TGIN: Translation-Based Graph Inference Network for Few-Shot Relational Triplet Extraction

Jiaxin Wang, Lingling Zhang, Jun Liu, *Senior Member, IEEE*, Kunming Ma, Wenjun Wu, Xiang Zhao, Yaqiang Wu, and Yi Huang

***Abstract*— Extracting relational triplets aims at detecting entity pairs and their semantic relations. Compared with pipeline models, joint models can reduce error propagation and achieve better performance. However, all of these models require large amounts of training data, therefore performing poorly on many long-tail relations in reality with insufficient data. In this article, we propose a novel end-to-end model, called TGIN, for few-shot triplet extraction. The core of TGIN is a multilayer heterogeneous graph with two types of nodes (entity node and relation node) and three types of edges (relation–entity edge, entity–entity edge, and relation–relation edge). On the one hand, this heterogeneous graph with entities and relations as nodes can intuitively extract relational triplets jointly, thereby reducing error propagation. On the other hand, it enables the triplet information of limited labeled data to interact better, thus maximizing the advantage of this information for few-shot triplet extraction. Moreover, we devise a graph aggregation and update method that utilizes translation algebraic operations to mine semantic features while retaining structure features between entities and relations, thereby improving the robustness of the TGIN in a few-shot setting. After updating the node and edge features through layers, TGIN propagates the label information from a few labeled examples to unlabeled examples, thus inferring triplets from these unlabeled examples. Extensive experiments on three reconstructed datasets demonstrate that TGIN can significantly improve the accuracy of triplet extraction by 2.34%∼10.74% compared with the state-of-the-art baselines. To the best of our knowledge, we are the first to introduce a heterogeneous graph for few-shot relational triplet extraction.**

***Index Terms*— Few-shot learning (FSL), heterogeneous graph, relational triplet extraction, translation.**

Manuscript received 28 December 2021; revised 21 June 2022 and 30 September 2022; accepted 23 October 2022. This work was supported in part by the National Key Research and Development Program of China under Grant 2020AAA0108800; in part by the National Natural Science Foundation of China under Grants 62137002, 61937001, 62192781, 62176209, 62176207, 62106190, and 62250009; in part by the Innovative Research Group of the National Natural Science Foundation of China under Grant 61721002; in part by the Innovation Research Team of Ministry of Education (IRT_17R86) through the MoE-CMCC "Artifical Intelligence" Project under Grant MCM20190701; in part by the Consulting Research Project of the Chinese Academy of Engineering "The Online and Offline Mixed Educational Service System for 'The Belt and Road' Training in MOOC China"; in part by the "LENOVO-XJTU" Intelligent Industry Joint Laboratory Project; in part by the CCF-Lenovo Blue Ocean Research Fund; in part by the Project of China Knowledge Centre for Engineering Science and Technology; in part by the Foundation of the Key National Defense Science and Technology Laboratory under Grant 6142101210201; in part by NSFC under Grant 61872446; in part by the Science and Technology Innovation Program of Hunan Province under Grant 2020RC4046; and in part by the Fundamental Research Funds for the Central Universities under Grants xhj032021013-02, xzy022021048, and xpt012022033. *(Corresponding author: Lingling Zhang.)*

Jiaxin Wang, Lingling Zhang, and Jun Liu are with the School of Computer Science and Technology, Xi'an Jiaotong University, Xi'an, Shaanxi 710049, China (e-mail: jiaxinwangg@outlook.com; zhanglling@xjtu.edu.cn; liukeen@xjtu.edu.cn).

Kunming Ma and Wenjun Wu are with the Shaanxi Provincial Key Laboratory of Big Data Knowledge Engineering and the National Engineering Laboratory for Big Data Analytics, Xi'an Jiaotong University, Xi'an, Shaanxi 710049, China (e-mail: mkm_xjtu@qq.com; nickjunwork@163.com).

Xiang Zhao is with the Laboratory of Big Data and Decision, National University of Defense Technology, Changsha, Hunan 410073, China (e-mail: xiangzhao@nudt.edu.cn).

Yaqiang Wu is with Lenovo Research, Beijing 100094, China (e-mail: wuyqe@lenovo.com).

Yi Huang is with the JIUTIAN Team, China Mobile Research Institute, Beijing 100032, China (e-mail: huangyi@chinamobile.com).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TNNLS.2022.3218981.

Digital Object Identifier 10.1109/TNNLS.2022.3218981

NOMENCLATURE

$W, \langle h, r, t\rangle$	Sentence and relational triplet in this sentence.
$\mathscr{D}_{\text{train}}$ and $\mathscr{D}_{\text{test}}$	Training set and test set.
$\mathcal{S}$ and $\mathcal{Q}$	Support set and query set.
$\mathcal{G} = (\mathcal{V}, \mathcal{E})$	Graph data.
$\mathcal{V}$ $(\mathcal{V}^u \cup \mathcal{V}^r)$ and $\mathcal{E}$ $(\mathcal{E}^{ru} \cup \mathcal{E}^{uu} \cup \mathcal{E}^{rr})$	Node set and edge set.
$v_i^u \in \mathcal{V}^u$ and $v_j^r \in \mathcal{V}^r$	One entity node and one relation node.
$\mathbf{v}_i^u \in \mathbb{R}^d$ and $\mathbf{v}_j^r \in \mathbb{R}^d$	Embedding of entity node and relation node.
$E(v_j^r, v_i^u) \in \mathcal{E}^{ru}$, $E(v_i^u, v_{i'}^u) \in \mathcal{E}^{uu}$, and $E(v_j^r, v_{j'}^r) \in \mathcal{E}^{rr}$	One relation–relation edge, one entity–entity edge, and one relation–relation edge.
$\mathbf{e}_{ji}^{ru} = \left[e_{ji}^{(1),ru}; e_{ji}^{(2),ru}\right]$, $e_{ii'}^{uu}$, and $e_{jj'}^{rr}$	Edge feature of $E(v_j^r, v_i^u)$, $E(v_i^u, v_{i'}^u)$, and $E(v_j^r, v_{j'}^r)$.
$\mathcal{N}(\cdot, \cdot)$	Neighboring function that output all neighboring nodes for the target node.
$\mathscr{T}_h(v_i^u)$, $\mathscr{T}_t(v_i^u)$, and $\mathscr{T}_r(v_j^r)$	Triplet set $(\langle v_i^u, \cdot, \cdot\rangle)$, triplet set $(\langle \cdot, \cdot, v_i^u\rangle)$, and triplet set $(\langle \cdot, v_j^r, \cdot\rangle)$.
$\boldsymbol{\phi}_{h,v_i^u}$	Aggregation effects on v_i^u from neighboring triplets (follow $\boldsymbol{h} \leftarrow \boldsymbol{t} - \boldsymbol{r}$).

$\boldsymbol{\phi}_{t,v_i^u}$	Aggregation effects on v_i^u from neighboring triplets (follow $\boldsymbol{t} \leftarrow \boldsymbol{h} + \boldsymbol{r}$).
$\boldsymbol{\phi}_{r,v_j^r}$	Aggregation effects on v_j^r from neighboring triplets (follow $\boldsymbol{r} \leftarrow \boldsymbol{t} - \boldsymbol{h}$).
$\boldsymbol{\psi}_{v_j^r,v_{j'}^r}$	Aggregation effects on v_j^r from neighboring relations.
$f_s(\langle v_i^u, v_j^r, v_{i'}^u\rangle)$	Score function for triplet $\langle v_i^u, v_j^r, v_{i'}^u\rangle$.
$\mathcal{T}_{rh}(v_j^r, v_i^u)$ and $\mathcal{T}_{rt}(v_j^r, v_i^u)$	Triplet set $(\langle v_i^u, v_j^r, \cdot\rangle)$ and triplet set $(\langle \cdot, v_j^r, v_i^u\rangle)$.
$\mathcal{T}_G$ and $\mathcal{T}'_G$	Correct triplet set and incorrect triplet set in $\mathcal{G}$.

I. Introduction

EXTRACTING relational triplets is one of the most essential technology for knowledge graph construction [1]. It aims to detect entity mentions and extract their semantic relations to form triplets from unstructured text. For example, when given a sentence *Saint Paul is located in the United States*, a relational triplet extraction system will produce an output as ⟨*Saint Paul, Located in, United States*⟩, where *Saint Paul*, *Located in*, and *United States* are a head entity, relation, and tail entity, respectively. Various models have been proposed for relational triplet extraction, which can be divided into two categories: pipeline models and joint models. Pipeline models [2], [3] split this task into two subtasks: named entity recognition (NER) [4], [5] and relation classification (RC) [6]. NER is first performed to recognize entities, and then, RC is conducted to classify the relation types between entities. Pipeline models have a major problem of error propagation, that is, if an error occurs in the former subtask, it will directly accumulate in the later subtask, thereby reducing the models' performance on triplet extraction. Joint models [7], [8], [9], [10], [11] employ an end-to-end approach to extract entities and relations with joint training, which alleviates error propagation and achieves higher performance than pipeline models. In this article, we focus on joint models.

Previous joint models are always based on supervised learning, and their performance depends on large amounts of labeled training data. However, as shown in Fig. 1, we give the distribution result of relation frequency in the widely used relational triplet extraction dataset, i.e., New York Times (NYT) [12]. Statistics show that 42% relations appear less than ten times. This indicates that relations tend to have long-tail distribution, which means that a large number of relations, such as *Buried in*, will suffer from data deficiency compared with the most popular relations, such as *Born in*. Therefore, when dealing with long-tail relations, the performance of these models will drop dramatically because of training data deficiency. In this sense, models are required to handle relational triplet extraction with a few labeled training examples, namely, few-shot learning (FSL). Most of the existing approaches with FSL concentrate on a single task, which means that they do FSL only for NRE [13], [14] or RC [15], [16], [17], [18]. Hence, jointly extract relational triplet in a few-shot setting still needs to be investigated. The multiprototype embedding (MPE) network [19] is the only model proposed for this task. It designs the head (tail) entity prototype and the relation prototype for triplet extraction by averaging the features of

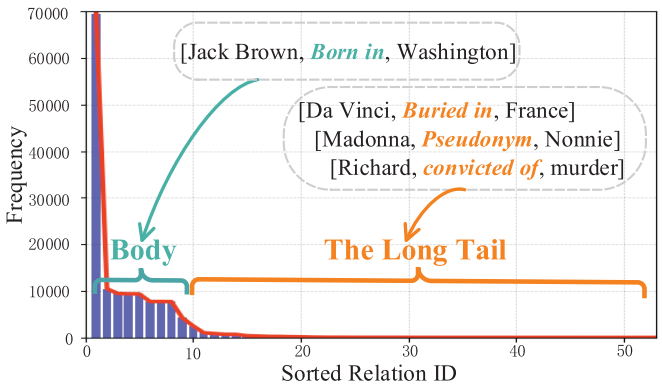


Fig. 1. Histogram of relation frequency in the NYT dataset. We can observe that there is a large portion of relations that have a few examples.

all instances in the same category. However, the accuracy of MPE is limited because this indiscriminate averaging will causes much information loss, e.g., for relation *religion*, the head entities can be *person* or *religious buildings* and, hence, cannot be treated equally by averaging. FSL algorithms have shown that they require full exploitation of a few labeled instances [20], [21], and graph networks can retain the features for all instances. Thus, the use of graph networks has the potential for jointly extracting entities and relations in a few-shot setting [22], [23], [24].

In this work, to maximize the advantage of a few labeled instances and reduce error propagation for few-shot relational triplet extraction, we propose a translation-based graph inference network (TGIN). Specifically, TGIN is designed as a heterogeneous graph consisting of two types of nodes (entity node and relation node) and three types of edges (relation–entity edge, entity–entity edge, and relation–relation edge). On the one hand, this heterogeneous graph with entities and relations as nodes can intuitively extract relational triplets jointly, thereby reducing error propagation. On the other hand, it allows the triplet information of limited labeled data to interact better, thus maximizing the advantage of this information for few-shot triplet extraction.

Furthermore, we devise a translation-based graph aggregation and update method for TGIN. "Translation" [25] is a principle in knowledge graphs, which interprets relations and entities in a learned low-dimensional embedding space as follows: if ⟨*head, relation, tail*⟩ holds, then the embedding of the $tail$ entity should be close to the embedding of the *head* entity plus some vector that depends on the $relation$. As this principle is designed to embed tail entities with given relations and head entities for knowledge graph representation [26], [27], it cannot extract triplets directly from texts but reflects the structural features of entities and relations within triplets, i.e., $tail \approx head + relation$. Inspired by this, we extend the "translation" to translation algebraic operations for the update of TGIN, which helps label propagation with the unit of triplets, thus facilitating triplet extraction. Specifically, we consider a group of nodes can form a triplet $\langle \boldsymbol{h}, \boldsymbol{r}, \boldsymbol{t}\rangle$, in which $\boldsymbol{h}$, $\boldsymbol{r}$, and $\boldsymbol{t}$ represent head entity node, relation node, and tail entity node, respectively. The features of $\boldsymbol{h}$ are approximately equal to $\boldsymbol{t}$ minus $\boldsymbol{r}$, i.e., $\boldsymbol{h} \leftarrow \boldsymbol{t} - \boldsymbol{r}$, and similarly, for

$\boldsymbol{r} \leftarrow \boldsymbol{t} - \boldsymbol{h}$, $\boldsymbol{t} \leftarrow \boldsymbol{h} + \boldsymbol{r}$. The empirical results validate algebraic operations can improve the robustness of the model by 1.43%∼2.48% in terms of triplet performance with limited training instances. Finally, a series of comparative experiments demonstrate the effectiveness of each decomposition of TGIN, with an overall performance superiority of 2.34%∼10.74% in terms of accuracy for few-shot relational triplet extraction.

To sum up, the three main contributions of this work are summarized as follows.

1) We propose an innovative joint framework with a heterogeneous graph named TGIN for few-shot relational triplet extraction. To the best of our knowledge, we are the first to introduce a heterogeneous graph network for this task to address the problems of error propagation and data deficiency simultaneously.
2) We devise a translation-based graph aggregation and update method that allows TGIN to mine semantic features and maintain structural information of triplets, thereby improving the robustness of TGIN in a few-shot setting.
3) Extensive experiments on three datasets are conducted to compare the proposed TGIN with state-of-the-art models, as well as its ablation variants. The results suggest that TGIN achieves a significant improvement of 2.34%∼10.74% in terms of accuracy for few-shot relational triplet extraction.

The remainder of this article is organized as follows. Section II reviews the related work on relational triplet extraction, few-shot relational triplet extraction, and the heterogeneous graph network. Section III gives the problem definition of few-shot relational triplet extraction. Section IV describes the detail of our proposed model and learning procedure. Section V presents experimental evaluations on three datasets. Finally, in Section VI, we summarize the conclusions.

II. Related Work

A. Relational Triplet Extraction

Early research efforts on relational triplet extraction are based on pipeline methods [2], [3] that conduct NER [4], [5] and RC [6] as two subtasks. Therefore, these methods inevitably propagate errors and, hence, affect the accuracy of triplet extraction [7].

As pipeline methods suffer from these problems, researchers have proposed joint methods [8], [10] based on neural networks. For instance, Zeng et al. [28] presented a sequence-to-sequence learning method with copy mechanism to jointly extract relational triplets from sentences. Wei et al. [9] introduced a cascade binary tagging framework to achieve triplet extraction and converted this task into an end-to-end binary tagging problem. Fu et al. [7] considered the interaction between entities and relations via relation-weighted graph convolutional networks (GCNs) to better extract triplets. Meanwhile, Guo et al. [29] proposed attention-guided GCNs (AGGCNs), which use structural information from dependency trees to enhance semantic information for relation extraction. Recently, Zhao et al. [10] creatively combined subaware relation prediction and question generation together for entity-relation extraction. Nevertheless, all these supervised methods notoriously depend on large amounts of labeled data, which limits the model performance due to long-tail distributions [30] and annotation cost [31]. In this way, few-shot relational triplet extraction needs to be explored.

B. Few-Shot Relational Triplet Extraction

Models for FSL need to rapidly learn with only a few labeled data [32]. Metalearning [33] is a common strategy for FSL, which extracts some transferable knowledge from auxiliary tasks and then helps the model to solve the target few-shot problem [34]. In this sense, few-shot relational triplet extraction aims to learn a model based on metalearning that can perform well for triplet prediction in the case where only a few labeled samples are given.

However, most existing studies only engross in few-shot NER or few-shot RC but lack the capability of extracting relational triplets in a few-shot scenario. The prototypical network (ProtoNet) [20] is a popular model for FSL, which averages the embedding of each support class to obtain prototype representations and calculates the distance between a query instance and the prototype. Based on this idea, Gao et al. [16] proposed a hybrid attention-based ProtoNet (HATT) for few-shot RC, while Fritzler et al. [35] tried to utilize the ProtoNet to achieve few-shot NER. Rather than referring ProtoNet, Ye and Ling [15] considered a multilevel matching and aggregation network (MLMAN) to classify relations in a few-shot setting. Hou et al. [13] constructed a collapsed dependency transfer and label-enhanced task-adaptive projection network to model the dependencies between entity labels for few-shot NER.

Currently, the MPE network [19] is the only model for few-shot relational triplet extraction. It first adopts a conditional random field (CRF) [36] to identify the head and tail entities in a sentence, then designs a head (tail) entity prototype also inspired by ProtoNet for different sentences with the same relation, and extracts relations from these entity prototypes. However, MPE has two major drawbacks. For one thing, the CRF in MPE only identifies the head and tail entities, which means that it ignores other entities in a sentence. These overlooked entities may act as head or tail entities in other sentences. As a result, the accuracy of the model is limited. For another, a head (tail) prototype is obtained by averaging the embedding of all head (tail) entities with the same relation, which is unreasonable, e.g., for relation *religion*, the head entities can be *person* or *religious buildings* and, hence, cannot be treated equally.

C. Heterogeneous Graph Neural Network

The heterogeneous graph neural network (HGNN) [37] has multiple types of nodes or edges and can deal with real-world heterogeneous data. Its superiority has been verified in many natural language processing (NLP) tasks. Specifically, Hong et al. [38] designed a type-aware attention heterogeneous information network (HIN) for graph representation learning. Tu et al. [39] employed HGNN-based message-passing algorithms for multihop reading comprehension. Hu et al. [40] presented a flexible HIN framework for text classification, which can capture the relations between texts and address the semantic sparsity. Wang et al. [41] proposed an HGNN-based

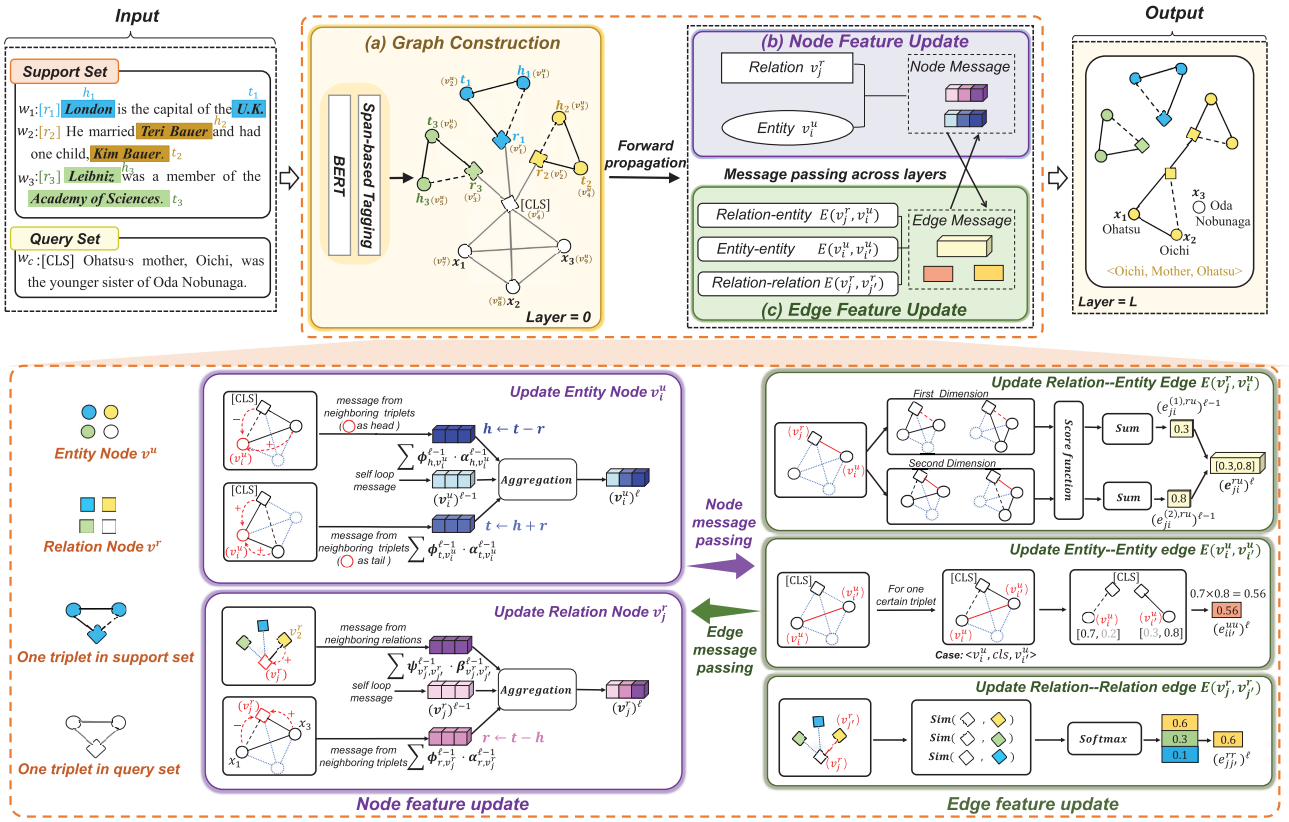


Fig. 2. Overview of TGIN for a 3-way 1-shot problem with one query sentence. TGIN includes three parts: (a) graph construction (yellow part), (b) node feature update (purple part), and (c) edge feature update (green part). Our TGIN first uses $\text{BERT}_{\text{BASE}}$ to encode sentences into embedding and adopts the span-based tagging method to identify all entities in the query sentence. Then, TGIN builds a heterogeneous graph on this basis. This heterogeneous graph contains two types of nodes (relation node and entity node) and three types of edges (relation–entity edge, entity–entity edge, and relation–relation edge). TGIN devises a translation-based graph aggregation and update method, which updates nodes and edges features with units of triplets. After forward propagation through layers, TGIN can infer the relational triplets from a given query sentence. For example, TGIN generates a triplet as (*Ocichi, Mother, Ohatus*) from the query sentence in this figure.

network for extractive summarization, which contains semantic nodes that act as the intermediary between sentences and enrich the cross-sentence relations, and Zeng et al. [42] utilized HGNN for document-level relation extraction. Taking inspiration from the above success, we first introduce HGNN to the task of few-shot relational triplet extraction, and creatively aggregate and update this HGNN by translation algebraic operations.

III. Problem Definition

The task of few-shot relational triplet extraction aims to learn a model that can extract triplets for given sentences with a few labeled instances. For brevity, we study a sentence with one relation and an associated triplet. More concretely, we define the form of each instance as $(W, \langle h, r, t\rangle)$, where W is a sentence, and $\langle h, r, t\rangle$ denotes the relational triplet extracted from this sentence. h, t, and r represent the head entity, tail entity, and their relation, respectively. In this task, we split all instances into two datasets, $\mathscr{D}_{\text{train}}$ and $\mathscr{D}_{\text{test}}$, according to their relation categories. Note that the relation sets $\mathcal{R}_{\text{train}}$ and $\mathcal{R}_{\text{test}}$ of these two datasets are disjoint with each other.

Specifically, in each training episode, we sample support set $\mathcal{S} \subset \mathscr{D}_{\text{train}}$ and a query set $\mathcal{Q} \subset \mathscr{D}_{\text{train}}$. $\mathcal{S} = \{(W_i, \langle h_i, r_i, t_i\rangle) | r_i \in \mathcal{R}_{(\cdot)}\}_{i=1}^{NK}$ contains N classes with K instances for each class (N and K are quite small in FSL), and $\mathcal{Q} = \{(W_c, \langle h_c, r_c, t_c\rangle) | r_c \in \mathcal{R}_{(\cdot)}\}_{c=NK+1}^{NK+C}$ has C instances. Here, $\mathcal{R}_{(\cdot)}$ represents $\mathcal{R}_{\text{train}}$. Our goal is to train a model that can predict the triplet $\langle h_c, r_c, t_c\rangle$ accurately in $\mathcal{Q}$ with $\mathcal{S}$. This target problem is also called the N-way K-shot problem. We set $\mathcal{R}_{(\cdot)}$ as $\mathcal{R}_{\text{test}}$ after finished training and still sample $\mathcal{S} \subset \mathscr{D}_{\text{test}}$ and $\mathcal{Q} \subset \mathscr{D}_{\text{test}}$ as each episode for test. Nomenclature shows explanations of important notations in this work.

IV. Proposed Model

In this section, we introduce our TGIN in detail. As illustrated in Fig. 2, TGIN includes three parts: 1) graph construction; 2) node feature update; and 3) edge feature update. Specifically, TGIN first identifies all entities and obtains the initialized features of entities and relations from unstructured sentences. Then, these features are fed into a heterogeneous graph consisting of a number of layers, where each layer is composed of a node-feature-update block and an edge-feature-update block. TGIN uses node features to update edge features in the same layer and then aggregates all of these features to update node features in the next layer. After features propagate and update through layers, TGIN infers relational triplets from unstructured sentences by calculating the score of each triplet according to the edge and node features in the final layer. Section IV-A describes how to identify entities and initialize

graph features. After that, we demonstrate the node feature update method in Section IV-B and the edge feature update method in Section IV-C. Finally, we introduce the training and prediction process in Section IV-D.

A. Graph Construction

The construction of TGIN is based on $N \times K$ support sentences and one query sentence, assuming that TGIN is denoted as $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ and $\mathcal{E}$ represent the node set and the edge set, respectively. Following is the construction process, including entity recognition, node feature initialization, and edge feature initialization.

1) Entity Recognition: We first utilize the pretrained language model $\text{BERT}_{\text{BASE}}$ [43] that applies multihead attention to capture the contextual information for input sentences. Given a sentence W with T tokens $\{w_1, w_2, \ldots, w_T\}$, we map W as $\text{BERT}_{\text{BASE}}$ input form: $\{w_0, w_1, w_2, \ldots, w_T, w_{T+1}\}$, where w_0 and w_{T+1} are the start of the sentence named "[CLS]" and the end of the sentence named "[SEP]," respectively. After $\text{BERT}_{\text{BASE}}$, we can get corresponding embedding $\mathcal{A} = \{a_0, a_1, \ldots, a_T, a_{T+1}\}$ of this sentence, where a_0 and a_{T+1} denote the [CLS] and [SEP] token embedding, respectively.

For one sentence $W_i \in \mathcal{S}$, we only focus on the known head entity h_i and tail entity t_i. For one sentence $W_c \in \mathcal{Q}$, we adopt a span-based tagging model following prior works [9], [11] to extract M entities $\{x_1, x_2, \ldots, x_M\}$ (note that M is variable). To be specific, we build each token a binary tag $(1/0)$, where "1" indicates that the current token position is a start or an end of an entity, and "0" indicates other positions besides these. We apply two binary classifiers with a sigmoid output layer to predict the probability of the ith token a_i as a start or an end position. The detailed operations of the span-based tagging on a_i are given as follows:

$$\begin{aligned} p_i^s &= \text{sigmoid}(\mathbf{W}_s \cdot a_i + \mathbf{b}_s) \\ p_i^e &= \text{sigmoid}(\mathbf{W}_e \cdot a_i + \mathbf{b}_e) \end{aligned} \tag{1}$$

where p_i^s and p_i^e represent the probability of the ith token being the start and end positions for an entity, respectively. $\mathbf{W}_{(\cdot)}$ is the trainable weights, and $\mathbf{b}_{(\cdot)}$ represents the bias.

2) Node Feature Initialization: The node set $\mathcal{V} = \mathcal{V}^u \cup \mathcal{V}^r$, where $\mathcal{V}^u = \{v_1^u, v_2^u, \ldots, v_{2NK+M}^u\}$ and $\mathcal{V}^r = \{v_1^r, v_2^r, \ldots, v_{NK+1}^r\}$ represent $2NK + M$ entity nodes and $NK + 1$ relation nodes, respectively. For an **entity node** $v_i^u \in \mathcal{V}^u$, the initialized features are obtained by taking out the representations of corresponding positions from sentence embedding, denoted as $\mathbf{v}_i^u \in \mathbb{R}^d$. Note that, due to the different lengths of the entities, we use the first token embedding of an entity to represent itself [44]. For a **relation node** $v_j^r \in \mathcal{V}^r$, we use the information of the entire sentence to represent its initialized features, denoted as $\mathbf{v}_j^r \in \mathbb{R}^d$. The whole information of one sentence is obtained by, h_0, the output of $\text{BERT}_{\text{BASE}}$'s first token [CLS].

3) Edge Feature Initialization: We define $E(\cdot, \cdot)$ to denote the edge between two input nodes. The edge set $\mathcal{E} = \mathcal{E}^{ru} \cup \mathcal{E}^{uu} \cup \mathcal{E}^{rr}$ contains three types of edges: relation–entity edge $E(v_j^r, v_i^u) \in \mathcal{E}^{ru}$, entity–entity edge $E(v_i^u, v_{i'}^u) \in \mathcal{E}^{uu}$, and relation–relation edge $E(v_j^r, v_{j'}^r) \in \mathcal{E}^{rr}$. Note that all these edges are undirected, e.g., $E(v_j^r, v_i^u) = E(v_i^u, v_j^r)$. Moreover, there are two characteristics of relations and entities. One is that relations are more independent compared to entities, which means that different sentences can convey the same relation. For example, sentences W_2 and W_c in Fig. 2 use completely different words, but both convey the relation of *mother*. The other is that the associations between entities in different sentences are weak. For example, although sentences W_2 and W_c express the same relation, the entity *Teri Bauer* in W_2 does not have any associations with entity *Oichi* in sentence W_c. According to the above two characteristics, our proposed TGIN is not a fully connected graph; we only build $E(v_i^r, v_j^u)$ and $E(v_i^u, v_{i'}^u)$ for nodes within a sentence and then build $E(v_j^r, v_{j'}^r)$ for all relation nodes. By doing this, the potential interactions between entities and relations in different sentences are completed by relation nodes, which reduces the noise that may be introduced directly establishing $E(v_i^r, v_j^u)$ and $E(v_i^u, v_{i'}^u)$ between nodes from different sentences.

We define **relation–entity edge** $E(v_j^r, v_i^u)$ as a 2-D vector connecting relation node v_j^r and entity node v_i^u, denoted by $\mathbf{e}_{ji}^{ru} = [e_{ji}^{(1),ru}; e_{ji}^{(2),ru}]$. Here, the first dimension and the second dimension are real numbers ranging from 0 to 1, representing the probability that v_i^u acts as the head entity of v_j^r and v_i^u is regarded as the tail entity of this relation node, respectively. In the support set $\mathcal{S}$, the initialized features $(\mathbf{e}_{ji}^{ru})^0 = [1; 0]$ indicate that v_i^u is an exact head entity of v_j^r; $(\mathbf{e}_{ji}^{ru})^0 = [0; 1]$ means v_i^u as the tail of v_j^r. In the query set $\mathcal{Q}$, we initialize $(\mathbf{e}_{ji}^{ru})^0 = [0.5; 0.5]$. As illustrated in Fig. 3, this initialization method ensures that the incorrect triplets (Case 1) do not convey information for node aggregation, while correct triplets (Case 2) satisfy translation algebraic operations.

Entity–entity edge $E(v_i^u, v_{i'}^u)$ is inseparable from the relation–entity edges $E(v_j^r, v_i^u)$ and $E(v_j^r, v_{i'}^u)$. It means that the correlation between two entity nodes depends on the relation node to which they are simultaneously connected; thus, we consider the entity–entity edge within triplets. v_i^u, $v_{i'}^u$, and v_j^r can be grouped together to form a triplet of two cases as $\langle v_i^u, v_j^r, v_{i'}^u \rangle$ and $\langle v_{i'}^u, v_j^r, v_i^u \rangle$. We denote the initialized features of $E(v_i^u, v_{i'}^u)$ as a 1-D vector and calculate it separately in the mentioned two cases

$$\left(e_{ii'}^{uu}\right)^0 = \begin{cases} \left(e_{ji}^{(1),ru}\right)^0 \cdot \left(e_{ji'}^{(2),ru}\right)^0, & \text{if } \left\langle v_i^u, v_j^r, v_{i'}^u \right\rangle \\ \left(e_{ji}^{(2),ru}\right)^0 \cdot \left(e_{ji'}^{(1),ru}\right)^0, & \text{if } \left\langle v_{i'}^u, v_j^r, v_i^u \right\rangle. \end{cases} \tag{2}$$

Relation–relation edge $E(v_j^r, v_{j'}^r)$ reflects the similarity of two relation nodes. We define the features of relation–relation edge as a 1-D vector

$$(e_{jj'}^{rr})^0 = \begin{cases} 1, & \text{if } y_j = y_{j'} \text{ and } j, j' \le N \times K \\ 0, & \text{if } y_j \ne y_{j'} \text{ and } j, j' \le N \times K \\ 0.5, & \text{otherwise} \end{cases} \tag{3}$$

where y_j and $y_{j'}$ are the labels of v_j^r and $v_{j'}^r$, respectively.

B. Node Feature Update

The data flow of node feature update is illustrated in Fig. 2. Different from other existing graph neural networks that update the given node by aggregating its neighboring nodes at one time [29], we devise an aggregation method based on translation algebraic operations to update one given node

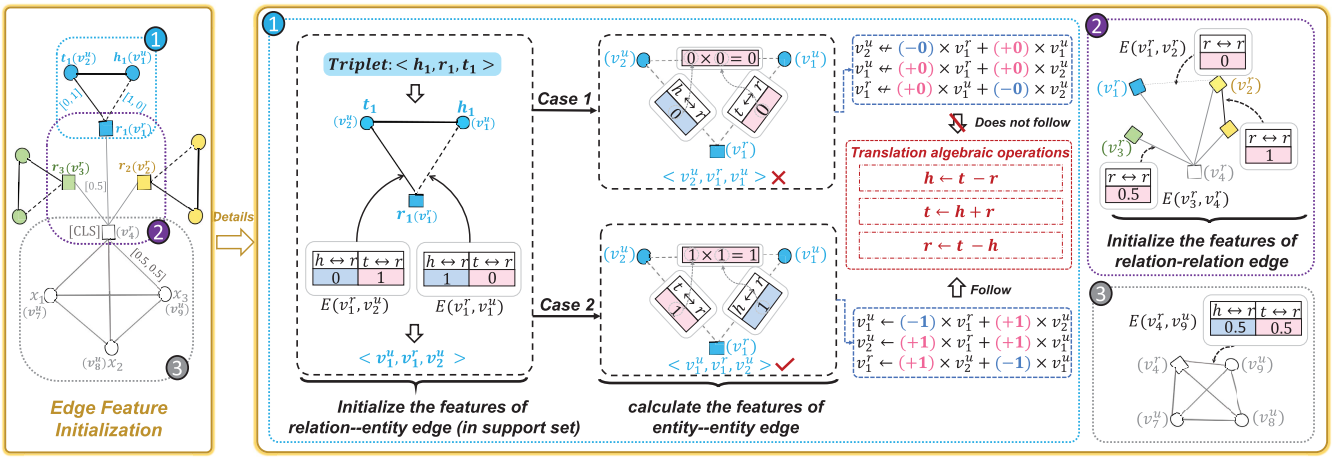


Fig. 3. Details of edge feature initialization. The initialized feature of relation–entity edge is a 2-D vector, where the first dimension and the second dimension represent that the probability of this edge is the head entity–relation ($h \leftrightarrow r$) and the tail entity–relation ($t \leftrightarrow r$), respectively. The feature of an entity–entity edge is calculated by the product of relation–entity edges within a triplet. The blue part (marked with ①) illustrates how relation–entity and entity–entity edges are initialized in a support set. There are two possible combinations of v_1^u, v_1^r, and v_2^u: $\langle v_1^u, v_1^r, v_2^u\rangle$ and $\langle v_2^u, v_1^r, v_1^u\rangle$. It is clear from $\langle h_1, r_1, t_1\rangle$ that only $\langle v_1^u, v_1^r, v_2^u\rangle$ is the correct triplet. Thus, we set $E(v_1^r, v_2^u) = [0; 1]$ and $E(v_1^r, v_1^u) = [1; 0]$ to ensure that the initialized nodes and edges features of $\langle v_1^u, v_1^r, v_2^u\rangle$ can follow translation algebraic operations (case 2), whereas $\langle v_2^u, v_1^r, v_1^u\rangle$ cannot (case 1). The gray part (marked with ③) illustrates how relation–entity edges are initialized in the query set. The purple part (marked with ②) illustrates how relation–relation edges are initialized. Note that all edges are undirected, e.g., $E(v_1^r, v_1^u) = E(v_1^u, v_1^r)$.

by aggregating its neighboring nodes with units of triplets. Specifically, we define the neighborhood function as $\mathcal{N}(\cdot,\cdot)$, which takes a node set and a given node as input and outputs all neighboring nodes from this node set for the given node.

For a given **entity node** v_i^u, its neighboring nodes include the relation node set $\mathcal{N}(\mathcal{V}^r, v_i^u)$ and the entity node set $\mathcal{N}(\mathcal{V}^u, v_i^u)$. All these neighboring nodes can form two types of triplet sets

$$\begin{aligned}\mathscr{T}_h(v_i^u) &= \left\{\langle v_i^u, v_j^r, v_{i'}^u\rangle \middle| v_{i'}^u \in \mathcal{N}(\mathcal{V}^u, v_i^u), v_k^r \in \mathcal{N}(\mathcal{V}^r, v_i^u)\right\}\\ \mathscr{T}_t(v_i^u) &= \left\{\langle v_{i'}^u, v_j^r, v_i^u\rangle \middle| v_{i'}^u \in \mathcal{N}(\mathcal{V}^u, v_i^u), v_j^r \in \mathcal{N}(\mathcal{V}^r, v_i^u)\right\}.\end{aligned} \tag{4}$$

Here, $\mathscr{T}_h(v_i^u)$ represents the triplet set where v_i^u acts as the head entity, and $\mathscr{T}_t(v_i^u)$ is the set where v_i^u is regarded as the tail entity. According to the translation algebraic operations, the aggregation effects on v_i^u from one of these two types of triplets in layer $\ell-1$ are calculated as follows:

$$\boldsymbol{\phi}_{h,v_i^u}^{\ell-1}\left(v_i^u \leftarrow v_{i'}^u, v_j^r\right) = \left(e_{ii'}^{uu}\right)^{\ell-1}\cdot\left(\mathbf{v}_{i'}^u\right)^{\ell-1} - \left(e_{ji}^{(1),ru}\right)^{\ell-1}\cdot\left(\mathbf{v}_j^r\right)^{\ell-1} \tag{5}$$

$$\boldsymbol{\phi}_{t,v_i^u}^{\ell-1}\left(v_i^u \leftarrow v_{i'}^u, v_j^r\right) = \left(e_{i'i}^{uu}\right)^{\ell-1}\cdot\left(\mathbf{v}_{i'}^u\right)^{\ell-1} + \left(e_{ji}^{(2),ru}\right)^{\ell-1}\cdot\left(\mathbf{v}_j^r\right)^{\ell-1} \tag{6}$$

where $\boldsymbol{\phi}_{h,v_i^u}^{\ell-1}$ in (5) follows the translation algebraic operation as $\boldsymbol{h \leftarrow t - r}$ and denotes the aggregation effect on v_i^u from $\langle v_i^u, v_j^r, v_{i'}^u\rangle \in \mathscr{T}_h(v_i^u)$. Likewise, $\boldsymbol{\phi}_{t,v_i^u}^{\ell-1}$ in (6) obeys the operation of $\boldsymbol{t \leftarrow h + r}$ and represents the aggregation effect on v_i^u from $\langle v_{i'}^u, v_j^r, v_i^u\rangle \in \mathscr{T}_t(v_i^u)$.

For a given **relation node** v_j^r, its neighboring nodes include the entity node set $\mathcal{N}(\mathcal{V}^u, v_j^r)$ and the relation node set $\mathcal{N}(\mathcal{V}^r, v_j^r)$. First, the entity node set can form the triplet set

$$\mathscr{T}_r(v_j^r) = \left\{\langle v_i^u, v_j^r, v_{i'}^u\rangle \middle| v_i^u, v_{i'}^u \in \mathcal{N}(\mathcal{V}^u, v_j^r)\right\} \tag{7}$$

and the aggregation effect on v_j^r from $\langle v_{i'}^u, v_j^r, v_i^u\rangle \in \mathscr{T}_r(v_j^r)$ can be represented as

$$\boldsymbol{\phi}_{r,v_j^r}^{\ell-1}\left(v_j^r \leftarrow v_i^u, v_{i'}^u\right) = \left(e_{ji}^{(2),ru}\right)^{\ell-1}\cdot\left(\mathbf{v}_i^u\right)^{\ell-1} - \left(e_{ji'}^{(1),ru}\right)^{\ell-1}\cdot\left(\mathbf{v}_{i'}^u\right)^{\ell-1} \tag{8}$$

where $\boldsymbol{\phi}_{r,v_j^r}^{\ell-1}$ follows the operation of $\boldsymbol{r \leftarrow t - h}$. Next, the aggregation effect on v_j^r from each neighboring relation node $v_{j'}^r \in \mathcal{N}(\mathcal{V}^r, v_j^r)$ is calculated as

$$\boldsymbol{\psi}_{v_j^r, v_{j'}^r}^{\ell-1} = \left(e_{jj'}^{rr}\right)^{\ell-1}\cdot\left(\mathbf{v}_{j'}^r\right)^{\ell-1}. \tag{9}$$

In (5), (6), (8), and (9), $(\mathbf{v}_*^*)^{\ell-1}$ and $(e_*^*)^{\ell-1}$ represent node features and edge features in layer $\ell-1$, respectively. Here, the edge feature is used as the aggregation strength like an attention mechanism for node feature updates.

Then, we adopt a softmax function to calculate the normalized weights for each $\boldsymbol{\phi}_{h,v_i^u}^{\ell-1}$, $\boldsymbol{\phi}_{t,v_i^u}^{\ell-1}$, $\boldsymbol{\phi}_{r,v_j^r}^{\ell-1}$, and $\boldsymbol{\psi}_{v_j^r,v_{j'}^r}^{\ell-1}$ denoted as $\boldsymbol{\alpha}_{h,v_i^u}^{\ell-1}$, $\boldsymbol{\alpha}_{t,v_i^u}^{\ell-1}$, $\boldsymbol{\alpha}_{r,v_j^r}^{\ell-1}$, and $\boldsymbol{\beta}_{v_j^r,v_{j'}^r}^{\ell-1}$, respectively. These weights represent how much information is being propagated from one unit to a given node. As a result, the updating process for an entity node v_i^u is achieved by

$$\left(\mathbf{v}_i^u\right)^{\ell} = f_v\left(\left(\mathbf{v}_i^u\right)^{\ell-1} + \overbrace{\sum_{\mathscr{T}_h(v_i^u)} \boldsymbol{\alpha}_{h,v_i^u}^{\ell-1}\boldsymbol{\phi}_{h,v_i^u}^{\ell-1}}^{\text{Message from Triplet set } \mathscr{T}_h(v_i^u)} + \overbrace{\sum_{\mathscr{T}_t(v_i^u)} \boldsymbol{\alpha}_{t,v_i^u}^{\ell-1}\boldsymbol{\phi}_{t,v_i^u}^{\ell-1}}^{\text{Message from Triplet set } \mathscr{T}_t(v_i^u)}\right) \tag{10}$$

and the updating process for a relation node v_j^r is

$$\left(\mathbf{v}_j^r\right)^\ell = f_v\left((\mathbf{v}_j^r)^{\ell-1} + \overbrace{\sum_{r_{j'}\in\mathcal{N}(\mathcal{V}^r,v_j^r)} \boldsymbol{\beta}^{\ell-1}_{v_j^r,v_{j'}^r}\boldsymbol{\psi}^{\ell-1}_{v_j^r,v_{j'}^r}}^{\text{Message from Neighboring Relations}} + \overbrace{\sum_{\mathscr{T}_r(v_j^r)} \boldsymbol{\alpha}^{\ell-1}_{r,v_j^r}\boldsymbol{\phi}^{\ell-1}_{r,v_j^r}}^{\text{Message from Triplet set } \mathscr{T}_r(v_j^r)}\right) \tag{11}$$

where $(\mathbf{v}_i^u)^\ell$ and $(\mathbf{v}_j^r)^\ell$ denote the updated entity node features and the relation node features in layer ℓ, respectively. $f_v(\cdot)$ is a node update network with two linear layers and an activation function of ReLU.

C. Edge Feature Update

Consistent with node feature updates, edge features are still updated with the units of triplets. We formulate a score function $f_s^\ell(\cdot)$ to measure the trustworthiness of triplet $\langle v_i^u, v_j^r, v_{i'}^u\rangle$ in layer ℓ as

$$\begin{aligned} s^\ell\left(\left\langle v_i^u, v_j^r, v_{i'}^u\right\rangle\right) &= -\left\|\left(\mathbf{v}_i^u\right)^\ell + \left(\mathbf{v}_j^r\right)^\ell - \left(\mathbf{v}_{i'}^u\right)^\ell\right\|_2^2 \\ f_s^\ell\left(\left\langle v_i^u, v_j^r, v_{i'}^u\right\rangle\right) &= \frac{\exp\left(s^\ell\left(\left\langle v_i^u, v_j^r, v_{i'}^u\right\rangle\right)\right)}{\sum_{\mathscr{T}_r\left(v_j^r\right)}\exp\left(s^\ell\left(\left\langle \cdot, v_j^r, \cdot\right\rangle\right)\right)} \end{aligned} \tag{12}$$

where $\|\cdot\|_2$ calculates the 2-norm of a vector, and $\mathscr{T}_r(v_j^r) = \{\langle\cdot, v_j^r, \cdot\rangle | v_j^r \in \mathcal{V}^r\}$ denotes the triplet set with relation v_j^r. Apparently, $f_s^\ell(\cdot)$ will be greater when the features of nodes in a triplet can better satisfy translation algebraic operations. On this basis, three types of edges are updated with the reobtained node features, as illustrated in Fig. 2, and the details are described as follows.

For a **relation–entity edge** $E(v_j^r, v_i^u)$, let $\mathscr{T}_{rh}(v_j^r, v_i^u) = \{\langle v_i^u, v_j^r, \cdot\rangle | v_j^r \in \mathcal{V}^r, v_i^u \in \mathcal{N}(\mathcal{V}^u, v_j^r)\}$ and $\mathscr{T}_{rt}(v_j^r, v_i^u) = \{\langle\cdot, v_j^r, v_i^u\rangle | v_j^r \in \mathcal{V}^r, v_i^u \in \mathcal{N}(\mathcal{V}^u, v_j^r)\}$ denote the triplet sets that v_i^u acts as the head entity of v_j^r and u_i is regarded as the tail entity of v_j^r, respectively. $\mathcal{N}(\cdot,\cdot)$ is the neighboring function defined in Section IV-B. The feature vector $(\mathbf{e}_{ji}^{ru})^\ell$ for this type of edge in layer ℓ is calculated as

$$\begin{aligned} \left(\mathbf{e}_{ji}^{ru}\right)^\ell &= \left[\sum_{\mathscr{T}_{rh}\left(v_j^r, v_i^u\right)} f_s^\ell\left(\left\langle v_i^u, v_j^r, \cdot\right\rangle\right); \sum_{\mathscr{T}_{rt}\left(v_j^r, v_i^u\right)} f_s^\ell\left(\left\langle \cdot, v_j^r, v_i^u\right\rangle\right)\right] \\ &= \left[\left(e_{ji}^{(1),ru}\right)^\ell; \left(e_{ji}^{(2),ru}\right)^\ell\right] \end{aligned} \tag{13}$$

where the value $(e_{ji}^{(1),ru})^\ell$ of the first dimension is obtained by the sum of triplet scores when v_i^u acts as the head entity of v_j^r in layer ℓ. Analogously, $(e_{ji}^{(2),ru})^\ell$ for the second dimension is the sum of triplet scores when v_i^u is regarded as the tail entity of v_j^r in layer ℓ.

Upon that, the features of an **entity–entity edge** $E(v_i^u, v_{i'}^u)$ in layer ℓ can be updated by

$$\left(e_{ii'}^{uu}\right)^\ell = \begin{cases} \left(e_{ji}^{(1),ru}\right)^\ell \cdot \left(e_{ji'}^{(2),ru}\right)^\ell, & \text{if } \left\langle v_i^u, v_j^r, v_{i'}^u\right\rangle \\ \left(e_{ji}^{(2),ru}\right)^\ell \cdot \left(e_{ji'}^{(1),ru}\right)^\ell, & \text{if } \left\langle v_{i'}^u, v_j^r, v_i^u\right\rangle. \end{cases} \tag{14}$$

For a **relation–relation edge** $E(v_j^r, v_{j'}^r)$, we update its features by adopting a simple but effective dot production to calculate the similarity between two relation nodes

$$\left(e_{jj'}^{rr}\right)^\ell = \frac{\exp\left(\left(\mathbf{v}_j^r\right)^\ell \cdot \left(\mathbf{v}_{j'}^r\right)^\ell\right)}{\sum_{j\neq j'',\, j''=1}^{NK}\exp\left(\left(\mathbf{v}_j^r\right)^\ell \cdot \left(\mathbf{v}_{j''}^r\right)^\ell\right)} \tag{15}$$

where $(\mathbf{v}_j^r)^\ell$, $(\mathbf{v}_{j'}^r)^\ell$ and $(\mathbf{v}_{j''}^r)^\ell$ denote the features of v_j^r, $v_{j'}^r$ and $v_{j''}^r$ in layer ℓ.

D. Learning and Prediction

We will elaborate on the training loss function and illustrate how to make predictions on query sentences in this section.

1) Training Function: We optimize TGIN by considering node features and edge features simultaneously. The loss function can be given as

$$\mathcal{L} = \lambda_n\mathcal{L}_n + \lambda_e\mathcal{L}_e \tag{16}$$

where $\mathcal{L}_n$ and $\mathcal{L}_e$ stand for node loss and edge loss, respectively. λ_n and λ_e are the weights to balance their importance. The details of $\mathcal{L}_n$ and $\mathcal{L}_e$ are given as follows.

First, node loss $\mathcal{L}_n$ aims to facilitate the representations of entity and relation nodes in the positive triplets to satisfy translation algebraic operations, but the negative triplets do not. Specifically, we assume that $\mathscr{T}_G = \{\langle v^h, v^r, v^t\rangle | v^h, v^t \in \mathcal{N}(\mathcal{V}^u, v^r), v^r \in \mathcal{V}^r\}$ and $\mathscr{T}_G' = \{\langle v^h, v^r, v^t\rangle'\}$ denote the positive triplet set and the negative triplet set in $\mathcal{G}$, respectively. $\mathcal{L}_n$ makes the triplet score of $\langle v^h, v^r, v^t\rangle$ greater than $\langle v^h, v^r, v^t\rangle'$

$$\begin{aligned} \mathcal{L}_n = \sum_{\ell=1}^{L}\sum_{\mathscr{T}_G'}\sum_{\mathscr{T}_G} \max\Big(0,\, & f_s^\ell\left(\left\langle v^h, v^r, v^t\right\rangle'\right) \\ & + \Theta - f_s^\ell\left(\left\langle v^h, v^r, v^t\right\rangle\right)\Big) \end{aligned} \tag{17}$$

where $f_s^\ell(\cdot)$ is defined in (12). L denotes the layer number of $\mathcal{G}$, and $\Theta > 0$ is a margin hyperparameter to be tuned. The construction of $\mathscr{T}_G'$ is in (18), where the head or tail entity in the positive triplet is replaced by a random entity vector, i.e., $v_\dagger^h \notin \mathcal{N}(\mathcal{V}^u, v^r)$, $v_\dagger^t \notin \mathcal{N}(\mathcal{V}^u, v^r)$ (but not simultaneously)

$$\begin{aligned} \mathscr{T}_G' &= \left\{\left\langle v^h, v^r, v^t\right\rangle'\right\} \\ &= \left\{\left\langle v_\dagger^h, v^r, v^t\right\rangle | v^t \in \mathcal{N}\left(\mathcal{V}^u, v^r\right), v^r \in \mathcal{V}^r\right\} \\ &\quad \cup \left\{\left\langle v^h, v^r, v_\dagger^t\right\rangle | v^h \in \mathcal{N}\left(\mathcal{V}^u, v^r\right), v^r \in \mathcal{V}^r\right\}. \end{aligned} \tag{18}$$

Next, edge loss $\mathcal{L}_e$ aims to constrain the uncertain edges to ground truth and is defined as follows:

$$\mathcal{L}_e = \sum_{\ell=1}^{L}\mathcal{L}_{CE}(Y, \hat{e}) \tag{19}$$

where $\mathcal{L}_{CE}$ is the binary cross-entropy loss. $\hat{e}$ and Y represent predictive edge labels and ground-truth edge labels, respectively. Finally, we summarize the steps of the learning process for TGIN in Algorithm 1.

Algorithm 1 Learning Process of TGIN

Input: $\mathcal{S} = \{(W_i, \langle h_i, r_i, t_i\rangle)|r_i \in \mathcal{R}\}_{i=1}^{NK}$, $\mathcal{Q} = \{W_c\}_{c=NK+1}^{NK+C}$
Output: The network parameters of TGIN.

Obtain M entities $\{x_1, x_2, \ldots, u_M\}$ from $W_c \in \mathcal{Q}$;
Construct $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \mathcal{V}^u \cup \mathcal{V}^r$; $\mathcal{E} = \mathcal{E}^{ru} \cup \mathcal{E}^{uu} \cup \mathcal{E}^{rr}$;
Initialize entity nodes $\mathcal{V}^u = \{v_1^u, \cdots, v_{2NK+M}^u\}$ and relation nodes $\mathcal{V}^r = \{v_1^r, \cdots, v_{NK+1}^r\}$;
Initialize relation–entity edge $E(v_j^r, v_i^u) \in \mathcal{E}^{ru}$, entity–entity edge $E(v_i^u, v_{i'}^u) \in \mathcal{E}^{uu}$, and relation–relation edge $E(v_j^r, v_{j'}^r) \in \mathcal{E}^{rr}$;
while TGIN does not converge **do**
  for $\ell = 1 \to L$ **do**
    for $i = 1 \to 2NK + M$ **do**
      Calculate the aggregation effect from neighboring triplets on v_i^u by Equation (5)(6);
      Update the features of v_i^u by Equation (10);
    end for
    for $j = 1 \to NK + 1$ **do**
      Calculate the aggregation effect from neighboring triplets on v_j^r by Equation (8);
      Calculate the aggregation effect from neighboring relations by Equation (9);
      Update the features of v_j^r by Equation (11);
    end for
    for $j = 1 \to NK + 1$ **do**
      for $i = 1 \to 2NK + M$ **do**
        Calculate each triplet score by Equation (12);
        Update the features of $E(v_j^r, v_i^u)$ by Equation (13);
      end for
    end for
    for $j = 1 \to NK + 1$ **do**
      for $(i, i') = 1 \to NK + M, i \neq i'$ **do**
        Update the features of $E(v_i^u, v_{i'}^u)$ by Equation (14);
      end for
    end for
    for $(j, j') = 1 \to NK + 1, j \neq j'$ **do**
      Update the features of $E(v_j^r, v_{j'}^r)$ by Equation (15);
    end for
  end for
  Calculate $\mathcal{L}$ for this episode by Equation (16)(17)(19);
  Let $\mathcal{L}$ to be minimized in the next episode.
end while
Return network parameters of TGIN.

2) Prediction on Query Sentences: The class of relation node v_c^r in a query sentence can be predicted by $(e_{cj}^{rr})^L$ in the last layer L as

$$p_c^r = \text{Softmax}\left(\sum_{j\neq c, j=1}^{NK} \left(e_{cj}^{rr}\right)^L\right) \tag{20}$$

where p_c^r is the probability that v_c^r belongs to each relation category. The positive triplet $\langle v_c^h, v_c^r, v_c^t\rangle$ for relation node v_c^r can be selected by (12) in layer L as

$$\left\langle v_c^h, v_c^r, v_c^t\right\rangle = \arg\max_{\mathcal{T}_r(v_c^r)} \left(f_s^L\left(\langle\cdot, v_c^r, \cdot\rangle\right)\right) \tag{21}$$

where $\mathcal{T}_r(v_c^r) = \{\langle\cdot, v_c^r, \cdot\rangle|v_c^r \in \mathcal{V}^r\}$ represents the triplet set with relation v_c^r.

TABLE I
STATISTICS OF THREE DATASETS

Categories	FewRel	NYT-single	TACRED
#Training Relations	50	5	14
#Validation Relations	15	4	8
# Test Relations	15	5	8
# Sentences per Relation	700	200/60	250/80
# Triplets per Sentence	1	1	1

V. EXPERIMENTS

In this section, we describe the experimental settings, which include datasets in Section V-A, comparison models in Section V-B, and the implementation details of experiments in Section V-C. Then, we evaluate the overall performance of TGIN in Section V-D, followed by ablation studies in Section V-E. Moreover, we analyze the sensitivity of model parameters in Section V-F and present case studies to show the validity of TGIN in Section V-G.

A. Datasets

We reconstruct three datasets to satisfy the few-shot relational triplet extraction task. The statistical results of the datasets are shown in Table I. Note that there are no overlapping classes among training, validation, and testing in these datasets.

1) FewRel [48]: The only dataset for few-shot RC is annotated by crowd workers. It is derived from Wikipedia and contains 70 000 instances in 100 relation categories, that is, each relation has 700 instances. The head and tail entities corresponding to the relation of each instance have been marked. However, only 80 relations of FewRel are released. Therefore, we randomly select 50 for training, 15 for validation, and 15 for testing in the following studies.

2) NYT-Single: It is a subset of the public dataset NYT [12], where the row data are extracted from NYT articles. NYT has been widely used for evaluating entity relation extraction models in the common task. We choose out the instances that contain only one relation (one triplet) from NYT and, finally, obtain instances with 14 relation categories. Due to the number of sentences for the relation types being various, we select five relations with a number of sentences less than 100 as the test and randomly select five relations of the remaining nine relations as training and four relations as validation. Specifically, each relation for the training set, validation set, and test set has 200, 200, and 60 instances, respectively.

3) TACRED [49]: It is a human-annotated relation extraction dataset. It is built over newswire and web text from the corpus used in the yearly TAC Knowledge Base Population (TAC KBP) challenges. We remove the relation types with less than 50 instances and, finally, obtain instances with 30 relation categories. Due to the number of instances for the relation types being various, we select 14 relations with a

number of instances of more than 250 as the training set and randomly select eight relations of the remaining 16 relations as the validation set and eight relations as the test set. More concretely, each relation for the training set, the validation set, and the test set has 250, 250, and 80 instances, respectively.

B. Compared Models

Eight baselines are categorized into three groups.

- First Only Few-Shot Relational Triplet Extraction Method (MPE [19]): To the best of our knowledge, MPE is the only existing model for the few-shot relational triplet extraction task in the previous works, and the code is not available, so we only extract the results from the original paper.
- Second We use five few-shot RC methods: MatchNet [34], ProtoNet [20], GNN [45], HATT [16], and MLMAN [15]. In order to make fair comparisons, we obtain the source code of these five methods from Github and then replace their encoder with $\text{BERT}_{\text{BASE}}$. Subsequently, we employ these models after span-based tagging and retrain them on our reconstructed datasets for few-shot relational triplet extraction. The details of these six baselines are listed as follows.
- Third To verify the performance of our TGIN comprehensively, we also use two general joint entity relation extraction models, PFN [46] and GRTE [47], that achieve excellent results with large amounts of labeled training data. We retrain these models on the training set and fine-tune them on the test set using a few labeled data to verify their performance in a few-shot setting.

1) MPE [19]: Use $\text{BERT}_{\text{BASE}}$ as an encoder, and adopt a new CRF [36] to identify the head and tail entities in a query sentence. Then, it utilizes a prototype-based metric-learning network to classify relations.

2) MatchNet [34]: It is a network via an LSTM as the read-attention method over the whole support instances; then, classify the category for a query instance by calculating cosine similarity.

3) ProtoNet [20]: A typical metric-learning-based model that assumes that each class has a prototype.

4) GNN [45]: It is a graph neural network that encodes each support instance and query instance as a node in the graph.

5) HATT [16]: An HATT uses instance-level and feature-level attention schemes to enhance the representation of prototypes.

6) MLMAN [15]: It is a MLMAN for FSL that enhances the representation of support and query instances via local matching and aggregation, and further calculates the class matching score between the query instance and each candidate class.

7) PFN [46]: It is a joint entity relation extraction model that uses $\text{BERT}_{\text{BASE}}$ as an encoder and designs a filter network for entity recognition and relation extraction.

8) GRTE [47]: It is a joint entity relation extraction model that makes full use of the associations of relations and entities, and achieves state of the art with an amount of labeled training data on the NYT dataset based on the $\text{BERT}_{\text{BASE}}$ backbone.

C. Implementation Details

We use the Adam optimizer [50] with an initial learning rate of 5×10^{-4} and a weight decay of 10^{-6}. For both datasets, we investigate in the N-way K-shot scenarios where one training episode containing $N \times K$ support instances and one query instance. We train all models on the training set and use grid search to determine the best hyperparameters on the validation set and testing on the test set.

We employ TGIN as a two-layer heterogeneous graph. For FewRel, we set $\lambda_n/\lambda_e = 2$ and train 30 000 iterations. 5-way and 10-way experiments are conducted, and the minibatch size is 4. The learning rate is reduced in half every 5000 episodes. For TACRED, we set $\lambda_n/\lambda_e = 1.3$ and conduct 5-way and 3-way experiments. The minibatch size is 2, and iterations are 3000 with the learning rate halved for every 500 episodes. For NYT-single, we set $\lambda_n/\lambda_e = 1$ and conduct 5-way and 3-way experiments. The minibatch size is 2, and the learning rate is halved for every 200 episodes because NYT-single is a smaller dataset and requires fewer iterations, which are set as 1000. In most of our experiments, Θ is set to 2.0. The analysis of λ_n, λ_e, and the graph layer L can be found in Section V-F.

We use accuracy to evaluate the model performance. Specifically, the performance of entity pairs, relations, and triplets means whether the model could identify the entity pairs, classify the relation categories, and extract triplets correctly for given query sentences, respectively. All results are averaged over five independent runs.

D. Overall Results

The comparative results on three datasets are presented in Tables II–IV. Note that the best and second-best results are marked in bold and underline, respectively. We can make the following observations.

1) The proposed TGIN outperforms the other eight baseline models with significantly higher accuracy on the performance of triplets for three datasets. Importantly, the triplet accuracy of TGIN on FewRel achieves 8.97% and 10.74% improvements over the only existing work MPE in 5-way 5-shot and 10-way 10-shot settings, respectively. This demonstrates that TGIN can better apply to few-shot relational triplet extraction.
2) In all four settings of the three datasets, PFN and GRTE perform poorly and with 0% accuracy in many cases. On the one hand, these models always maintain a token table for each relation, and the items in this table usually denote the start and end token positions of two entities that possess this specific relation. Basically, it means that these models require relation-aware tables for triplet extraction. Since the relation types that we used for training and testing are disjoint, using these relation-aware tables for few-shot relation triplet extraction is ineffective. On the other hand, these models are not transferable. Although the training is done on the training set, the models are still invalid because only a few labeled instances are used for fine-tuning on the test set.
3) In all experimental settings and datasets, TGIN outperforms GNN that only considers entities or relations as nodes by a substantial margin. This demonstrates

TABLE II

RESULTS OF THE COMPARED MODELS ON FEWREL (%). RESULTS WITH * ARE EXTRACTED FROM THEIR ORIGINAL PAPERS. RESULTS WITH † REPRESENT THAT WE RETRAIN THE MODELS FOR THIS TASK. NOTE THAT THE BEST AND SECOND-BEST RESULTS ARE MARKED IN BOLD AND UNDERLINE, RESPECTIVELY

Model	5-way 1-shot			5-way 5-shot			10-way 1-shot			10-way 10-shot		
	Ent	Rel	Tri	Ent	Rel	Tri	Ent	Rel	Tri	Ent	Rel	Tri
MPE* [19]	-	-	-	25.03	**93.81**	23.34	-	-	-	14.85	**84.58**	12.08
MatchNet† [34]	18.71	75.82	15.41	20.74	84.59	18.67	12.52	59.36	8.23	20.03	77.81	16.29
Proto† [20]	19.44	77.56	15.92	25.13	87.36	21.17	14.51	65.72	10.44	19.75	76.03	15.36
GNN† [45]	21.62	78.35	17.79	26.03	88.40	24.52	15.73	66.91	11.40	20.46	77.69	16.14
HATT† [16]	23.01	80.13	19.65	26.42	89.92	24.83	15.58	67.08	11.68	20.83	78.62	16.97
MLMAN† [15]	23.42	82.51	20.39	30.46	91.76	28.52	20.35	70.65	15.38	23.31	81.94	19.20
PFN† [46]	3.50	3.63	0.13	3.53	3.84	0.16	0.72	2.50	0.00	1.70	2.66	0.00
GRTE† [47]	3.46	3.65	0.15	3.63	4.11	0.20	0.80	2.49	0.00	1.92	2.68	0.00
TGIN	**27.54**	**83.67**	**23.96**	**33.57**	93.05	**32.31**	**22.83**	**72.29**	**17.25**	**26.57**	83.71	**22.82**

TABLE III

RESULTS OF THE COMPARED MODELS ON NTY-SINGLE(%). THE CODE OF MPE IS NOT RELEASED, AND THERE IS NO EXPERIMENTAL DATA OF MPE ON NYT-SINGLE. RESULTS WITH † REPRESENT THAT WE RETRAIN THE MODELS FOR THIS TASK. NOTE THAT THE BEST AND SECOND-BEST RESULTS ARE MARKED IN BOLD AND UNDERLINE, RESPECTIVELY

Model	5-way 1-shot			5-way 5-shot			3-way 1-shot			3-way 5-shot		
	Ent	Rel	Tri	Ent	Rel	Tri	Ent	Rel	Tri	Ent	Rel	Tri
MatchNet† [34]	14.66	63.73	9.74	16.31	82.33	14.56	16.60	79.73	13.70	18.57	86.73	17.47
Proto† [20]	15.73	66.52	11.53	18.29	82.70	15.77	18.25	83.49	15.84	19.13	88.42	17.84
GNN† [45]	16.20	67.83	11.88	19.46	83.05	16.91	18.76	84.53	16.52	19.38	89.13	18.47
HATT† [16]	19.01	70.72	14.60	20.84	83.65	18.15	19.47	86.66	17.28	21.79	89.96	20.59
MLMAN† [15]	20.42	71.51	15.94	21.76	84.27	19.46	20.82	87.34	19.39	22.68	91.46	21.63
PFN† [46]	2.11	3.18	0.00	2.30	3.92	0.00	2.73	3.70	0.00	2.80	3.75	0.00
GRTE† [47]	2.50	3.35	0.00	2.42	4.00	0.00	2.75	3.91	0.00	2.90	3.90	0.00
TGIN	**22.16**	**74.28**	**17.35**	**24.35**	**85.12**	**21.80**	**22.05**	**89.40**	**21.53**	**24.75**	**93.24**	**23.88**

TABLE IV

RESULTS OF THE COMPARED MODELS ON TACRED (%). THE CODE OF MPE IS NOT RELEASED, AND THERE IS NO EXPERIMENTAL DATA OF MPE ON NYT-SINGLE. RESULTS WITH † REPRESENT THAT WE RETRAIN THE MODELS FOR THIS TASK. NOTE THAT THE BEST AND SECOND-BEST RESULTS ARE MARKED IN BOLD AND UNDERLINE, RESPECTIVELY

Model	5-way 1-shot			5-way 5-shot			3-way 1-shot			3-way 5-shot		
	Ent	Rel	Tri	Ent	Rel	Tri	Ent	Rel	Tri	Ent	Rel	Tri
MatchNet† [34]	15.50	68.26	11.83	19.10	84.10	17.72	22.03	85.00	19.92	25.30	89.95	23.80
Proto† [20]	17.02	70.55	13.15	21.10	85.23	19.19	23.33	87.12	20.60	26.79	90.30	25.29
GNN† [45]	17.51	71.45	13.80	22.29	87.02	20.30	24.07	88.44	22.26	27.44	91.98	26.60
HATT† [16]	22.73	73.17	17.06	25.19	89.66	23.16	27.20	89.92	25.57	28.08	92.40	27.67
MLMAN† [15]	22.94	75.25	18.44	26.65	91.05	25.03	29.00	92.80	27.37	30.50	92.87	28.85
PFN† [46]	2.50	3.33	0.00	2.56	3.85	0.00	2.74	4.00	0.00	2.77	4.30	0.16
GRTE† [47]	2.53	3.30	0.00	2.55	4.03	0.00	2.65	4.00	0.00	2.75	4.00	0.16
TGIN	**24.33**	**79.20**	**20.36**	**28.35**	**92.11**	**27.71**	**31.36**	**93.69**	**30.35**	**32.26**	**93.88**	**31.85**

that TGIN, which includes both entities and relations as nodes, is more effective than GNN for mining the features of limited data in few-shot scenarios.

4) MatchNet performs poorly in all settings for this task. This may be due to the fact that it treats all support instances equally, so it is difficult to capture the unique features associated with different entities or relations. HATT and MLMAN achieve relatively better results with hybrid attention and multilevel attention. Compared with the above method, our TGIN devises a graph aggregation and update method based on translation algebraic operations to simultaneously mine semantic

TABLE V

ACCURACY OF MPE AND TGIN$^{\diamond}$. "Δ" REPRESENTS THE CHANGE VALUE, AND ALL RESULTS ARE AVERAGED OVER FIVE RUNS

	5-way 5-shot			10-way 10-shot		
	Ent	Rel	Tri	Ent	Rel	Tri
MPE	**25.03**	**93.81**	23.34	**14.85**	**84.58**	12.08
TGIN$^{\diamond}$	24.61	92.83	**23.80**	14.36	83.52	**12.74**
Δ	↓0.42	↓0.98	↑0.46	↓0.49	↓1.06	↑0.66

features and retain structure features between entities and relations, which yields the best results.

5) In addition, under the settings of 5-way 5-shot and 10-way 10-shot, the accuracy of relations is 0.67% and 0.87% lower than MPE, whereas the accuracy of entity pairs or triplets achieves significant improvements over MPE. As a result, TGIN is able to extract entity pairs and triplets more efficiently than MPE. However, MPE utilizes the CRF, while TGIN utilizes the span-based tagging for entity recognition, so further investigation is necessary to determine whether this phenomenon is only due to span-based tagging. We replaced the span-based tagging of TGIN with CRF in ablation studies described in Section V-E.

E. Ablation Studies

To further analyze TGIN, we conduct ablation studies to verify the effectiveness of different parts and strategies in TGIN. Note that, in each group of experiments, the settings are identical except for the variables of interest. We construct four variants of TGIN as follows.

1) TGIN$^{\diamond}$: The variant model that replaces span-based tagging with CRF for entity recognition.

2) TGIN^{-R}: The variant model that removes relation nodes related to the query sentences. We directly connect the entities obtained from a query sentence with the relation nodes from the support sentences.

3) TGIN^{-NU}: The variant model in which nodes update without following translation algebraic operations. We achieve this by breaking (5), (6), and (8).

4) TGIN^{-EU}: The variant model in which edges update without following translation algebraic operations. We achieve this by breaking the score function of (12).

First, we evaluate the performance of TGIN$^{\diamond}$ and MPE on FewRel, and the results are illustrated in Table V. It is interesting to note that, although the TGIN$^{\diamond}$ performs slightly lower than MPE on entity pairs and relations, it still improves the performance of triplets by 0.46% in 5-way 5-shot and 0.66% in 10-way 10-shot. This may be attributed to the construction and update methods of TGIN, which makes it capable of making better inferences about triplets.

Second, we compare the accuracy of entity pairs on TGIN and TGIN$^{\diamond}$ when the training iterations are varying. As shown in Fig. 4, we can observe that TGIN leads to better performance than TGIN$^{\diamond}$ on both FewRel and NYT-single. On the FewRel, TGIN$^{\diamond}$ needs 2000 iterations to achieve the accuracy to 9.63%, while TGIN needs less than 100 iterations. Likewise, on the NYT-single, TGIN$^{\diamond}$ needs more than 1200 iterations to make the accuracy of entity pairs to 8.12%, while TGIN only needs a few iterations. The reason may be that TGIN

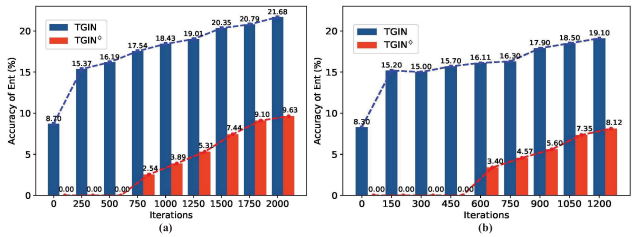


Fig. 4. Accuracy of entity pairs with TGIN and TGIN$^{\diamond}$ on different datasets, including (a) on FewRel and (b) on NYT-single. The experiment setting is 5-way 1-shot; we show the accuracy on the test set averaged over five runs.

TABLE VI

ACCURACY (%) ON DIFFERENT VARIANTS OF TGIN IN 5-WAY 1-SHOT SETTING. "Δ" REPRESENTS THE CHANGE VALUE OF THE VARIANTS COMPARED WITH TGIN, AND ALL RESULTS ARE AVERAGED OVER FIVE RUNS

	FewRel			NYT-single		
	Ent	Rel	Tri	Ent	Rel	Tri
TGIN	**27.54**	**83.67**	**23.96**	**22.16**	**74.28**	**17.35**
TGIN$^{\diamond}$	18.32	82.44	15.91	17.05	72.43	13.50
Δ	↓9.22	↓1.23	↓8.05	↓5.11	↓1.85	↓3.85
TGIN^{-R}	24.51	70.65	18.54	18.52	58.79	11.46
Δ	↓3.03	↓13.02	↓5.42	↓3.64	↓15.49	↓5.89
TGIN^{-NU}	25.32	82.20	21.64	19.62	72.80	14.87
Δ	↓2.22	↓1.47	↓2.32	↓2.54	↓1.48	↓2.48
TGIN^{-EU}	26.46	82.84	22.53	20.31	73.65	15.62
Δ	↓1.08	↓0.83	↓1.43	↓2.03	↓0.63	↓1.73

TABLE VII

ACCURACY (%) BY DIFFERENT NUMBERS OF TGIN LAYER IN 5-WAY 1-SHOT SETTING. "L" REPRESENTS THE LAYER, AND ALL RESULTS ARE AVERAGED OVER FIVE RUNS

	FewRel			NYT-single		
	Ent	Rel	Tri	Ent	Rel	Tri
TGIN (L=1)	26.41	82.44	22.50	21.42	74.03	16.75
TGIN (L=2)	**27.54**	**83.67**	**23.96**	**22.16**	**74.28**	**17.35**
TGIN (L=3)	27.51	83.62	23.81	22.03	74.10	16.98

is simple and has fewer parameters than TGIN$^{\diamond}$, which is easier to train. Third, we compare the performance of TGIN and its four variants in a 5-way 1-shot setting on two datasets, as shown in Table VI. Breaking translation algebraic operations in either node feature update or edge feature update dramatically decreases the performance of triplets, e.g., the accuracy of triplets for TGIN^{-NU}/TGIN^{-EU} drops by 2.32%/1.43% and 2.48%/1.73% on the FewRel and NYT-single datasets, respectively. This indicates not only that the update method of nodes is vital for capturing interactions between entities and relations, as well as extracting triplets, but also that the update method of edges still contributes to TGIN for a better prediction. Moreover, taking away the relation node generated by a query sentence makes the model performance drop terribly, e.g., TGIN^{-R} not only reduces the accuracy of relations on two datasets by 13.02% and 15.49% but it also reduces the accuracy of triplets by 5.42% and 5.89%. This illustrates that relation nodes are essential for the

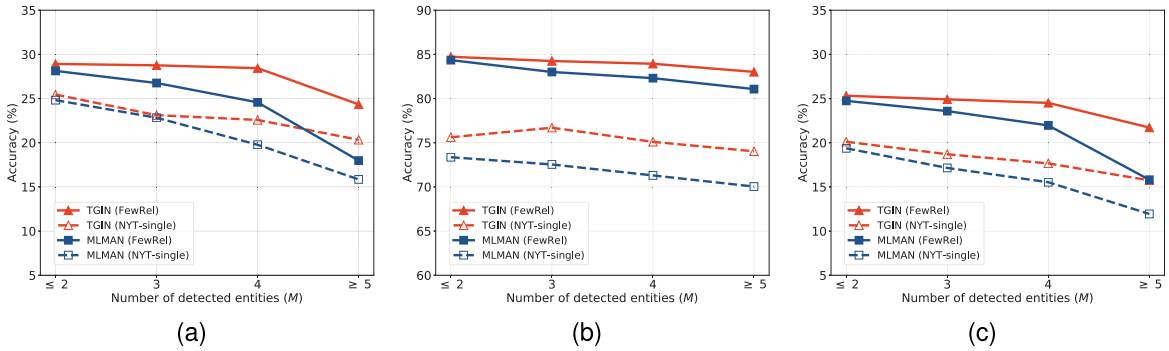


Fig. 5. Comparison of MLMAN and TGIN against different numbers of detected entities M on 5-way 1-shot setting. (a) Accuracy of entity pairs. (b) Accuracy of relations. (c) Accuracy of triplets.

performance of relations, as well as promoting information exchange between entities in different sentences to improve the performance of entity pairs and triplets.

From the above studies, we can conclude that both the construction method and the aggregation method of the TGIN are effective, as they can make full use of semantic information and contribute to the few-shot relational triplet extraction.

F. Parameters Sensitivity Analysis

1) Analysis on Detected Entities M: To further investigate the capability of TGIN to extract relational triplets when the number of detected entities M differs, we use MLMAN, which achieves the second-best result in Tables II–IV as a comparative model, and split the sentences in the test sets into four classes according to M ($\leq$2, 3, 4, and $\geq$5). As shown in Fig. 5, the TGIN achieves excellent performance in the overall four classes. We also notice that, from the easy class ($M \leq 2$) to the most difficult class ($M \geq 5$), though it is not surprising to find that the performance of these two models decreases, the accuracy for MLMAN drops significantly faster than TGIN. Since MLMAN uses the same span-based tagging model as TGIN, the possible reason might be that the construction and aggregation methods of TGIN can ensure our model maintains robustness and stability although there are more entities in a sentence.

2) Analysis on Graph Layers L: To evaluate the effect of layers stack, we vary the number of layers, and the experimental results are shown in Table VII. The number of layers is also regarded as the model depth. From Table VII, we can observe that, by increasing the depth of TGIN from one to two, the model performance is improved. However, when increasing the depth of TGIN is greater than two, the accuracy on the FewRel and NYT-single will decline. In other words, two propagation layers are sufficient to produce the triplets, and deeper layers might introduce noise and lead to overfitting. The observations show that our TGIN does not require a too complex network to predict relational triplets.

3) Analysis on Loss Weights λ_n and λ_e: As shown in Table VIII, TGIN can be constrained by using the node loss or the edge loss alone. We can observe that removing the node loss makes TGIN a serious decay than removing the edge loss on the performance of entity pairs. However, removing the edge loss has a greater impact on the performance of relations, which is 0.78% and 0.22% lower than removing the node loss on the fewRel and NYT-single, respectively. This indicates that the node loss mainly constrains the performance of entity pairs, while the edge loss is important for the performance of relations. Using the node loss and the edge loss together can make TGIN a better performance. To further investigate how each loss function affects the performance of TGIN, we evaluate the results by assigning $\lambda_n = 0$, $\lambda_e = 0$, and $\lambda_n/\lambda_e = \{1/3, 1/2, 1, 2, 3\}$, respectively, and the results are illustrated in Fig. 6. In our experiments, the model yields the best result when $\lambda_n/\lambda_e = 2$ and $\lambda_n/\lambda_e = 1$ are employed on the FewRel and NYT-single, respectively.

4) Analysis on Computational Efficiency: Table IX compares the computational efficiency between TGIN and other baselines. To be fair, we run all models on a Tesla A100

TABLE VIII

EVALUATING THE EFFECT OF EDGE LOSS FUNCTION AND NODE LOSS FUNCTION FOR TGIN IN 5-WAY 1-SHOT SETTING. ALL RESULTS ARE AVERAGED OVER FIVE RUNS

Loss Function		FewRel			NYT-single		
Node Loss	Edge Loss	Ent	Rel	Tri	Ent	Rel	Tri
√	×	27.06	81.73	22.63	20.81	72.25	16.14
×	√	24.33	82.51	21.19	16.93	72.47	13.55
√	√	**27.54**	**83.67**	**23.96**	**22.16**	**74.28**	**17.35**

TABLE IX

COMPUTATIONAL EFFICIENCY. PARAMETERS$_{\text{all}}$ IS THE NUMBER OF PARAMETERS FOR THE ENTIRE MODEL. PARAMETERS$_{\text{tuned}}$ REFERS TO THE PROPORTION OF TUNED PARAMETERS IN THE TOTAL MODEL PARAMETERS. EPOCH TIME IS THE AVERAGE TIME (MINUTE) THAT THE MODEL TAKES FOR AN EPOCH. MODELS WITH † REPRESENT THAT WE REPLACE THEIR ENCODER WITH BERT$_{\text{BASE}}$ AND APPLY THEM FOR THE TASK OF FEW-SHOT RELATIONAL TRIPLET EXTRACTION. TRIPLET ACC IS THE ACCURACY OF TRIPLETS ON FEWREL IN 5-WAY 1-SHOT SETTING

Model	Parameters$_{all}$	Parameters$_{tuned}$	Epoch time	Triplet Acc.
MatchNet†	110,623,488	3.79%	8.45	15.41
Proto†	110,034,432	3.21%	9.04	15.92
GNN†	112,743,188	5.74%	15.80	17.79
HATT†	110,035,968	3.22%	10.20	19.65
MLMAN†	113,580,288	6.24%	17.90	20.39
PFN	115,628,476	7.81%	19.52	0.13
GRTE	119,387,328	9.56%	22.40	0.15
TGIN	113,572,608	6.17%	17.10	23.96

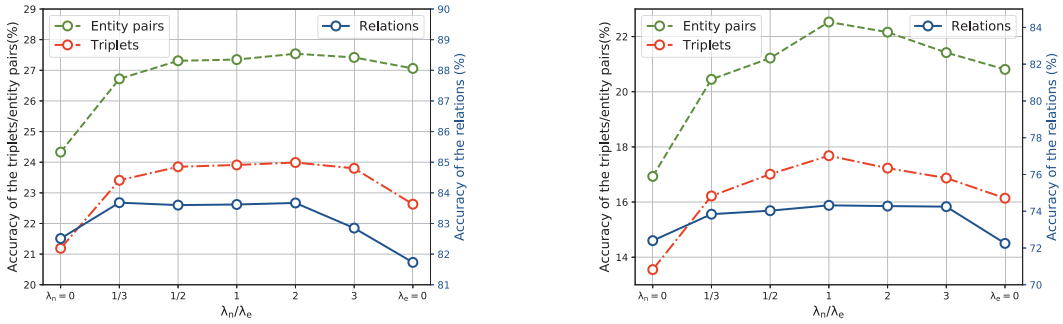


Fig. 6. Analysis of loss weights λ_n/λ_e in 5-way 1-shot setting. (a) Results on FewRel. (b) Results on NYT-single.

TABLE X

TRIPLETS INFERRED BY DIFFERENT MODELS. THE HEAD ENTITY, RELATION, AND TAIL ENTITY IN A TRIPLET ARE MARKED WITH RED, BLUE, AND GREEN, RESPECTIVELY. THE WRONG IDENTIFICATION PARTS ARE MARKED WITH ✗

Query Sentences	**Ground-Truth**	**MLMAN**	**TGIN$^{\diamond}$**	**TGIN**
The city was particularly favoured by the roman emperors Trajan, who restored and established its public buildings.	<Trajan, country of citizenship, roman>	<Trajan, owned by, ✗ roman>	<Trajan, country of citizenship, roman>	<Trajan, country of citizenship, roman>
To the west of the fortress are the yongding river and the marco emphpolo bridge.	<marco polo bridge, crosses, yongding river>	<marco polo bridge, crosses, yongding river>	<marco polo, ✗ crosses, yongding ✗>	<marco polo bridge, crosses, yongding river>
Kekuiapoiwa liliha married kīwalaō and their child was queen keōpūolani, consort of Kamehameha i and mother of two kings.	< kīwalaō, child, keōpūolani>	<Kekuiapoiwa liliha, child, kīwalaō ✗>	<Kekuiapoiwa, ✗ child, keopuolani ✗>	<Kekuiapoiwa liliha child, keōpūolani >

and use the dataset of FewRel in the 5-way 1-shot setting, as well as set batch size to 4 for all models. In fact, for all the models that use $\text{BERT}_{\text{BASE}}$ (or other kinds of pretrained language model) as their basic encoders, $\text{BERT}_{\text{BASE}}$ usually takes up the most of model parameters. Besides, the parameter number of TGIN is slightly larger than MatchNet, Proto, and HATT, which is mainly due to the use of a heterogeneous graph. However, compared with GNN that uses a graph too, we can see that TGIN has only 0.7% more parameters, yet the accuracy rate improves by 34.68% (17.79% → 23.96%). Compared to the general entity relation extraction models PFN and GRTE, TGIN has smaller parameters and achieves excellent results. Similarly, compared to the MLMAN, TGIN not only has smaller parameters but also increases the accuracy rate by 15.7% (20.39% → 23.96%). Moreover, we can also see that TGIN achieves a very competitive training speed. There are two main reasons for these observations. First, TGIN is not a fully connected graph, which reduces the complexity of the graph. Second, the feature update of TGIN mainly relies on translation algebraic operations, rather than a large number of learnable parameters, which also reduces the complexity of the model.

G. Case Studies

We select two comparison models for case studies.

1) *MLMAN:* Since the source code of the only existing few-shot relational triplet extraction model (MPE) has not been released, MLMAN with the best performance from other baselines is selected as one comparison model to observe the superiority of TGIN.
2) *TGIN$^{\diamond}$:* To illustrate the advantages of span tagging over CRF, we choose TGIN$^{\diamond}$ as the other comparison model.

Table X shows three typical qualitative results, which can be observed as follows. First, compared to TGIN, MLMAN is not better at distinguishing the relations between entity entities in complex clauses, e.g., in the first query sentence, the ground-truth relation is *country of citizenship*. TGIN can extract this relation correctly, but MLMAN extracts this error to *owned by*. This shows that the structure of TGIN has better relational inference capability. Second, TGIN$^{\diamond}$ cannot correctly identify the entity boundaries compared with TGIN, e.g., TGIN$^{\diamond}$ misidentifies the entity *marco polo bridge* as *marco polo* in the second query sentence. This indicates that using the span-based tagging method can improve the accuracy of entity extraction than CRF, especially when detecting boundaries for an entity composed of multiple words. Finally, although TGIN extracts the triplet ⟨*Kekuiapoiwa liliha, child, keōpūolani*⟩ different from the ground-truth ⟨*kīwalaō, child, keōpūolani*⟩, it is also the logically correct triplet for the third query sentence. Hence, TGIN can infer the correct triplets from the given sentences, but the completeness of extraction results still needs to be improved, which will be our focus in future research.

VI. Conclusion

In this article, we propose a novel joint model TGIN for the task of few-shot relational triplet extraction. The core of TGIN is a multilayer heterogeneous graph that can not only extract relational triplets jointly but also enables the triplet information of limited labeled data to interact better, thus maximizing the advantage of this information in a few-shot setting. After features propagate through layers, TGIN passes the label information from a few labeled examples to unlabeled examples and solves the problem of extracting relational triplets for long-tail relations. The edge loss and the node loss are used to jointly update the parameters of TGIN. Extensive experiments demonstrate that TGIN outperforms recent state-of-the-art few-shot algorithms by 2.34%∼10.74% in terms of accuracy for relational triplet extraction. In the future, we will continue to improve the completeness of extraction results in a few-shot setting.

Acknowledgment

The authors sincerely thank all anonymous reviewers for their valuable and constructive comments.

References

[1] K. A. Khatib, Y. Hou, H. Wachsmuth, C. Jochim, F. Bonin, and B. Stein, "End-to-end argumentation knowledge graph construction," in *Proc. AAAI Conf. Artif. Intell.*, 2020, pp. 7367–7374.

[2] Y. S. Chan and D. Roth, "Exploiting syntactico-semantic structures for relation extraction," in *Proc.49th Annu. Meeting Assoc. Comput. Linguistics, Hum. Lang. Technol.*, 2011, pp. 551–560.

[3] G. Bekoulis, J. Deleu, T. Demeester, and C. Develder, "Reconstructing the house from the ad: Structured prediction on real estate classifieds," in *Proc. 15th Conf. Eur. Chapter Assoc. Comput. Linguistics* vol. 2, 2017, pp. 274–279.

[4] K. Shaalan, "A survey of Arabic named entity recognition and classification," *Comput. Linguistics*, vol. 40, no. 2, pp. 469–510, Jun. 2014.

[5] S. Zhao, M. Hu, Z. Cai, H. Chen, and F. Liu, "Dynamic modeling cross- and self-lattice attention network for Chinese NER," in *Proc. AAAI*, 2021, pp. 14515–14523.

[6] Y. Lin, Z. Liu, and M. Sun, "Neural relation extraction with multi-lingual attention," in *Proc. 55th Annu. Meeting Assoc. Comput. Linguistics*, vol. 1, 2017, pp. 34–43.

[7] T.-J. Fu, P.-H. Li, and W.-Y. Ma, "GraphRel: Modeling text as relational graphs for joint entity and relation extraction," in *Proc. 57th Annu. Meeting Assoc. Comput. Linguistics*, 2019, pp. 1409–1418.

[8] Z. Tan, X. Zhao, W. Wang, and W. Xiao, "Jointly extracting multiple triplets with multilayer translation constraints," in *Proc. AAAI*, 2019, pp. 7080–7087.

[9] Z. Wei, J. Su, Y. Wang, Y. Tian, and Y. Chang, "A novel cascade binary tagging framework for relational triple extraction," in *Proc. 58th Annu. Meeting Assoc. Comput. Linguistics*, 2020, pp. 1476–1488.

[10] T. Zhao, Z. Yan, Y. Cao, and Z. Li, "A unified multi-task learning framework for joint extraction of entities and relations," in *Proc. AAAI*, 2021, pp. 14524–14531.

[11] X. Li, X. Luo, C. Dong, D. Yang, B. Luan, and Z. He, "TDEER: An efficient translating decoding schema for joint extraction of entities and relations," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2021, pp. 8055–8064.

[12] S. Riedel, L. Yao, and A. McCallum, "Modeling relations and their mentions without labeled text," in *Proc. Joint Eur. Conf. Mach. Learn. Knowl. Discovery Databases*, vol. 6323, 2010, pp. 148–163.

[13] Y. Hou et al., "Few-shot slot tagging with collapsed dependency transfer and label-enhanced task-adaptive projection network," in *Proc. 58th Annu. Meeting Assoc. Comput. Linguistics*, 2020, pp. 1381–1393.

[14] R. Wang et al., "Few-shot class-incremental learning for named entity recognition," in *Proc. 60th Annu. Meeting Assoc. Comput. Linguistics*, vol. 1, S. Muresan, P. Nakov, and A. Villavicencio, Eds. Stroudsburg, PA, USA: Association for Computational Linguistics, 2022, pp. 571–582.

[15] Z.-X. Ye and Z.-H. Ling, "Multi-level matching and aggregation network for few-shot relation classification," in *Proc. 57th Annu. Meeting Assoc. Comput. Linguistics*, 2019, pp. 2872–2881.

[16] T. Gao, X. Han, Z. Liu, and M. Sun, "Hybrid attention-based prototypical networks for noisy few-shot relation classification," in *Proc. AAAI*, 2019, pp. 6407–6414.

[17] M. Qu, T. Gao, L. A. C. Xhonneux, and J. Tang, "Few-shot relation extraction via Bayesian meta-learning on relation graphs," in *Proc. ICML*, vol. 119, 2020, pp. 7867–7876.

[18] J. Han, B. Cheng, and W. Lu, "Exploring task difficulty for few-shot relation extraction," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, M. Moens, X. Huang, L. Specia, and S. W. Yih, Eds. Association for Computational Linguistics, 2021, pp. 2605–2616.

[19] H. Yu, N. Zhang, S. Deng, H. Ye, W. Zhang, and H. Chen, "Bridging text and knowledge with multi-prototype embedding for few-shot relational triple extraction," in *Proc. 28th Int. Conf. Comput. Linguistics*, 2020, pp. 6399–6410.

[20] J. Snell, K. Swersky, and R. Zemel, "Prototypical networks for few-shot learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 4077–4087.

[21] M. Boudiaf, I. M. Ziko, J. Rony, J. Dolz, P. Piantanida, and I. B. Ayed, "Information maximization for few-shot learning," in *Proc. NIPS*, 2020, pp. 1–13.

[22] S. Xu and Y. Xiang, "Frog-GNN: Multi-perspective aggregation based graph neural network for few-shot text classification," *Exp. Syst. Appl.*, vol. 176, Aug. 2021, Art. no. 114795.

[23] J. Kim, T. Kim, S. Kim, and C. D. Yoo, "Edge-labeling graph neural network for few-shot learning," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 11–20.

[24] C. Zhang et al., "Few-shot learning on graphs," in *Proc. 31st Int. Joint Conf. Artif. Intell.*, Jul. 2022, pp. 5662–5669.

[25] A. Bordes, N. Usunier, A. García-Durán, J. Weston, and O. Yakhnenko, "Translating embeddings for modeling multi-relational data," in *Proc. NIPS*, 2013, pp. 2787–2795.

[26] T. Ebisu and R. Ichise, "Generalized translation-based embedding of knowledge graph," *IEEE Trans. Knowl. Data Eng.*, vol. 32, no. 5, pp. 941–951, May 2020.

[27] H. Huang, G. Long, T. Shen, J. Jiang, and C. Zhang, "RatE: Relation-adaptive translating embedding for knowledge graph completion," in *Proc. 28th Int. Conf. Comput. Linguistics*, 2020, pp. 556–567.

[28] X. Zeng, D. Zeng, S. He, K. Liu, and J. Zhao, "Extracting relational facts by an end-to-end neural model with copy mechanism," in *Proc. 56th Annu. Meeting Assoc. Comput. Linguistics*, vol. 1, 2018, pp. 506–514.

[29] Z. Guo, Y. Zhang, and W. Lu, "Attention guided graph convolutional networks for relation extraction," in *Proc. 57th Annu. Meeting Assoc. Comput. Linguistics*, 2019, pp. 241–251.

[30] W. Xiong, M. Yu, S. Chang, X. Guo, and W. Y. Wang, "One-shot relational learning for knowledge graphs," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2018, pp. 1980–1990.

[31] B. Kratzwald, S. Feuerriegel, and H. Sun, "Learning a cost-effective annotation policy for question answering," in *Proc. Conf. Empirical Methods Natural Lang. Process. (EMNLP)*, 2020, pp. 3051–3062.

[32] Y. Wang, Q. Yao, J. T. Kwok, and L. M. Ni, "Generalizing from a few examples: A survey on few-shot learning," *ACM Comput. Surv.*, vol. 53, no. 3, p. 63, Jun. 2020.

[33] N. Mishra, M. Rohaninejad, X. Chen, and P. Abbeel, "A simple neural attentive meta-learner," in *Proc. ICLR*, 2018, pp. 1–17.

[34] O. Vinyals, C. Blundell, T. Lillicrap, K. Kavukcuoglu, and D. Wierstra, "Matching networks for one shot learning," in *Proc. NIPS*, 2016, pp. 3630–3638.

[35] A. Fritzler, V. Logacheva, and M. Kretov, "Few-shot classification in named entity recognition task," in *Proc. 34th ACM/SIGAPP Symp. Appl. Comput.*, Apr. 2019, pp. 993–1000.

[36] J. D. Lafferty, A. McCallum, and F. C. N. Pereira, "Conditional random fields: Probabilistic models for segmenting and labeling sequence data," in *Proc. ICML*, 2001, pp. 282–289.

[37] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and S. Y. Philip, "A comprehensive survey on graph neural networks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 1, pp. 4–24, Mar. 2021.

[38] H. Hong, H. Guo, Y. Lin, X. Yang, Z. Li, and J. Ye, "An attention-based graph neural network for heterogeneous structural learning," in *Proc. AAAI*, 2020, pp. 4132–4139.

[39] M. Tu, G. Wang, J. Huang, Y. Tang, X. He, and B. Zhou, "Multi-hop reading comprehension across multiple documents by reasoning over heterogeneous graphs," in *Proc. 57th Annu. Meeting Assoc. Comput. Linguistics*, 2019, pp. 2704–2713.

[40] H. Linmei, T. Yang, C. Shi, H. Ji, and X. Li, "Heterogeneous graph attention networks for semi-supervised short text classification," in *Proc. Conf. Empirical Methods Natural Lang. Process. 9th Int. Joint Conf. Natural Lang. Process. (EMNLP-IJCNLP)*, 2019, pp. 4820–4829.
[41] D. Wang, P. Liu, Y. Zheng, X. Qiu, and X. Huang, "Heterogeneous graph neural networks for extractive document summarization," in *Proc. 58th Annu. Meeting Assoc. Comput. Linguistics*, 2020, pp. 6209–6219.
[42] S. Zeng, R. Xu, B. Chang, and L. Li, "Double graph based reasoning for document-level relation extraction," in *Proc. Conf. Empirical Methods Natural Lang. Process. (EMNLP)*, 2020, pp. 1630–1640.
[43] J. Devlin, M. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proc. NAACL-HLT*, 2019, pp. 4171–4186.
[44] L. Baldini Soares, N. FitzGerald, J. Ling, and T. Kwiatkowski, "Matching the blanks: Distributional similarity for relation learning," in *Proc. 57th Annu. Meeting Assoc. Comput. Linguistics*, 2019, pp. 2895–2905.
[45] V. G. Satorras and J. B. Estrach, "Few-shot learning with graph neural networks," in *Proc. ICLR*, 2018, pp. 1–13.
[46] Z. Yan, C. Zhang, J. Fu, Q. Zhang, and Z. Wei, "A partition filter network for joint entity and relation extraction," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2021, pp. 185–197.
[47] F. Ren et al., "A novel global feature-oriented relational triple extraction model based on table filling," in *Proc. EMNLP*, 2021, pp. 2646–2656.
[48] X. Han et al., "Fewrel: A large-scale supervised few-shot relation classification dataset with state-of-the-art evaluation," in *Proc. EMNLP*, 2018, pp. 4803–4809.
[49] Y. Zhang, V. Zhong, D. Chen, G. Angeli, and C. D. Manning, "Position-aware attention and supervised data improve slot filling," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2017, pp. 35–45.
[50] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. ICLR*, 2015, pp. 1–15.

Jiaxin Wang received the B.S. and M.S. degrees in communication and information systems from Northwestern Polytechnical University, Xi'an, Shaanxi, China, in 2017 and 2020, respectively. She is currently pursuing the Ph.D. degree with the School of Computer Science and Technology, Xi'an Jiaotong University, Xi'an.

Her research interests include natural language processing, knowledge graphs, and few-shot learning.

Lingling Zhang received the Ph.D. degree in computing science from Xi'an Jiaotong University, Xi'an, China, in 2020.

She was a Visiting Student with the School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, USA, working with Prof. A. Hauptmann. She is currently an Assistant Professor of computer science with Xi'an Jiaotong University. Her research interests include cross-media information mining, computer vision, zero-shot learning, and few-shot learning.

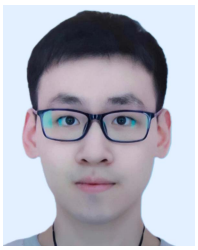
Jun Liu (Senior Member, IEEE) received the B.S. degree in computer science and technology and the Ph.D. degree in systems engineering from Xi'an Jiaotong University, Xi'an, Shaanxi, China, in 1995 and 2004, respectively.

He is currently a Professor with the Department of Computer Science and Technology, Xi'an Jiaotong University. He has authored hundreds of research articles in various journals and conference proceedings. His research interests include natural language processing and e-learning.

Dr. Liu has won the Best Paper Awards at the IEEE International Symposium on Software Reliability Engineering (ISSRE) 2016, the IEEE International Conference on Big Knowledge (ICBK) 2018, and the National Software Application Conference (NASAC) 2018. He has served as a Guest Editor for many technical journals, such as *Information Fusion* and the IEEE SYSTEMS JOURNAL. He also acted as the conference/workshop/track chair at numerous conferences.

Kunming Ma received the B.S. degree from Xi'an Jiaotong University, Xi'an, Shaanxi, China, in 2020, where he is currently pursuing the M.S. degree with the School of Computer Science and Technology.

His research interests include information extraction and knowledge graphs.



Wenjun Wu received the B.S. degree from Xi'an Jiaotong University, Xi'an, Shaanxi, China, in 2020, where he is currently pursuing the Ph.D. degree in computer science.

His research interests include cross-modal information mining, diagram understanding, and few-shot learning.

Xiang Zhao received the Ph.D. degree from The University of New South Wales, Sydney, NSW, Australia, in 2014.

He is currently an Associate Professor with the National University of Defense Technology, Changsha, China, where he is the Head of the Knowledge Systems Engineering Group. His research interests include graph data management and mining, with a special focus on knowledge graphs.

Yaqiang Wu is currently a Research Director with Lenovo Research, Beijing, China, and oversees multiple research areas, including cloud computing, soft engineering, and AI-empowered applications. His research has focused on computer vision, multimedia, and machine learning.

Dr. Wu and his team won several competitions in text detection and recognition at the International Conference on Document Analysis and Recognition (ICDAR) and the International Conference on Pattern Recognition (ICPR) in recent years.

Yi Huang is currently a Senior Researcher with the AI and Intelligent Operation Center, China Mobile Research Institute, Beijing, China. He has over 20 professional publications and 30 patents. His research interests are dialog systems and knowledge engineering.

Dr. Huang is a Frequent Reviewer and an Organizer for major natural language international conferences and journals, such as the Annual Meeting of the Association for Computational Linguistics (ACL) and the International Joint Conference on Artificial Intelligence (IJCAI).