Context-aware Entity Typing in Knowledge Graphs

Weiran Pan[1], **Wei Wei**[1*] and **Xian-Ling Mao**[2]
[1]Cognitive Computing and Intelligent Information Processing Laboratory, School of Computer Science and Technology, Huazhong University of Science and Technology
[2]Department of Computer Science and Technology, Beijing Institute of Technology
panwr789@gmail.com, weiw@hust.edu.cn, maoxl@bit.edu.cn

Abstract

Knowledge graph entity typing aims to infer entities' missing types in knowledge graphs which is an important but under-explored issue. This paper proposes a novel method for this task by utilizing entities' contextual information. Specifically, we design two inference mechanisms: i) N2T: independently use each neighbor of an entity to infer its type; ii) Agg2T: aggregate the neighbors of an entity to infer its type. Those mechanisms will produce multiple inference results, and an exponentially weighted pooling method is used to generate the final inference result. Furthermore, we propose a novel loss function to alleviate the false-negative problem during training. Experiments on two real-world KGs demonstrate the effectiveness of our method. The source code and data of this paper can be obtained from https://github.com/CCIIPLab/CET.

1 Introduction

Knowledge graphs (KGs) store world knowledge in a structured way. They consist of collections of triples in the form of (head entity, relation, tail entity), and entities are labeled with types (see Figure 1). The entity type information on knowledge graph has applications in many NLP tasks including entity linking (Gupta et al., 2017), question answering (Bordes et al., 2014) and fine-grained entity typing in text (Ling and Weld, 2012, Choi et al., 2018, Zhou et al., 2018). An entity can have multiple types, and the entity type information on the knowledge graph is usually incomplete. In this paper, we focus on *Knowledge Graph Entity Typing* (KGET), which aims to infer entities' missing types in knowledge graphs.

Existing methods for the KGET task can be divided into embedding-based methods and graph convolutional networks (GCNs) for the multi-relational graph. Knowledge graph embedding (KGE) is representative of embedding-based methods. Treating entities' known types as special triples with a unique relation "has type", *e.g.*, (Barack Obama, has type, person), the KGET task can be understood as a subtask of knowledge graph completion. Consequently, KGE methods can infer entities' missing types by completing (entity, has type, ?). Recently, two embedding-based KGET models based on KGE have been proposed: ETE (Moon et al., 2017b) and ConnectE (Zhao et al., 2020). They first obtain entity embeddings from KGE methods, then use them to infer entities' missing types. GCNs for the multi-relational graph can aggregate the rich information in entities' neighbors to infer entities' missing types.

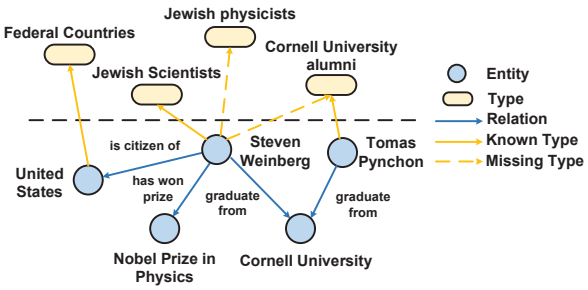

Figure 1: A knowledge graph fragment. Some types of entity *Steven Weinberg* are missing.

Existing methods usually encode all attributes of an entity into one embedding, then use this representation to conduct inference. However, when judging whether an entity has a particular type, only some attributes of this entity may be helpful while the others remain useless. For example, in Figure 1, only the neighbor (graduate from, Cornell University) can indicate the central entity *Steven Weinberg* have type *Cornell University alumni*. We argue that always considering all attributes of an entity during inference may introduce irrelevant information as noise and ultimately reduce the accuracy of entity typing.

Besides the above-mentioned shortcoming, ETE

*Corresponding author: Wei Wei.

Findings of the Association for Computational Linguistics: EMNLP 2021, pages 2240–2250
November 7–11, 2021.

and ConnectE also ignore entities' known type information when training entity embeddings, which is important for entity typing. For instance, in Figure 1 there are no known triples which can indicate entity *Steven Weinberg* has type *Jewish physicists*. In this case, the model needs to utilize the known type information. *Steven Weinberg* has type *Jewish Scientists*, and in the known triples exists (Steven Weinberg, has won prize, Nobel Prize in Physics). Combining the two, we can infer *Steven Weinberg* has type *Jewish physicists*. In short, these two KGET models have difficult using the entities' known types to infer the missing ones. In the experiment, we found this seriously affect their performance.

To overcome those shortcomings in existing methods, we propose a novel method for the KGET task, called CET (Context-aware Entity Typing). Specifically, CET contains two inference mechanisms: i) N2T: independently use each neighbor of an entity to infer its type; ii) Agg2T: aggregate the neighbors of an entity to infer its type. According to our observation, one neighbor usually represents a specific attribute of the central entity. Thus, the N2T mechanism allows CET to consider each attribute of an entity during inference individually. In contrast, previous works mix various attributes of an entity into one embedding for inference. Therefore, we believe CET can produce more accurate entity typing results than existing methods. Moreover, some complex types like *21st-century American novelists* involve multiple semantic aspects of an entity. It's difficult to infer those types only using a single neighbor. Therefore, we further propose the Agg2T mechanism, which simultaneously considers multiple attributes of the central entity during inference by aggregating neighbors. We also treat the known types of the central entity as its neighbors to use them to infer the missing types. To aggregate the inference results generated by N2T and Agg2T mechanism, we adopt a carefully designed pooling method similar to softpool (Stergiou et al., 2021). Experiments show that this pooling method can produce stable and interpretable inference results.

In addition, we face serious false-negative problem during training. Some (entity, type) pairs are valid but happen to be missing in current knowledge graphs. Treating them as negative samples will seriously affect model performance. We propose a novel loss function to alleviate this. To sum up, our contributions are three-fold:

- We propose CET, a novel and flexible method for inferring entities' missing types in knowledge graphs, which fully utilize the neighbor information in an independent-based mechanism and aggregated-based mechanism.
- We design a novel loss function to alleviate the false-negative problem during training.
- Experiments on two real-world knowledge graphs demonstrate the superiority of our proposed method over other state-of-the-art algorithms, and the inference process of our method is interpretable.

2 Related Work

Embedding-based methods. Moon et al. (2017b) propose to learn type embedding for knowledge graph entity typing and build two methodologies: i) Synchronous training: Adding entities' known types to knowledge graphs in the form of triples with a unique relation "has type", *e.g.,* (Barack Obama, has type, person), Knowledge Graph Embedding (KGE) methods (Nickel et al., 2011, Bordes et al., 2013, Nickel et al., 2016) can learn the embeddings of entities and types simultaneously. KGE methods can infer entities' missing types by completing (entity, has type, ?). ii) Asynchronous training: The model first obtains entities' embeddings from KGE methods, then minimizes the L1 distance between the entities' and their corresponding types' embeddings while keeping the entities' embeddings fixed. During inference, the smaller L1 distance between an entity and a specific type means the entity is more likely to have this type. Moon et al. (2017b) observe that there will be only one type of relation associate with types in synchronous training. They claim this lack of diversity of relations means that synchronous training methods have difficulty solving the KGET task. So they proposed a model called ETE, which follows the asynchronous training strategy and uses CONTE (Moon et al., 2017a) to obtain entity embeddings.

Zhao et al. (2020) propose ConnectE, a more advanced KGET model which contains two inference mechanisms. One is called E2T, which uses a linear transformation to project the entities' embeddings into type embedding space. Another is called TRT, which uses the neighbors' types to infer the central entities' missing types. TRT is based

on the assumption that the relationship can remain unchanged when replacing the entities in the triple to their corresponding types. For instance, if triple (Barack Obama, born in, Honolulu) holds, a new triple (person, born in, location) should also hold. ConnectE also follows the asynchronous training strategy, which first uses TransE to obtain entities' embedding then fixes them to train E2T and TRT.

ETE and ConnectE do not consider entities' known types when training entities' embeddings, which means they do not encode the known type information into entities' embeddings. Therefore, both of them have difficulty using entities' known types to infer the missing ones, which seriously affects their performance. Our experiments support this claim.

GCNs for Multi-Relational Graph. The TRT mechanism in ConnectE attempts to use entities' neighbors to infer entities' missing types. However, TRT only utilizes the neighbors' types. To fully utilize the information in entities' neighbors, GCNs for multi-relational graphs can be used to encode entities' neighbors. Schlichtkrull et al. (2018) proposed R-GCN, an extension of GCNs for relational graphs. R-GCN aggregate the information in neighbors using the relation-specific filter. Weighted Graph Convolutional Network (Shang et al., 2019) utilizes learnable relational specific scalar weights to aggregate neighbors. Vashishth et al. (2020) proposed a more generalized framework by leverage composition operators from KGE techniques during GCN aggregation. In the KGET task, the entities' missing types can be inferred by performing multi-label classification on entities' embeddings obtained by GCNs.

Existing methods usually encode all attributes of an entity into one embedding during inference. We argue this will introduce noise as sometimes only part of attributes of an entity is helpful for the KGET task while the others may be useless. To overcome this shortcoming, we propose the N2T mechanism. By independently uses each neighbor of an entity to infer its missing types, the N2T mechanism allows our model to consider each attribute of an entity during inference individually. This can reduce the impact of irrelevant information on entity typing. Also, we treat entities' known types as neighbors which means our model can use them to infer entities' missing types.

Others. Note that embedding knowledge graphs containing concepts (ontologies) and modeling the relationship between concepts (Lv et al., 2018, Hao et al., 2019) are not the goal of this paper. We concentrate on inferring entities' missing types. Some other works on KGET (Neelakantan and Chang, 2015, Jin et al., 2018) mainly focus on how to combine additional information, such as the text description of the entities, to infer the missing types. Our work only uses the information on the knowledge graphs to infer the missing types of entities, which is more universal.

3 Method

In this section, we introduce our proposed method in detail. We first introduce the notations used in this paper. Afterward, we introduce two inference mechanisms used in our method. Finally, we introduce a novel loss function that can alleviate the false negative problem during training.

3.1 Notations

Let $\mathcal{G} = \{(s, r, o)\} \subseteq \mathcal{E} \times \mathcal{R} \times \mathcal{E}$ be a knowledge graph where $\mathcal{E}$ and $\mathcal{R}$ are the entity set and the relation set, respectively. The known type information on the knowledge graph is represented as $\mathcal{I} = \{(e, t)\} \subseteq \mathcal{E} \times \mathcal{T}$. Let L be the number of types. We number the type from 1 to L and use type i to refer to the i-th type.

We add the known type information to the knowledge graphs. If an entity e has type t, we add an edge *(e, has type, t)* to KG, where *has type* is a newly added relationship. For the convenience of discussion, if edge (s, r, o) exists in KG, we add its inverted edge (o, r^{-1}, s) to KG where r^{-1} is the reverse relation of r. Let $\mathcal{G}'$ be the KG after adding known type information and inverted edges. After adding those inverted edges, we only consider outgoing edges when discussing entities' neighbors. The neighbors set of u can be represented as $\mathcal{N}(u) = \{(n_r, n_e)|(u, n_r, n_e) \in \mathcal{G}'\}$. We use bold $\mathbf{n}_r, \mathbf{n}_e$ to represent the embedding of neighbor relation and neighbor entity, respectively. Let k be the dimension of the embeddings. The neighbors mentioned later all refer to those neighbors on $\mathcal{G}'$ which include the neighbors in the knowledge graph and the entities' know types.

3.2 Proposed Method

Our proposed method contains two inference mechanisms. One is to use each neighbor to infer the central entity's type independently, called N2T. Another is to aggregate neighbor information then

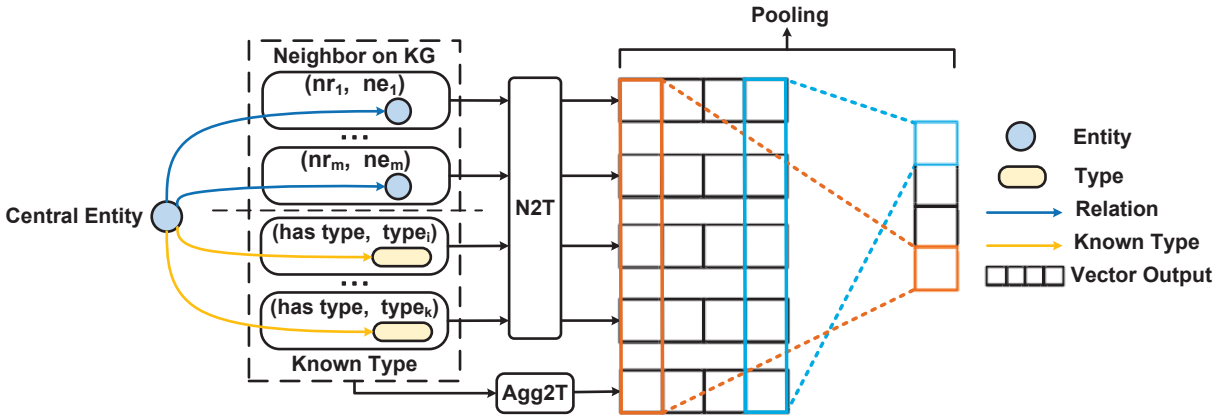

Figure 2: The overall architecture of CET. The N2T mechanism independently uses each neighbor to infer entities' missing types. The Agg2T mechanism aggregates neighbors' information then conducts inference. The final inference result is generated by an exponentially weighted pooling method.

conduct inference, called Agg2T. And we use an exponentially weighted pooling method to generate the final inference result. The overall architecture is shown in Figure 2.

N2T mechanism. We observe a strong correlation between the neighbors and the central entity's type. For instance, the neighbor (is affiliated to, Los Angeles Lakers) can indicate the central entity has type *Los Angeles Lakers player*. Meanwhile, different neighbors may correspond to different types. Therefore, we propose the N2T mechanism that independently uses each neighbor to infer the missing types of central entities. It's worth noting that when judging whether an entity has a particular type, sometimes only a few neighbors are helpful while the others remain useless. The N2T mechanism focuses on a single neighbor during inference, reducing the interference of irrelevant information on entity typing. In practice, CET follows the translating assumption in TransE to obtain the neighbor embedding[1], then conducts non-linear activation[2] on neighbor embedding and sent it to a linear layer:

$$\mathbf{R}^{N2T}_{(n_r,n_e)} = \mathbf{W}\mathrm{Relu}(\mathbf{n}_e - \mathbf{n}_r) + \mathbf{b}, \quad (1)$$

where $\mathbf{W} \in \mathbb{R}^{L\times k}, \mathbf{b} \in \mathbb{R}^L$ are the learning parameters and $\mathbf{R}^{N2T}_{(n_r,n_e)} \in \mathbb{R}^L$ is the relevance score calculated by the N2T mechanism, where the i-th entry represents the relevance score between neighbor (n_e, n_r) and type i. The higher $R^{N2T}_{(n_r,n_e),i}$ means the neighbor (n_e, n_r) is more relevant to type i, which indicates the central entity is more likely to have type i.

Agg2T mechanism. It's difficult to infer some complex types like *21st-century American novelists* and *Film directors from New York City* from a single neighbor. Therefore, we further propose the Agg2T mechanism which aggregate entities' neighbors to infer entities' missing types:

$$\mathbf{h}_u = \frac{1}{|\mathcal{N}(u)|}\sum_{(n_r,n_e)\in\mathcal{N}(u)}(\mathbf{n}_e - \mathbf{n}_r), \quad (2)$$

$$\mathbf{R}^{Agg2T}_u = \mathbf{W}\mathrm{Relu}(\mathbf{h}_u) + \mathbf{b}, \quad (3)$$

where $\mathbf{h}_u \in \mathbb{R}^k$ is the aggregated representation of u's neighbors and $\mathbf{R}^{Agg2T}_u \in \mathbb{R}^L$ stores the relevance scores with all types. Here we chose a simple non-parameterized mean aggregation operation to verify the effectiveness of our method. Actually, CET is a highly flexible method that can work with the existing GCN-based method by replacing the aggregation operation in the Agg2T mechanism. We leave those analyses as future work.

Pooling approach. The N2T mechanism and the Agg2T mechanism will generate multiple entity typing results for every entity. To generate the final entity typing result, a pooling method is needed. Mean-pooling is not recommended as some types can only be indicated by a few neighbors. Max-pooling seems to be a suitable choice. However, we find its performance is not ideal. Choosing the max value as the final result makes only a small part of the input get the gradient which means some embeddings may not be sufficiently trained. As a result, the model may fail to represent every attribute of an entity accurately. In practice, we adopt

[1]The original relation r and its reversed relation r^{-1} share the same set of parameters and their embeddings satisfy $\mathbf{r} = -\mathbf{r}^{-1}$.

[2]Non-linear activation is not necessary, but we found that adding it can achieve better results.

an exponentially weighted pooling method similar to softpool (Stergiou et al., 2021):

$$R_{u,i} = \text{pool}(\{R^{Agg2T}_{u,i}, R^{N2T}_{(n_r,n_e),i} \mid \forall (n_e, n_r) \in \mathcal{N}(u)\}), for\ i \in 1, 2, \ldots, L, \quad (4)$$

$$\text{pool}(\{x_1, x_2, ..., x_n\}) = \sum_{i=1}^{n} w_i x_i, \quad (5)$$

$$w_i = \frac{\exp \alpha x_i}{\sum_{k=1}^{n} \exp \alpha x_k}, \quad (6)$$

where $R_{u,i} \in \mathbb{R}$ is the relevance score between entity u and type i. $\alpha \in \mathbb{R}^+$ is a hyperparameter that controls the temperature of the pooling process. The higher $R_{u,i}$ means entity u is more likely to have type i. This pooling method has a similar effect to max-pooling but can generate a gradient for every input which ensures every embedding gets sufficient training.

Neighbor sampling. If we use all the neighbors during training, the model may learn to use available type information to infer themselves, *e.g.,* using neighbor (has type, person) to infer the entity has type *person*. The model can perfectly fit the training set in this way, result in a severe overfitting problem. One solution is to perform the following mask operation before the equation (6):

$$\begin{cases} R^{N2T}_{(has\ type,i),i} = -\infty, & i \in 1, 2, \ldots, L \\ R^{Agg2T}_{u,i} = -\infty, & if\ (u,i) \in \mathcal{I} \end{cases} \quad (7)$$

Another solution is to perform neighbor sampling: dynamically sample entities' neighbors with replacement during training. We find both methods have similar performance while performing neighbor sampling can significantly save training time, so we finally settle with neighbor sampling. Sampling fewer neighbors can lead to faster training speed, but at the expense of performance degradation. In practice, we find that sampling ten neighbors can usually achieve a good balance between speed and performance. We only conduct neighbor sampling during training; all neighbors of the entity are used during inference.

3.3 Optimization

We hope that $R_{u,i}$ as high as possible if entity u has type i (positive samples), while $R_{u,i}$ as low as possible if entity u does not has type i (negative samples). The known (entity, type) pairs in $\mathcal{I}$ can be used as positive samples. To gather the negative samples, a simple choice is to treat all the nonexistent (entity, type) pairs in $\mathcal{I}$ as negative samples. Then we can use the binary cross-entropy (BCE) as loss function:

$$p_{u,i} = \sigma(R_{u,i}), \quad (8)$$

$$\mathcal{L} = -\sum_{(u,i)\in\mathcal{I}} \log p_{u,i} - \sum_{(u,i)\notin\mathcal{I}} \log(1 - p_{u,i}). \quad (9)$$

However, some (entity, type) pairs are valid but happen to be missing in current knowledge graphs. Actually, the entities' missing types which we want to infer belongs to this category. This brings serious false negative problems during training. To overcome this, we propose the following false-negative aware (FNA) loss function:

$$\mathcal{L} = -\sum_{(u,i)\notin\mathcal{I}} \beta(p_{u,i} - p^2_{u,i}) \log(1 - p_{u,i}) - \sum_{(u,i)\in\mathcal{I}} \log p_{u,i}, \quad (10)$$

where β is a hyper-parameter used to control the overall weight of negative samples. The FNA loss function will assign lower weight to those negative samples with too large or too small relevance scores. Those negative samples with too large relevance scores are possibly false negative samples, and those with too small relevance scores are easy ones. These two kinds of negative samples do not provide helpful information, so we give them a lower weight.

Dataset	FB15kET	YAGO43kET
# Entities	14951	42334
# Relations	1345	37
# Types	3584	45182
# Train. triples	483142	331686
# Train. tuples	136618	375853
# Valid	15848	43111
# Test	15847	43119

Table 1: Statistics of used datasets.

4 Experiment

In this section, we evaluate and analyze the proposed method on two real-world KGs. We introduce datasets and experiment settings in Section 4.1, present the main result in Section 4.2. The ablation study can be found in Section 4.3. Section 4.4 provides some cases to further analyze our method.

4.1 Datasets and Experiment Setup

Datasets. We conduct experiments on two real-world knowledge graphs, *i.e.,* FB15k (Bordes et al., 2013), YAGO43k (Moon et al., 2017b) which are subsets of Freebase (Bollacker et al., 2008) and YAGO (Suchanek et al., 2007), respectively. Moon et al. collected entities' types in both datasets and added them into the original datasets in the form of *(entity, entity type)*. The datasets after adding those entity-type tuples are called FB15kET and YAGO43kET. Their training sets consist of original triples in FB15k and YAGO43k with some entity-type tuples, and the other entity-type tuples are served as validation and test sets. The statistics of the datasets are shown in Table 1 [3].

Hyper-parameter Settings. We perform stochastic minibatch training and use Adam (Kingma and Ba, 2015) as the optimizer. The hyper-parameters are tuned according to the MRR on the validation set. The search space for the grid search are set as follows: embedding dim $k \in \{50, 100\}$, pooling temperature $\alpha \in \{0.5, 1.0\}$, negative samples weight $\beta \in \{1.0, 2.0, 4.0\}$ and learning rate $lr \in \{0.001, 0.005, 0.01\}$. We also tried to adjust the batch size but this had no impact so we fixed the batch size to 128. The embeddings of entities, relations, and types are uniformly initialized, using a uniform distribution:$[-10/k, 10/k]$ (k is the dimension of embeddings). The best model was selected by early stopping using the MRR on validation sets, computed every 25 epochs with a maximum of 1000 epochs. The optima configurations are: $\{k = 100, \alpha = 0.5, \beta = 4.0, lr = 0.001\}$ on FB15kET; $\{k = 100, \alpha = 0.5, \beta = 2.0, lr = 0.001\}$ on YAGO43kET.

Evaluation Protocol. For each test sample (*e*, *t*) in test set. We first calculate the relevance score between *e* and every type and then rank all the types in descending order of relevance score. Following (Bordes et al., 2013), we evaluate model performance in the filtered setting: All the known types of *e* in the training, validation, and test sets are removed from the ranking. Finally, we can obtain the exact rank of the correct type *t* in all types. We use Mean Rank (MR), Mean Reciprocal Rank (MRR), and Hits at 1/3/10 as evaluation metrics.

Baselines. We compare our model with six state-of-the-art models, which can be divided into three groups. Models in the first group are KGE models which treat the KGET task as a special sub-task of knowledge graph completion, including TarnsE (Bordes et al., 2013), ComplEx (Trouillon et al., 2016) and RotatE (Sun et al., 2019). The second group are recently proposed KGET models including ETE (Moon et al., 2017b) and ConnectE (Zhao et al., 2020). And for GCNs for multi-relational graph we choose R-GCN (Schlichtkrull et al., 2018) as baseline. To make a fair comparison, R-GCN has similar experiment settings with CET: treating entities' known types as neighbors and performing the neighbor sampling during training. Hyper-parameter settings of those baselines can be found in Appendix A.

Implementation. All the KGE baselines in this paper are implemented using DGL-KE (Zheng et al., 2020). Our model and R-GCN are implemented using the DGL framework (PyTorch as backends). All the experiments were run on a single 1080Ti system with 32GB RAM.

4.2 Main Results

Table 2 summarizes our result on FB15kET and YAGO43kET. We implement TransE, ComplEx, and RotatE using the self-adversarial negative sampling (Sun et al., 2019) and L3 regularization (Lacroix et al., 2018), which leads to better results than the reported results in the previous paper. We can see our model outperforms all baselines on almost all metrics. Meanwhile, after using the FNA loss, the performance significantly improved. Not only our model, but R-GCN can also benefit from this, which further proved the effectiveness of FNA loss.

The performance of TransE, ComplEx, and RotatE is limited by their entity representation strategy. These KGE methods encode all attributes of an entity into one embedding for inference. However, when judging whether an entity has a particular type, the irrelevant attributes may interfere with the inference result. CET overcomes this shortcoming by using the N2T mechanism and achieves better performance.

Similar to the KGE methods, R-GCN also suffers from the noise introduced by irrelevant information. R-GCN aggregates entities' neighbors to infer entities' missing types. However, sometimes a type can only be indicated by a few neighbors. This kind of rare information is easily overwhelmed by other irrelevant information during

[3] we exclude the data which contains unseen types in the training set from validation set and test set.

Model	FB15kET					YAGO43kET				
	MRR	**MR**	**Hit@1**	**Hit@3**	**Hit@10**	**MRR**	**MR**	**Hit@1**	**Hit@3**	**Hit@10**
TransE	0.618	**18**	0.504	0.686	0.835	0.427	393	0.304	0.497	0.663
ComplEx	0.595	20	0.463	0.680	0.841	0.435	631	0.316	0.504	0.658
RotatE	0.632	**18**	0.523	0.699	0.840	0.462	316	0.339	0.537	0.695
ETE*	0.500	-	0.385	0.553	0.719	0.230	-	0.137	0.263	0.422
ConnectE*	0.590	-	0.496	0.643	0.799	0.280	-	0.160	0.309	0.479
R-GCN (BCE)	0.662	19	0.571	0.711	0.836	0.357	366	0.266	0.392	0.533
R-GCN (FNA)	0.679	20	0.597	0.722	0.843	0.372	397	0.281	0.409	0.549
CET (BCE)	0.682	19	0.593	0.733	0.852	0.472	**239**	0.362	0.540	0.669
CET (FNA)	**0.697**	19	**0.613**	**0.745**	**0.856**	**0.503**	250	**0.398**	**0.567**	**0.696**

Table 2: Results of several models on FB15kET and YAGO43kET datasets. Best results are in **bold**. [*]: Results are taken from original papers. ConnectE has three different training settings, here we report the best one.

Model				FB15kET					YAGO43kET				
N2T	**TAN**	**Agg2T**	**FNA**	**MRR**	**MR**	**Hit@1**	**Hit@3**	**Hit@10**	**MRR**	**MR**	**Hit@1**	**Hit@3**	**Hit@10**
✓	✓	✓	✓	**0.697**	**19**	**0.613**	**0.745**	**0.856**	**0.503**	250	**0.398**	**0.567**	**0.696**
✓	✓	✓		0.682	**19**	0.593	0.733	0.852	0.472	**239**	0.362	0.540	0.669
✓	✓			0.679	**19**	0.591	0.730	0.850	0.460	272	0.348	0.528	0.664
✓				0.663	21	0.575	0.710	0.836	0.431	505	0.331	0.491	0.615

Table 3: Results of ablation study. Models without FNA loss function use BCE loss function instead.

R-GCN's aggregation process. This phenomenon is rarely observed on FB15kET but is common on YAGO43kET. This can explain why R-GCN outperforms other baselines on FB15kET but has a poor performance on YAGO43kET.

ETE and ConnectE are largely left behind, especially on YAGO43kET. This is because these two methods have difficulty using entities' known types to infer the missing ones. Compared with FB15k, YAGO43k has a sparser graph structure and fewer types of relations (see Table 1). Therefore, in YAGO43kET, ignoring entities' known types and only using the (entity, relation, entity) triples to train entity embeddings can hardly fully model various attributes of each entity. As a result, the performance gap between ETE/ConnectE and other methods is more pronounced on YAGO43kET. This result demonstrates that using entities' known types to infer the missing ones is crucial in the KGET task. We will further illustrate this in Section 4.3.

4.3 Ablation Study

Our model includes two inference mechanisms: N2T and Agg2T. Treating entities' known types as neighbors (TAN, short for types as neighbors) and the false negative aware loss function (FNA) can also improve the performance. To understand each component's effect on the model, we conduct the ablation study on FB15kET and YAGO43kET datasets. The result is reported in Table 3. We can see the full model (the first row) outperforms all the ablated models on almost all metrics, illustrating every component's effectiveness in our model.

Model	**MRR**	**MR**	**Hit@1**	**Hit@3**	**Hit@10**
mean	0.396	338	0.300	0.440	0.578
max	0.462	327	0.366	0.517	0.636
ewp	**0.503**	**250**	**0.398**	**0.567**	**0.696**

Table 4: Comparison of different pooling methods.

Impact of N2T. Only using the N2T mechanism, our model still achieves competitive results against other state-of-the-art baselines. This indicates that independently considering entities' different attributes during inference can reduce the noise and produce accurate entity typing results.

Impact of TAN. Treating entities' known types as neighbors allows CET utilize entities' known type to infer the missing ones. This strategy is especially effective on datasets containing rich entity-type information such as YAGO43kET.

Impact of Agg2T. Agg2T mechanism is designed

Inference		Top 3 Relevant Information Source	
Entity	**Type**	**Information Source**	**Relevance Score**
Bob Dylan	Pulitzer Prize winners	(has won prize, Pulitzer Prize)	6.93
		(has won prize, Quill Award)	-0.44
		(has type, American poets)	-0.72
Ian Fleming	English writers	(has type, English short story writers)	3.36
		(has type, English novelists)	2.96
		(has type, English spy fiction writers)	2.15
Ben Gazzara	American male television actors	Aggregation	3.04
		(has type, American male film actors)	-0.53
		(has type, American television actors)	-0.61

Table 5: Representative entity typing examples. We present the top 3 relevant information sources for entity typing and their relevance scores. *Aggregation* stands for the aggregation of neighbors, the information source used in the Agg2T mechanism.

to infer those complex types. Types like *21st-century American novelists* that involve multiple attributes of entities and require joint inference by multiple neighbors almost only appear in YAGO43kET. So it is natural that the Agg2T mechanism has little effect on FB15kET but improves model performance on YAGO43kET.

Impact of FNA. The false-negative aware loss function can bring significant performance improvement, which proves its effectiveness.

We also compared several pooling methods on YAGO43kET. The result is summarized in Table 4. *mean*, *max*, *ewp* stand for mean pooling, maximum pooling and exponential weighted pooling, respectively. We can see that exponentially weighted pooling outperforms other pooling methods, which is consistent with our previous analysis.

4.4 Case Study

In Tabel 5, we select three representative inferences made by our model. These examples show how CET used the N2T and Agg2T mechanisms to infer entities' missing types, and the inference process is interpretable.

In the first example, our model mainly uses the neighbor (*has won prize*, *Pulitzer Prize*) to conduct inference. This is intuitive because the correlation between other neighbors and the candidate type *Pulitzer Prize winners* is indeed not strong. In the second example, our model uses several entities' known types to conduct inference. This inference process is reasonable and can be described in natural language: Ian Fleming is an English short story writer, so he is also an English writer. In the first two examples, the model mainly uses the N2T mechanism. However, in the last example, the type *American male television actors* involves multiple attributes of the entity, which the N2T mechanism cannot infer. Therefore, we can see our model uses the aggregation of neighbors to complete the inference, which is consistent with our previous analysis.

In addition, we provide some N2T examples in Table 6 to show the mapping from neighbors to types, and the results are intuitive.

Neighbors	Top 3 Relevant Types	
	Type	**Relevance Score**
(plays for, A.C. Milan)	A.C Milan players	6.32
	Serie A footballers	4.68
	Living people	2.71
(has won prize, Nobel Prize in Chemistry)	Nobel laureates in Chemistry	3.33
	scientist	2.35
	20th-century chemists	1.54
(type, American rock singers)	American rock singers	6.37
	American singers	3.87
	rock singers	1.89

Table 6: Three most relevant types with a particular neighbor.

5 Conclusion

This paper describes a novel knowledge graph entity typing method called CET, which utilizes the entities' contextual information to infer entities' missing types. We design two inference mechanisms, one is to independently use each neighbor of an entity to infer its types, another is to aggre-

gate entities' neighbors than conduct inference. In addition, we propose a novel loss function to alleviate the false negative problem during training. Our method is highly flexible, and we are considering introducing advanced graph convolutional network technology into our method.

Acknowledgments

We would like to thank all the anonymous reviewers for their insightful and valuable suggestions, which help improve this paper's quality. This work is supported by Cognitive Computing and Intelligent Information Processing (CCIIP) Laboratory, School of Computer Science and Technology, Huazhong University of Science and Technology.

References

Kurt D. Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2008, Vancouver, BC, Canada, June 10-12, 2008*, pages 1247–1250. ACM.

Antoine Bordes, Sumit Chopra, and Jason Weston. 2014. Question answering with subgraph embeddings. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 615–620. ACL.

Antoine Bordes, Nicolas Usunier, Alberto García-Durán, Jason Weston, and Oksana Yakhnenko. 2013. 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*, pages 2787–2795.

Eunsol Choi, Omer Levy, Yejin Choi, and Luke Zettlemoyer. 2018. Ultra-fine entity typing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 1: Long Papers*, pages 87–96. Association for Computational Linguistics.

Nitish Gupta, Sameer Singh, and Dan Roth. 2017. Entity linking via joint encoding of types, descriptions, and context. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017*, pages 2681–2690. Association for Computational Linguistics.

Junheng Hao, Muhao Chen, Wenchao Yu, Yizhou Sun, and Wei Wang. 2019. Universal representation learning of knowledge bases by jointly embedding instances and ontological concepts. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2019, Anchorage, AK, USA, August 4-8, 2019*, pages 1709–1719. ACM.

Hailong Jin, Lei Hou, Juanzi Li, and Tiansi Dong. 2018. Attributed and predictive entity embedding for fine-grained entity typing in knowledge bases. In *Proceedings of the 27th International Conference on Computational Linguistics, COLING 2018, Santa Fe, New Mexico, USA, August 20-26, 2018*, pages 282–292. Association for Computational Linguistics.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Timothée Lacroix, Nicolas Usunier, and Guillaume Obozinski. 2018. Canonical tensor decomposition for knowledge base completion. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pages 2869–2878. PMLR.

Xiao Ling and Daniel S. Weld. 2012. Fine-grained entity recognition. In *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence, July 22-26, 2012, Toronto, Ontario, Canada*. AAAI Press.

Xin Lv, Lei Hou, Juanzi Li, and Zhiyuan Liu. 2018. Differentiating concepts and instances for knowledge graph embedding. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 1971–1979. Association for Computational Linguistics.

Changsung Moon, Steve Harenberg, John Slankas, and Nagiza F. Samatova. 2017a. Learning contextual embeddings for knowledge graph completion. In *21st Pacific Asia Conference on Information Systems, PACIS 2017, Langkawi, Malaysia, July 16-20, 2017*, page 248.

Changsung Moon, Paul Jones, and Nagiza F. Samatova. 2017b. Learning entity type embeddings for knowledge graph completion. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management, CIKM 2017, Singapore, November 06 - 10, 2017*, pages 2215–2218. ACM.

Arvind Neelakantan and Ming-Wei Chang. 2015. Inferring missing entity type instances for knowledge base completion: New dataset and methods. In *NAACL HLT 2015, The 2015 Conference of the*

North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Denver, Colorado, USA, May 31 - June 5, 2015, pages 515–525. The Association for Computational Linguistics.

Maximilian Nickel, Lorenzo Rosasco, and Tomaso A. Poggio. 2016. Holographic embeddings of knowledge graphs. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA*, pages 1955–1961. AAAI Press.

Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. 2011. 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*, pages 809–816. Omnipress.

Michael Sejr Schlichtkrull, Thomas N. Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. 2018. Modeling relational data with graph convolutional networks. In *The Semantic Web - 15th International Conference, ESWC 2018, Heraklion, Crete, Greece, June 3-7, 2018, Proceedings*, volume 10843 of *Lecture Notes in Computer Science*, pages 593–607. Springer.

Chao Shang, Yun Tang, Jing Huang, Jinbo Bi, Xiaodong He, and Bowen Zhou. 2019. End-to-end structure-aware convolutional networks for knowledge base completion. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*, pages 3060–3067. AAAI Press.

Alexandros Stergiou, Ronald Poppe, and Grigorios Kalliatakis. 2021. Refining activation downsampling with softpool. *CoRR*, abs/2101.00440.

Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. 2007. Yago: a core of semantic knowledge. In *Proceedings of the 16th International Conference on World Wide Web, WWW 2007, Banff, Alberta, Canada, May 8-12, 2007*, pages 697–706. ACM.

Zhiqing Sun, Zhi-Hong Deng, Jian-Yun Nie, and Jian Tang. 2019. 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.

Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. 2016. 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*, volume 48 of *JMLR Workshop and Conference Proceedings*, pages 2071–2080. JMLR.org.

Shikhar Vashishth, Soumya Sanyal, Vikram Nitin, and Partha P. Talukdar. 2020. 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.

Yu Zhao, Anxiang Zhang, Ruobing Xie, Kang Liu, and Xiaojie Wang. 2020. Connecting embeddings for knowledge graph entity typing. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 6419–6428. Association for Computational Linguistics.

Da Zheng, Xiang Song, Chao Ma, Zeyuan Tan, Zihao Ye, Jin Dong, Hao Xiong, Zheng Zhang, and George Karypis. 2020. Dgl-ke: Training knowledge graph embeddings at scale. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '20, page 739–748, New York, NY, USA. Association for Computing Machinery.

Ben Zhou, Daniel Khashabi, Chen-Tse Tsai, and Dan Roth. 2018. Zero-shot open entity typing as type-compatible grounding. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 2065–2076. Association for Computational Linguistics.

A Hyper-parameter Settings

R-GCN. We use one layer R-GCN with 100-dimension embeddings in our experiment. The hyper-parameters are tuned according to the MRR on the validation set. The search space for the grid search are set as follows: learning rate $lr \in \{0.001, 0.005, 0.01\}$, activation function $\varphi \in none, relu, tanh$, and the weight of negative samples in FNA loss $\beta \in \{1, 2, 3, 4\}$. The input embedding is randomly initialized with a uniform distribution [-0.1, 0.1], and the training batch size is fixed to 128. Using *basis-* or *block-diagonal-* decomposition do not improve results but removing the *self-loop* improve performance. Table 7 summarizes the best configuration.

Dataset	lr	β	φ	self-loop
FB15kET	0.001	3	none	FALSE
YAGO43kET	0.001	2	none	FALSE

Table 7: The best configuration for R-GCN.

KGE methods. For KGE methods, we use 200-dimension embeddings in our experiment. We use random search to tune the hyper-parameters for

KGE methods. Table 8 summarizes the search space. *neg_num* is the number of negative samples for every positive sample; α is the temperature in the self-adversarial negative sampling; *lr* is the learning rate; λ is the regularization coefficient in L3 regularization; γ is a fixed margin in *logsigmoid* loss function and it also controls the initialization of the embeddings. We fix the training batch size to 1024.

Model	neg_num	α	lr	λ	γ
TransE	{128, 256, 512}	[0.5, 2.0]	[0.005, 0.2]	[1e-7, 1e-5]	[5, 15]
ComplEx	{128, 256, 512}	[0.5, 2.0]	[0.005, 0.2]	[1e-7, 1e-5]	[60, 80]
RotatE	{128, 256, 512}	[0.5, 2.0]	[0.005, 0.2]	[1e-7, 1e-5]	[5, 20]

Table 8: Search space for KGE methods.

We run 100 trails for each model, and every trial runs 50000 steps. The best configuration was selected according to the MRR on the validation set. Table 9 summarizes the best configuration for each model.

Dataset	Model	neg_num	α	lr	λ	γ
FB15kET	TransE	256	1.98	0.023	7.20E-06	6.5
	ComplEx	512	2.00	0.148	6.20E-06	66.8
	RotatE	512	1.91	0.0168	3.50E-06	6.0
YAGO43kET	TransE	512	1.99	0.05	4.40E-06	10.5
	ComplEx	512	1.99	0.154	2.00E-06	62.4
	RotatE	256	1.74	0.0344	2.40E-06	11.8

Table 9: The best configuration for KGE methods.