AsyncET: Asynchronous Learning for Knowledge Graph Entity Typing with Auxiliary Relations

Submission 1872

ORCiD ID: Submission 1872 https://orcid.org/....-....-....-....

Abstract. Knowledge graph entity typing (KGET) is a task to predict the missing entity types in knowledge graphs (KG). Previously, KG embedding (KGE) methods tried to solve the KGET task by introducing an auxiliary relation, 'hasType', to model the relationship between entities and their types. However, a single auxiliary relation has limited expressiveness for diverse entity-type patterns. We improve the expressiveness of KGE methods by introducing multiple auxiliary relations in this work. Similar entity types are grouped to reduce the number of auxiliary relations and improve their capability to model entity-type patterns with different granularities. With the presence of multiple auxiliary relations, we propose a method adopting an **Async**hronous learning scheme for **E**ntity **T**yping, named AsyncET, which updates the entity and type embeddings alternatively to keep the learned entity embedding up-to-date and informative for entity type prediction. Experiments are conducted on two commonly used KGET datasets to show that the performance of KGE methods on the KGET task can be substantially improved by the proposed multiple auxiliary relations and asynchronous embedding learning. Furthermore, our method has a significant advantage over state-of-the-art methods in model sizes and time complexity.

1 Introduction

Knowledge graph (KG) stores human-readable knowledge in a graph-structured format, where nodes and edges denote entities and relations, respectively. There are multiple relation types in KGs to describe the relationship between two entities. A (head entity, relation, tail entity) factual triple is a basic component in KGs. In addition to different relation types, each entity also comes with multiple types to describe the high-level abstractions of an entity[1]. Fig. 1 shows an example KG containing entity type information. As shown in the figure, each entity can be labeled with multiple types. For example, the entity "*Mark Twain*" has types "*writer*" and "*lecturer*" at the same time. Entity types are crucial in several artificial intelligence (AI) and natural language processing (NLP) applications, such as drug discovery [11], entity alignment, [23, 8], and entity linking [5]. In real-world applications, entity types could be missing, e.g. having type "*writer*" without type "*person*", due to prediction errors from information extraction models [24, 25]. Such missing types can be inferred from the existing information in KG. For example, in Fig. 1, we can infer that "*Mark Twain*" has a missing type "*person*" given that there is a known type "*writer*" and the relation "*born in*". Thus, knowledge graph entity typing (KGET) is a task to predict the missing types based on the observed types and triples in the KGs. KGET methods can serve as refinement mechanisms for real-world knowledge bases.

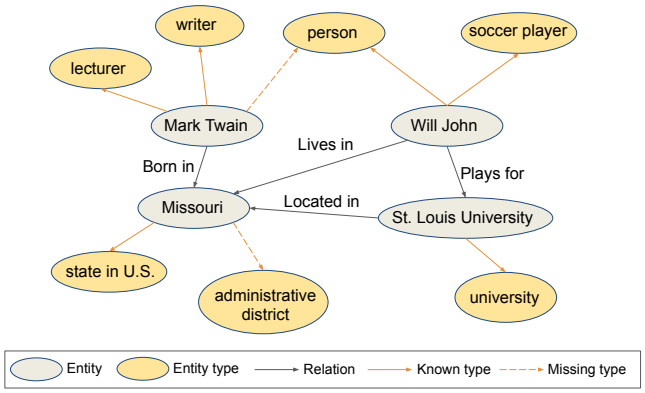


Figure 1: An example KG with missing entity types.

Knowledge graph embedding (KGE) methods achieve great success in predicting missing triples in KGs [9], and they are extended to solve the KGET task in [15]. Since the type labels are stored in the format of tuples (entity, type), an auxiliary relation, *hasType*, is first introduced to convert the typing tuples into triples (entity, *hasType*, type), and, then, a KGE method [14] is adopted to predict missing types. Although such a method is time- and parameter-efficient, it does not perform well since the relationship between entities and types is too diverse to be modeled by a single relation. In addition, such a method does not consider the neighborhood information. This affects the performance of entity type prediction as well. Other methods are proposed to improve the model's expressiveness. Embedding-based methods, such as ConnectE [26], first learn embeddings for entities and types separately using KGE methods. Then, a linear projection matrix is learned to minimize the distance between the entity and the type spaces. Another work leverages multi-relational graph convolution networks (R-GCN) [18] to encode the neighborhood information into entity embeddings. Attention mechanism is also explored in [27] to control the contributions of neighbors when predicting an entity type. Afterward, multi-layer perceptrons (MLPs) are cascaded to predict the entity types based on the learned entity embeddings. The KGET task is therefore formulated as a multi-label classification problem. Although GCN-based methods offer superior performance, they are not applicable in a resource-constrained environment, such as mobile/edge devices and real-time prediction [10], due to their high inference time complexity and large model sizes. In addition, training

[1] We refer to "entity type" when using the term "type" in the remainder of this paper.

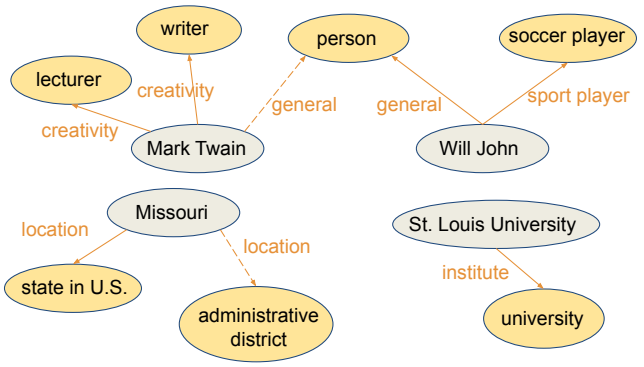


Figure 2: Illustration of using multiple auxiliary relations to model the relationship between entities and entity types.

GCNs could be time- and memory-consuming. Their applicability to large-scale KGs is challenging. It is desired to develop a KGET method of low inference time complexity, small model sizes, and good performance. This is the objective of this work.

As argued above, a single auxiliary relation is not sufficient to model the relationship between entities and types. Here, we introduce more auxiliary relations to improve the expressiveness of KGE models on the entity-type prediction task. This idea is illustrated in Fig. 2, where we show an example of using multiple auxiliary relations to model the entity-type relationship. It is intuitive that we should use different auxiliary relations to model typing relationship for type "*administrative district*" and type "*person*" since they describe two different concepts of entities. Similarly, type "*writer*" and type "*soccer player*" should adopt different auxiliary relations since they are semantically different. They should not be close to each other in the embedding space. On the other hand, for other types, such as "*writer*" and "*lecturer*", they co-occur with each other more frequently. Thus, they can adopt the same auxiliary relation for model simplicity.

Along this direction, we introduce multiple auxiliary relations based on the "context" of types. The context of a type is defined as a collection of attributes of its entities. It can be viewed as a discrete representation of the type. As such, the local KG structure is implicitly encoded when auxiliary relations are used. Next, we propose a method adopting an **Async**hronous learning scheme for **E**ntity **T**yping, named AsyncET, to obtain better embeddings for entities and types for the entity type prediction task. The training process consists of two stages: 1) link prediction, and 2) type prediction. Entity embeddings are first optimized by training with only factual triples to predict missing links. Then, the typing information is used to train type embeddings and fine-tune entity embeddings by predicting missing types. Two training stages alternate as the training progresses. The asynchronous training schema keeps the learned entity embedding up-to-date and informative for entity type prediction. Experiments conducted on two KGET datasets demonstrate that the performance of the KGET task can be substantially improved by the proposed multiple auxiliary relations and asynchronous training framework. Furthermore, AsyncET has a significant advantage over existing KGET methods in model sizes and time complexity.

The main contributions of this paper are summarized below.

- We introduce a novel strategy, called multiple auxiliary relations, to model relationships between entities and types.
- We propose a new asynchronous embedding learning framework, named AsyncET, to obtain better entity and type embeddings for the KGET task.
- AsyncET can improve the performance of KGE models on the KGET task substantially while being more efficient in inference time and model size.

2 Related Work

2.1 Embedding-based Methods

KGE methods rely on an auxiliary relation, *hasType*, to form typing triples (entity, *hasType*, type) so as to solve the KGET task. Synchronous training is often adopted to mix the factual triples and typing triples when training embeddings. Such methods have advantages in inference and model parameter efficiency. However, the performance is difficult to improve due to overly simplifying the relationship between entities and types. Other embedding-based methods tend to learn an entity space and a type space separately. Then, a mapping between two vector spaces is learned to predict the missing connections between entities and types. ETE [15] tried to minimize the distance between the learned entity and type space through the L1-norm. ConnectE [26] adopts a linear projection matrix to connect entity and type embedding spaces. JOIE [6] proposes two training objectives, cross-view grouping and cross-view transformation, to make sure entities with similar types are embedded closely. TransC [12] encodes entities and types in the same embedding space as high-dimensional balls. Several constraints are imposed to maintain the hierarchy among entity types. CORE [3] learns KGE in a complex subspace [19, 20] for entities and types individually. Then, a linear regression problem is solved to link entities with their corresponding types. Although the embedding-based methods generally contain fewer model parameters and have lower inference time, their performance highly depends on the expressiveness of the model used to describe the entity-type relations.

2.2 Deep Neural Network Methods

The neighborhood information is important in the KGET task since the types of an entity can often be determined by the neighboring entities and types. Following this line of thought, multi-relational GCNs [18, 28, 21] are proposed for the KGET task. First, entity embeddings are aggregated from the neighboring entities and types in GCNs. Then, multi-layer perceptrons (MLPs) are used to predict missing entity types by solving a multi-label classification problem. Since not all neighbors contribute to entity type prediction equally, an attention mechanism has been used to achieve better performance in recent work. For example, ConnectE-MRGAT [27] adopts graph attention networks (GATs) [22, 16] to solve the KGET task. CET [17] uses two attention mechanisms (i.e., N2T and Agg2T) to aggregate the neighborhood information. AttET [29] adopt a type-specific attention mechanism to improve the quality of entity embeddings. TET [7] uses a transformer as the entity encoder to aggregate the neighboring information. Although deep neural network methods can achieve superior performance, their inference complexity and model sizes are much larger than those of embedding-based methods.

3 Methodology

3.1 Notations

We use $\mathcal{G}$ to denote a KG containing a collection of factual triples; namely,

$$\mathcal{G} = \{(e^{head}, r, e^{tail}) \mid e^{head}, e^{tail} \in \mathcal{E}, r \in \mathcal{R}\}, \quad (1)$$

where $\mathcal{E}$ and $\mathcal{R}$ represent sets of entities and relations in the KG, respectively. The type information is denoted as

$$\mathcal{I} = \{(e, t) \mid e \in \mathcal{E}, t \in \mathcal{T}\}, \tag{2}$$

where $\mathcal{T}$ is a set of entity types in the KG. In order to group similar types based on the attributes of their associated entities, we define the context of type t as a collection of relations that co-occur with entities of type t

$$\mathcal{C}_t = \{r \mid (e^{head}, r, e^{tail}) \in \mathcal{G}, (e^{head}, t) \in \mathcal{I}\}. \tag{3}$$

For example, the context of type *person* is {*Born in*, *Lives in*, *Plays for*, ... }. It contains all attributes an entity of type *person* can have. The context of type t can be seen as a discrete representation for t that encodes the local structure of the KG.

3.2 Auxiliary Relations

Previous KGE models have only one auxiliary relation - *hasType*. They convert a typing tuple, (e, t), into a typing triple, $(e, \textit{hasType}, t)$. However, a single relation is not sufficient to model diverse entity-type patterns. Here, we aim to find a mapping such that, given entity type t, an auxiliary relation $p = Aux(t)$ is assigned to form new typing triples (e, p, t), where $t \in \mathcal{T}$, $p \in \mathcal{P}$, and $\mathcal{P}$ denotes a set of auxiliary relations in the KG. The objective is to maximize the capabilities of auxiliary relations to model every entity-type relationship. We compare three methods for the design of auxiliary relations below.

Bijective assignment. A straightforward solution to enhance the expressiveness of auxiliary relations is to assign a unique auxiliary relation to each type, called the bijective assignment. It can model diverse typing patterns well by exhaustively modeling every possible typing pattern in the KG. However, when the KG contains a large number of types, this assignment has several shortcomings. First, the model optimization is less stable since the number of model parameters increases significantly. Second, it is too fine-grained to perform well on the test dataset. Third, its inference time is much longer. Therefore, it is essential to group similar types and assigns auxiliary relations to each group of types.

Taxonomy-based assignment. Taxonomy is a hierarchical organization of concepts in KGs. For example, type "/film/producer" in Freebase [1] is a type under category "film" with the subcategory "producer". To group types based on the taxonomy, we can group them based on the first layer of the taxonomy. For instance, "/film/producer" will belong to the "film" group, and "/sports/sports_team" will belong to the "sports" group. However, such a taxonomy might not be available for some KGs, say, YAGO subsets [13]. Furthermore, the first-layer-taxonomy-based assignment may not have enough granularity for some types. To address these issues, we propose a new assignment method below.

Efficient assignment. To strike a balance between a small number of auxiliary relations, $|\mathcal{P}|$, and high expressiveness of entity-type modeling, we maximize similarities among the types in the same group and minimize similarities among different groups. Mathematically, we adopt the Jaccard similarity between the type contexts to define similarities between types. It can be written in the form of

$$Sim(t, t') = \frac{|\mathcal{C}_t \cap \mathcal{C}_{t'}|}{|\mathcal{C}_t \cup \mathcal{C}_{t'}|}. \tag{4}$$

Then, based on the well-defined similarity function between types, grouping similar types with the minimum number of groups can be formulated as a min-max optimization problem. Such an optimization problem is NP-Hard. Here, we develop a greedy algorithm to find an approximate optimal assignment as elaborated in the following. First, we identify several anchor types to be the centroid of each group. Initially, the anchor type set is empty, and all types are not covered. The method iteratively selects the type, which has not yet been selected, with the largest intersection of context and the uncovered relations $|\mathcal{C}_t \cap \mathcal{U}|$ as a new anchor. The iteration ends when the union of all anchors' contexts is equal to $\mathcal{R}$. The process of finding anchor types is depicted in Algorithm 1. Then, we assign a unique auxiliary relation to each anchor type. The non-anchor types will find their most similar anchor types and share the same auxiliary relations with the anchor type.

Algorithm 1 Find anchors for efficient auxiliary relation assignment

Initialization:
  The uncovered relations in the KG: $\mathcal{U} = \mathcal{R}$
  The set of existing anchor types: $\mathcal{A} = \varnothing$
Iteration:
  while $\mathcal{U} \neq \varnothing$ **do**
    $t = \text{argmax}_{t \in \mathcal{T}} |\mathcal{C}_t \cap \mathcal{U}|$
    $\mathcal{U} \leftarrow \mathcal{U} \setminus \mathcal{C}_t$
    $\mathcal{A} \leftarrow \mathcal{A} \cup \{t\}$
  end while

To verify whether the proposed algorithm can generate reasonable results, we show examples of the grouped entity types in Table 1. We see from the table that auxiliary relation # 2 is assigned to types of persons who work in the entertainment industry, auxiliary relation # 20 is assigned to types that are mostly sports teams, and auxiliary relation # 27 is assigned to geographical locations. Similar types are successfully grouped together while types of distant semantic meanings are separated.

Table 1: Examples of auxiliary relations and the corresponding entity types using the proposed efficient assignment, where anchor types are marked in boldface.

Auxiliary Relation	Entity Types
# 2	**/film/producer** /TV/tv_director /film/writer /film/film_story_contributor
# 20	**/sports/sports_team** /soccer/football_team /baseball/baseball_team /sports/school_sports_team
# 27	**/location/administrative_division** /film/film_location /fictional_universe/fictional_setting /location/citytown

3.3 *AsyncET: Asynchronous Embedding Learning for Knowledge Graph Entity Typing*

After auxiliary relations are defined, typing tuples (e, t) can be converted into typing triples (e, p, t). Such typing triples form a typing graph

$$\mathcal{TG} = \{(e, p, t) \mid (e, t) \in \mathcal{I}, p = Aux(t)\}. \tag{5}$$

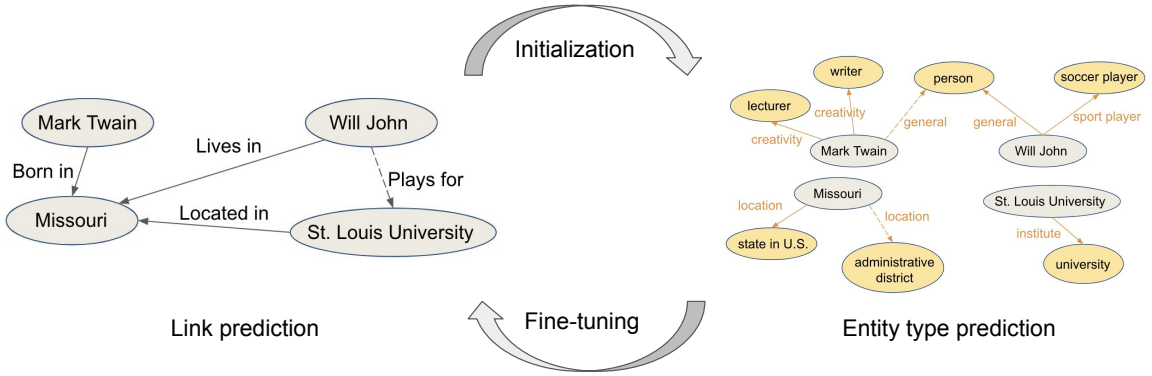


Figure 3: A diagram of the training process in AsyncET.

Instead of mixing the original triples and the newly-constructed typing triples together in embedding learning, we optimize the entity and type embeddings on the original KG $\mathcal{G}$ and the typing graph $\mathcal{TG}$ alternatively. That is, the embedding learning process is divided into two stages. In stage 1, the entity embeddings are trained on $\mathcal{G}$ using a link prediction task. The learned entity embeddings serve as an initialization for embedding learning in stage 2, where we use the typing graph $\mathcal{TG}$ to learn type embeddings and fine-tune entity embeddings with typing triples. The two training stages are optimized alternatively. Fig. 3 illustrates the training process of asynchronous embedding learning. Details of each training stage are elaborated below.

Stage 1: Link prediction. The goal of this stage is to obtain a good initialization of entity embeddings that can be used to predict the missing types. We follow the training loss in [19] with the self-adversarial negative sampling. The link prediction loss is defined as

$$\begin{aligned}\mathcal{L}_{lp} = & -\log(\sigma(f(\boldsymbol{e}^{\boldsymbol{head}}, \boldsymbol{r}, \boldsymbol{e}^{\boldsymbol{tail}}))) \\ & -\sum_{i=1}^{n} p(e_i', r, e_i'') \log(\sigma(-f(\boldsymbol{e}_i', \boldsymbol{r}, \boldsymbol{e}_i''))),\end{aligned} \tag{6}$$

where (e_i', r, e_i'') is the negative samples generated by corrupting the head and tail entities, $f(\boldsymbol{e}^{\boldsymbol{head}}, \boldsymbol{r}, \boldsymbol{e}^{\boldsymbol{tail}})$ is the scoring function in the KGE model, and $\boldsymbol{e}^{\boldsymbol{head}}, \boldsymbol{r}, \boldsymbol{e}^{\boldsymbol{tail}}$ are embeddings for the head entity, relation, and tail entity, respectively. The self-adversarial negative sampling distribution is defined as

$$p(e_j', r, e_j'') = \frac{\exp(\alpha f(\boldsymbol{e}_j', \boldsymbol{r}, \boldsymbol{e}_j''))}{\sum_{i=1}^{n} \exp(\alpha f(\boldsymbol{e}_i', \boldsymbol{r}, \boldsymbol{e}_i''))}, \tag{7}$$

where α is the temperature to control the smoothness of the softmax function. As a result, negative samples with lower scores are assigned smaller weights for optimization as they are well-trained already, and the model can focus on optimizing the hard cases.

Stage 2: Entity type prediction. In this stage, we fine-tune the entity embeddings and train the type embeddings using only typing triples. We adopt a loss similar to (6) to predict the missing entity types.

$$\begin{aligned}\mathcal{L}_{tp} = & -\log(\sigma(f(\boldsymbol{e}, \boldsymbol{p}, \boldsymbol{t}))) \\ & -\sum_{i=1}^{n} p(e, p_i', t_i') \log(\sigma(-f(\boldsymbol{e}, \boldsymbol{p}_i', \boldsymbol{t}_i'))),\end{aligned} \tag{8}$$

where (e, p_i', t_i') is a negative sample for entity type prediction generated by replacing the valid types with a random type t_i' and the corresponding auxiliary relation p_i'. The auxiliary relations are assigned based on mappings, $p = Aux(t)$ and $p_i' = Aux(t_i')$. Since the number of entity types is much fewer than the number of entities (i.e. $|\mathcal{T}| << |\mathcal{E}|$), false negatives, (i.e. $(e, p_i', t_i') \in \mathcal{TG}$) are more prevalent for entity-type prediction. To address this issue, we adopt false-negative-aware negative sampling distribution introduced in [17]. It can be written as

$$p(e, p_j', t_j') = x - x^2, \tag{9}$$

where

$$x = \sigma(-f(\boldsymbol{e}, \boldsymbol{p}_i', \boldsymbol{t}_i')). \tag{10}$$

Then, negative samples with the highest scores are assigned lower weights as they are possibly false negatives. Similar to the self-adversarial loss, negative samples with the lowest scores are already well-trained, so they are assigned smaller negative sampling weights.

Table 2: Dataset statistics.

		FB15k-ET	YAGO43k-ET
$\mathcal{G}$	# entities	14,951	42,334
	# relations	1,345	37
	# triples	483,142	331,686
$\mathcal{TG}$	# types	3,584	45,182
	# train	136,618	375,853
	# valid	15,848	43,111
	# test	15,847	43,119
	# p-bijective	3,584	45,182
	# p-taxonomy	89	-
	# p-efficient	54	10

4 Experiments

4.1 Experimental Setup

Datasets. We adopt two KGET datasets, FB15k-ET and YAGO43k-ET [15], for evaluation. FB15k-ET is derived from a link prediction dataset, FB15K [2], extracted from Freebase [1] by adding typing tuples. Freebase contains general relations between real-world entities. YAGO43k-ET is derived from another link prediction dataset, YAGO43k [14], extracted from YAGO [13] by adding typing tuples. There are mostly attributes and relations for persons in YAGO. The

Table 3: Results on KGET datasets, where the best performance in each column is shown in boldface, and the second-best performance is underlined.

Models	FB15kET				YAGO43kET			
	MRR	H@1	H@3	H@10	MRR	H@1	H@3	H@10
TransE [2]	0.618	0.504	0.686	0.835	0.427	0.304	0.497	0.663
RotatE [19]	0.632	0.523	0.699	0.840	0.462	0.339	0.537	0.695
CompoundE [4]	0.640	0.525	0.719	0.859	0.480	0.364	0.558	0.703
ETE [15]	0.500	0.385	0.553	0.719	0.230	0.137	0.263	0.422
ConnectE [26]	0.590	0.496	0.643	0.799	0.280	0.160	0.309	0.479
CET [17]	**0.697**	**0.613**	<u>0.745</u>	0.856	**0.503**	**0.398**	<u>0.567</u>	0.696
ConnectE-MRGAT [27]	0.630	0.562	0.662	0.804	0.320	0.243	0.343	0.482
AttET [29]	0.620	0.517	0.677	0.821	0.350	0.244	0.413	0.565
AsyncET-TransE (Ours)	0.659	0.552	0.729	0.859	0.452	0.341	0.518	0.684
AsyncET-RotatE (Ours)	0.668	0.564	0.735	<u>0.864</u>	0.471	0.359	0.556	<u>0.717</u>
AsnycET-CompoundE (Ours)	<u>0.688</u>	<u>0.581</u>	**0.755**	**0.885**	<u>0.492</u>	<u>0.380</u>	**0.574**	**0.721**

dataset statistics are summarized in Table 2, where p-bijective, p-taxonomy, and p-efficient denote the auxiliary relations obtained from the bijective assignment, the taxonomy-based assignment, and the efficient assignment described in Sec. 3.2, respectively. Only FB15k-ET contains taxonomy labels for the types.

Implementation details. We select three representative KGE methods as the scoring functions to evaluate the effectiveness of AsyncET. Specifically, we select TransE [2], RotatE [19], and CompoundE [4]. TransE models relations as translation in the vector space. It has several limitations in expressiveness since it does not model symmetric relations well. RotatE models relations as rotation in the complex embedding space. CompoundE is a recently proposed KGE method that generalizes the majority of distance-based methods by including compounding geometric transformations such as translation, rotation, and scaling. It also operates in the complex embedding space. The scoring functions of three KGE methods for AsyncE are written below.

- TransE [2]:

$$f(\boldsymbol{e}^{\boldsymbol{head}}, \boldsymbol{r}, \boldsymbol{e}^{\boldsymbol{tail}}) = \gamma - \|\boldsymbol{e}^{\boldsymbol{head}} + \boldsymbol{r} - \boldsymbol{e}^{\boldsymbol{tail}}\|,$$

  where γ is the margin. It's a hyperparameter that can be tuned.
- RotatE [19]:

$$f(\boldsymbol{e}^{\boldsymbol{head}}, \boldsymbol{r}, \boldsymbol{e}^{\boldsymbol{tail}}) = \gamma - \|\boldsymbol{e}^{\boldsymbol{head}} \circ \boldsymbol{r} - \boldsymbol{e}^{\boldsymbol{tail}}\|,$$

  where $\circ$ denotes the rotation operation in the complex embedding space.
- CompoundE [4]:

$$f(\boldsymbol{e}^{\boldsymbol{head}}, \boldsymbol{r}, \boldsymbol{e}^{\boldsymbol{tail}}) = \gamma - \|\mathbf{T}_{\mathbf{r}} \cdot \mathbf{R}(\theta_{\mathbf{r}}) \cdot \mathbf{S}_{\mathbf{r}} \cdot \mathbf{e}^{\boldsymbol{head}} - \mathbf{e}^{\boldsymbol{tail}}\|,$$

  where $\mathbf{T}_{\mathbf{r}}$, $\mathbf{R}(\theta_{\mathbf{r}})$, $\mathbf{S}_{\mathbf{r}}$ denote the translation, rotation, and scaling operations in CompoundE, respectively.

For both datasets, we select the best hyper-parameters from a certain search space under the embedding dimension $d = 500$ based on the performance of the validation set for entity type prediction. The search space is given below:

- Number of negative samples $n_{neg} \in \{128, 256_{\star\diamond}, 512\}$;
- Learning rate $lr \in \{0.01_{\diamond}, 0.001_{\star}, 0.0001\}$;
- Softmax temperature $\alpha \in \{0.5, 1.0_{\star\diamond}\}$;
- Margin $\gamma \in \{8.0_{\star}, 12.0, 16.0, 20.0_{\diamond}\}$.

The hyper-parameter settings adopted for FB15k-ET and YAGO43k-ET are marked with $\star$ and $\diamond$, respectively. We also conduct an ablation study on the performance against the number of alternate steps between two training stages in asynchronous embedding learning in Sec. 4.4. Based on the study, we alternate two training stages every 16 steps for both datasets. All experiments are conducted using one NVIDIA Tesla P100 GPU.

Evaluation metrics. For the KGET task, the goal is to predict the missing types given an entity, i.e. $(e, ?)$. However, in AsyncET, we convert the tuples into triples so the test queries become $(e, ?, ?)$. Since each entity type is only modeled by one auxiliary relation, we evaluate the joint plausibility $f(e, p', t'), \forall t' \in \mathcal{T}$, where $p' = Aux(t')$ for a given query. The valid entity types should be ranked as high as possible compared to all other candidates. Following the convention in [2], we adopt the filtered setting, where all entity types in the KG serve as candidates except for those observed ones. Several commonly used ranking metrics are adopted, including the Mean Reciprocal Rank (MRR) and Hits@k (k=1, 3, and 10).

4.2 Main Results

The experimental results on two KGET datasets are given in Table 3, models are clustered into three groups: 1) KGE methods trained using a single auxiliary relation (i.e., *hasType*), 2) other type-embedding methods and models using graph neural networks, 3) proposed AsyncET with TransE, RotatE, and CompoundE scoring functions. For group 3, we report the best performance using p-bijective, p-taxonomy, or p-efficient. Detailed comparison of the effects of different auxiliary relations will be discussed in Sec. 4.3. We have the following observations from the table. AsyncET is significantly better than KGE methods trained with only one auxiliary relation. CET performs better in exact matches (i.e. H@1) than AsyncET since they adopt a graph neural network to learn entity embeddings from neighbors. However, the performance of some type-embedding methods, such as ETE and ConnectE, is much worse than that of AsyncET for both datasets since they do not encode the neighboring information effectively. Recent methods, such as CET, ConnectE-MRGAT, and AttET, adopt attention mechanisms to obtain context-aware entity embeddings to predict the missing types. In AsyncET, entities' attributes are encoded through auxiliary relations and asynchronous training. AsyncET using scoring functions of TransE, RotatE, and CompoundE outperforms attention-based methods in all metrics except for CET.

Table 4: Ablation study on asynchronous representation learning and different auxiliary relations. The MRR performance is reported. The best performance in each column is shown in boldface.

		FB15kET			YAGO43kET		
Training	*Aux. Rel.*	TransE	RotatE	CompoundE	TransE	RotatE	CompoundE
Syn.	*hasType*	0.618	0.632	0.640	0.427	0.462	0.480
Syn.	p-bijective	0.532	0.534	0.581	0.362	0.388	0.407
Syn.	p-taxonomy	0.545	0.550	0.603	-	-	-
Syn.	p-efficient	0.565	0.564	0.625	0.418	0.438	0.455
Asyn.	*hasType*	0.621	0.624	0.638	0.443	0.458	0.474
Asyn.	p-bijective	**0.659**	**0.668**	**0.688**	0.391	0.418	0.442
Asyn.	p-taxonomy	0.633	0.641	0.664	-	-	-
Asyn.	p-efficient	0.654	0.661	0.682	**0.452**	**0.471**	**0.492**

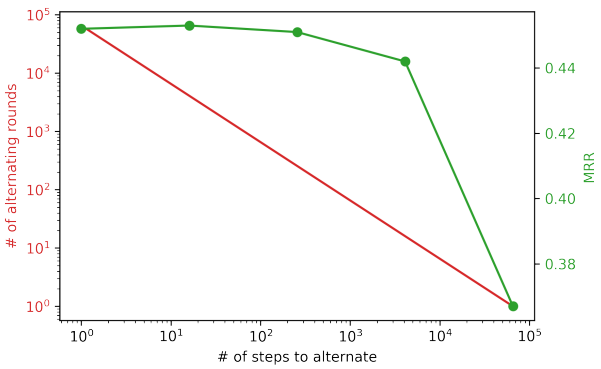


Figure 4: The MRR performance as a function of the number of alternating rounds between two stages in asynchronous representation learning.

Furthermore, AsyncET-CompoundE can even outperform CET in H@3 and H@10.

4.3 Ablation Study

To analyze the effectiveness of asynchronous training and different auxiliary relation assignments, we conduct an ablation study in Table 4, where the MRR performance of several methods for two datasets is reported. We compare the following:

- Synchronous vs. asynchronous training;
- Single auxiliary relation *hasType* vs. multiple auxiliary relations with p-bijective, p-taxonomy, and p-efficient designs;
- TransE, RotatE, or CompoundE scoring functions.

As shown in the table, asynchronous training consistently outperforms synchronous training. The performance improvement of asynchronous training when using only one auxiliary relation, *hasType*, is not significant since the second stage of the embedding learning is trained on a single-relational graph. When there are multiple auxiliary relations, mixing typing triples with original factual triples in embedding training using the synchronous framework still yields poor performance. This could be attributed to the fact that KGE methods are difficult to train when there are too many relations. The performance improves significantly when we decompose the training process into two stages under the asynchronous framework.

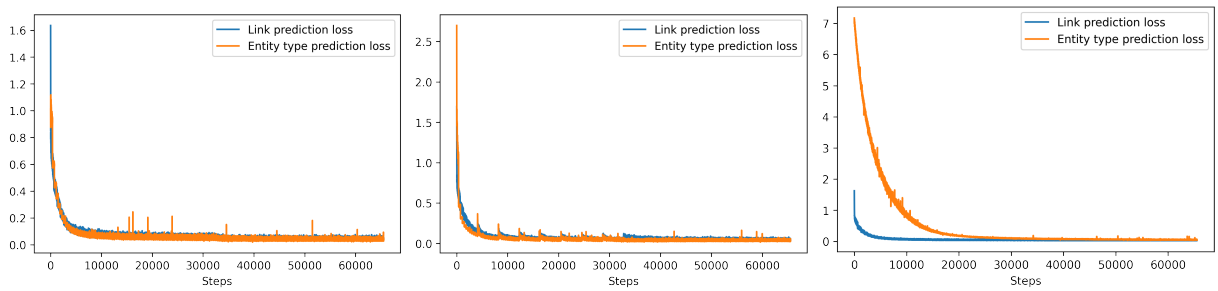
When using multiple auxiliary relations to model the typing relationship, the bijective assignment works well in the dataset with fewer entity types, e.g. FB15k-ET. However, it performs worse than the efficient assignment on the dataset with more entity types, e.g. YAGO43kET. In datasets with many entity types, the bijective assignment introduces larger amounts of new parameters, making the embedding training more difficult to converge. The efficient assignment can achieve the best performance on YAGO43kET, showing that such an assignment can capture the context in the KG and similarities among entity types well. Surprisingly, the taxonomy-based assignment does not perform well on both datasets. One possible reason is that grouping entity types based on only the first layer of the taxonomy ignores the granularities of different entity-type patterns. Such a problem might be solved by considering more layers in the taxonomy.

4.4 Number of Alternating Rounds

In asynchronous learning, we alternate between the link prediction stage (Stage 1) and the entity type prediction stage (Stage 2) after a few stochastic gradient descent steps. The link prediction loss in Eq. (6) and the entity type prediction loss in Eq. (8) are minimized in Stage 1 and Stage 2, respectively. The training process begins with Stage 1 and switches to Stage 2 after $N_{s,1}$ stochastic gradient descent steps. Similarly, it conducts $N_{s,2}$ stochastic gradient descent steps and then switches back to Stage 1. We call one entire cycle of performing Stage 1 and Stage 2 once an alternating round. In our experiments, we set $N_{s,1} = N_{s,2} = N_s$ and use N_r to denote the number of alternating rounds.

We plot the MRR performance on YAGO43kET using TransE as the scoring function as a function of N_s and N_r in Fig. 4 in the green line. We conduct experiments using N_s steps, with $N_s = 1, 16, 256, 4096, 65536$ in one round. Besides, we set $N_s \times N_r = 65,536$. The relation of N_s and N_r is shown by the red line. The smaller N_s being set means the stage alternation is more frequent. We see that the performance is better when alternating between the two stages more frequently. Clearly, asynchronous training contributes better entity and type embedding quality when there are more frequent interactions between the entity and type embedding spaces.

We also plot the training loss curves as a function of alternating rounds in Fig. 5. It shows that the training loss can be lower with more alternating rounds. In addition, both the link prediction loss and the entity type prediction loss are successfully reduced during the training. In other words, the two training stages are mutually beneficial. When alternating between the two stages only once, as shown in Fig. 5 (c), the loss curves go down slowlier than the other two cases. Thus, more alternating rounds help convergence. We conclude that alternating between two stages can fine-tune entity and type embeddings to approach the global optimal in optimization.



(a) # of alternating rounds = 256 (b) # of alternating rounds = 16 (c) # of alternating rounds = 1

Figure 5: Training loss curves with respect to different numbers of alternating rounds.

Table 5: Inference time and memory complexity of KGET methods.

Model	Inference time complexity	Memory complexity
ConnectE [26]	$O(Td_e^2 d_t)$	$O((E+R)d_e + Td_t + d_e d_t)$
R-GCN [18]	$O(TBL(E+T)d^2)$	$O((E+R+T)d + BLd^2 + BLR)$
WGCN [28]	$O(TL(E+T)d^2)$	$O((E+R+T)d + Ld^2 + LR)$
CET [17]	$O(T(Td^2 + 2T))$	$O((E+R+3T)d)$
KGE Models	$O(Td)$	$O((E+R+T)d)$
AsyncET (Ours)	$O(2Td)$	$O((E+R+T+P)d)$

4.5 *Complexity Analysis*

We conduct complexity analysis on the inference time and the number of model parameters for several representative methods in Table 5, where d, E, R, T, and P denote the embedding dimension, numbers of entities, relations, entity types, and auxiliary relations, respectively. For ConnectE, the embedding dimensions for entities and types are denoted as d_e and d_t, respectively. For GCN-based methods, B is the number of bases to decompose the propagation matrix in GCN layers, and L is the number of GCN layers. We see that KGE methods are the most efficient methods in terms of inference time complexity under the same embedding dimension. ConnectE is also an embedding-based method but it tries to learn a matrix to connect the entity and type space. Thus, its time complexity and number of model parameters are proportional to d^2, which are larger than KGE methods. The complexity of GCN methods is correlated with the number of layers and the number of nodes in the graph. As a result, the inference time complexity is proportional to $(E+T)Ld^2$, which is highly inefficient in testing. The complexity of AsyncET is similar to KGE methods except it needs additional model parameters to store embeddings for auxiliary relations. For time complexity, AsyncET requires twice the inference time as that of KGE methods since it considers joint probabilities $f(e, p', t')$ in candidate ranking. KGE methods only need to calculate the conditional probabilities $f(e, t' \mid \mathit{hasType})$.

4.6 *Qualitative Analysis*

We show some examples of predicted types in Table 6. In all three examples, the groundtruth ranks among the top three. In the first example, for entity *Mihail Majearu*, the model can successfully rank the groundtruth at the top one. In addition, the top three predicted types are all persons and the 2nd prediction is also related to football. In the second example, for entity *David Cross* who is a comedian, the model can rank the groundtruth at the top one again. The other two entity types are also relevant to the entity. They are valid types. In the last example, for entity *Zhejiang*, although the groundtruth only ranks as the third, the first two predicted types are both relevant to the entities. Note that *Zhejiang* is a province instead of a city in China. The granularity of the entity is not predicted correctly in the top two choices.

Table 6: Top 3 predicted entity types by AsyncET for entities in YAGO43kET. Groundtruth is marked in boldface.

Entity	Top 3 Type Predictions
Mihail Majearu	**Romanian footballers** Alkmaar players People from Glasgow
David Cross	**Jewish actors** 21st-century American actors American humorists
Zhejiang	Cities in Zhejiang Provincial capitals in China **Administrative divisions of China**

5 Conclusion and Future Work

Multiple auxiliary relations were proposed to solve the KGET task in this work. Three methods for the design of auxiliary relations were compared. Among the three, the efficient assignment is recommended since it is scalable to datasets containing many entity types. In addition, asynchronous embedding learning was proposed to achieve better entity and type embeddings by predicting missing links and types alternatively. It was shown by experimental results that AsyncET outperforms SOTA in H@3 and H@10 with much lower inference complexity and fewer model parameters. As future extensions, we will investigate how the sparsity of the typing information affects the performance of KGET methods. We aim to not only develop a time- and parameter-efficient model, but achieve less performance degradation when trained with fewer labeled entity types.

References

[1] Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor, 'Freebase: a collaboratively created graph database for structuring human knowledge', in *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pp. 1247–1250, (2008).

[2] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko, 'Translating embeddings for modeling multi-relational data', *Advances in neural information processing systems*, **26**, (2013).

[3] Xiou Ge, Yun-Cheng Wang, Bin Wang, and C-C Jay Kuo, 'Core: A knowledge graph entity type prediction method via complex space regression and embedding', *arXiv preprint arXiv:2112.10067*, (2021).

[4] Xiou Ge, Yun-Cheng Wang, Bin Wang, and C-C Jay Kuo, 'Compounde: Knowledge graph embedding with translation, rotation and scaling compound operations', *arXiv preprint arXiv:2207.05324*, (2022).

[5] Nitish Gupta, Sameer Singh, and Dan Roth, 'Entity linking via joint encoding of types, descriptions, and context', in *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pp. 2681–2690, (2017).

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

[7] Zhiwei Hu, Víctor Gutiérrez-Basulto, Zhiliang Xiang, Ru Li, and Jeff Z Pan, 'Transformer-based entity typing in knowledge graphs', *arXiv preprint arXiv:2210.11151*, (2022).

[8] Hongren Huang, Chen Li, Xutan Peng, Lifang He, Shu Guo, Hao Peng, Lihong Wang, and Jianxin Li, 'Cross-knowledge-graph entity alignment via relation prediction', *Knowledge-Based Systems*, **240**, 107813, (2022).

[9] Shaoxiong Ji, Shirui Pan, Erik Cambria, Pekka Marttinen, and S Yu Philip, 'A survey on knowledge graphs: Representation, acquisition, and applications', *IEEE Transactions on Neural Networks and Learning Systems*, (2021).

[10] C-C Jay Kuo and Azad M Madni, 'Green learning: Introduction, examples and outlook', *Journal of Visual Communication and Image Representation*, 103685, (2022).

[11] Yu Lin, Saurabh Mehta, Hande Küçük-McGinty, John Paul Turner, Dusica Vidovic, Michele Forlin, Amar Koleti, Dac-Trung Nguyen, Lars Juhl Jensen, Rajarshi Guha, et al., 'Drug target ontology to classify and integrate drug discovery data', *Journal of biomedical semantics*, **8**(1), 1–16, (2017).

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

[13] Farzaneh Mahdisoltani, Joanna Biega, and Fabian Suchanek, 'Yago3: A knowledge base from multilingual wikipedias', in *7th biennial conference on innovative data systems research*. CIDR Conference, (2014).

[14] Changsung Moon, Steve Harenberg, John Slankas, and Nagiza F Samatova, 'Learning contextual embeddings for knowledge graph completion', (2017).

[15] Changsung Moon, Paul Jones, and Nagiza F Samatova, 'Learning entity type embeddings for knowledge graph completion', in *Proceedings of the 2017 ACM on conference on information and knowledge management*, pp. 2215–2218, (2017).

[16] Deepak Nathani, Jatin Chauhan, Charu Sharma, and Manohar Kaul, 'Learning attention-based embeddings for relation prediction in knowledge graphs', *arXiv preprint arXiv:1906.01195*, (2019).

[17] Weiran Pan, Wei Wei, and Xian-Ling Mao, 'Context-aware entity typing in knowledge graphs', *arXiv preprint arXiv:2109.07990*, (2021).

[18] Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling, 'Modeling relational data with graph convolutional networks', in *European semantic web conference*, pp. 593–607. Springer, (2018).

[19] Zhiqing Sun, Zhi-Hong Deng, Jian-Yun Nie, and Jian Tang, 'RotatE: Knowledge graph embedding by relational rotation in complex space', in *International Conference on Learning Representations*, (2019).

[20] Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard, 'Complex embeddings for simple link prediction', in *International conference on machine learning*, pp. 2071–2080. PMLR, (2016).

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

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

[23] Yuejia Xiang, Ziheng Zhang, Jiaoyan Chen, Xi Chen, Zhenxi Lin, and Yefeng Zheng, 'Ontoea: ontology-guided entity alignment via joint knowledge graph embedding', *arXiv preprint arXiv:2105.07688*, (2021).

[24] Yadollah Yaghoobzadeh, Heike Adel, and Hinrich Schütze, 'Noise mitigation for neural entity typing and relation extraction', *arXiv preprint arXiv:1612.07495*, (2016).

[25] Yadollah Yaghoobzadeh and Hinrich Schütze, 'Corpus-level fine-grained entity typing using contextual information', *arXiv preprint arXiv:1606.07901*, (2016).

[26] Yu Zhao, Anxiang Zhang, Ruobing Xie, Kang Liu, and Xiaojie Wang, 'Connecting embeddings for knowledge graph entity typing', *arXiv preprint arXiv:2007.10873*, (2020).

[27] Yu Zhao, Han Zhou, Anxiang Zhang, Ruobing Xie, Qing Li, and Fuzhen Zhuang, 'Connecting embeddings based on multiplex relational graph attention networks for knowledge graph entity typing', *IEEE Transactions on Knowledge and Data Engineering*, (2022).

[28] Yunxiang Zhao, Jianzhong Qi, Qingwei Liu, and Rui Zhang, 'Wgcn: graph convolutional networks with weighted structural features', in *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 624–633, (2021).

[29] Jianhuan Zhuo, Qiannan Zhu, Yinliang Yue, Yuhong Zhao, and Weisi Han, 'A neighborhood-attention fine-grained entity typing for knowledge graph completion', in *Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining*, pp. 1525–1533, (2022).