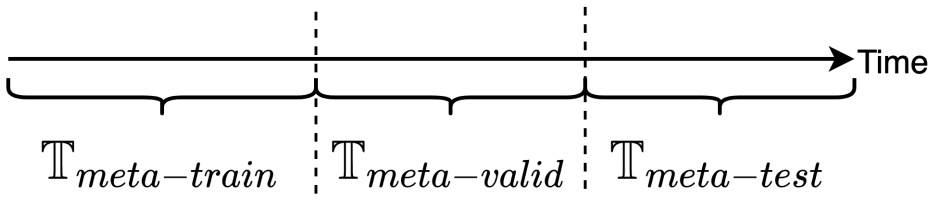

Figure. 4. Time span of meta learning sets ($\mathbb{T}_{meta-train}, \mathbb{T}_{meta-valid}, \mathbb{T}_{meta-test}$) for the extrapolated LP.

knowledge, this is the first work generalizing TKG interpolated LP to the one-shot setting for sparse relations LP tasks;

- We redefine the one-shot TKG extrapolated LP task and we conduct both subject and object entity prediction on the quadruples regarding sparse relations;
- We propose four new large-scale datasets for one-shot relational learning on TKGs. For every sparse relation, we have a substantial number of associated TKG facts, which greatly alleviates instability in model training and promotes reliable evaluation;
- We propose a model learning **m**eta representations of **o**ne-**s**ho**t** relations for solving both tasks in TKGs (MOST). We evaluate our model on all four new datasets and compare it with a bunch of baselines. Our model achieves state-of-the-art performance on all datasets in both tasks.

1.5 Outline of the Thesis

After the introduction, Chapter 2 reviews the background knowledge and the related works. Chapter 3 explains how we design our datasets and model. Details of the datasets and the results of our model experience will be introduced in Chapter 4. Finally, the conclusion is drawn and the outlook for future research is provided in Chapter 5.

2 Background and Related Work

2.1 Knowledge Graph Embedding Methods

Knowledge graph embedding (KGE) learns the embedding representation of entities and relations in a knowledge graph and serves as a key driver for KG reasoning tasks, e.g. KG link prediction. A line of KGE methods treat relations as translations between subject and object entities [5, 24, 35, 1], while another series of models calculate the plausibility of potential semantics of entities and relations based on tensor factorization[30, 51, 2]. On top of them, because graph neural networks (GNNs)[9, 19] can utilize structural information of KGs, a group of researchers develop neural-based relational graph encoders for KGE [33, 38]. Moreover, some researchers consider that multiple kinds of relations exist in the knowledge graph, and develop a line of rotation models, which treat relations as a rotation between the subject and object entities [36, 47]. We then introduce the representative models in each category separately.

2.1.1 Translation Models

For a KG triplet (s, r, o), translation model calculates the distance between the translated subject s and the object o by comparing the embedding representations $\mathbf{h_s} + \mathbf{h_r}$ with $\mathbf{h_o}$ where $\mathbf{h_s}$, $\mathbf{h_o}$ and $\mathbf{h_r}$ denote the embedding representation of subject s, object o and relation r in the KG triplet (s, r, o), respectively. Figure. 5 shows how the earliest translation model, TransE [5], calculate the distance.

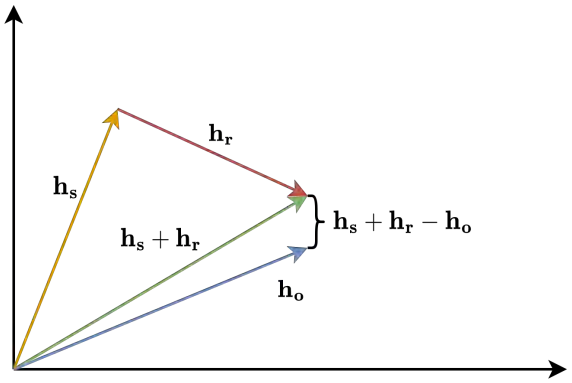

Figure. 5. TransE score function [5].

The yellow, blue and red arrows represent the embedding representation of subject

$\mathbf{h_s}$, object $\mathbf{h_o}$ and relation $\mathbf{h_r}$, respectively. The green arrow denotes the translated subject representation $\mathbf{h_s} + \mathbf{h_r}$. TransE considers the relation r as the translation relation between the subject and object entities, the distance is calculated by $\mathbf{h_s} + \mathbf{h_r} - \mathbf{h_o}$, which represents the plausibility of the KG triplet (s, r, o). The scoring function is defined as:

$$f(s, r, o) = \|\mathbf{h_s} + \mathbf{h_r} - \mathbf{h_o}\| \tag{2}$$

with the optimization objective of minimizing the scoring function and the norm can be either L1 norm or L2 norm. The smaller value represents higher plausibility of the KG triplet (s, r, o). TransE is able to solve the 1-1 category of relations, but not well for the 1-N, N-1, N-N relations. When the subject entity and the relation are the same, TransE assumes that all object entities have the same embedding information. For example, for the two triples (*Jonathan Nolan*, *directed*, *Revenant*) and (*Jonathan Nolan*, *directed*, *Westworld*), two object entities *Revenant* and *Westworld* have the same representation by TransE as the two triplets have the same subject entity and relation. To address this problem, TransR [43] introduces separated spaces for entities and relations. After projection, the embedding information of the object entity will be different, even though the subject entity and the relation are the same.

In addition to TransE and TransH, there are other Trans models that take into account the probabilistic and sparse nature of entities and relations. In general, Trans models treat relations as translations between subject and object entities, solving the N-to-N problem that exists in knowledge graphs.

2.1.2 Factorization Models

KGE models can be decomposed as three-way tensor $\mathcal{X}$ from the perspective of factorization. With relation matrix $\mathbf{M_r}$, an overall principle of tensor factorization can be referred to as:

$$f(s, r, o) = \mathbf{h_s}^T \mathbf{M}_r \mathbf{h_o} \tag{3}$$

Nickel et al. [30] proposed the three-way rank-r factorization model RESCAL. RESCAL represents relations using full-rank matrices. The information of entities and relations can deeply interact. However, RESCAL is prone to overfitting and can be very complex as the dimensionality of the relation matrix increases, making it difficult to apply to large-scale knowledge graphs. DistMult [51] relaxes the constraints on the relational matrix by representing the relational matrix $\mathbf{M_r}$ using

a diagonal matrix, and redefine the loss function:

$$f(s,r,o) = \mathbf{h_s}^T diag(\mathbf{M}_r)\mathbf{h_o} \tag{4}$$

However, DisMult oversimplifies the RESCAL model, resulting in only being able to solve the symmetric relations that exist in the knowledge graph. ComplEx [37] extends DisMult to a complex space and defines the scoring function as:

$$f(s,r,o) = Re\left(\mathbf{h_s}^T diag(\mathbf{M}_r)\mathbf{h_o}\right) \tag{5}$$

where all $\mathbf{h_s}$, $\mathbf{h_o}$ are represented by complex numbers, $\bar{\mathbf{h}_\mathbf{s}}$ denotes the conjugate complex, and $Re(\cdot)$ denotes the real part of the complex obtained. ComplEx can solve both symmetric and asymmetric relations by projecting embeddings into complex vector space.

Models in the factorization family such as RESCAL, DistMult, and ComplEx can be transformed from one into another with certain constraints [41].

2.1.3 Neural Network Models

Most translation models and factorization models are proposed before 2016. In recent years, with the popularity of neural networks, there are also models that use neural networks to solve KGE problems, including ConvE, CapsE.

ConvE first converts the head entities and relations into vectors, then uses the convolutional and fully connected layers to obtain interaction information. After that, ConvE uses it with the matrix W and tail entities to compute the plausibility score of the current events. ConvE employs the score function:

$$f\left(vec(f([\mathbf{h_s}, \mathbf{h_r}] * w))\mathbf{W}\right)\mathbf{h_o} \tag{6}$$

where $\mathbf{h_s}$,$\mathbf{h_r}$ and $\mathbf{h_o}$ denote the representation of subject, relation and object, w is the convolutional kernel, $\mathbf{W}$ is the weight matrix, and vec is the vectorization operator.

As an emerging direction, Graph neural networks (GNNs) show a great potential in non-Euclidean data. And GNNs also have a natural connection with KGE. We will discuss this in the later sections.

2.1.4 Rotation Models

rotation models hold the opinion that there are multiple types of relations in the knowledge base, such as symmetry(e.g., marriage), antisymmetry(e.g., filiation), inversion(e.g., hypernym and hyponym), composition(e.g., my mother's husband is my father) relations [36] , but previous models such as TransE[5], and RESCAL[30] are unable to resolve these relations.

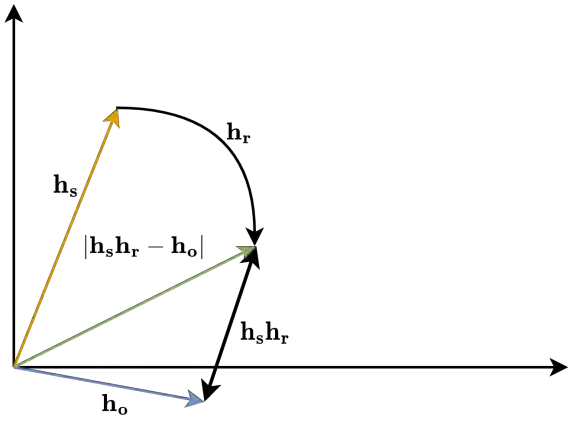

Figure. 6. RotatE score function [36].

As Figure 6 shows, RotatE proposes modeling in complex space, treating the relation as a rotation between subject and object entities, and the score function is defined as:

$$f_r(\mathbf{h_s}, \mathbf{h_o}) = ||\mathbf{h_s} \circ \mathbf{h_r} - \mathbf{h_o}|| \tag{7}$$

where $\{\mathbf{h_s}, \mathbf{h_r}, \mathbf{h_o}\} = e^{i\theta} = cos\theta + isin\theta$. RotatE has been theoretically proven to be able to solve different relations, e.g. symmetric, antisymmetric, flip. In addition, RotatE also believes that many triples are obviously wrong during the training process, thus RotatE proposes a self-adversarial negative sampling method to make the wrong samples more obvious.

RotatE project the embedding space in a two-dimensional complex plane space, so it can be naturally extended to a higher-dimensional complex plane space. QuatE [53] uses quaternions for rotation and defines the score function as:

$$f_r(\mathbf{h_s}, \mathbf{h_o}) = \mathbf{h_s} \otimes ||\mathbf{h_r}|| * \mathbf{h_o} \tag{8}$$

$\mathbf{h_s}, \mathbf{h_r}, \mathbf{h_o}$ are all quaternion and $||\cdot||$ is the norm operator, $\otimes$,$*$ represent the Hamilton product and inner product, respectively.

It can be seen from the RotatE and QuatE that the characteristics of rotation are useful when solving the symmetric/antisymmetric, flip, and combination relations in the knowledge base.

2.2 Deep Neural Networks

Deep Neural Network (DNN) is now one of the most representative models in the field of artificial intelligence due to its outstanding performance on a wide range of different tasks. The deep belief networks [14] proposed by Geoffrey E. Hinton et al. in 2006 can be seen as the foundation of modern DNN models. In the human brain, there are 100 billion neurons to receive and send electric signals. DNN try to imitate this process by using the basic component, the neuron.

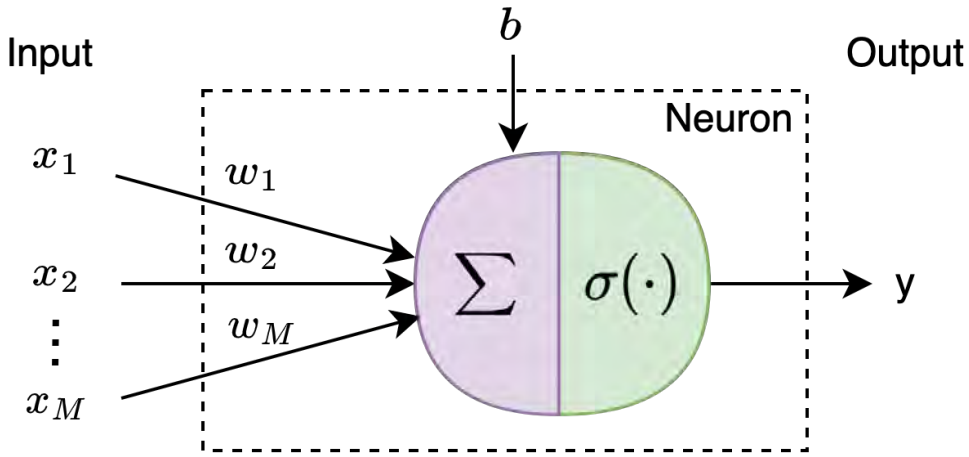

Figure. 7. An example of Neuron.

An example of a neuron is shown in Figure 7 and the computation in a neuron consists of three steps, i.e. weighted sum, add bias, and non-linear transformation.

$$y = \sigma\left(\sum_{k=1}^{M} w_k x_k + b\right) \tag{9}$$

For each input x, it will be multiplied with a weight w, then a bias term b will be added in the weighted sum. The intermediate number $\sum_{k=1}^{M} w_k x_k + b$ is a linear combination of all the inputs. With an activation function $\sigma(\cdot)$, the non-linearity is introduced to the linear combination, which is more suitable for complex tasks.

A neural network is formed with a number of neurons. Figure 8 shows an example of three layer neural network with one hidden layer. More general, each neural network has an input layer, several hidden layers and an output layer. The output of the neural network can be computed in the following way:

$$y = \sigma\left(W_2^T \sigma(W_1^T x + b_1) + b_2\right) \tag{10}$$

where x denotes all inputs and W_1, W_2, b_1, b_2 denote the learnable parameters, i.e. weight and biases, in each layer of the neural network.

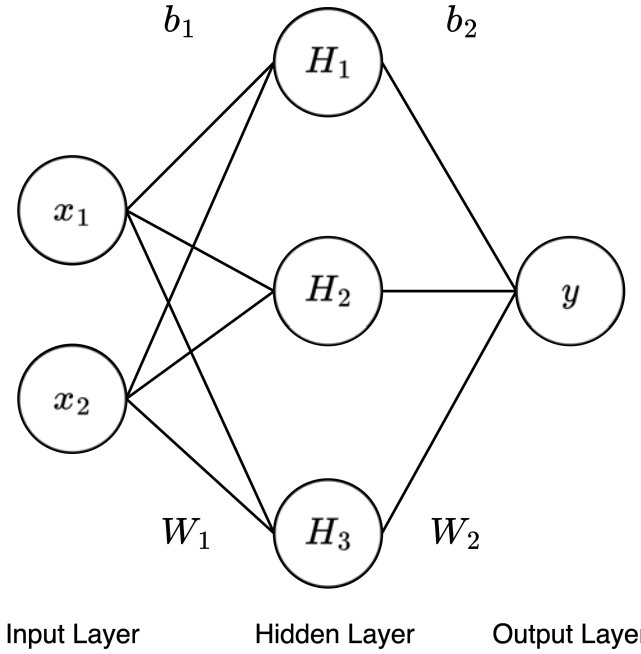

Figure. 8. An example of two layer neural network.

In real world cases, the number of the hidden layers is greatly larger than one. This makes the neural network deeper and a neural network becomes a DNN.

2.3 Graph Neural Networks

Over the past few years, the rise and application of deep learning (DL) have successfully driven research in pattern recognition and data mining. Although traditional deep learning methods have been applied to extract features from Euclidean space data with great success, many real-world application scenarios generate data

from non-Euclidean spaces. Traditional deep learning methods still struggle to perform satisfactorily with non-Euclidean space data. For example, in e-commerce, a graph-based learning system can use the interaction between users and products to make very accurate recommendations [45].

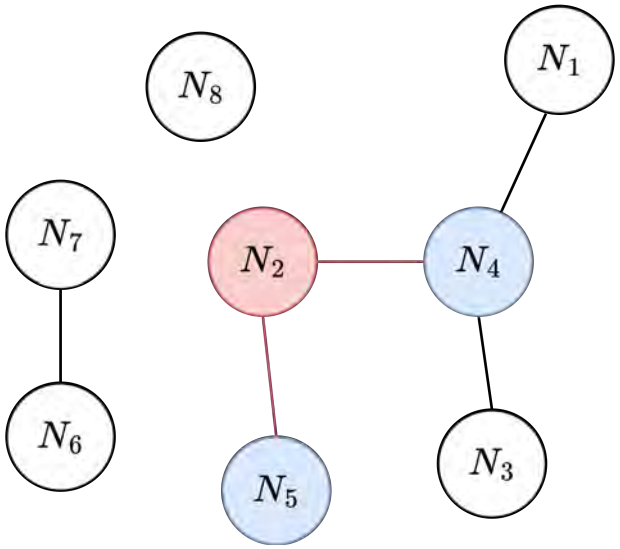

Figure. 9. An example of Graph $\mathcal{G}$.

Figure 9 shows an example of a graph. The graph $\mathcal{G}$ has eight nodes and the edges between nodes are denoted with lines. It can be observed that nodes in $\mathcal{G}$ have both dependent and independent relations. As for the red node N_2, we mark its neighbor with blue and the neighbor is defined as the set of nodes that connect with N_2. The neighbor size of N_2 is two, since the neighbor of it consists of two nodes. As for node N_8, it is independent with any other nodes in $\mathcal{G}$.

However, compared to text and images, this network type of unstructured data is very complex and the difficulties in dealing with it include:

- The size of the graph is arbitrary, the topology of the graph is complex and there is no spatial localization as in the case of images.
- Each graph has an unordered node with variable size and each node in the graph has a different number of neighboring nodes, which makes important operations (e.g. convolution) no longer suitable for direct use.
- Graphs are often dynamic and contain multi-modal features.

These problems have led to the emergence and development of graph neural networks (GNNs). GNNs take advantage of this special graph data structure and have flourished in recent years.

2.4 Graph Convolutional Networks

Due to the special data structure, we can not directly apply convolution operation in graph data. GCNs extend convolution operations from traditional data (e.g. images) to graph data. The main idea is to learn a mapping of function $f(\cdot)$, through which a node in the mapping graph can aggregate its own features with those of its neighbors to generate a new representation of the node. GCNs are the basis for many complex graph neural network models. GCNs can be further divided into two main categories, spectral-based and spatial-based. Spectral-based methods define graph convolution by introducing filters from the perspective of graph signal processing, where the graph convolution operation is interpreted as removing noise from the graph signal. The spatial-based approach represents graph convolution as the aggregation of feature information from the neighborhood, and when the algorithm of the graph convolution network operates at the node level, the graph pooling module can be interleaved with the graph convolution layer to coarsen the graph into high-level substructures.

2.4.1 Spectral-based Graph Convolutional Network

Given a graph $\mathcal{G} = (\mathcal{N}, \mathcal{E})$, where $\mathcal{N}$, $\mathcal{E}$ denote nodes and edges in the graph, respectively. In spectral-based graph neural networks, graphs are assumed to be undirected graphs, and a robust mathematical representation of undirected graphs is the regularized graph Laplacian matrix, i.e.

$$\mathbf{L} = \mathbf{I} - \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2} \tag{11}$$

where $\mathbf{D}$ is the diagonal matrix and $\mathbf{D}_{ii} = \sum_j(\mathbf{A_{i,j}})$, $\mathbf{A} \in \mathbb{R}^{|\mathcal{N}|\times|\mathcal{N}|}$ is the adjacency matrix of $\mathcal{G}$. regularised Laplacian matrices have the property of being real symmetric and semi-positive definite. Using this property, the regularised Laplacian matrix can be decomposed into:

$$\mathbf{L} = \mathbf{U}\Lambda\mathbf{U}^T \tag{12}$$

$\mathbf{U} = [u_0, u_1, ..., u_{n-1}] \in \mathbb{R}^{|\mathcal{N}|\times|\mathcal{N}|}$ is a diagonal matrix where the values on the diagonal are the eigenvalues of L. The eigenvectors of the regularized Laplacian matrix form a set of orthogonal bases.

Generally, spectral-based graph convolutional networks can be represented:

$$\mathbf{L}^{k+1} = \mathbf{U}_{filter}\mathbf{U}^T\mathbf{L}^k \tag{13}$$

In the k^{th} graph convolutional layer, the inputs $\mathbf{L}^k$ will first be projected onto an orthogonal space whose base is composed of the eigenvectors of the Laplacian matrix by multiplying by $\mathbf{U}^T$. After that, a filter $\mathbf{U}_{filter}$ will be used to process the projected inputs. In the end, the processed inputs will be projected back to the original space. Existing spectral-based graph convolution network models, e.g., Spectral CNN [7], Chebyshev Spectral CNN (ChebNet) [9] and Adaptive Graph Convolution Network (AGCN) [23] all follow this pattern. The key difference between them lies in the different filters chosen.

A common disadvantage of spectral-based graph convolution neural network approaches is that they require the entire graph to be loaded into memory in order to perform graph convolution, which is not efficient when processing large graphs.

2.4.2 Spatial-based Graph Convolutional Networks

The idea of spatially based graph convolutional neural networks is mainly derived from traditional convolutional neural networks for image convolution operations, the difference is that spatially based graph convolutional neural networks define the graph convolution based on the spatial relation of the nodes. Figure 10 depicts the difference between the two convolutions.

An image can be regarded as a special form of graphs, where each pixel represents a node, as shown in left part of Figure 10, we use a pixels matrix to illustrate a part of an image. Each pixel is directly connected to the pixels in its neighborhood. Through a 3 x 3 filter (kernel), the neighborhood of each node is the eight pixels around it. The eight blue pixels represent the read node's neighbors. A filter is then applied to this 3×3 window by applying a weighted average to the pixel values of the central red node and its neighboring nodes on each channel. Due to the specific order of neighboring nodes, trainable weights can be shared at different locations. Similarly, for a general graph, space-based graph convolution aggregates the central node representation and the neighboring node representation to obtain

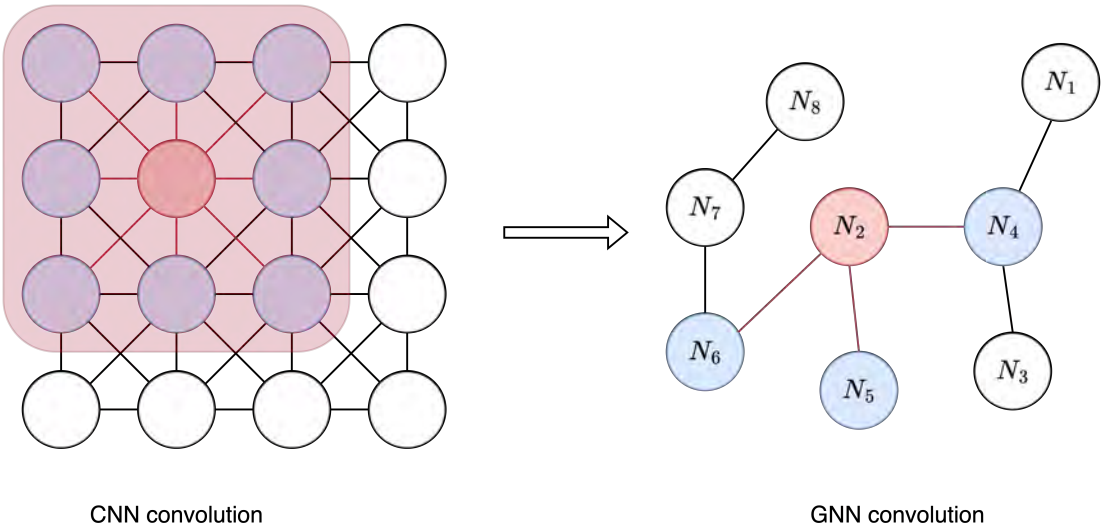

Figure. 10. CNN convolution and GNN convolution [16].

a new representation of that node, as shown in the right part of Figure 10. The hidden layer representation of each node v is computed by the following equation:

$$\mathbf{h}_v^{k+1} = \sigma\left(\sum_{u \in \mathbb{N}_v} \frac{1}{c_v} \mathbf{W}^k \mathbf{h}_u^k\right) \tag{14}$$

where $\frac{1}{c_v}$ is the regularization constants, $\mathbf{h}_v$, $\mathbf{h}_u$, $\mathbf{W}$ are representations for the central node and its neighbors, and the weight between them, respectively.

2.5 Graph Attention Networks

The attention mechanism [39] is now widely used in sequence-based tasks and has the advantage of being able to amplify the impact of the most important parts of the data. This feature has proved useful for many tasks, such as machine translation and natural language understanding.

2.5.1 Graph Attention Network

Graph Attention Network is a spatially based graph convolutional network that uses the attention mechanism [39] to determine the weights of node neighbor-

hoods when aggregating feature information. The graph convolutional operation is defined as:

$$\mathbf{L}_v^k = \sigma\left(\sum_{u \in \mathbb{N}_v} \alpha_{vu}^k \mathbf{W}^k \mathbf{L}_u^{k-1}\right) \tag{15}$$

where $\mathbf{L_u^{k-1}}$ is the output last layer and α_{vu} is the attention weight, which represents the strength of the connection between neighbor u and central node v. To learn the attention weights in different subspaces, GAT can also use multi-head attention:

$$\mathbf{L}_v^k = ||_{n=1}^N \sigma\left(\sum_{u \in \mathbb{N}(v) \cup v} \alpha_{vu}^{nk} \mathbf{W}^k \mathbf{L}_u^{k-1}\right) \tag{16}$$

where N is the number of attention head, α_{vu}^{nk} is the n^{th} attention weight between node v and node u in the k^{th} layer.

2.5.2 Gated Attention Network

While a multi-head attention aggregator can explore multiple representation subspaces between a central node and its neighborhood, not all of these subspaces are equally important. Some subspaces may not even exist at some nodes. Inputting an output that captures the attention of a useless representation can mislead the final prediction of the model.

The Gated Attention Network (GAAN) also employs a multi-head attention mechanism to update the hidden state of nodes, it further calculates an additional soft gate between 0 (low importance) and 1 (high importance), assigning a different importance to each head. The update rule is defined as:

$$\begin{aligned} \mathbf{y}_v &= \phi_{\theta_0}\left(\mathbf{x}_v \oplus \|_{n=1}^N (\mathbf{f}_v^n \sum_{u \in \mathbb{N}_v} w_{v,u}^n \phi_{\theta_v^n}^h(\mathbf{z}_u))\right) \\ \mathbf{f}_v &= \psi_f\left(\mathbf{x}_v, \mathbf{z}_{u \in \mathbb{N}_v}\right) \end{aligned} \tag{17}$$

where N is the number of attention head, $w_{v,u}^n$ is the n^{th} attention weight between nodes v and its neighbor node u, $\mathbf{f}_v$, a scalar, is the gate value at the n^{th} head of

node v. To ensure that the added gate does not introduce too many additional parameters, GAAN use a convolutional network $\psi_f(\cdot)$, which takes the features of the central node and neighboring nodes to generate the gate values.

Both GAT and GAAN can be considered as spatial-based graph convolutional networks. It is an advantage that they are able to learn the importance weights of their neighbors adaptively. However, the computational cost and memory consumption increase rapidly with the computation of attention weights between each pair of neighbors.

2.6 Multi-Relational Graph Convolutional Networks

In the real world, knowledge graphs are multi-relational graphs. Traditional GCN algorithms are widely used for homogeneous graphs, which are far from adequate for knowledge graph needs.

To solve this problem, Michael et al. propose R-GCN [5] by learning the relation representation together with node representations in multi-relational graphs. In traditional GCN, the weight $\mathbf{W}^k$ in 14, is shared with all the nodes. In contrast, R-GCN adopts different weights for different relations, and nodes that share the same relation type use the same mapping weight $\mathbf{W}_r$. The hidden layer representation of each node v is updated:

$$\mathbf{h}_v^{k+1} = \sigma\left(\mathbf{W}_0^k \mathbf{h}_v^k + \sum_r \sum_{u \in \mathbb{N}_v^r} \frac{1}{c_{v,r}} \mathbf{W}_r^k \mathbf{h}_u^k\right) \tag{18}$$

where $c_{v,r}$ is a constant, $\mathbb{N}_v^r$ denotes the set of neighbor of node v that share the same relation r, $\mathbf{h}$ denotes the node representations, $\mathbf{W}_r$ is the specific weight for the relation r, and $\mathbf{W}_0$ is the weight for the self-connection of each node.

However, R-GCN has a huge drawback of introducing too many relationship matrix $\mathbf{W}_r$ as the number of relations increases. This leads to an explosion of parameters thus the model can not be trained.

On top of R-GCN, COMPGCN learns entity and relation representations for multi-relational graphs in the following way:

$$\mathbf{h}_v^{k+1} = f\left(\sum_r \sum_{u \in \mathbb{N}_v^r} \mathbf{W}_{\lambda_r}^k \phi(\mathbf{h}_u^k, \mathbf{h}_r^k)\right) \tag{19}$$

where $\phi(\cdot)$ denotes a composition operators and $\mathbf{W}_{\lambda_r}^k$ is the weight matrix to encode the relations.

Three different composition operator are introduced:

- Subtraction: $\phi(\mathbf{h}_u, \mathbf{h}_r) = \mathbf{h}_u - \mathbf{h}_r$
- Multiplication: $\phi(\mathbf{h}_u, \mathbf{h}_r) = \mathbf{h}_u * \mathbf{h}_r$
- Circular-correlation: $\phi(\mathbf{h}_u, \mathbf{h}_r) = \mathbf{h}_u \star \mathbf{h}_r$

Subtraction, multiplication and circular-correlation correspond to TransE [5], DistMult [51] and HolE [29], respectively. In terms of experimental results, TransE is the fastest but least effective and HolE is the slowest but most effective. Besides the forward relation r, COMPGCN also adds inverse and self-connect relations to enhance the model. Weight matrix $\mathbf{W}_{\lambda_r}$ is defined as:

- Forward: $\mathbf{W}_{\lambda_r} = \mathbf{W}_O, r \in \mathcal{R}$
- Inverse: $\mathbf{W}_{\lambda_r} = \mathbf{W}_I, r \in \mathcal{R}_{inv}$
- Self-connect: $\mathbf{W}_{\lambda_r} = \mathbf{W}_S, r = \textit{self-connect}$

where $\mathcal{R}$ denotes the set of relations r, $\mathcal{R}_{inv}$ denotes all the inverse relations r_{-1}. In each layer of iteration, in addition to updates to the node embedding, the relation embedding also needs to be updated. COMPGCN uses a relatively simple way to update the relation embedding by introducing $\mathbf{W}_{rel}$.

$$\mathbf{h}_r^{k+1} = \mathbf{W}_{rel}^k \mathbf{h}_r^k \tag{20}$$

COMPGCN uses the encoder-decoder framework proposed by R-GCN. As for decoders, COMPGCN adopts existing KG models, e.g. ConvE [10] and Distmult [51]. Since COMPGCN introduces relation embedding in the encoder stage, it performs better on KG link prediction task.

2.7 Temporal Knowledge Graph Embedding Methods

In recent years, an increasing interest has been shown in developing temporal KGE methods for TKGs. Many existing methods derive time-aware KG scoring functions. Lacroix et al. [20] propose TComplEx by extending the ComplEx embedding model with new regularization components that take into account the temporal information. Ma et al. [26] propose a pattern to generalize static knowledge graphs to temporal knowledge graphs. Leblay et al. [21] show the importance of incorporating side information in the learning process.

Another series of methods employ recurrent modules to autoregressively encoder temporal dependencies between TKG events. RE-NET [17] uses a recurrent event encoder to model the time-conditional joint probability distribution of event sequences and equips the event encoding with a neighborhood aggregator to model concurrent events within the time window associated with each entity. It infers graphical sequences of future timestamps by sequentially sampling from the learned joint distribution. Experiments demonstrate a significant advantage of RE-NET for multi-step inference of future timestamps. In TKGs, temporal information is sparse and the distribution of entity is with great variance. Wu et al. [44] propose TeMP to explicitly solve the problem with temporal dynamics models combined with graph neural networks and data imputation.

Apart from them, it has been proven to be effective to sample temporal neighboring graph for TKG entities and learn contextualized time-aware entity representations. Han et al. [12] propose the model xERTE, which can reason about query-related subgraphs of the TKG and jointly model graph structure with temporal contextual information, ultimately predicting links of future events. T-GAP [18] proposes a new encoder that can efficiently capture query-related information from the temporal knowledge graph. Ding et al. [11] propose a parameter-efficient model TARGCN to utilize the information from the whole temporal context. By considering temporal information, temporal KGE methods outperform static KGE methods in TKG reasoning tasks, e.g., TKG completion.

2.8 Few-shot Relational Learning Methods for Knowledge Graphs

Classification of relations between entities is an important part of the Knowledge Graph Completion (KGC) task. The current approach assumes that a large number of training samples exist for each relations for training, but in reality, there are

a large number of long-tailed relations in the knowledge graph training set, which makes training not as effective as expected. Moreover, predictions for arbitrary new relations can not be made without sufficient training data. To solve this problem, Xiong et al. [46] propose a meta-learning based method Gmatching as the first work introducing few-shot relational learning into the context of KG. Gmatching employs a neighbor encoder to learn a representation of the target entity through the one-hop structure of the knowledge graph. The matching processor uses a matching metric function that can compute the similarity score between the query samples and the support sample. On top of Gmatching, Chen et al. [8] follow the meta-learning framework and propose MetaR. Instead of obtaining relation representation through the process of knowledge graph encoder, MetaR employs a relation-meta learner to learn relational meta-information from subject and object entities. Sheng et al. [34] propose a few-shot relation learning model (FSRL) to solve the small samples link prediction task. FSRL first employs a relation-aware heterogeneous neighbor encoder to learn the entities representations, then a recurrent autoencoder network is designed to model the relations between small samples of entities. After obtaining the representation of reference set , a matching network can eventually be used to find similar entity pairs. Based on Gmatching, FAAN [34] presents an adaptive neighbor encoder for entities to extract information from the examples. It further employs a transformer based encoder to learn adaptive query-aware entity representations, which helps to better differentiate supporting information from entities' neighborhoods. Similar to FAAN, GANA [31] presents a gated attentional aggregator for learning contextualized entity representations. A novel MTransH scoring function is designed for modeling complex relations and it contributes greatly to the model performance.

OAT [27] is the first method developed for one-shot relational learning for TKGs. A Transformer-based [39] history encoder is employed to encode historical information and generate time-aware entity representations. Coupled with a multi-layer feed-forward neural network, entity representations of the one-shot example are used to compute the plausibility scores of TKG facts. In this work, Mirtaher et al. propose an extrapolation LP task for TKGs in the one-shot setting, together with two datasets constructed from the subsets of two benchmark TKG databases, i.e., ICEWS [6] and GDELT [22].

3 Meta Representations of One-shot Relations

We propose a metric-based meta-learning model, i.e., **m**eta representations of **o**ne-**s**ho**t** relations (MOST), to solve both one-shot TKG interpolated and extrapo-

lated LP. MOST consists of three components: (1) Meta-information extractor, (2) Meta-relational decoder and (3) Metric function.

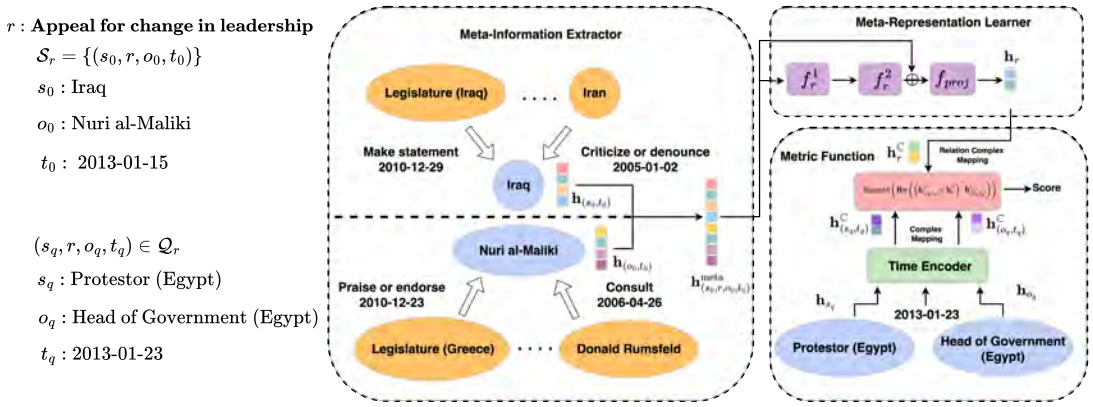

Figure. 11. Overview of MOST.

Figure 11 shows the overview of MOST. Given a support quadruple (s_0, r, o_0, t_0) of the sparse relation r, MOST extracts the meta-information from the time-aware representations of support entities s_0, o_0 with a meta-information extractor, and uses a meta-representation learner to learn r's meta-representation based on the support quadruple. Then MOST employs a metric function to compute the plausibility scores of TKG quadruples concerning r. Assume we have a sparse relation *Appeal for change in leadership*, an associated support quadruple (*Iraq*, *Appeal for change in leadership*, *Nuri al-Maliki*, 2013-01-15) $\in \mathcal{S}_r$ and an associated query quadruple (*Protestor (Egypt)*, *Appeal for change in leadership*, *Head of Government (Egypt)*, 2013-01-23) $\in \mathcal{Q}_r$. In Time-aware relational graph encoder, MOST computes contextualized time-aware representations of the nearest observable temporal neighbors of support entities, i.e., *Iraq* and *Nuri al-Maliki*. Then MOST generates a meta-representation for the sparse relation by concatenating support entity representations and going through the meta-representation learner. When the model derives a link prediction query (*Protestor (Egypt)*, *Appeal for change in leadership*, ?, 2013-01-23), all candidate entity representations are updated with a time encoder and the scoring function will compute the plausibility scores of the quadruples with different candidates. We then introduce more details about each component below.

3.1 Meta-information Extractor

For all the relations $\mathcal{R} = \{\mathcal{R}_{sp}, \mathcal{R}_{freq}\}$ in a TKG, $\mathcal{R}_{sp}$ denotes all the sparse relations and $\mathcal{R}_{freq}$ denotes all the frequent relations. For a meta learning task

T_r, we have a support quadruple $(s_0, r, o_0, t_0) \in \mathcal{S}_r$, and several query quadruples $(s_q, r, o_q, t_q) \in \mathcal{Q}_r$. First we infer the representation of r with the support quadruple by utilizing the contextualized entity representation of s_0 and o_0. To learn the contextualized time-aware entity representations of support entities, MOST first employs a time-aware relational graph encoder to find the temporal neighbors for every support entity (s_0 or o_0) by searching for available background facts whose object entity corresponds to the support entity. e.g., s_0's temporal neighborhood is denoted as $\mathcal{N}_{s_0} = \{(e', r', t') | r' \in \mathcal{R}_{freq}, (e', r', s_0, t') \in \mathcal{G}'\}$.

We keep a fixed number of temporal neighbors nearest to the support timestamp t_0.(The kept neighbors are prior to t_0 for one-shot TKG extrapolated LP) and we set the number of sampled neighbors as a tunable hyperparameter. We denote the filtered neighborhood as $\tilde{\mathcal{N}}_{s_0}$ and $\tilde{\mathcal{N}}_{o_0}$. By aggregating the information provided by their temporal neighbors, MOST calculates the time-aware entity representations $\mathbf{h}_{(s_0,t_0)}$, $\mathbf{h}_{(o_0,t_0)}$ of s_0, o_0 as follows:

$$\begin{aligned}
\mathbf{h}_{(s_0,t_0)} &= \mathbf{h}_{s_0} + \delta_1 \sigma\Big(\frac{1}{|\tilde{\mathcal{N}}_{s_0}|} \sum_{(e',r',t') \in \tilde{\mathcal{N}}_{s_0}} \mathbf{W}_g \left(f(\mathbf{h}_{e'} \| \mathbf{\Phi}(t')) \circ \mathbf{h}_{r'}\right)\Big), \\
\mathbf{h}_{(o_0,t_0)} &= \mathbf{h}_{o_0} + \delta_1 \sigma\Big(\frac{1}{|\tilde{\mathcal{N}}_{o_0}|} \sum_{(e',r',t') \in \tilde{\mathcal{N}}_{o_0}} \mathbf{W}_g \left(f(\mathbf{h}_{e'} \| \mathbf{\Phi}(t')) \circ \mathbf{h}_{r'}\right)\Big).
\end{aligned} \tag{21}$$

where d is the dimension of the representations. $\mathbf{h}_{s_0} \in \mathbb{R}^d$ and $\mathbf{h}_{o_0} \in \mathbb{R}^d$ denote the time-invariant entity representations of s_0 and o_0. $\mathbf{h}_{r'} \in \mathbb{R}^d$ denotes the relation representation of the frequent relation r'. $\circ$, $\|$ represent Hadamard product and concatenation operation, respectively. $\mathbf{W}_g \in \mathbb{R}^{d \times d}$ is a weight matrix that processes the information in the graph aggregation. $f : \mathbb{R}^{2d} \mapsto \mathbb{R}^d$ is a layer of feed-forward neural network. δ_1 is a trainable parameter indicating how much information from the temporal neighbors is included in updating entity representations. σ is an activation function. $\mathbf{\Phi}(t')$ denotes the time encoding function that encodes timestamp t' as $\mathbf{\Phi}(t') = \sqrt{\frac{1}{d}}[cos(\omega_1 t' + \phi_1), \ldots, cos(\omega_d t' + \phi_d))]$, where $\omega_1 \ldots \omega_d$ and $\phi_1 \ldots \phi_d$ are trainable parameters. We call our model with this time encoder as MOST-TA. Besides, we also design another model variant MOST-TD by inputting $t_0 - t'$ instead of absolute t' into the time encoder.

$$\begin{aligned}
\mathbf{h}_{(s_0,t_0)} &= \mathbf{h}_{s_0} + \delta_1 \sigma\Big(\frac{1}{|\tilde{\mathcal{N}}_{s_0}|} \sum_{(e',r',t') \in \tilde{\mathcal{N}}_{s_0}} \mathbf{W}_g \left(f(\mathbf{h}_{e'} \| \mathbf{\Phi}(t_o - t')) \circ \mathbf{h}_{r'}\right)\Big), \\
\mathbf{h}_{(o_0,t_0)} &= \mathbf{h}_{o_0} + \delta_1 \sigma\Big(\frac{1}{|\tilde{\mathcal{N}}_{o_0}|} \sum_{(e',r',t') \in \tilde{\mathcal{N}}_{o_0}} \mathbf{W}_g \left(f(\mathbf{h}_{e'} \| \mathbf{\Phi}(t_0 - t')) \circ \mathbf{h}_{r'}\right)\Big).
\end{aligned} \tag{22}$$

After obtaining $\mathbf{h}_{(s_0,t_0)}$ and $\mathbf{h}_{(o_0,t_0)}$, we calculate the meta-information of r as follows:

$$\mathbf{h}^{\text{meta}}_{(s_0,r,o_0,t_0)} = \mathbf{h}_{(s_0,t_0)} \| \mathbf{h}_{(o_0,t_0)}. \tag{23}$$

$\mathbf{h}^{\text{meta}}_{(s_0,r,o_0,t_0)} \in \mathbb{R}^{2d}$ represents the meta-information of r, given the support quadruple (s_0, r, o_0, t_0).

3.2 Meta-representation Learner

MOST employs the meta-representation learner to derive the meta-representation of r given the meta-information. $\mathbf{h}^{\text{meta}}_{(s_0,r,o_0,t_0)}$

$$\mathbf{h}_r = f_{\text{proj}}\left(\mathbf{h}^{\text{meta}}_{(s_0,r,o_0,t_0)} + f_r^2\left(\sigma\left(f_r^1\left(\mathbf{h}^{\text{meta}}_{(s_0,r,o_0,t_0)}\right)\right)\right)\right), \tag{24}$$

where $f_r^1 : \mathbb{R}^{2d} \mapsto \mathbb{R}^{4d}$, $f_r^2 : \mathbb{R}^{4d} \mapsto \mathbb{R}^{2d}$, $f_{\text{proj}} : \mathbb{R}^{2d} \mapsto \mathbb{R}^{\frac{d}{2}}$ are three single layer neural networks. The meta-representation $\mathbf{h}_r \in \mathbb{R}^{\frac{d}{2}}$ will then be used in the metric function to compute the scores of the TKG quadruples.

3.3 Metric Function

MOST employs the metric function to compute the plausibility score for the query quadruple (s_q, r, o_q, t_q). We first derive time-aware entity representations $\mathbf{h}_{(s_q,t_q)}$, $\mathbf{h}_{(o_q,t_q)}$ of the query entities s_q, o_q at t_q as follows:

$$\begin{aligned} \mathbf{h}_{(s_q,t_q)} &= \mathbf{h}_{s_q} + \delta_2 f(\mathbf{h}_{s_q} \| \mathbf{\Phi}(t_q)), \\ \mathbf{h}_{(o_q,t_q)} &= \mathbf{h}_{o_q} + \delta_2 f(\mathbf{h}_{o_q} \| \mathbf{\Phi}(t_q)). \end{aligned} \tag{25}$$

$\mathbf{h}_{s_q} \in \mathbb{R}^d$ and $\mathbf{h}_{o_q} \in \mathbb{R}^d$ denote the time-invariant entity representations of s_q and o_q. δ_2 is a trainable parameter that controls the amount of the injected temporal information. Unlike Equation 21, we do not search temporal neighbors from the background graph for query entities, so no aggregation is performed. Equation 25 shows how we compute representations for the query entity in MOST-TA. Similarly, MOST-TD adapts Equation 25 to the following Equation 26 to enable time difference learning,

$$\begin{aligned} \mathbf{h}_{(s_q,t_q)} &= \mathbf{h}_{s_q} + \delta_2 f(\mathbf{h}_{s_q} \| \mathbf{\Phi}(t_q - t_0)), \\ \mathbf{h}_{(o_q,t_q)} &= \mathbf{h}_{o_q} + \delta_2 f(\mathbf{h}_{o_q} \| \mathbf{\Phi}(t_q - t_0)). \end{aligned} \tag{26}$$

Then we calculate the meta-representation of r. Inspired by the KG scoring function RotatE [35], we treat the meta-representation of r as element-wise rotation

in the complex plane. To do this, we project $\mathbf{h}_r$ into a complex space to get the complex vector $\mathbf{h}_r^{\mathbb{C}} \in \mathbb{C}^{\frac{d}{2}}$.

$$\begin{aligned} \tilde{\mathbf{h}}_r &= \frac{\pi}{\|\mathbf{h}_r\|_\infty}\mathbf{h}_r, \\ \mathbf{h}_r^{\mathbb{C}}[j] &= \cos\left(\tilde{\mathbf{h}}_r[j]\right) + \sqrt{-1}\sin\left(\tilde{\mathbf{h}}_r[j]\right), \quad 1 <= j <= \frac{d}{2}. \end{aligned} \tag{27}$$

$\mathbf{h}_r^{\mathbb{C}}[j]$ and $\tilde{\mathbf{h}}_r[j]$ denote the jth element of the vectors $\mathbf{h}_r^{\mathbb{C}}$ and $\tilde{\mathbf{h}}_r$. $\|\mathbf{h}_r\|_\infty$ denotes the infinity norm of the vector $\mathbf{h}_r$. We also map the query entity representations $\mathbf{h}_{(s_q,t_q)}$, $\mathbf{h}_{(o_q,t_q)}$ to another complex space $\mathbb{C}^{\frac{d}{2}}$ to get $\mathbf{h}^{\mathbb{C}}_{(s_q,t_q)}$, $\mathbf{h}^{\mathbb{C}}_{(o_q,t_q)}$. For each mapped vector, we take the first half of the original vector from $\mathbb{R}^d$ as the real part and the second half as imaginary part . For example, for the vector $\mathbf{v} = [2,3]^\top \in \mathbb{R}^2$, we map it to $\mathbf{v}^{\mathbb{C}} = [2+3\sqrt{-1}]^\top \in \mathbb{C}^1$. According to [35], unitary complex number can be considered as a rotation in the complex plane, while $\mathbf{h}^{\mathbb{C}}_{(s_q,t_q)} \circ \mathbf{h}^{\mathbb{C}}_r$ can be interpreted as doing element-wise rotation from the query subject s_q in the complex space. We give the complete form of our metric function ψ as

$$\psi\left(q|\mathcal{S}_r\right) = \text{Sigmoid}\left(\mathbf{Re}\left(\left(\mathbf{h}^{\mathbb{C}}_{(s_q,t_q)} \circ \mathbf{h}^{\mathbb{C}}_r\right)^\top \bar{\mathbf{h}}^{\mathbb{C}}_{(o_q,t_q)}\right)\right), \tag{28}$$

where $q = (s_q, r, o_q, t_q)$ and Sigmoid denotes the sigmoid function that maps the score to a value between 0 and 1. **Re** means taking the real part of the complex number and $\bar{\mathbf{h}}^{\mathbb{C}}_{(o_q,t_q)}$ means the complex conjugate of $\mathbf{h}^{\mathbb{C}}_{(o_q,t_q)}$. ψ takes the real part of the dot product (Hermitian product) between the representations of the rotated query support $\mathbf{h}^{\mathbb{C}}_{(s_q,t_q)} \circ \mathbf{h}^{\mathbb{C}}_r$ and the query object $\mathbf{h}^{\mathbb{C}}_{(o_q,t_q)}$ as the score.

3.4 Parameter Learning

We train MOST with episodic training. In each episode, we first randomly choose one sparse relation r. Then we sample one r-related quadruple as the support quadruple $\mathcal{S}_r = (s_0, r, o_0, t_0)$, and collect a group of quadruples containing r as query set $\mathcal{Q}_r$. For each of the quadruple in the query set, i.e., $q = (s_q, r, o_q, t_q) \in \mathcal{Q}_r$, we switch q's object entity o_q to every other entity $e \in (\mathcal{E} \setminus \{o_q\})$ in the TKG (where $\mathcal{E}$ denotes the set of all entities in this TKG) and construct $|\mathcal{E}|-1$ polluted quadruples $\{q^-\}$ for q. And we use the binary cross entropy loss to optimize our model.

$$\mathcal{L} = \frac{1}{|\mathcal{Q}_r|}\sum_q \frac{1}{|\mathcal{E}|}\left(l_{q^+} + \sum_{q^-} l_{q^-}\right), \tag{29}$$

Algorithm 1: One-Shot Episodic Training

Training sparse relations $\mathcal{R}_{sp}^{train}$
for episode = 1: M **do**
  Shuffle relations in $\mathcal{R}_{sp}^{train}$
  Sample sparse relation r from $\mathcal{R}_{sp}^{train}$
  Sample a (s_0, r, o_0, t_0) to make the support set $\mathcal{S}_r$
  if One-Shot Interpolated LP **then**
    Sample a batch of query quadruples $\mathcal{Q}_r = \{(s_q, r, o_q, t_q)\}$
  else `// One-Shot Extrapolated LP`
    Sample a batch of query quadruples $\mathcal{Q}_r = \{(s_q, r, o_q, t_q) | t_0 < t_q\}$
  Compute $\mathbf{h}_{(s_0,t_0)}$, $\mathbf{h}_{(o_0,t_0)}$ with graph encoder
  Compute meta-information $\mathbf{h}^{\text{meta}}_{(s_0,r,o_0,t_0)}$
  Learn meta-representation $\mathbf{h}_r$ with meta-representation learner
  Pollute each $q \in \mathcal{Q}_r$ and generate polluted quadruples $\{q^-\}$
  Compute time-aware representations for entities in all q and $\{q^-\}$ `// Equation 25`
  Compute scores for all q and $\{q^-\}$ with metric function ψ
  Calculate the loss $\mathcal{L}$
  Update model parameters using gradient of loss $\bigtriangledown\mathcal{L}$

where $l_{q^+} = y_q \log(\psi(q|\mathcal{S}_r)) + (1-y_q)\log(1-\psi(q|\mathcal{S}_r))$ and $l_{q^-} = y_{q^-} \log(\psi(q^-|\mathcal{S}_r)) + (1-y_{q^-})\log(1-\psi(q^-|\mathcal{S}_r))$ denote the binary cross entropy loss of q and q^-, respectively. $y_q = 1$ and $y_{q^-} = 0$ and for $q \in \mathcal{Q}_r$, we want to maximize the positive score $\psi(q|\mathcal{S}_r)$, and minimize the negative score $\psi(q^-|\mathcal{S}_r)$. We describe our one-shot training procedure with Algorithm 1.

4 Experiments

Under the extrapolation setting, the existing datasets have the problem of an extremely small number of associated quadruples. Thus, We propose four new large-scale datasets with a substantial number of associated TKG quadruples. We then evaluate MOST by performing interpolated and extrapolated LP under the one-shot setting and compare MOST's performance with several baselines and conduct ablation studies to show the superiority of our model components.

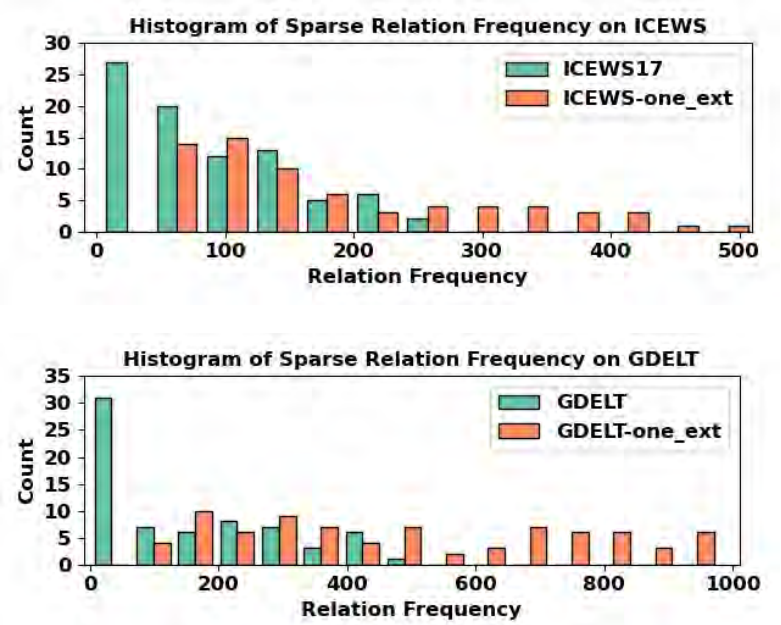

Figure. 12. Sparse Relation frequency comparison between ICEWS-one_ext and ICEWS17; GDELT-one_ext and GDELT.

4.1 Datasets

4.1.1 Problem with Previous Datasets

Integrated Crisis Early Warning System (ICEWS) [6] is a series of datasets containing political events over time. Global Database of Events, Language, and Tone (GDELT) [22] is a database recording all the events driving our global society every second of every day. Based on these two TKG databases, Mirtaheri et al. [27] propose two one-shot extrapolation LP datasets, ICEWS17 and GDELT. They first set upper and lower frequency thresholds (50 to 500 for ICEWS17, 50 to 700 for GDELT) for datasets and then select the frequency relations between the thresholds as sparse relations. A significant number of quadruples regarding sparse relations are removed to prevent time overlaps among meta-learning sets ($\mathbb{T}_{meta-train}$, $\mathbb{T}_{meta-valid}$, $\mathbb{T}_{meta-test}$). For example, assume a sparse relation r is selected, $T_r \in \mathbb{T}_{meta-train}$, and the ending timestamp of the meta-training set is t'. Then all the quadruples in $\{(s, r, o, t)|s, o \in \mathcal{E}, t > t'\}$ are removed from the dataset. For worse situation, if r's frequency is close to the lower threshold before removal, it is very likely the number of associated quadruples left in $\{(s, r, o, t)|s, o \in \mathcal{E}, t \leq t'\}$

becomes extremely small after removal. This tiny query set $\mathcal{Q}_r$ will cause instability during training. Similarly, for $T_r \in \mathbb{T}_{meta-valid}$ or $T_r \in \mathbb{T}_{meta-test}$, evaluation on a tiny query set $\mathcal{Q}_r$ also makes it hard to accurately determine the performance of the model. As shown in Figure. 12, a large part of sparse relations in ICEWS17 and GDELT have very few associated quadruples. Specifically, in ICEWS17, 31 out of 85 sparse relations have less than 50 associated quadruples. In GDELT, 24 out of 69 sparse relations have less than 50 associated quadruples. Moreover, 4 out of 14 test relations have even less than ten associated quadruples in ICEWS17, and this also applies to 11 out of 14 test relations in GDELT. This introduces instability in model training and evaluation.

4.1.2 Our Datasets

To fix the problem with ICEWS17 and GDELT [27], we construct two new large-scale extrapolation LP datasets, i.e., ICEWS-one_ext and GDELT-one_ext by taking subsets from ICEWS [6] and GDELT [22]. Furthermore, we also construct two interpolation datasets, i.e., ICEWS-one_int and GDELT-one_int for interpolated LP in the one-shot setting. Details about data construction are as follows:

(A) Extrapolation Datasets

1. We take ICEWS05-15 [1] and GDELT [2] as the databases for dataset construction.

2. For each database, by tracking every relation's frequency of occurrence, we divide all relations into two groups, i.e., frequent relations and sparse relations. Relations occurring between 100 and 1000 times in ICEWS05-15, and 200 and 2000 times for GDELT are taken as sparse relations. Those occurring more than 1000 times in ICEWS05-15 and more than 2000 times in GDELT are considered as frequent relations.

3. For each database, the quadruples containing its frequent relations form the background graph $\mathcal{G}'$. We split sparse relations into meta-train/meta-valid/meta-test groups, and remove a number of quadruples to avoid time overlap between every two sparse relation groups (following [27]. After quadruple removal, if the number of a sparse relation's associated quadruples is smaller than 50 for ICEWS, 100 for GDELT, we discard all the quadruples concerning this sparse

[1] https://github.com/mniepert/mmkb/tree/master/TemporalKGs
[2] https://github.com/INK-USC/RE-Net/tree/master/data

relation. The remaining quadruples containing sparse relations are kept for meta-learning process.

ICEWS-one_ext contains timestamped political facts happening from 2005 to 2015, while GDELT-one_ext contains global social facts from Jan. 1st, 2018 to Jan. 31st, 2018. We take the relations with higher frequency as frequent relations $\mathcal{R}_{freq}$ and build background graphs $\mathcal{G}'$ with all the quadruples containing them. Following [27], we then remove a part of quadruples associated with sparse relations to prevent time overlaps among meta-learning sets. After removal, we further discard the relations with too few associated quadruples (less than 50 for ICEWS-one_ext, 100 for GDELT-one_ext). In this way, we prevent including meta-tasks T_r with an extremely small query set $\mathcal{Q}_r$.

From Figure 12, we observe that ICEWS-one_ext and GDELT-one_ext have a substantial number of associated quadruples for each sparse relation, which greatly alleviates instability in model training and evaluation.

Similarly, we construct two more datasets, i.e., ICEWS-one_int and GDELT-one_int, for interpolated LP in the one-shot setting.

(B) Interpolation Datasets

1. We take ICEWS05-15 and GDELT as the databases for dataset construction.

2. For each database, by tracking every relation's frequency of occurrence, we divide all relations into two groups, i.e., frequent relations and sparse relations. Relations occurring between 50 and 500 times in ICEWS05-15, and 100 and 1000 times for GDELT are taken as sparse relations. Those occurring more than 500 times in ICEWS05-15 and more than 1000 times in GDELT are considered as frequent relations.

3. For each database, the quadruples containing the frequent relations form the background graph $\mathcal{G}'$. We split its sparse relations into meta-train/meta-valid/meta-test groups, and the quadruples containing the sparse relations are kept for meta-learning process.

Since in interpolated LP, there is no constraint on the support timestamp t_0, we do not remove quadruples to eliminate time overlaps among meta-learning sets. We set the upper and lower thresholds of sparse relations' frequency to 50 and 500 for ICEWS-one_int, 100 and 1000 for GDELT-one_int, and then split these relations into train/valid/test groups.

We present the statistics of our four datasets in Table 1.

Table 1: Dataset statistics.

Dataset	$\lvert\mathcal{E}\rvert$	$\lvert\mathcal{R}\rvert$	$\lvert\mathcal{T}\rvert$	$\lvert\mathbb{T}\rvert$
ICEWS05-15-one_ext	7,934	109	4,017	53/6/11
ICEWS05-15-one_int	10,356	155	4,017	74/9/10
GDELT-one_ext	6,647	155	2,751	55/7/11
GDELT-one_int	7,677	181	2,751	64/8/8

In Table 1, $|\mathbb{T}|$ denotes the number of meta-learning tasks in $\mathbb{T}_{meta-training}$, $\mathbb{T}_{meta-valid}$, $\mathbb{T}_{meta-test}$. $|\mathcal{E}|$, $|\mathcal{R}|$ and $|\mathcal{T}$ denotes the number of entities, relations, and timestamps in each dataset, respectively.

4.2 Baseline Methods

We compare our model with two groups of baseline methods on both interpolated and extrapolated LP in the one-shot setting.

4.2.1 Few-shot Relational Learning Methods

For static KG FSL methods, we consider five models, i.e., Gmatching [46], MetaR [8], FSRL [52], FAAN [34], GANA [31]. For TKG FSL method, we consider OAT [27]. In [27], static KG FSL methods are trained and evaluated on an unweighted static KG derived from collapsing the original TKG, which greatly decreases the inductive bias brought by the original TKG and causes poor performance of these methods. We hold the view that this setting is unfair. In our work, we provide static KG FSL methods with all the facts in the original datasets and only neglect time information, i.e., neglecting t in (s, r, o, t). More experimental comparisons are provided in the section 4.5.1.

4.2.2 Temporal Knowledge Graph Embedding Methods

We select three TKG interpolation methods, i.e., TNTComplEx [20], ATiSE [49], TeLM [48], and three TKG extrapolation methods, i.e., TANGO [13], CyGNet [54], xERTE [12] as our baselines.

For each interpolation dataset, we build a training set for these methods by adding all the quadruples of the background graph $\mathcal{G}'$ and the quadruples associated with all the meta-training relations $\mathcal{R}_{sp}^{train}$. For each extrapolation dataset, we build a training set by adding all the background quadruples and all the quadruples concerning every $r \in \mathcal{R}_{sp}^{train}$ during meta-training time $\mathcal{G}'_{train}$. We do not include any quadruple regarding $r \in \{\mathcal{R}_{sp}^{valid}, \mathcal{R}_{sp}^{test}\}$ in the training set. But the models have access to the support quadruples ($\mathcal{S}_r$, $r \in \{\mathcal{R}_{sp}^{valid}, \mathcal{R}_{sp}^{test}\}$) during inference. We test TKG embedding baselines with the same quadruples tested by FSL methods to ensure a fair comparison.

4.3 Implementation Details

Table 2: Hyperparameter searching strategy.

Hyperparameter	Search Space
Time Encoding Strategy	{TA, TD}
Embedding Size	{50, 100, 200}
# Aggregation Step	{1, 2}
Activation Function	{Tanh, ReLU, LeakyReLU}
Dropout	{0.2, 0.3, 0.5}
# Temporal Neighbor	{64, 128, 512}
Batch Size	{64, 128}

We implement all experiments with PyTorch [32] on a single NVIDIA Tesla T4. We use the hyperparameter searching strategy stated in Table 2. For every dataset, we do 648 trials, and let our model run for 10000 batches. We select the trial leading to the best performance on the meta-validation set and take this hyperparameter setting as our best configuration. We train our model five times and report averaged results. The best hyperparameter settings are reported in Table 4. The GPU memory usage is reported in Table 3.

For baseline methods, we use the official implementation of TNTComplEx [3],

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

ATiSE [4], TeLM [5], TANGO [6], CyGNet [7], xERTE [8], GMatching [9], MetaR [10], FSRL [11], FAAN [12], GANA [13], and OAT [14]. We pretrain Distmult [51] on the whole background graph of every interpolation dataset, and on the background graph before the end of meta-training set of every extrapolation dataset. We initialize the entity representations of KG FSL methods with the pretrained embeddings. We provide the hyperparameter settings of all baseline methods in Table 5 and Table 6. We refer to the best hyperparameter settings of baseline methods reported in their original papers.

Table 3: GPU memory usage.

Datasets	**ICEWS-one_ext**	**ICEWS-one_int**	**GDELT-one_ext**	**GDELT-one_int**
Model	GPU Memory	GPU Memory	GPU Memory	GPU Memory
MOST-TA	3327MB	3327MB	2967MB	2545MB
MOST-TD	5759MB	3327MB	3315MB	2967MB

Table 4: Best hyperparameter settings on each dataset.

Datasets	**ICEWS-one_ext**	**ICEWS-one_int**	**GDELT-one_ext**	**GDELT-one_int**
Hyperparameter				
Time Encoding Strategy	TD	TA	TD	TA
Embedding Size	200	100	100	50
# Aggregation Step	1	1	1	1
Activation Function	ReLU	ReLU	LeakyReLU	LeakyReLU
Dropout	0.2	0.2	0.3	0.3
# Temporal Neighbor	512	512	512	512
Batch Size	64	64	64	64

[4]https://github.com/soledad921/ATISE
[5]https://github.com/soledad921/TeLM
[6]https://github.com/TemporalKGTeam/TANGO
[7]https://github.com/CunchaoZ/CyGNet
[8]https://github.com/TemporalKGTeam/xERTE
[9]https://github.com/xwhan/One-shot-Relational-Learning
[10]https://github.com/AnselCmy/MetaR
[11]https://github.com/chuxuzhang/AAAI2020_FSRL
[12]https://github.com/JiaweiSheng/FAAN
[13]https://github.com/ngl567/GANA-FewShotKGC
[14]https://openreview.net/forum?id=GF8wO8MFQOr

Table 5: Hyperparameter settings of interpolation baselines.

Datasets	**ICEWS-one_int**			**GDELT-one_int**		
Hyperparameter	Embedding Size	# Negative Sample	Batch Size	Embedding Size	# Negative Sample	Batch Size
TNTComplEx	256	-	1000	312	-	1000
ATiSE	500	10	512	500	10	512
TeLM	4000	-	1000	4000	-	1000
GANA	100	1	1024	100	1	1024
MetaR	100	1	1024	100	1	1024
GMatching	100	1	128	100	1	128
FSRL	100	1	128	100	1	128
FAAN	100	1	128	100	1	128
OAT	50	1	100	50	1	100

Table 6: Hyperparameter settings of extrapolation baselines.

Datasets	**ICEWS-one_ext**			**GDELT-one_ext**		
Hyperparameter	Embedding Size	# Negative Sample	Batch Size	Embedding Size	# Negative Sample	Batch Size
TANGO	200	-	-	200	-	-
CyGNet	200	-	1024	200	-	1024
xERTE	256	-	128	128	-	128
GANA	100	1	1024	100	1	1024
MetaR	100	1	1024	100	1	1024
GMatching	100	1	128	100	1	128
FSRL	100	1	128	100	1	128
FAAN	100	1	128	100	1	128
OAT	50	1	100	50	1	100

4.4 Evaluation Metrics

We use Mean Reciprocal Rank (MRR) and Hits@k to evaluate the model performance on extrapolated link prediction. Previous KG FSL methods only report object prediction results. To achieve comprehensive results, for each test quadruple $(s_q, r_q, o_q, t_q) \in \mathcal{Q}_r$, $r_q \in \mathcal{R}_{sp}^{test}$, we derive two link prediction queries: $(s_q, r_q, ?, t_q)$ and $(?, r_q, o_q, t_q)$. Following [12], we transform $(?, r_q, o_q, t_q)$ into $(o_q, r^{-1}, ?, t_q)$ (r^{-1} denotes the reciprocal relation of r), and perform object prediction. We compute the rank of the ground truth missing entities (s_q or o_q) for every link prediction query. Let ψ_{s_q} and ψ_{o_q} denote the rank of $(?, r_q, o_q, t_q)$ and $(s_q, r_q, ?, t_q)$, respectively. We compute MRR by averaging the ranks among all the test quadruples:

$$\frac{1}{\sum\limits_{r_q \in \mathcal{R}_{sp}^{test}} 2|\mathcal{Q}_{r_q}|} \sum_{r_q \in \mathcal{R}_{sp}^{test}} \sum_{\tilde{q} \in \mathcal{Q}_{r_q}} \left(\frac{1}{\psi_{s_q}} + \frac{1}{\psi_{o_q}} \right), \tag{30}$$

where $\tilde{q}$ denotes a test quadruple (s_q, r_q, o_q, t_q). Hits@1/3/5/10 denote the proportions of the predicted links where ground truth entities are ranked as top 1, top 3, top 5, and top 10, respectively. The larger the two metrics are, the better the model performs on the task.

4.5 Experiment Results

4.5.1 Unfair Evaluation for Static KG FSL Methods

As mentioned in section 4.2, collapsing a TKG into an unweighted static KG will cause unfair evaluation for static KG FSL methods. For example, in the original TKG, there exist n identical events {(*Jonathan Nolan*, *live in*, *London*, t_1), ..., (*Jonathan Nolan*, *live in*, *London*, t_n)} that happen at n different timestamps. If n is a large number, these n repeated events will introduce a strong inductive bias showing that the entities, *Jonathan Nolan* and *London*, are likely to be highly correlated. Collapsing the original TKG into an unweighted static KG will lose great amounts of information for static KG FSL methods and force the models to learn more from weakly correlated entities.

Table 7: Interpolated LP results on collapsed unweighted KGs. Evaluation metrics are MRR and Hits@1/3/5/10 (%).

Datasets	**ICEWS-one_int**					**GDELT-one_int**				
Model	MRR	Hits@1	Hits@3	Hits@5	Hits@10	MRR	Hits@1	Hits@3	Hits@5	Hits@10
GANA	9.49	3.14	12.74	16.27	21.47	5.08	2.86	5.08	6.32	9.12
MetaR	17.73	0.00	28.96	38.77	49.13	7.94	0.11	10.41	14.99	22.14
GMatching	21.63	10.44	24.36	33.34	45.52	11.61	5.98	11.61	15.41	22.43
FSRL	21.09	9.90	23.99	32.34	43.84	11.08	5.67	11.12	14.61	21.32
FAAN	23.05	11.95	26.87	34.76	45.84	12.62	6.66	13.42	16.82	23.72

Table 8: Extrapolated LP results on collapsed unweighted KGs. Evaluation metrics are MRR and Hits@1/3/5/10 (%).

Datasets	**ICEWS-one_ext**					**GDELT-one_ext**				
Model	MRR	Hits@1	Hits@3	Hits@5	Hits@10	MRR	Hits@1	Hits@3	Hits@5	Hits@10
GANA	17.10	5.55	22.84	29.28	38.88	9.75	0.69	13.31	18.26	25.48
MetaR	15.28	2.46	21.74	29.35	39.98	8.11	0.08	11.23	15.61	26.30
GMatching	15.99	8.29	16.97	22.90	32.99	11.82	7.27	11.44	14.67	22.24
FSRL	11.96	5.62	10.63	16.32	26.54	9.69	7.21	9.09	10.93	14.17
FAAN	19.51	11.31	21.94	27.50	34.56	12.81	7.80	13.02	16.25	21.28

To empirically prove our assertion, we collapse our datasets into unweighted static KGs and rerun all static KG FSL methods on them. We retrain Distmult on the unweighted background graphs for embedding initialization. We report the experimental results in Table 7 and Table 8. By comparing with Table 9 and Table 10, we observe that in most cases, collapsing TKGs into unweighted KGs worsens the performance of static KG FSL methods greatly.

4.5.2 Model Results

Table 9: Interpolated LP results for one-shot relational learning on ICEWS-one_int and GDELT-one_int. Evaluation metrics are MRR and Hits@1/3/5/10. The best results are marked in bold.

Datasets	**ICEWS-one_int**					**GDELT-one_int**				
Model	MRR	Hits@1	Hits@3	Hits@5	Hits@10	MRR	Hits@1	Hits@3	Hits@5	Hits@10
TNTComplEx	23.34	14.57	27.21	31.54	36.88	11.95	6.76	11.98	15.58	21.78
ATiSE	34.40	22.03	39.51	49.25	60.57	7.77	5.10	6.72	8.13	12.13
TeLM	35.38	24.42	39.21	47.74	59.12	10.41	5.97	10.37	13.28	18.87
GANA	13.83	6.07	19.21	24.00	27.23	5.89	2.53	6.54	8.35	12.20
MetaR	27.69	7.88	41.02	52.58	61.78	9.91	0.18	14.61	19.62	26.79
GMatching	30.59	15.46	39.33	48.80	58.62	12.53	6.55	12.80	17.14	24.15
FSRL	33.98	18.94	44.48	52.61	59.82	14.11	7.61	14.67	19.56	27.54
FAAN	35.48	23.27	43.36	49.45	57.73	14.77	7.67	16.19	21.35	27.11
OAT	11.55	5.47	10.09	14.81	23.42	12.28	7.70	12.59	15.18	21.47
MOST-TA	**47.79**	**39.91**	51.79	**57.01**	62.25	**17.71**	11.56	**19.07**	**23.25**	**29.76**
MOST-TD	47.60	39.43	**51.98**	56.83	**62.38**	17.36	**11.67**	18.18	22.74	28.63

Table 9 and Table 10 report the experimental results of one-shot interpolated and extrapolated LP, respectively. We find that static KG FSL methods can achieve competitive or even better performance compared with traditional TKG embedding methods, implying the effectiveness of FSL in modeling sparse relations in KGs.

MOST outperforms baseline methods on all datasets in both LP tasks. While MOST-TA performs better than MOST-TD in the interpolation task, MOST-TD outperforms MOST-TA in the extrapolation task. We have the following explanation.

In the interpolated LP, every timestamp is observable during training, which enables the time encoder to learn information from all the timestamps. However, in the extrapolated LP, meta-training set does not span across the whole timeline due to the time constraint, when we sample the temporal neighbors during inference, there might be some timestamps unseen in the meta-training set, which leads to the degenerated model performance. For extrapolation tasks, time differences modeling achieves better results since almost all encountered time differences during inference are already seen and learned by the model during meta-training.

For static KG FSL methods, i.e., Gmatching, MetaR, FSRL, FAAN, GANA, their performances in both LP tasks are not so good since they do not incorporate temporal information. Traditional TKG embedding methods, i.e., TNTComplEx,

Table 10: Extrapolated LP results for one-shot relational learning on ICEWS-one_ext and GDELT-one_ext. Evaluation metrics are MRR and Hits@1/3/5/10. The best results are marked in bold.

Datasets	**ICEWS-one_ext**					**GDELT-one_ext**				
Model	MRR	Hits@1	Hits@3	Hits@5	Hits@10	MRR	Hits@1	Hits@3	Hits@5	Hits@10
TANGO	10.23	3.94	11.40	15.88	25.78	13.88	9.61	13.17	16.93	22.29
CyGNet	22.30	12.61	25.51	30.46	39.13	9.42	4.87	9.74	13.13	16.81
xERTE	30.02	19.79	36.63	42.13	51.16	16.38	10.88	18.23	22.19	27.76
GANA	11.34	3.70	15.52	19.25	25.67	7.12	4.85	7.02	8.89	11.13
MetaR	23.50	9.01	32.95	40.18	48.73	9.66	0.03	13.79	19.52	26.30
GMatching	20.30	12.35	21.06	28.80	38.02	12.26	8.41	11.44	13.76	19.01
FSRL	18.06	12.09	17.68	21.06	32.23	6.96	2.52	8.81	11.58	14.13
FAAN	25.73	15.86	29.14	35.95	43.73	14.36	8.71	15.31	18.46	23.71
OAT	13.28	9.51	12.69	18.02	21.48	14.06	6.71	13.43	18.59	28.11
MOST-TA	32.94	26.35	34.64	39.97	47.19	15.69	10.14	16.49	20.54	26.38
MOST-TD	**38.46**	**31.51**	**40.73**	**46.02**	**52.32**	**17.36**	**11.64**	**18.37**	**22.46**	**28.15**

ATiSE, TeLM, TANGO, CyGNet, xERTE, are not specially designed to capture information in the one-shot setting and generalize to the events of the associated spare relation. The TKG FSL method OAT is designed for extrapolated LP. It includes temporal information by employing a snapshot encoder that sequentially encodes a fixed number of historical graph snapshots right before the query timestamp. For the interpolation task, OAT will not get the temporal information coming after the query timestamp, which causes degenerated performance. For the extrapolation task, since OAT has a fixed history length, the temporal information is not enough and the model performance degenerated. MOST search for the nearest temporal neighbors in the meta-information extractor and there is no constraint on how far away these neighbors are, which helps to incorporate temporal neighbors in a better way. Besides, OAT employs cosine similarity for score computation, while MOST employs a metric function to compute the score. It is also worth noting that OAT performs much worse on ICEWS-based datasets (Table 9 and Table 10). It is due to the characteristics of databases. As discussed in [44], ICEWS database is much sparser than GDELT. This implies that it is hard to capture enough information when only considering a fixed number of graph snapshots, which causes worse performance on ICEWS-based datasets.

Table 11: Ablation studies of MOST-TA variants on ICEWS-one_int and ICEWS-one_ext. The best results are marked in bold.

Datasets	**ICEWS-one_int**			**ICEWS-one_ext**		
Variants	MRR	Hits@1	Hits@10	MRR	Hits@1	Hits@10
A	9.21	3.05	21.95	9.87	3.04	26.69
B1	45.00	35.40	61.83	29.98	20.17	46.30
B2	42.99	32.14	61.47	31.78	24.40	46.19
C1	1.00	0.82	0.89	9.33	5.16	17.38
C2	16.27	7.36	32.47	26.13	17.97	43.65
MOST-TA	**47.79**	**39.91**	**62.25**	**32.94**	**26.35**	**47.19**

Table 12: Ablation studies of MOST-TD variants on ICEWS-one_int and ICEWS-one_ext. The best results are marked in bold.

Datasets	**ICEWS-one_int**			**ICEWS-one_ext**		
Variants	MRR	Hits@1	Hits@10	MRR	Hits@1	Hits@10
A	9.21	3.05	21.95	9.87	3.04	26.69
B1	44.91	34.56	60.40	35.45	27.87	51.77
B2	42.11	30.45	61.79	37.73	30.62	52.32
C1	0.95	0.62	0.97	10.46	6.17	15.77
C2	23.68	13.87	44.40	27.35	18.82	43.78
MOST-TD	**47.60**	**39.43**	**62.38**	**38.46**	**31.51**	**52.32**

4.5.3 Ablation Study

To validate the effectiveness of model, we derive model variants for both MOST-TA and MOST-TD, and conduct several ablation studies on ICEWS-one_int and ICEWS-one_ext. We present the experimental results in Table 11 and Table 12. We devise model variants from the following angles:

(A) Excluding temporal information: In A, we remove the time encoder $\mathbf{\Phi}$ in all model components (Equation 21, 22 and 25), creating a model without using any temporal information. Note that without $\mathbf{\Phi}$, MOST-TA equals MOST-TD. we first change Equation 21 to Equation 31.

$$\begin{aligned}\mathbf{h}_{(s_0,t_0)} &= \mathbf{h}_{s_0} + \delta_1\sigma(\frac{1}{|\tilde{\mathcal{N}}_{s_0}|}\sum_{(e',r',t')\in\tilde{\mathcal{N}}_{s_0}}\mathbf{W}_g\left(\mathbf{h}_{e'}\circ\mathbf{h}_{r'}\right)),\\ \mathbf{h}_{(o_0,t_0)} &= \mathbf{h}_{o_0} + \delta_1\sigma(\frac{1}{|\tilde{\mathcal{N}}_{o_0}|}\sum_{(e',r',t')\in\tilde{\mathcal{N}}_{o_0}}\mathbf{W}_g\left(\mathbf{h}_{e'}\circ\mathbf{h}_{r'}\right)).\end{aligned} \tag{31}$$

In Equation 31 we randomly sample a fixed number of the temporal neighbors rather than getting the neighbors nearest to the support timestamp to avoid any temporal information. We also change Equation 25 to Equation 32.

$$\begin{aligned}\mathbf{h}_{(s_q,t_q)} &= \mathbf{h}_{s_q} + \delta_2\mathbf{h}_{s_q},\\ \mathbf{h}_{(o_q,t_q)} &= \mathbf{h}_{o_q} + \delta_2\mathbf{h}_{o_q}.\end{aligned} \tag{32}$$

We use $\mathbf{h}_{(s_0,t_0)}$, $\mathbf{h}_{(o_0,t_0)}$ in Equation 31 to compute the meta-representation of r, and we map $\mathbf{h}_{(s_q,t_q)}$, $\mathbf{h}_{(o_q,t_q)}$ in Equation 32 to the complex plane to compute score ψ. Here we just denote these representations with timestamps, e.g., with t_0 in $\mathbf{h}_{(s_0,t_0)}$, but they do not contain any temporal information.

The results show that it is crucial to utilize temporal information and MOST also heavily relies on temporal information.

(B) Changing graph aggregation function in meta-information extractor: In B1, we use mean pooling over the representations of temporal neighbors. For MOST-TA, we change the graph aggregation function Equation 21 to Equation

33.

$$\begin{aligned}\mathbf{h}_{(s_0,t_0)} &= \frac{1}{|\tilde{\mathcal{N}}_{s_0}|} \sum_{(e',r',t')\in\tilde{\mathcal{N}}_{s_0}} f(\mathbf{h}_{e'} \| \mathbf{\Phi}(t')), \\ \mathbf{h}_{(o_0,t_0)} &= \frac{1}{|\tilde{\mathcal{N}}_{o_0}|} \sum_{(e',r',t')\in\tilde{\mathcal{N}}_{o_0}} f(\mathbf{h}_{e'} \| \mathbf{\Phi}(t')).\end{aligned} \tag{33}$$

For MOST-TD, we change the graph aggregation function Equation 22 to Equation 34.

$$\begin{aligned}\mathbf{h}_{(s_0,t_0)} &= \frac{1}{|\tilde{\mathcal{N}}_{s_0}|} \sum_{(e',r',t')\in\tilde{\mathcal{N}}_{s_0}} f(\mathbf{h}_{e'} \| \mathbf{\Phi}(t_0 - t')), \\ \mathbf{h}_{(o_0,t_0)} &= \frac{1}{|\tilde{\mathcal{N}}_{o_0}|} \sum_{(e',r',t')\in\tilde{\mathcal{N}}_{o_0}} f(\mathbf{h}_{e'} \| \mathbf{\Phi}(t_0 - t')).\end{aligned} \tag{34}$$

In B2, for MOST-TA, we change the graph aggregation function Equation 21 to Equation 35.

$$\begin{aligned}\mathbf{h}_{(s_0,t_0)} &= \frac{1}{|\tilde{\mathcal{N}}_{s_0}|} \sum_{(e',r',t')\in\tilde{\mathcal{N}}_{s_0}} \mathbf{W}_{r'} \left(f(\mathbf{h}_{e'} \| \mathbf{\Phi}(t')) \right), \\ \mathbf{h}_{(o_0,t_0)} &= \frac{1}{|\tilde{\mathcal{N}}_{o_0}|} \sum_{(e',r',t')\in\tilde{\mathcal{N}}_{o_0}} \mathbf{W}_{r'} \left(f(\mathbf{h}_{e'} \| \mathbf{\Phi}(t')) \right).\end{aligned} \tag{35}$$

here $\mathbf{W}_{r'}$ is a weight matrix modeling the relation r'. For MOST-TD, we change the graph aggregation function Equation 22 to Equation 36.

$$\begin{aligned}\mathbf{h}_{(s_0,t_0)} &= \frac{1}{|\tilde{\mathcal{N}}_{s_0}|} \sum_{(e',r',t')\in\tilde{\mathcal{N}}_{s_0}} \mathbf{W}_{r'} \left(f(\mathbf{h}_{e'} \| \mathbf{\Phi}(t_0 - t')) \right), \\ \mathbf{h}_{(o_0,t_0)} &= \frac{1}{|\tilde{\mathcal{N}}_{o_0}|} \sum_{(e',r',t')\in\tilde{\mathcal{N}}_{o_0}} \mathbf{W}_{r'} \left(f(\mathbf{h}_{e'} \| \mathbf{\Phi}(t_0 - t')) \right).\end{aligned} \tag{36}$$

The results show that the aggregation function has a great ability in extracting meta-information.

(C) Changing metric function: In C1, we replace our metric function ψ with RotatE [35] together with time-aware representations.

$$\text{score} = \| \mathbf{h}^{\mathbb{C}}_{(s_q,t_q)} \circ \mathbf{h}^{\mathbb{C}}_{r} - \mathbf{h}^{\mathbb{C}}_{(o_q,t_q)} \|_1, \tag{37}$$

where $\| \cdot \|_1$ is the L1-norm.

In C2, we replace ψ with the LSTM-based matcher proposed in [46]. We notice that there is no meta-representation of sparse relation in [46]. The LSTM-based matcher only calculates a score representing the similarity between support entity pairs (s_0, o_0) and the query entity pairs (s_q, o_q). Following [46], we perform two steps of matching. Each step of matching is defined as

$$\begin{aligned} \mathbf{h}'_{k+1}, \mathbf{c}_{k+1} &= \text{LSTM}(\mathbf{h}_{\text{query}}, [\mathbf{h}_k \| \mathbf{h}_{\text{support}}, \mathbf{c}_k]), \\ \mathbf{h}_{k+1} &= \mathbf{h}'_{k+1} + \mathbf{h}_{\text{query}}, \\ \text{score}_{k+1} &= \mathbf{h}_{k+1}^{\top} \mathbf{h}_{\text{support}}, \end{aligned} \tag{38}$$

where $\text{LSTM}(\mathbf{x}, [\mathbf{h}, \mathbf{c}])$ is a standard LSTM cell [15] with input $\mathbf{x}$, hidden state $\mathbf{h}$, and cell state $\mathbf{c}$. $\mathbf{h}_{\text{support}} = \mathbf{h}_{(s_0,t_0)} \| \mathbf{h}_{(o_0,t_0)}$, and $\mathbf{h}_{\text{query}} = \mathbf{h}_{(s_q,t_q)} \| \mathbf{h}_{(o_q,t_q)}$.

We find that our metric function works much better in one-shot TKG LP.

4.5.4 Performance over Different Sparse Relations

Table 13: One-shot TKG interpolated LP performance over each sparse relation on ICEWS-one_int and GDELT-one_int. The best results are marked in bold. The second best results are underlined.

ICEWS-one_int					GDELT-one_int				
		MRR					MRR		
Relation	**Frequency**	MOST-TA	FAAN	TeLM	**Relation**	**Frequency**	MOST-TA	FAAN	TeLM
Threaten to reduce or break relations	65	**35.25**	28.13	21.79	Receive deployment of peacekeepers	108	**35.94**	9.52	10.62
Demonstrate for leadership change	89	**53.65**	40.86	44.08	Ban political parties or politicians	167	9.46	**11.54**	5.30
Express intent to yield	92	**50.50**	36.38	26.06	Attempt to assassinate	177	**12.23**	7.65	6.36
Increase police alert status	118	75.75	53.06	**78.31**	Receive inspectors	234	**17.07**	15.13	1.26
Appeal for material aid	146	**52.69**	49.37	37.96	Demand change in institutions, regime	411	**14.16**	7.00	15.42
Impose blockade, restrict movement	175	**63.83**	49.41	56.58	Threaten political dissent	675	**18.21**	13.77	16.15
Impose restrictions on political freedoms	282	**38.45**	34.33	23.00	Declare truce, ceasefire	748	**19.65**	13.41	6.33
Acknowledge or claim responsibility	269	**39.54**	25.66	38.48	Give ultimatum	752	**17.98**	16.84	11.47
Share intelligence or information	349	**29.75**	15.18	27.21					
Defy norms, law	436	**57.84**	43.37	29.33					

We also compare MOST with three strong baselines regarding the performance over different sparse relations. We choose TeLM and xERTE because they are the strongest traditional TKG interpolation and extrapolation baselines. We also choose FAAN since it outperforms almost all baselines in our main results. We do not compare with OAT because its performance is not strong enough compared with the above mentioned baselines, even though it is the only method developed for one-shot TKG LP.

Table 13 and Table 14 show MOST has strong robustness over different sparse

Table 14: One-shot TKG extrapolated LP performance over each sparse relation on ICEWS-one_ext and GDELT-one_ext. The best results are marked in bold. The second best results are underlined.

ICEWS-one_ext					GDELT-one_ext				
		MRR					MRR		
Relation	Frequency	MOST-TD	FAAN	xERTE	Relation	Frequency	MOST-TD	FAAN	xERTE
Provide military protection or peacekeeping	55	**32.85**	17.63	17.10	Investigate crime, corruption	149	15.34	**15.90**	15.29
Accuse of human rights abuses	57	**21.39**	4.99	17.07	Express intent to de-escalate military engagement	153	**19.58**	15.83	18.22
Appeal for change in leadership	65	22.05	21.60	**30.98**	Express intent to settle dispute	160	**22.19**	8.01	21.58
Acknowledge or claim responsibility	67	**26.31**	23.30	25.23	Protest violently, riot	176	**18.72**	11.66	13.42
Share intelligence or information	93	**30.41**	18.52	26.35	Carry out suicide bombing	180	**10.17**	5.06	10.01
Rally opposition against	107	21.67	15.40	**24.33**	Seize or damage property	212	17.04	11.43	**17.29**
Express intent to provide material aid	127	**37.13**	28.26	29.09	Veto	279	22.78	**23.15**	22.67
Appeal for intelligence cooperation	128	42.91	**43.88**	41.67	Demand intelligence cooperation	312	14.75	**15.61**	15.31
Provide military aid	130	**33.51**	9.74	18.59	Engage in political dissent	321	**15.67**	10.21	11.83
Mobilize or increase armed forces	180	**59.61**	32.96	28.99	Appeal for economic aid	348	15.21	**16.17**	15.55
Bring lawsuit against	184	**47.47**	36.28	45.99	Express intent to provide economic aid	359	**19.69**	17.67	18.72

relations in both one-shot TKG interpolated LP and extrapolated LP as it outperforms FAAN, TeLM and xERTE in almost all relations.

4.5.5 Performance over Different Support-Query Time Differences

Figure 13 and Figure 14 show that our model is robust to support-query time differences. We also keep the above mentioned models as baselines. For each sparse relation r in the meta-test set, we compute the time difference $|t_q - t_0|$, where t_0, t_q are support and query timestamp, respectively. The number of test quadruples with the same difference $|t_q - t_0|$ is relatively small. Inspired by previous work [27], we aggregate every 140 hours on ICEWS-one_int, every 24 hours on ICEWS-one_ext, every 40 hours on GDELT-one_int and every 6 hours on GDELT-one_ext for better visualization. In Figure 13a to 13d, it can be observed that our model performance dominates FAAN and TeLM on both interpolated LP datasets. From Figure 14a to 14d, our model also outperforms FAAN and xERTE almost in all points on both extrapolated LP datasets.

4.5.6 Time Cost Analysis

Table 15: Test time (min) comparison among MOST and the strongest baselines on ICEWS-based datasets.

Model	MOST-TA	MOST-TD	FAAN	TeLM	xERTE
ICEWS-one_int	0.10	0.11	35.93	0.02	-
ICEWS-one_ext	0.20	0.23	27.20	-	5.23

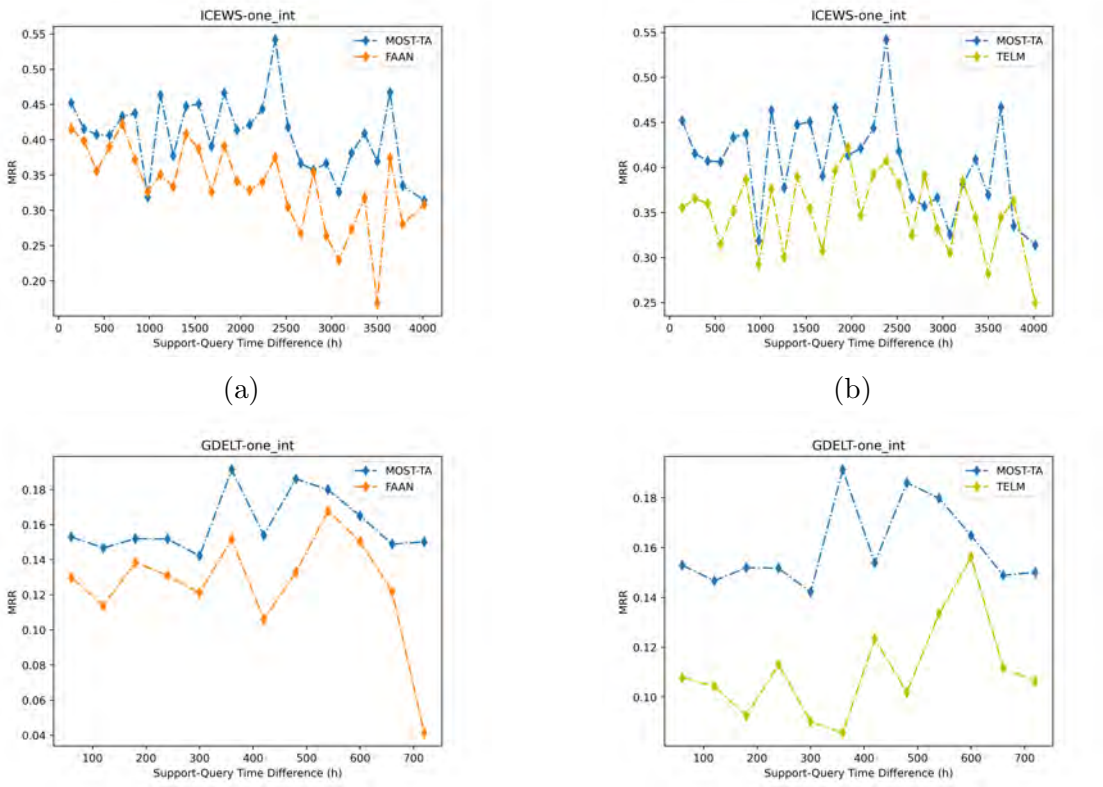

Figure. 13. Performance comparison between MOST and baselines over different support-query time differences $|t_q - t_0|$ on ICEWS-one_int. (a) MOST-TA vs. FAAN on ICEWS-one_int; (b) MOST-TA vs. TeLM on ICEWS-one_int; (c) MOST-TA vs. FAAN on GDELT-one_int; (d) MOST-TA vs. TeLM on GDELT-one_int.

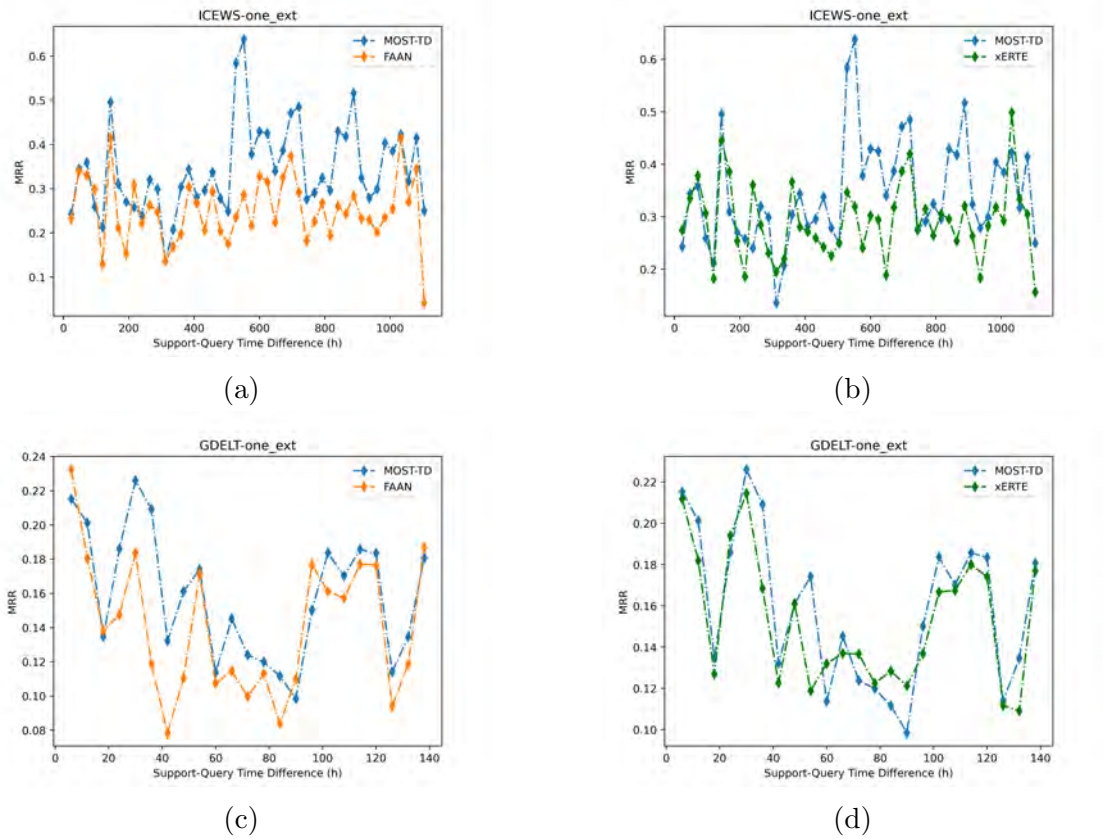

Figure. 14. Performance comparison between MOST and baselines over different support-query time differences $|t_q - t_0|$ on ICEWS-one_ext. (a) MOST-TD vs. FAAN on ICEWS-one_ext; (b) MOST-TD vs. xERTE on ICEWS-one_ext; (c) MOST-TD vs. FAAN on GDELT-one_ext; (d) MOST-TD vs. xERTE on GDELT-one_ext.

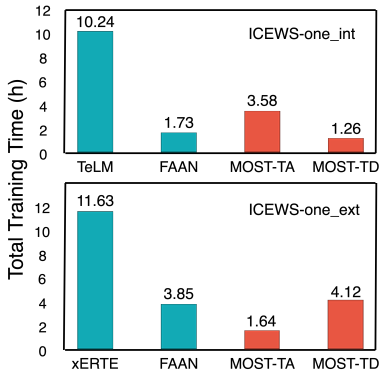

Figure. 15. Training time comparison among MOST and the strongest baselines on ICEWS-based datasets.

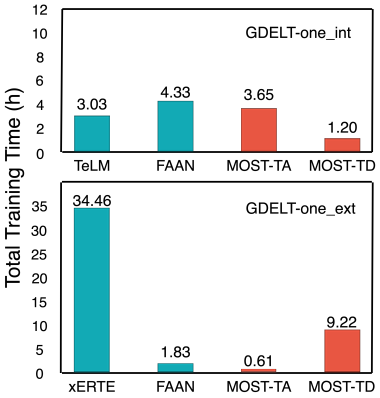

Figure. 16. Training time comparison among MOST and the strongest baselines on GDELT-based datasets.

We report in Figure 15 the time cost of MOST and several strong baselines on both ICEWS-based datasets. We observe that MOST achieves the best performance on LP tasks while keeping a low time cost. Though MOST-TD and MOST-TA achieve weaker performance than their counterpart in the interpolated and extrapolated LP, respectively, they require much shorter training time and can still achieve superior performance as reported in Table 9 and Table 10. Similar to Figure 15, in Figure 16, we report the total training time comparison among MOST and several strong baselines on GDELT-one_int and GDELT-one_ext. Note that static KG FSL methods employ pretrained KG embeddings for initialization. We do not include this time cost in the numbers presented in Figure 15 and Figure 16. MOST does not require pretraining and it also keeps low time costs while training GDELT-based datasets.

Except for training time, evaluation time is also a critical factor affecting the total time cost of model development. We report the evaluation time of all methods on meta-test sets in Table 16 and Table 17. We find that MOST keeps extremely low time consumption during evaluation. This greatly accelerates the process of model development.

We attribute the high training time efficiency of MOST to the employment of binary cross entropy loss. We treat every entity other than the ground truth missing entity as a negative sample, instead of sampling a number of negative samples for each LP query. We avoid the time cost during sampling and we also

Table 16: Test time (min) comparison of all methods on interpolation datasets.

Datasets	**ICEWS-one_int**	**GDELT-one_int**
Model		
TNTComplEx	0.02	0.03
ATiSE	2.20	2.77
TeLM	0.02	0.04
GANA	9.77	11.12
MetaR	8.01	7.33
GMatching	52.31	43.23
FSRL	32.21	23.66
FAAN	35.93	41.08
OAT	612.34	781.49
MOST-TA	0.10	0.27
MOST-TD	0.11	0.24

Table 17: Test time (min) comparison of all methods on extrapolation datasets.

Datasets	**ICEWS-one_ext**	**GDELT-one_ext**
Model		
TANGO	3.76	4.54
CyGNet	1.68	5.56
xERTE	5.23	12.41
GANA	3.64	7.86
MetaR	4.13	8.99
GMatching	19.61	18.91
FSRL	10.06	18.28
FAAN	27.20	33.78
OAT	1112.17	1507.78
MOST-TA	0.20	0.29
MOST-TD	0.23	0.46

jointly learn the representations of all entities when we perform prediction for every LP query. For evaluation, during score computation, we do not compute contextualized entity representations for all candidates. Instead, we incorporate temporal information with a simple time encoding layer for all the entities together. Some of the previous methods, e.g., OAT, compute the score for each candidate entity by going through the whole model (e.g. going through the whole model for $|\mathcal{E}|$ times if there exist $|\mathcal{E}|$ entities). However, in our work, we only need to go through the whole model for one time, thus cutting great time costs during evaluation.

5 Conclusion and Limitations

5.1 Conclusion

TKGs are known to be highly incomplete and a large portion of relations occur only a handful of times. In this work, we employ the one-shot learning method to tackle this problem. We propose four new large-scale TKG datasets for one-shot relational learning. Compared with previous datasets, our new datasets have a substantial number of associated TKG facts, which greatly alleviates model training and evaluation instability. Furthermore, we extend the interpolated and extrapolated LP tasks to the one-shot setting, and propose a model learning **m**eta representations of **o**ne-**s**ho**t** relations (MOST) for solving both tasks. Our model achieves state-of-the-art performance on all datasets in both tasks while keeping a low time cost.

We derive a meta-information extractor in order to learn contextualized entity representations. As for an entity e, a fixed number of temporal neighbors are sampled. MOST then aggregates the information provided by its temporal neighbors to compute its time-aware representation. Inspired by the work of Xu et al. [50], we encode time information in two different ways, i.e. absolute timestamp t' and time difference $t_0 - t'$ for different LP tasks. We learn an adaptively regularized meta representation for the sparse relation r from the time-aware representations of support entities in the one-shot examples, and then map the entity representations to the complex space. After that, we further employ a metric function for predicting missing entities.

Experimental results show that MOST achieves state-of-the-art performance and outperforms all the baselines on both one-shot LP tasks, showing its great potential

in the area of LP.

5.2 Limitations

Our work only considers the one-shot scenario without generalizing it to a larger shot size. Additional modules are required to distinguish graph information from more than one support example rather than directly using all the information provided by the only support example. We leave for future work to solve the low-shot TKG relational learning problem, e.g., 3-shot, 5-shot.

References

[1] Ralph Abboud et al. "BoxE: A Box Embedding Model for Knowledge Base Completion". In: *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*. Ed. by Hugo Larochelle et al. 2020. URL: https://proceedings.neurips.cc/paper/2020/hash/6dbbe6abe5f14af882ff977fc3f3550Abstract.html.

[2] Ivana Balazevic, Carl Allen, and Timothy M. Hospedales. "TuckER: Tensor Factorization for Knowledge Graph Completion". In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*. Ed. by Kentaro Inui et al. Association for Computational Linguistics, 2019, pp. 5184–5193. DOI: 10.18653/v1/D19-1522. URL: https://doi.org/10.18653/v1/D19-1522.

[3] Tim Berners-Lee, James Hendler, and Ora Lassila. "The semantic web". In: *Scientific american* 284.5 (2001), pp. 34–43.

[4] Tim Berners-Lee et al. "The World-Wide Web". In: *Commun. ACM* 37.8 (Aug. 1994), pp. 76–82. ISSN: 0001-0782. DOI: 10.1145/179606.179671. URL: https://doi.org/10.1145/179606.179671.

[5] Antoine Bordes et al. "Translating Embeddings for Modeling Multi-relational Data". In: *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States*. Ed. by Christopher J. C. Burges et al. 2013, pp. 2787–2795. URL: https://proceedings.neurips.cc/paper/2013/hash/1cecc7a77928ca8133fa24680a88d2fAbstract.html.

[6] Elizabeth Boschee et al. *ICEWS Coded Event Data*. Version V29. 2015. DOI: 10.7910/DVN/28075. URL: https://doi.org/10.7910/DVN/28075.

[7] Joan Bruna et al. "Spectral Networks and Locally Connected Networks on Graphs". In: *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*. Ed. by Yoshua Bengio and Yann LeCun. 2014. URL: http://arxiv.org/abs/1312.6203.

[8] Mingyang Chen et al. “Meta Relational Learning for Few-Shot Link Prediction in Knowledge Graphs”. In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*. Ed. by Kentaro Inui et al. Association for Computational Linguistics, 2019, pp. 4216–4225. DOI: 10.18653/v1/D19-1431. URL: https://doi.org/10.18653/v1/D19-1431.

[9] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. “Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering”. In: *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*. Ed. by Daniel D. Lee et al. 2016, pp. 3837–3845. URL: https://proceedings.neurips.cc/paper/2016/hash/04df4d434d481c5bb723be1b6df1ee6
Abstract.html.

[10] Tim Dettmers et al. “Convolutional 2D Knowledge Graph Embeddings”. In: *CoRR* abs/1707.01476 (2017). arXiv: 1707.01476. URL: http://arxiv.org/abs/1707.01476.

[11] Zifeng Ding et al. “A Simple But Powerful Graph Encoder for Temporal Knowledge Graph Completion”. In: *CoRR* abs/2112.07791 (2021). arXiv: 2112.07791. URL: https://arxiv.org/abs/2112.07791.

[12] Zhen Han et al. “Explainable Subgraph Reasoning for Forecasting on Temporal Knowledge Graphs”. In: *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. URL: https://openreview.net/forum?id=pGIHq1m7PU.

[13] Zhen Han et al. “Learning Neural Ordinary Equations for Forecasting Future Links on Temporal Knowledge Graphs”. In: *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 7-11 November, 2021*. Ed. by Marie-Francine Moens et al. Association for Computational Linguistics, 2021, pp. 8352–8364. DOI: 10.18653/v1/2021.emnlp-main.658. URL: https://doi.org/10.18653/v1/2021.emnlp-main.658.

[14] Geoffrey E. Hinton, Simon Osindero, and Yee Whye Teh. “A Fast Learning Algorithm for Deep Belief Nets”. In: *Neural Comput.* 18.7 (2006), pp. 1527–1554. DOI: 10.1162/neco.2006.18.7.1527. URL: https://doi.org/10.1162/neco.2006.18.7.1527.

[15] Sepp Hochreiter and Jürgen Schmidhuber. “Long Short-Term Memory”. In: *Neural Comput.* 9.8 (1997), pp. 1735–1780. DOI: 10.1162/neco.1997.9.8.1735. URL: https://doi.org/10.1162/neco.1997.9.8.1735.

[16] Shaoxiong Ji et al. “A survey on knowledge graphs: Representation, acquisition, and applications”. In: *IEEE Transactions on Neural Networks and Learning Systems* 33.2 (2021), pp. 494–514.

[17] Woojeong Jin et al. “Recurrent Event Network: Autoregressive Structure Inferenceover Temporal Knowledge Graphs”. In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Online: Association for Computational Linguistics, Nov. 2020, pp. 6669–6683. DOI: 10.18653/v1/2020.emnlp-main.541. URL: https://aclanthology.org/2020.emnlp-main.541.

[18] Jaehun Jung, Jinhong Jung, and U Kang. “Learning to Walk across Time for Interpretable Temporal Knowledge Graph Completion”. In: *KDD ’21: The 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Virtual Event, Singapore, August 14-18, 2021*. Ed. by Feida Zhu, Beng Chin Ooi, and Chunyan Miao. ACM, 2021, pp. 786–795. DOI: 10.1145/3447548.3467292. URL: https://doi.org/10.1145/3447548.3467292.

[19] Thomas N. Kipf and Max Welling. “Semi-Supervised Classification with Graph Convolutional Networks”. In: *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017. URL: https://openreview.net/forum?id=SJU4ayYgl.

[20] Timothée Lacroix, Guillaume Obozinski, and Nicolas Usunier. “Tensor Decompositions for Temporal Knowledge Base Completion”. In: *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. URL: https://openreview.net/forum?id=rke2P1BFwS.

[21] Julien Leblay and Melisachew Wudage Chekol. “Deriving Validity Time in Knowledge Graph”. In: *Companion of the The Web Conference 2018 on The Web Conference 2018, WWW 2018, Lyon , France, April 23-27, 2018*. Ed. by Pierre-Antoine Champin et al. ACM, 2018, pp. 1771–1776. DOI: 10.1145/3184558.3191639. URL: https://doi.org/10.1145/3184558.3191639.

[22] Kalev Leetaru and Philip A Schrodt. “Gdelt: Global data on events, location, and tone, 1979–2012”. In: *ISA annual convention*. Vol. 2. 4. Citeseer. 2013, pp. 1–49.

[23] Ruoyu Li et al. “Adaptive Graph Convolutional Neural Networks”. In: *CoRR* abs/1801.03226 (2018). arXiv: 1801.03226. URL: http://arxiv.org/abs/1801.03226.

[24] Yankai Lin et al. “Learning Entity and Relation Embeddings for Knowledge Graph Completion”. In: *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, January 25-30, 2015, Austin, Texas, USA*. Ed. by Blai Bonet and Sven Koenig. AAAI Press, 2015, pp. 2181–2187. URL: http://www.aaai.org/ocs/index.php/AAAI/AAAI15/paper/view/9571.

[25] Jiang Lu et al. “Learning from Very Few Samples: A Survey”. In: *CoRR* abs/2009.02653 (2020). arXiv: 2009.02653. URL: https://arxiv.org/abs/2009.02653.

[26] Yunpu Ma, Volker Tresp, and Erik A. Daxberger. “Embedding models for episodic knowledge graphs”. In: *J. Web Semant.* 59 (2019).

[27] Mehrnoosh Mirtaheri et al. “One-shot Learning for Temporal Knowledge Graphs”. In: *3rd Conference on Automated Knowledge Base Construction, AKBC 2021, Virtual, October 4-8, 2021*. Ed. by Danqi Chen et al. 2021. DOI: 10.24432/C55K56. URL: https://doi.org/10.24432/C55K56.

[28] Allen Newell, John C Shaw, and Herbert A Simon. “Report on a general problem solving program”. In: *IFIP congress*. Vol. 256. Pittsburgh, PA. 1959, p. 64.

[29] Maximilian Nickel, Lorenzo Rosasco, and Tomaso A. Poggio. “Holographic Embeddings of Knowledge Graphs”. In: *CoRR* abs/1510.04935 (2015). arXiv: 1510.04935. URL: http://arxiv.org/abs/1510.04935.

[30] Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. “A Three-Way Model for Collective Learning on Multi-Relational Data”. In: *Proceedings of the 28th International Conference on Machine Learning, ICML 2011, Bellevue, Washington, USA, June 28 - July 2, 2011*. Ed. by Lise Getoor and Tobias Scheffer. Omnipress, 2011, pp. 809–816. URL: https://icml.cc/2011/papers/438%5C_icmlpaper.pdf.

[31] Guanglin Niu et al. “Relational Learning with Gated and Attentive Neighbor Aggregator for Few-Shot Knowledge Graph Completion”. In: *SIGIR ’21: The 44th International ACM SIGIR Conference on Research and Development in Information Retrieval, Virtual Event, Canada, July 11-15, 2021*. Ed. by Fernando Diaz et al. ACM, 2021, pp. 213–222. DOI: 10.1145/3404835.3462925. URL: https://doi.org/10.1145/3404835.3462925.

[32] Adam Paszke et al. “PyTorch: An Imperative Style, High-Performance Deep Learning Library”. In: *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*. Ed. by Hanna M. Wallach et al. 2019, pp. 8024–8035. URL: https://proceedings.neurips.

cc/paper/2019/hash/bdbca288fee7f92f2bfa9f7012727740-Abstract.html.

[33] Michael Sejr Schlichtkrull et al. "Modeling Relational Data with Graph Convolutional Networks". In: *The Semantic Web - 15th International Conference, ESWC 2018, Heraklion, Crete, Greece, June 3-7, 2018, Proceedings*. Ed. by Aldo Gangemi et al. Vol. 10843. Lecture Notes in Computer Science. Springer, 2018, pp. 593–607. DOI: 10.1007/978-3-319-93417-4_38. URL: https://doi.org/10.1007/978-3-319-93417-4%5C_38.

[34] Jiawei Sheng et al. "Adaptive Attentional Network for Few-Shot Knowledge Graph Completion". In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*. Ed. by Bonnie Webber et al. Association for Computational Linguistics, 2020, pp. 1681–1691. DOI: 10.18653/v1/2020.emnlp-main.131. URL: https://doi.org/10.18653/v1/2020.emnlp-main.131.

[35] Zhiqing Sun et al. "RotatE: Knowledge Graph Embedding by Relational Rotation in Complex Space". In: *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. URL: https://openreview.net/forum?id=HkgEQnRqYQ.

[36] Zhiqing Sun et al. "Rotate: Knowledge graph embedding by relational rotation in complex space". In: *arXiv preprint arXiv:1902.10197* (2019).

[37] Théo Trouillon et al. "Complex Embeddings for Simple Link Prediction". In: *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*. Ed. by Maria-Florina Balcan and Kilian Q. Weinberger. Vol. 48. JMLR Workshop and Conference Proceedings. JMLR.org, 2016, pp. 2071–2080. URL: http://proceedings.mlr.press/v48/trouillon16.html.

[38] Shikhar Vashishth et al. "Composition-based Multi-Relational Graph Convolutional Networks". In: *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. URL: https://openreview.net/forum?id=BylA%5C_C4tPr.

[39] Ashish Vaswani et al. "Attention is All you Need". In: *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*. Ed. by Isabelle Guyon et al. 2017, pp. 5998–6008. URL: https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html.

[40] Oriol Vinyals et al. "Matching Networks for One Shot Learning". In: *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*. Ed. by Daniel D. Lee et al. 2016, pp. 3630–3638. URL: https://proceedings.neurips.cc/paper/2016/hash/90e1357833654983612fb05e3ec9148c-Abstract.html.

[41] Yanjie Wang, Rainer Gemulla, and Hui Li. "On Multi-Relational Link Prediction With Bilinear Models". In: *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*. Ed. by Sheila A. McIlraith and Kilian Q. Weinberger. AAAI Press, 2018, pp. 4227–4234. URL: https://www.aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/16900.

[42] Yaqing Wang et al. "Generalizing from a Few Examples: A Survey on Few-shot Learning". In: *ACM Comput. Surv.* 53.3 (2020), 63:1–63:34. DOI: 10.1145/3386252. URL: https://doi.org/10.1145/3386252.

[43] Zhen Wang et al. "Knowledge Graph Embedding by Translating on Hyperplanes". In: *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence, July 27 -31, 2014, Québec City, Québec, Canada*. Ed. by Carla E. Brodley and Peter Stone. AAAI Press, 2014, pp. 1112–1119. URL: http://www.aaai.org/ocs/index.php/AAAI/AAAI14/paper/view/8531.

[44] Jiapeng Wu et al. "TeMP: Temporal Message Passing for Temporal Knowledge Graph Completion". In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*. Ed. by Bonnie Webber et al. Association for Computational Linguistics, 2020, pp. 5730–5746. DOI: 10.18653/v1/2020.emnlp-main.462. URL: https://doi.org/10.18653/v1/2020.emnlp-main.462.

[45] Zonghan Wu et al. "A Comprehensive Survey on Graph Neural Networks". In: *IEEE Trans. Neural Networks Learn. Syst.* 32.1 (2021), pp. 4–24. DOI: 10.1109/TNNLS.2020.2978386. URL: https://doi.org/10.1109/TNNLS.2020.2978386.

[46] Wenhan Xiong et al. "One-Shot Relational Learning for Knowledge Graphs". In: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*. Ed. by Ellen Riloff et al. Association for Computational Linguistics, 2018, pp. 1980–1990. DOI: 10.18653/v1/d18-1223. URL: https://doi.org/10.18653/v1/d18-1223.

[47] Canran Xu and Ruijiang Li. "Relation embedding with dihedral group in knowledge graph". In: *arXiv preprint arXiv:1906.00687* (2019).

[48] Chengjin Xu et al. "Temporal Knowledge Graph Completion using a Linear Temporal Regularizer and Multivector Embeddings". In: *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2021, Online, June 6-11, 2021*. Ed. by Kristina Toutanova et al. Association for Computational Linguistics, 2021, pp. 2569–2578. DOI: 10.18653/v1/2021.naacl-main.202. URL: https://doi.org/10.18653/v1/2021.naacl-main.202.

[49] Chengjin Xu et al. "Temporal Knowledge Graph Embedding Model based on Additive Time Series Decomposition". In: *CoRR* abs/1911.07893 (2019). arXiv: 1911.07893. URL: http://arxiv.org/abs/1911.07893.

[50] Da Xu et al. "Inductive representation learning on temporal graphs". In: *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. URL: https://openreview.net/forum?id=rJeW1yHYwH.

[51] Bishan Yang et al. "Embedding Entities and Relations for Learning and Inference in Knowledge Bases". In: *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*. Ed. by Yoshua Bengio and Yann LeCun. 2015. URL: http://arxiv.org/abs/1412.6575.

[52] Chuxu Zhang et al. "Few-Shot Knowledge Graph Completion". In: *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*. AAAI Press, 2020, pp. 3041–3048. URL: https://ojs.aaai.org/index.php/AAAI/article/view/5698.

[53] Shuai Zhang et al. "Quaternion Knowledge Graph Embeddings". In: *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*. Ed. by Hanna M. Wallach et al. 2019, pp. 2731–2741. URL: https://proceedings.neurips.cc/paper/2019/hash/d961e9f236177d65d21100592edb0769-Abstract.html.

[54] Cunchao Zhu et al. "Learning from History: Modeling Temporal Knowledge Graphs with Sequential Copy-Generation Networks". In: *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The*

Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021. AAAI Press, 2021, pp. 4732–4740. URL: https://ojs.aaai.org/index.php/AAAI/article/view/16604.

Acknowledgement

I would like to thank all of you for helping me in writing this thesis. I am most grateful to my supervisor, Prof. Dr. Volker Tresp, who provided great support for my master's thesis project. Without his dedicated involvement in every step throughout the process, this paper would have never been accomplished. He is not only a knowledgeable supervisor in research but also an experienced elder in life. His advice and help will still be of great benefit to me in the future.

Secondly, I would like to express my sincere thanks to my tutor, Zifeng Ding. He provided me with a lot of help in various aspects, such as experiment design, project planning, result analysis, and thesis writing. He showed me in action how hardworking and uncompromising a researcher should be. When I encountered difficulties and wanted to give up. he encouraged me to continue. When I was confused, he showed me the way forward. He also taught me how to balance work and life, he is a real role model for me.

I am also deeply indebted to Prof. Dr. Christian Heumann, Prof. Dr. Christoph Kern, Ms. Maj-Catherine Botheroyd-Hobohm, Dr. Friedemann Steck, and Dr. Zhen Han. You have listened to me patiently and enlightened me in my most difficult moments and helped me think of ways when I struggled. You even helped me with financial pressures so that I could focus on my research.

Getting through my thesis required more than academic support, and I have many, many people to thank for listening to me and, at times, having to tolerate me over the past years. I cannot begin to express my gratitude and appreciation for their friendship. Yicui Kang, Jasmin, Coco, Zongyue Li, and Han Bao have been unwavering in their personal and professional support during the time I spent at the University.

Most importantly, none of this could have happened without my family. My mother, offered her encouragement through phone calls every week – despite my limited devotion to correspondence. With his brand of humor, my brother has been kind and supportive of me over the last several years. Every time I was ready to quit, you did not let me, and I am forever grateful. This dissertation stands as a testament to your unconditional love and encouragement.