World Wide Web (2023) 26:2453–2470
https://doi.org/10.1007/s11280-023-01155-1

A performant and incremental algorithm for knowledge graph entity typing

Zepeng Li[1] · Rikui Huang[1] · Minyu Zhai[1] · Zhenwen Zhang[1] · Bin Hu[1,2,3]

Received: 11 November 2022 / Revised: 1 February 2023 / Accepted: 14 February 2023 /
Published online: 30 March 2023

Abstract
Knowledge Graph Entity Typing (KGET) is a subtask of knowledge graph completion, which aims at inferring missing entity types by utilizing existing *type knowledge* and *triple knowledge* of the knowledge graph. Previous knowledge graph embedding (KGE) algorithms infer entity types through trained entity embeddings. However, for new unseen entities, KGE models encounter obstacles in inferring their types. In addition, it is also difficult for KGE models to improve the performance incrementally with the increase of added data. In this paper, we propose a statistic-based KGET algorithm which aims to take both performance and incrementality into consideration. The algorithm aggregates the neighborhood information and type co-occurrence information of target entities to infer their types. Specifically, we first compute the type probability distribution of the target entity in the semantic context of given fact triple. Then the probability information of fact triples involved in the target entity is aggregated. In addition to local neighborhood information, we also consider capturing global type co-occurrence information for target entities to enhance inference performance. Extensive experiments show that our algorithm outperforms previous statistics-based KGET algorithms and even some KGE models. Finally, we design an incremental inference experiment, which verifies the superiority of our algorithm in predicting the types of new entities, and the experiment also verifies that our algorithm has excellent incremental property.

Keywords Knowledge graph entity typing · Knowledge graph completion · Incremental inference

1 Introduction

Many large-scale Knowledge Graphs (KGs) were built for real-world applications such as recommendation systems [1] and question answering [2]. High-quality KGs can provide powerful support for many AI tasks [3]. For example, ERNIE 3.0 [4], which recently thrived on NLP tasks, claims that the model absorbed a well-designed knowledge graph with over

✉ Bin Hu
bh@lzu.edu.cn

Extended author information available on the last page of the article.

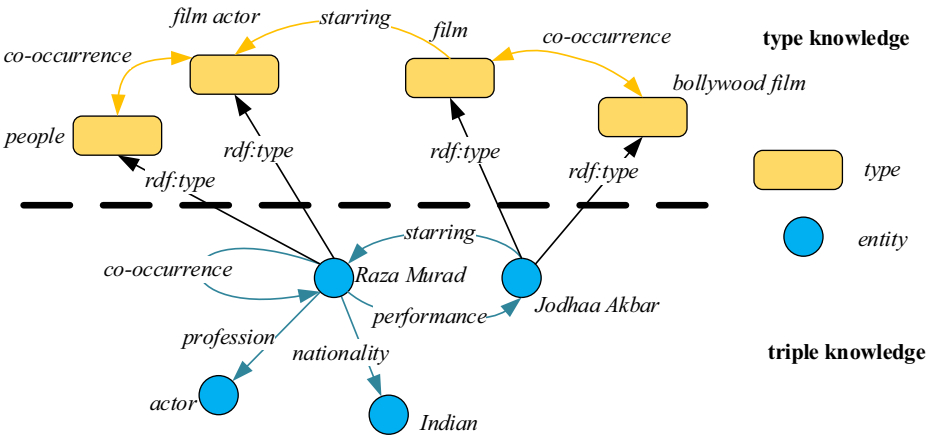

Figure 1 An example of knowledge graph

50 million entities. However, KGs usually suffer from *incompleteness* and miss important facts, jeopardizing their usefulness in downstream tasks. Therefore, it is critical to develop automatic methods of knowledge graph completion.

A KG is usually composed of a large number of fact triples (*subject*, *predicate*, *object*) (denoted (*s*, *p*, *o*)). The *subject* and *object* are entities, and the *predicate* is the relationship that connects the two entities. There is a special class of triples, whose predicates are *rdf:type*, use the object to indicate the entity type of the subject. We refer to these triples as *type knowledge*, and the remaining as *triple knowledge*. Figure 1 shows an example of a knowledge graph centered on an Indian film actor *Raza Murad*. Knowledge Graph Entity Typing (KGET) aims at inferring missing entity types by utilizing existing *type knowledge* and *triple knowledge* of the knowledge graph.

Previous KGE-based models achieved many achievements on KGET. They embed entity types and complete entity types through link prediction. High-quality type inference results can be obtained by relying on the entity embeddings trained by the KGE model. However, real-world knowledge graphs are often constantly updated. Take *7-lore* Knowledge Graph[1] as an example, the knowledge graph adds millions of entities a day. The KGE models does not have the entity information of these new additions during the training process, which causes KGE models to encounter many difficulties in the inference stage. [5] proposes the *inductive* concept to elaborate on this challenge. Although, some recent studies begin to focus on Out-of-Knowledge-Base Embedding [6–8], there is few KGET algorithm for new entities. On the other hand, the current KGE-based models cannot take advantage of incoming new data to improve performance, i.e. these algorithms are not incremental. Statistics-based algorithms have natural advantages in increment and efficiency, but the performance of existing statistics-based KGET algorithms is far inferior to KGE-based models.

In this paper, we aim to design a statistics-based KGET algorithm that takes into account both performance and incrementality. Specifically, for a target entity, we leverage its neighborhood information and its existing type information for inference. Inspired by [9], we assume that the relations connecting entities have semantic invariant properties from entity

[1] http://www.openkg.cn/dataset/7lore

to type. As shown in Figure 1, for the fact triple (*Jodhaa Akbar*, *starring*, *Raza Murad*), even if the subject and object are replaced with their corresponding types, the derived type triple (*film*, *starring*, *film actor*) still holds. For neighborhood information, it could be observed that *Raza Murad* is connected with the neighbor entity *Jodhaa Akbar* by the relation *starring*. If we know that entity *Jodhaa Akbar* is a film and has type *film*, we can infer the type triple (*film*, *starring*, *film actor*) and further infer that entity *Raza Murad* has type *film actor*. For type information, multiple different types often co-occur in an entity, such as *film actor* and *person*. If we know that *Raza Murad* already has the type of *film actor*, we can infer that *Raza Murad* has the type *person* by using co-occurrence information. We named the algorithm **PIANO**, which means **P**erformant and **I**ncremental algorithm with **A**ggregating **N**eighborhood and co-**O**ccurrence information for KGET. PIANO consists of two stages: **data statistics stage** and **information aggregation stage**.

Data statistics stage All type triples derived from fact triples will be counted. Intuitively, type triples with more occurrences are more likely to be considered facts. However, such intuitions may suffer from noise and imbalance in knowledge graphs. Therefore, we treat type-predicate pairs as queries and type triples as answers, and use the conditional probability of type triples under query pairs to measure the confidence of type triples. It is worth mentioning, we use a neat trick to add a self-loop with relationship *co-occurrence* to each entity for statistical type co-occurrence information. The entire data statistics process only needs to traverse the knowledge graph and its export type triples once.

Information aggregation stage At this stage, we consider two issues: i) How to aggregate the information of multiple type triples derived from a fact triple. ii) How to aggregate the information of multiple fact triples associated with a target entity. For these two issues, we adopt the strategy of mean and sum respectively. Finally, we get a score for the entity-type pair, which reflects the confidence that the entity has a type, the higher the score, the more likely the entity is to have the type.

Our **contributions** are as follows:

- We design a statistics-based algorithm for KGET named PIANO, which maintains both high performance and incremental property.
- We conduct KGET experiments on multiple datasets, and the experimental results show that PIANO far outperforms most previous statistics-based algorithms, and even outperforms KGE-based models.
- We design a specific incremental experiment to infer types of new entities and to verify the incremental properties of algorithms.

2 Related work

Statistics-based Algorithm In the semantic web community, there has been research related to KGET for a long time [10], which is called schema discovery. Schema discovery can be roughly classified into three categories: implicit schema discovery, explicit schema enrichment, and structural pattern discovery. Explicit schema enrichment into which KGET can be categorized. Among them SDType (Statistical Distribution of Types) [11] is the research most relevant to our work. SDType treats the predicates associated with the target entity as features, and tries to avoid propagation of errors of irrelevant instances through a weighted voting method. Its essential idea is type-constrained [12, 13], which means the

scope of entity type is constrained by predicate associated with the entity. For example, the types of subject and object entities connected by the *MarriedTo* predicate can usually only be person or subtypes of person. Fang et al. [14] is similar to SDType as it is based on the statistical distribution of types. The difference is that it infers the type based on the category information. However, not all knowledge graphs have category information available.

KGE-based Algorithm In the field of knowledge graph embedding, many works [15–19] have been proposed to embed entities and predicates into low-dimensional semantic spaces for downstream tasks. However, most of these models are used in the knowledge graph link prediction task, while ignoring the KGET task. Until recently, these KGE models were first applied to the KGET task by Moon et al. [20]. They tried two paradigms: One paradigm treats *rdf:type* triples as fact triples and directly uses the KGE models for type inference. Another paradigm is to also embed entity types into a low-dimensional semantic space, and perform type inference through the distance between entity embeddings and type embeddings. Thereafter, Zhao et al. [9] proposed the connecting embeddings model (ConnectE) and introduced the concepts of global type knowledge and local triple knowledge. The concept of global type knowledge considers entities with the same type to be close in the embedding space. The concept of local triple knowledge believes that the predicate has semantic invariance. That is, when the subject and object entities in the fact triple are replaced with the corresponding types, the derived type triple still holds. In addition, Pan et al. proposed CET [21] model, which utilizes the neighbor information in an independent-based mechanism and aggregated-based mechanism for type inference. Recently, several new KGE-based KGET models were proposed continuously, such as ConnectE-MRGAT [22], AttEt [23] and RACE2T [24].

In this paper, we propose a new statistics-based KGET algorithm PIANO. Different from SDType, PIANO utilizes not only the predicates associated with the target entity, but also the type information of neighbor entities for type inference. Incorporating the idea of aggregating neighborhood information based on representation learning, PIANO is able to possess satisfactory performance while maintaining the incremental property.

3 Methodology

To facilitate reading, we first define some notions. A KG G usually contains fact triples $(s, p, o) \in G$, where entities $s, o \in E$, and predicates $p \in P$. In KGET tasks, an entity e usually has some existing type declarations $t_e \in T$. We use $T(e)$ to represent the set of known type instances of entity e. Given a fact triple (s, p, o), based on the existing type declarations of its subject entity and object entity, we can derive a large number of type triples (t_s, p, t_o). All of the type triples derived from fact triples in G form a Type Graph (TG), denoted as G_t. Note that a type triple may appear in G_t repeatedly. For a given predicate p, we define the set of all subject types associated with p in G_t as domain, and the set of all object types as range, denoted as $D(p)$ and $R(p)$ respectively.

3.1 SDType++

Our goal is to be able to compute an entity-type matrix $M \in \mathbb{R}^{|E|\times|T|}$, where element M_{ij} represents the confidence score that the i^{th} entity has the j^{th} type. For a predicate p, if its domain frequently contains a type t, we believe that the subject s associated with the predicate p has a high probability of having that type. The intuition holds for its range.

To reflect this intuition, we first calculate the predicate-type incidence matrices $M^s_{p2t}, M^o_{p2t} \in \mathbb{R}^{|P|\times|T|}$. The elements in the matrices represent the times that the predicate and the subject(or the object) type appear together in G_t. For example, $M^s_{p2t}[p][t_s]$ indicates the number of $(t_s, p, -)$ in G_t. As we described in Figure 1, an entity in the KG may be associated with more than one fact triple. To this end, we calculate the entity-predicate incidence matrices $M^s_{e2p}, M^o_{e2p} \in \mathbb{R}^{|E|\times|P|}$. $M^s_{e2p}[e][p]$ indicates the number of $(e, p, -)$ in G. We refer to the algorithm as SDType++. Different from SDType [11], SDType++ not only considers what kind of predicate is associated with the target entity, but also considers that the target entity may be associated with the same predicate multiple times.

The statistical process of the previous matrix can be regarded as the data statistics stage of SDType++. Through matrix operations we can naturally aggregate all statistics and obtain entity-type matrix M:

$$M = M^s_{e2p} M^s_{p2t} + M^o_{e2p} M^o_{p2t} \tag{1}$$

3.2 PIANO

However, there are some problems with the intuition in SDType++. As shown in Table 1, no matter what the target entity and its neighbor entities are, as long as the entity is located in the object position of the predicate *starring*, the entity will be preferentially predicted to have type *tv actor*. Recalling Figure 1, we find that entity *Raza Murad* should be more likely to have type *film actor* than have *tv actor*. To compensate for this flaw, we fuse the information of neighbor entities of the target entity.

Data statistics stage First, we count the number of each type triple in the G_t. Considering data sparsity, we use a dictionary A to store these information. That is, $A[(t_s, p, t_o)]$ represents the number of (t_s, p, t_o) in G_t. In the process of calculating A, we can also calculate M^s_{p2t}, M^o_{p2t} above mentioned. In order to facilitate calculations and save computing resources, we use dictionaries Q_o and Q_s to store M^s_{p2t} and M^o_{p2t}, respectively. That is, $Q_o[(t_s, p)] = M^s_{p2t}[p][t_s]$ and $Q_s[(p, t_o)] = M^o_{p2t}[p][t_o]$. Second, we convert the above statistics into probability information. For each type triple (t_s, p, t_o), we calculate the conditional probability under the query of (t_s, p) and the query of (p, t_o):

$$p((t_s, p, t_o)|(t_s, p)) = A[(t_s, p, t_o)]/Q_o[(t_s, p)] \tag{2}$$

$$p((t_s, p, t_o)|(p, t_o)) = A[(t_s, p, t_o)]/Q_s[(p, t_o)] \tag{3}$$

For a triple (s, p, o), the value $p((t_s, p, t_o)|(t_s, p))$ indicates the probability that o has the type t_o under the query (t_s, p). The motivation for computing conditional probabilities is that the types of neighbor entities can constrain the types of target entities. For example, from common sense, we think that the $p((film, starring, film\ actor)|\ (film, starring))$ should

Table 1 Top 6 types of range of predicate *starring*

Type	Statistics
tv actor	7863
person	7841
film actor	7665
award nominee	7258
entity in film	6366
influence node	6326

be greater than $p((film, starring, tvactor)|\ (film, starring))$.That is, the probability of a film starring a film actor should be greater than the probability of a film starring a tv actor.

Information aggregation stage We first consider how to aggregate the information of multiple type triples derived from a fact triple. For the target entity, the information returned by each type triple is probabilistic information and the numbers of type triples derived by fact triples are different. For example, in Figure 1, *Indian* has 66 known type declarations, while *Jodhaa Akbar* has only 7. We use the averaging operation to keep the numerical value of the probability information between 0 and 1. Based on our common sense, we believe that *Jodhaa Akbar* is more helpful for predicting that *Raza Murad* have type *film actor*. Keeping numerical information in probabilistic form better reflects this intuition. In the second step, we consider how to aggregate all neighborhood information of the target entity e. For a target entity e, we define its subject neighborhood as $N_s = \{(s, p)|(s, p, e) \in G\}$ and its object neighborhood as $N_o = \{(p, o)|(e, p, o) \in G\}$. Similar to SDType++, the information aggregated by each fact triple can be regarded as a vote process, so we adopt a summation strategy when aggregating information in the second step. Our goal is still to calculate the entity-type matrix M. The element in M is the score of the target entity e on each type which is calculated as follows:

$$\begin{aligned} M[e][t] = & \sum_{(s,p)\in N_s} \frac{1}{|T(s)|} \sum_{t_s \in T(s)} p((t_s, p, t)|(t_s, p)) \\ & + \sum_{(p,o)\in N_o} \frac{1}{|T(o)|} \sum_{t_o \in T(o)} p((t, p, t_o)|(p, t_o)) \end{aligned} \tag{4}$$

where $|T(e)|$ represents the number of known type declaration of the entity e. $1/|T(e)|$ is the average factor.

Type co-occurrence information Furthermore, we also consider the global type co-occurrence. As mentioned in [20], 10% of entities in the FB15k dataset have the type */music/artist* but do not have the type */people/person* in the Freebase. Another perspective to the problem, most of entities having type */music/artist* in the dataset should also have type */people/person*. Formally, if a large amount of entities have type X and type Y simultaneously, then for a target entity that has type X, we think that it is likely to have type Y. We add a predicate *co-occurrence*for each entity in KG (as shown in Figure 1) to mine type co-occurrence information. The triple (e, *co-occurrence*, e) will derive lots of type triple (t_{e1}, *co-occurrence*, t_{e2}), where $t_{e1}, t_{e2} \in T(e)$. In order to explore the importance of neighborhood information and types co-occurrence information, we set a weight parameter ω. Equation (4) is updated as follows:

$$\begin{aligned} M[e][t] = & \sum_{(s,p)\in N_s} \frac{W_p}{|T(s)|} \sum_{t_s \in T(s)} p((t_s, p, t)|(t_s, p)) \\ & + \sum_{(p,o)\in N_o} \frac{W_p}{|T(o)|} \sum_{t_o \in T(o)} p((t, p, t_o)|(p, t_o)) \end{aligned} \tag{5}$$

where $W_p = \omega$ if p is *co-occurrence*, otherwise $W_p = 1$.

SDType++ is an extension of SDType, extending the qualitative study of entity types associated with predicate relations in SDType to a quantitative study. Further, PIANO not only relies on predicate relationship information, but also adds information on neighbouring entity types as constraints to predict the constraints of the target entity. Overall, PIANO

is an extension and generalisation of SDType and SDType++, which aims to improve the performance of prediction while retaining the efficiency of the statistical-based approach.

3.3 Complexity analysis

The algorithms SDType++ and PIANO both consist **data statistics stage** and **information aggregate stage**, we will analyze the time complexity of these two stages separately.

Data statistics stage Obviously, it only needs to traverse G_t once to get the required statistics. Therefore, the time complexity of the data statistics stage is $O(|G_t|)$. Extremely, $|G_t|$ may be $|T|^2 \times |G|$. It is worth to discuss whether there will be a quadratic explosion in the number of type triples in G_t. To eliminate this doubt, Figure 2 show the distribution of existing types number of entities in FB15kET and YAGO43kET. It can be clearly seen that the maximum number of types in the two datasets is about 100, and the number of types of most entities is below 20. This phenomenon is in line with the law of the real world, that is, a large number of entities can be classified with a few type catalogs. We denote the average types of all entities as α, $\alpha \ll |T|$. Hence, the time complexity of the data statistics stage is $O(\alpha^2 \times |G|)$.

Information aggregate stage For each entity, it needs to traverse all associated fact triples in the process of aggregating its neighborhood information. Furthermore, every time the fact triples are traversed, all possible types of the target entity need to be queried. Combined with the phenomenon in Figure 2, the time complexity of the type inference stage is $O(|G| \times |T| \times \alpha)$. Note that there is $|G| \gg |T|$ for the knowledge graph of the real world. Let's focus on the time complexity of the SDType++ algorithm. The SDType++ algorithm simplifies the process of aggregating neighborhood information into a matrix multiplication operation. According to the dimensions of the matrices, the time complexity of the SDType++ at this

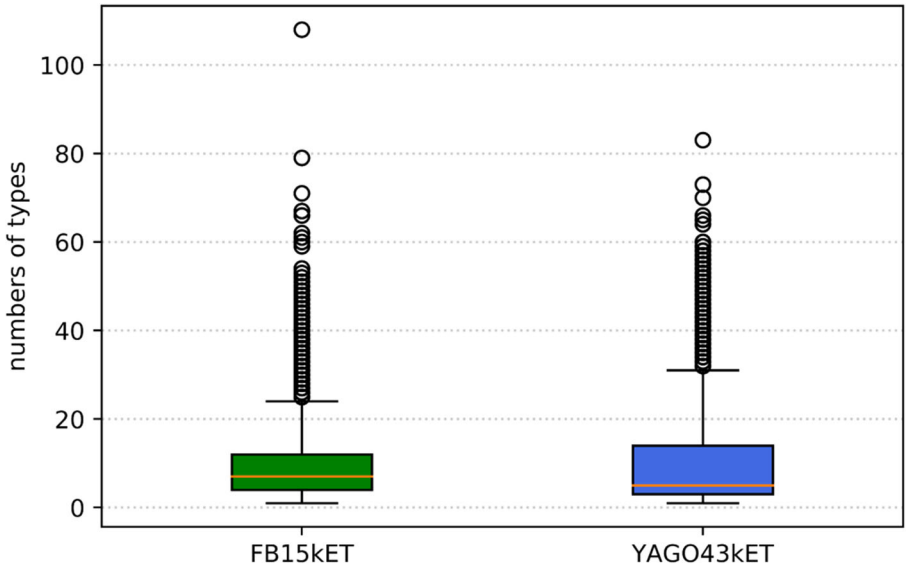

Figure 2 Distribution of types in FB15kET and YAGO43kET

stage can be calculated as $O(|E| \times |R| \times |T|)$. Although it seems that the complexity has increased (due to sparseness, usually, $|E| \times |R| > |G|$), the SDType++ algorithm is faster in actual calculations benefiting from mature matrix calculation tools.

In view of the fact that the value of α is usually very small, type inference stage is the main time cost of the two algorithms. In summary, we can deduce that the total time complexity of the SDType++ algorithm is $O(|E| \times |R| \times |T|)$ and the total time complexity of the PIANO is $O(|G| \times |T| \times \alpha)$.

4 Experiments

4.1 Datasets

For evaluation, we use two real-world datasets widely used in KGE literature which contain thousands of entity types. Each dataset contains a large number of fact triples (s, p, o) and type pairs (e, t_e), and is divided into training set, validation set and test set. Considering the difference between statistics-based algorithms and the KGE-based models, we only use the training sets as the prior knowledge for fairness, and use the validation set and the test set to evaluate the models or algorithm. We did not follow datasets that are widely used in the semantic web community, such as DBpedia and Histmunic [25]. Because these datasets contain relatively few types, it is not conducive to the evaluation of the KGET task. Table 2 shows the statistics of used datasets. The introduction of datasets is as follows:

FB15kET [20] is a subset of Freebase [26] which is a large fraction of content describing facts about movies, actors, awards, sports, and sport teams.

YAGO43kET [20] is a subset of YAGO [27] whose triples deal with descriptive attributes of people, such as citizenship, gender, and profession.

4.2 Knowledge graph entity typing task

This task aims to complete the entity-type pairs (e, t_e) when the types of the entity are missing.

Protocol We use the same experimental protocol as [9] and [20] described. For each entity-type pair (e, t_e) in the test set, experiment uses all type instances in T for replacement and obtains candidate pairs set $C = \{(e, t'_e)|t'_e \in T\}$. The scores of all pairs are then calculated

Table 2 Statistics of used datasets

Dataset	FB15kET	YAGO43kET
#Entity	14,951	42,335
#Predicate	1,345	37
#Type	3,851	45,182
#Train.triples	483,142	331,687
#Valid.triples	50,000	29,599
#Test.triples	59,071	29,593
#Train.tuples	136,618	375,853
#Valid.tuples	15,749	42,739
#Test.tuples	15,780	42,750

by energy function. Rank C according to the scores and get the ranking of (e, t_e). It looks like link prediction task. Finally, according to the ranking of correct pairs in the test set, we calculate the mean reciprocal rank (MRR) and the proportion of correct pairs ranked in the top n (H@n). Different from the experimental protocol in the previous literature, PIANO directly calculates the entity-type matrix M. The elements in the M are the scores of entity-type pairs, so we only need to sort the type list of each entity without replacing type instances. Considering that some pairs in C may be the other correct pairs in the training or validation set, the ranking obtained may be unreasonable. Such the setting is called 'Raw' in [15]. The setting to filter other all correct pairs before ranking is called 'Filter'. We only report the experimental results with 'Filter' setting in this paper.

$$MRR = \frac{1}{|Test|}\sum_{i=1}^{|Test|}\frac{1}{rank_i} \tag{6}$$

$$H@n = \frac{1}{|Test|}\sum_{i=1}^{|Test|} x_i \tag{7}$$

where $|Test|$ represents the number of entity-type pairs in the test set, $rank_i$ is the ranking of the i^{th} true entity-type pair, and $x_i = 1$ if $rank_i \leq n$, otherwise $x_i = 0$.

Parameter setting PIANO has only one parameter ω to adjust the weight of neighborhood information and global types co-occurrence information. We search for ω in the range of {0.1, 0.3, 1, 3, 10, 30, 100, 300, 1000} and select ω based on the H@10 of the validation set. Finally, the optimal ω is 30 for FB15kET, and 300 for YAGO43kET. Figure 3 shows the values of H@10 is taken for different values of ω on FB15kET and YAGO43kET.

Experimental results Table 3 shows evaluation of different models on FB15kET and YAGO43kET. The KGE-based models and the statistics-based algorithms are distinguished by a horizontal line, the upper part is the results based on the KGE models, and the lower part is the results of the statistics-based algorithms. For statistics-based algorithms, the improved SDType++ significantly outperforms SDType. This suggests that it is necessary to consider that the target entity may be associated with the same predicate multiple times. Remarkably, among the statistics-based algorithms, PIANO achieves the best results. This indicates that using the neighbor entity type information can effectively help the PIANO algorithm to perform type inference on the target entity.

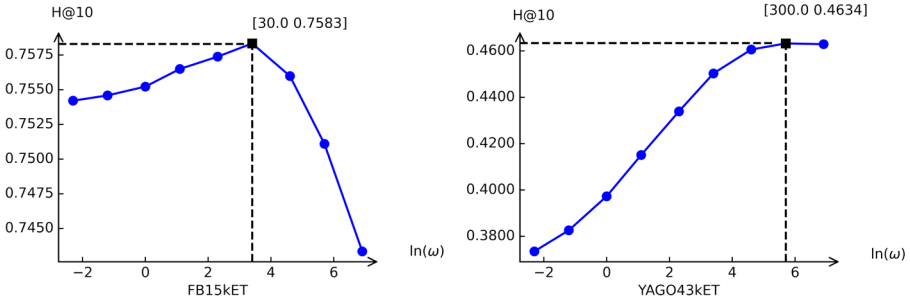

Figure 3 Line chart of H@10 for different values of ω in different datasets

Table 3 Entity type prediction results

Dataset		FB15kET				YAGO43kET			
METRICS		MRR	H@1	H@3	H@10	MRR	H@1	H@3	H@10
KGE-based	RESCAL [17]	0.190	0.097	0.196	0.376	0.080	0.042	0.083	0.153
	HOLE [16]	0.220	0.133	0.234	0.382	0.160	0.090	0.173	0.293
	TransE [15]	0.450	0.315	0.515	0.739	0.210	0.126	0.232	0.389
	ETE [20]	0.500	0.385	0.553	0.719	0.230	0.137	0.263	0.422
	ConnectE [9]	0.590	0.496	0.643	0.799	0.280	0.160	0.309	0.479
	ConnectE-MRGAT [22]	0.630	0.562	0.662	0.804	0.320	0.243	0.343	0.482
	AttEt [23]	0.620	0.517	0.677	0.821	0.350	0.244	0.413	0.565
	CET [21]	**0.697**	**0.613**	**0.745**	**0.856**	**0.503**	**0.398**	**0.567**	**0.696**
Stastistical-based	SDType [11]	0.096	0.029	0.079	0.221	0.003	0.0006	0.002	0.005
	SDType++(ours)	0.496	0.408	0.537	0.682	0.114	0.087	0.114	0.158
	PIANO(ours)	**0.568**	**0.471**	**0.615**	**0.764**	**0.315**	**0.238**	**0.337**	**0.464**

Evaluation of different models/algorithms on FB15kET and YAGO43kET. The bold values represent the best results in KGE-based models and stastistical-based models, respectively

Compared to KGE-based models, the PIANO algorithm outperforms most models. However, there is still a significant gap with few KGE-based models, especially CET. CET achieves the best results on all evaluation metrics on both datasets. We think there may be two reasons: i) CET also adopts the idea of aggregating neighborhood information, which shows that neighborhood information can help infer the type of target entity. ii) The KGE-based model maps entities and types into the semantic space and is able to better capture the semantic associations between entities and types. Unlike KGE-based approaches that seek higher prediction accuracy, PIANO is motivated from a practical point of view that in addition to seeking higher prediction accuracy, maintaining higher efficiency and sustaining incremental properties is more meaningful for a continuously updated knowledge graph. The experimental results show that the FB15kET results are better than those on the YAGO43kET. The difference between the two datasets leads to the difference in the experimental results. Compared with FB15kET, YAGO43kET contains more kinds of entity types and entities, but fewer kinds of predicates, i.e. YAGO43kET is more sparse than FB15kET, which makes the YAGO43kET dataset more challenging for the KGET task.

Ablation experiment and parameter sensitivity PIANO has only one parameter of ω. We attempt to explore the influence of neighborhood information and global types co-occurrence information of the PIANO. We compare the results of using the neighborhood information, using the co-occurrence information of global types, and the optimal parameter ω in Table 4, respectively. It can be seen that the PIANO with optimal ω achieves best results on both datasets.

We further analyse the sensitivity of parameter ω in PIANO based on the H@10 of the validation set in different datasets in Figure 3. In essence, the larger the value of ω, the more important the role of type co-occurrence information for PIANO. For FB15kET, the performance of PIANO increases with the increase of ω at first, and then decreases with the increase of ω value after reaching the optimal performance. For YAGO43kET, the

Table 4 Results of ablation experiment

	FB15kET		YAGO43kET	
	MRR	H@10	MRR	H@10
Neighbor information only	0.569	0.761	0.254	0.398
Type co-occurrence only	0.520	0.740	0.310	0.461
PIANO	**0.569**	**0.764**	**0.315**	**0.464**

The bold values represent the best results

performance of PIANO improves with the increase of the ω value, but eventually tends to be stable.

Combined with the analysis in Table 4 and Figure 3, we argue that the PIANO algorithm relies more on type co-occurrence information on the dataset YAGO43kET where the neighborhood information is more sparse. In contrast, on the dataset FB15kET with richer neighborhood information, excessive reliance on type co-occurrence information will degrade the performance of the algorithm. Neighborhood information and type co-occurrence information can complement each other to further improve the performance of the algorithm.

Case analysis Since PIANO uses statistics to infer types, the algorithm is highly interpretable. Table 5 shows the details in Figure 1 of inferring the types of *Raza Murad* relying on fact triple (Jodhaa Akbar, *starring*, Raza Murad) and global statistics. Each element in the table represents the calculated conditional probability (the number in parentheses indicates the predicted 'Raw' ranking by corresponding type triple). The last line averages the calculated conditional probabilities of all type triples derived from the fact triple. As we expected, PIANO lowered the predicted ranking of tv *actor* to the 4 from 1 and raised the predicted ranking of *film actor* ranking from 3 to 2. Compared with Table 1, the ranking results are more reasonable and more interpretable. We also show the prediction details of the type in the test set *influence node* and the type *entity in film* which is correct but does not exist in dataset.

Further, Table 6 shows the detail of aggregating neighborhood information of Figure 1. It can be found that almost all neighborhood information provides *person* information. As we expected, the film *Jodhaa Akbar* provides the most information for *film actor*. Unexpectedly, relying on *nationality* relations can also achieve close prediction results. We argue there may be bias issues in the dataset, which may lead a coarse-grained predictor can also achieve a not bad results.

4.3 Incremental inference experiment

In the real world, the KGs are not only large-scale, but also constantly dynamically updated. It is very likely that a large number of new entities that have never existed in the knowledge graph will be linked into the knowledge graph. How to predict the types of these new entities has become a question that remains to be explored.

When adding a new entity, the embedding-based models encounters a great obstacle to predicting type instances of the entity since they never trained the embedding of the new entity. In addition, the embedding-based model training process is often very time-consuming. Therefore, it is unrealistic to retrain the model once adding a new entity. In the

Table 5 The details of inferring the types of *Raza Murad* by the triple (*Jodhaa Akbar*, *starring*, *Raza Murad*) in Figure 1

Types of neighbor entity	Types of target entity					
	Types in dataset				Types not in dataset	
	Person	Award nominee	Film actor	Incluence node	Tv actor	Entity in film
Noinated work	0.054(2)	0.050(4)	0.053(3)	0.044(6)	0.054(1)	0.044(5)
Bollywood film	0.095(1)	0.082(4)	0.089(2)	0.082(3)	0.072(6)	0.058(7)
Film fare awards	0.097(1)	0.085(3)	0.091(2)	0.082(4)	0.070(6)	0.060(7)
Film	0.054(2)	0.050(4)	0.053(3)	0.044(5)	0.054(1)	0.044(6)
Netflix title	0.054(2)	0.050(4)	0.053(3)	0.044(6)	0.054(1)	0.044(5)
Topic	0.089(2)	0.085(3)	0.091(1)	0.071(4)	0.071(5)	0.062(7)
Winning work	0.054(2)	0.051(4)	0.053(3)	0.044(7)	0.054(1)	0.045(5)
Average	0.071(1)	0.065(3)	0.069(2)	0.058(7)	0.061(5)	0.051(6)

same scenario, PIANO could infer types of the new entity based on the existing statistical information and the entity-relation pairs connected with the new entity. If the degree of the new entity is d, then the complexity of this process is $O(d)$. Next, we will simulate real scenes to predict the types of these entities newly linked into the knowledge graph.

Protocol Suppose there is an existing KG, and now a new entity needs to be linked into the KG through some relations. Because it is a new entity, we know nothing about it. That is, we don't have its embedding and any types. Therefore, it is challenging to predict its type instances. We implement incremental inference experiment on FB15kET and YAGO43kET. In order to simulate such real scenes, we make a few changes to datasets. We use FB15kET and FB15k as an example to illustrate the transformation process, and the same on YAGO43k and YAGO43kET.

First, for all entities having types in the test set of FB15kET, we move all of their other entity-type pairs from the training set and the validation set to the test set. Analogously, for all entities having types in the modified validation set, we move the entity-type pairs from the training set to the validation set. We call the modified dataset FB15kET-I. In particular,

Table 6 The details of inferring the types of *Raza Murad* by the information of the neighborhood in Figure 1

Query of neighborhood	Types of target entity					
	Types in dataset				Types not in dataset	
	Person	Award nominee	Film actor	Incluence node	Tv actor	Entity in film
(Jodhaa Akbar,starring,?)	0.071(1)	0.065(3)	0.069(2)	0.058(7)	0.061(5)	0.051(6)
(?,performance,Jodhaa Akbar)	0.072(1)	0.067(3)	0.070(2)	0.064(4)	0.062(5)	0.048(7)
(?,nationality, Indian)	0.075(1)	0.057(3)	0.062(2)	0.047(5)	0.041(6)	0.037(7)
(?,profession, actor)	0.055(1)	0.048(3)	0.050(2)	0.036(7)	0.046(4)	0.039(6)
Sum	0.273(1)	0.236(3)	0.251(2)	0.205(5)	0.210(4)	0.175(7)

Table 7 Statistics of datasets after transformation

Dataset	FB15kET-I	YAGO43kET-I
#Entity	14,951	42,335
#Predicate	1,345	37
#Type	3,851	45,405
#prior.triples	23,440	16,263
#expa.triples	37,070	25,690
#eval.triples	136,839	113,692
#prior.tuples	17,829	49,762
#expa.tuples	24,356	52,218
#eval.tuples	118,231	330,615

we refer to the modified training set, validation set, and test set as the prior set, expansion set, and evaluation set, respectively.

Second, for each fact triple in training set of FB15k, we consider the following three cases according to its head entity and tail entity: (1) both the two entities are in the prior set of FB15kET-I; (2) one entity is in the prior set of FB15kET-I, and the other one is in the expansion set; (3) one entity is in the prior set of FB15kET-I, and the other one is in the evaluation set. The prior set, expansion set, and evaluation set of FB15k-I are composed of fact triples that line with the (1), (2), and (3), respectively.

Finally, we remove all entities, which are not in FB15k-I, and their related entity-type pairs of FB15kET-I. After processing, we use the prior set of FB15k-I and FB15kET-I to calculate prior statistical information, use the expansion set to simulate the ever-increasing data, and use the evaluation set to evaluate the experimental results. It is ensured that the entities in the expansion set and evaluation set of FB15kET-I are completely unknown before inference.

Table 7 shows the statistics of datasets after transformation.

Infer types for New Entities As a comparison, we design a process for ConnectE [9] and CET [21]to infer types of new entity. ConnectE utilizes TransE for training on fact triples to obtain entity and predicate embeddings. In order to better fit the ConnectE, we use the idea of TransE [15], i.e. $\mathbf{h+r} \approx \mathbf{t}$, to obtain the embedding of new entities. Specifically, we utilize all the fact triples connected by the target entity and TransE to obtain a series of generated embeddings. Average pooling of these generated entity embeddings is performed to obtain embeddings for new entities. With these embeddings of new entities, ConnectE can predict their types. Since CET includes a computational process utilizing neighborhood information, we utilize the N2T mechanism and Agg2T mechanism [21]in CET to directly perform type inference on the target entity. We also report the results of SDType and SDType++ predicting new entity types. For PIANO, we aggregate the neighborhood information of the new entities to implement the inference process.

Experimental results The experimental results under the new experimental protocol are shown in Table 8. In such a real scene, the performance of all methods are greatly reduced. It shows that inferring types for new entities is a challenging task. Through comparison, it can be found that the performance of PIANO is more stable, and the best results are obtained under the new experimental protocol. It is worth noting that the performance of CET under

Table 8 Results of inferring types of new entities

Dataset	FB15kET-I				YAGO43kET-I			
Metrics	MRR	H@1	H@3	H@10	MRR	H@1	H@3	H@10
ConnectE	0.39	0.288	0.425	0.605	0.075	0.054	0.080	0.117
CET	0.366	0.282	0.397	0.538	0.010	0.008	0.011	0.014
SDType	0.308	0.179	0.348	0.599	0.010	0.003	0.011	0.019
SDType++	0.417	0.331	0.441	0.597	0.086	0.065	0.087	0.122
PIANO	**0.446**	**0.362**	**0.482**	**0.620**	**0.144**	**0.103**	**0.153**	**0.222**

The bold values represent the best results

this experimental protocol degrades sharply and is even inferior to that of ConnectE. This is in stark contrast to the performance of CET in achieving SOTA results in Table 3 on the ordinary KGET task. We think there are mainly two possible reasons: i) KGE-based models rely on trained entity embeddings for type inference when performing the KGET task. When performing type inference on new entities, the model cannot utilize the information of the new entities, resulting in degraded inference performance. ii) Comprehensive consideration of Table 4 and Figure 3, the algorithm may over-rely on the co-occurrence information of entity types. We believe that CET may overfit during training, making the model over-rely on type co-occurrence information while ignoring the role of neighborhood information. Hence, it is difficult for CET to effectively apply the neighborhood information of the target entity for type inference.

In addition, we consider utilizing continuously updated data to improve the performance of PIANO. This process is unrealistic for KGE-based models, because the models must be retrained once new data is added. However, the retraining processes are extremely cumbersome and require a lot of expensive computing resources, while PIANO only needs to update the statistics.

We attempt to add 100 entities of the expansion sets in turn, and connect the corresponding triples to update the statistics. Every time we add new data, we re-evaluate the results on the evaluation set. We compare the performance changes of SDType, SDType++ and PIANO in the process of incremental data addition. Figures 4 and 5 show the performance changes of the three algorithms on FB15ET-I and YAGO43kET-I, respectively. In Figure 4, we can see that the performance of SDType decreases during the process of increasing data. We think that the process of adding data has associated more predicates to entities in the prior set, and that these predicates may confuse SDType. Although SDType++ and SDType

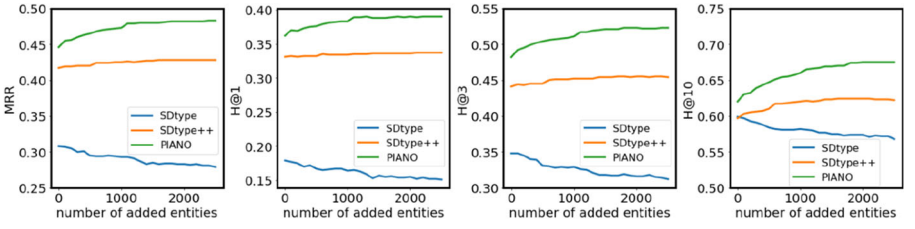

Figure 4 Comparison of H@1/3/10 and MRR improvements on the test set for the SDType, SDType++ and PIANO with adding new data on FB15kET-I

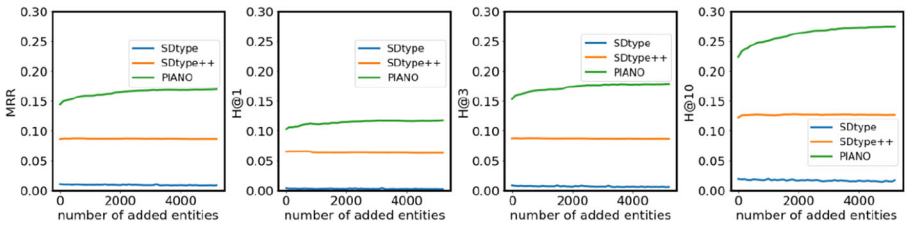

Figure 5 Comparison of H@1/3/10 and MRR improvements on the test set for the SDType, SDType++ and PIANO with adding new data on YAGO43kET-I

go through the same process, SDType++ can benefit from newly added data by distinguishing the number of predicates associated with the target entities. Compared with SDType++, which only has obvious incremental effect on H@10 indicators, PIANO can significantly benefit from the new data in all indicators. This phenomenon is more obvious in Figure 5, where the type data of YAGO43kET-I is numerous and sparse. In Figure 5, PIANO can still improve performance as new data is added, while SDType and SDType++ are almost insensitive. It shows that the addition of predicate data can bring gains to algorithms, but the gain is limited. The addition of the type information of neighborhood can continuously improve the ability of the algorithm to infer types of target entities.

In conclusion, although SDType has an incremental nature in theory, it encounters obstacles in a wide variety of data types. SDType++ alleviates the challenges SDType faces in incremental experimentation. PIANO has superior incremental properties and can continuously benefit from new data to improve performance.

Efficiency analysis of PIANO To measure the efficiency of PIANO, we compare the training time of the KGE-based methods CET, ConnectE and the computational time of statistics-based method PIANO in incremental experiments. Since the KGE-based methods need to

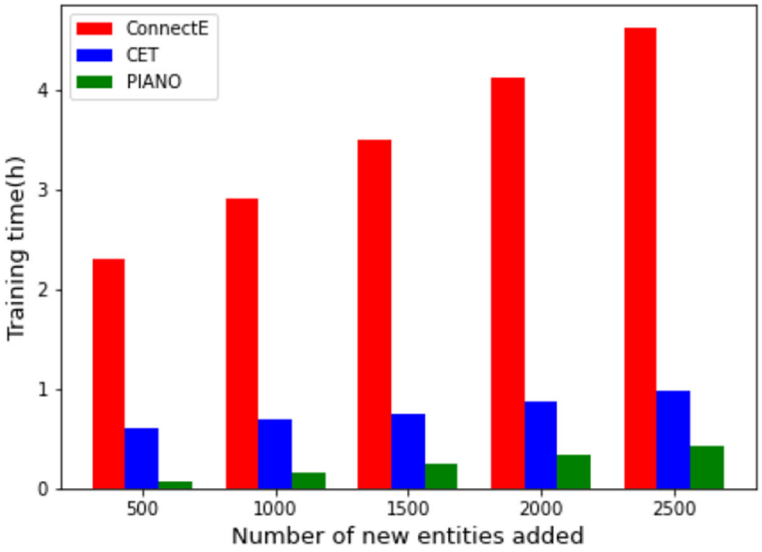

Figure 6 Comparison of training time for ConnectE, CET and PIANO with adding new data on FB15KET-I

retrain the models after adding new entities, we consider five subsets of FB15KET-I with the sizes of 500, 1000, 1500, 2000 and 2500, respectively. We use these five datasets as the train sets to train models and report the results in Figure 6. The models CET and ConnectE are conducted with one Intel(R) Core(TM) i7-8700 CPU and one Nvidia A100 GPU, while PIANO are only conducted with one Intel(R) Core(TM) i7-8700 CPU. From the Figure 6, we can observe that the computational time of PIANO is much smaller than the training time of the models CET and ConnectE, especially the model ConnectE, which further indicates the efficiency of our model.

5 Conclusion

In this paper, we propose a statistics-based knowledge graph entity typing algorithm **PIANO**. The algorithm performs type inference by aggregating the neighborhood information and type co-occurrence information of the target entity. The experimental results show that the PIANO algorithm achieves performance that can compete with the KGE-based models. We find that neighborhood information plays a better role in dense data, while type co-occurrence information plays a more important role in sparse data.

We also design incremental experiments to simulate the knowledge graph update process in real-world scenarios. The experimental results show that all the algorithms encounter a dramatic decline in performance when predicting types of new entities. Nevertheless, the PIANO algorithm retains a relatively stable performance. Experiments also verify that PIANO algorithm can improve the performance according to the continually added data, that is, the algorithm has outstanding incremental property.

Acknowledgements Thanks for the computing resources provided by the Supercomputing Center of Lanzhou University.

Author Contributions All authors contributed to the study conception and design. Material preparation, data collection and analysis were performed by Zepeng Li, Rikui Huang, Zhenwen Zhang and Bin Hu; The first draft of the manuscript was written by Zepeng Li, Rikui Huang and Minyu Zhai and all authors commented on previous versions of the manuscript. All authors read and approved the final manuscript.

Funding This work is supported by the National Key Research and Development Program of China (Grant No. 2021YFF1201203), the National Natural Science Foundation of China (Grant No. 62227807), the Natural Science Foundation of Gansu Province (Grant No. 20JR10RA605), and the Fundamental Research Funds for the Central Universities (lzujbky-2021-66).

Declarations

Consent for Publication All authors have checked the manuscript and have agreed to the submission.

Conflict of Interests The authors have no relevant financial or non-financial interests to disclose.

References

1. Guo, Q., Zhuang, F., Qin, C., Zhu, H., Xie, X., Xiong, H., He, Q.: A survey on knowledge graph-based recommender systems. IEEE Transactions on Knowledge and Data Engineering (Early Access) (2020)
2. Xiong, H., Wang, S., Tang, M., Wang, L., Lin, X.: Knowledge graph question answering with semantic oriented fusion model. Knowl.-Based Syst. **221**, 106954 (2021)
3. Ji, S., Pan, S., Cambria, E., Marttinen, P., Philip, S.Y.: A survey on knowledge graphs: Representation, acquisition, and applications. IEEE Trans. Neural Netw. Learn. Syst. 1–21 (2021)

4. Sun, Y., Wang, S., Feng, S., Ding, S., Pang, C., Shang, J., Liu, J., Chen, X., Zhao, Y., Lu, Y., Liu, W., Wu, Z., Gong, W., Liang, J., Shang, Z., Sun, P., Liu, W., Ouyang, X., Yu, D., Tian, H., Wu, H., Wang, H.: ERNIE 3.0: Large-scale Knowledge Enhanced Pre-training for Language Understanding and Generation (2021)
5. Hamilton, W.L., Ying, R., Leskovec, J.: Inductive representation learning on large graphs. In: Annual Conference on Neural Information Processing Systems (NeurIPS), pp. 1025–1035 (2017)
6. Teru, K., Denis, E., Hamilton, W.: Inductive relation prediction by subgraph reasoning. In: International Conference on Machine Learning (ICML), pp. 9448–9457 (2020)
7. Zhang, Y., Wang, W., Chen, W., Xu, J., Liu, A., Zhao, L.: Meta-learning based hyper-relation feature modeling for out-of-knowledge-base embedding. In: Proceedings of the 30th ACM International Conference on Information & Knowledge Management (CIKM), pp. 2637–2646 (2021)
8. Chen, M., Zhang, W., Zhu, Y., Zhou, H., Yuan, Z., Xu, C., Chen, H.: Meta-knowledge transfer for inductive knowledge graph embedding. arXiv:2110.14170 (2022)
9. Zhao, Y., Zhang, A., Xie, R., Liu, K., Wang, X.: Connecting embeddings for knowledge graph entity typing. In: Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL), pp. 6419–6428 (2020)
10. Kellou-Menouer, K., Kardoulakis, N., Troullinou, G., Kedad, Z., Plexousakis, D., Kondylakis, H.: A survey on semantic schema discovery. The VLDB J.,1–36 (2021)
11. Paulheim, H., Bizer, C.: Type inference on noisy RDF data. In: International Semantic Web Conference, pp. 510–525. Springer (2013)
12. Krompaß, D., Baier, S., Tresp, V.: Type-constrained representation learning in knowledge graphs. In: Proceedings of the International Semantic Web Conference (ISWC), pp. 640–655 (2015)
13. Krompaß, D., Nickel, M., Tresp, V.: Large-scale factorization of type-constrained multi-relational data. In: Proceedings of the International Conference on Data Science and Advanced Analytics (DSAA), pp. 18–24 (2014)
14. Fang, L., Miao, Q., Meng, Y.: Dbpedia entity type inference using categories. In: International Semantic Web Conference (Posters & Demos) (2016)
15. Bordes, A., Usunier, N., Garcia-Duran, A., Weston, J., Yakhnenko, O.: Translating embeddings for modeling multi-relational data. Adv. Neural Inf. Process. Syst. **26**, 2787–2795 (2013)
16. Nickel, M., Rosasco, L., Poggio, T.: Holographic embeddings of knowledge graphs. In: Proceedings of the AAAI Conference on Artificial Intelligence (AAAI), vol. 30, pp. 1955–1961 (2016)
17. Nickel, M., Tresp, V., Kriegel, H.-P.: A three-way model for collective learning on multi-relational data. In: Proceedings of the International Conference on Machine Learning (ICML), pp. 809–816 (2011)
18. Dettmers, T., Minervini, P., Stenetorp, P., Riedel, S.: Convolutional 2D knowledge graph embeddings. In: Proceedings of the AAAI Conference on Artificial Intelligence (AAAI), vol. 32, pp. 1811–1818 (2018)
19. Sun, Z., Deng, Z.-H., Nie, J.-Y., Tang, J.: Rotate: Knowledge graph embedding by relational rotation in complex space. In: International Conference on Learning Representations (ICLR) (2019)
20. Moon, C., Jones, P., Samatova, N.F.: Learning entity type embeddings for knowledge graph completion. In: Proceedings of the International Conference on Information and Knowledge Management (CIKM), pp. 2215–2218 (2017)
21. Pan, W., Wei, W., Mao, X.-L.: Context-aware entity typing in knowledge graphs. In: Findings of the Association for Computational Linguistics: EMNLP 2021, pp. 2240–2250 (2021)
22. Zhao, Y., Zhou, H., Zhang, A., Xie, R., Li, Q., Zhuang, F.: Connecting embeddings based on multiplex relational graph attention networks for knowledge graph entity typing. IEEE Trans. Knowl. Data Eng., 1–1. https://doi.org/10.1109/TKDE.2022.3142056 (2022)
23. Zhuo, J., Zhu, Q., Yue, Y., Zhao, Y., Han, W.: A neighborhood-attention fine-grained entity typing for knowledge graph completion. In: Proceedings of the 15th ACM International Conference on Web Search and Data Mining. WSDM '22, pp. 1525–1533. Association for Computing Machinery (2022). https://doi.org/10.1145/3488560.3498395
24. Zou, C., An, J., Li, G.: Knowledge graph entity type prediction with relational aggregation graph attention network. In: The Semantic Web: 19th International Conference, ESWC 2022, Hersonissos, Crete, Greece, May 29 – June 2, 2022, Proceedings, pp. 39–55. Springer (2022). https://doi.org/10.1007/978-3-031-06981-9_3
25. Kardoulakis, N., Kellou-Menouer, K., Troullinou, G., Kedad, Z., Plexousakis, D., Kondylakis, H.: Hint: Hybrid and incremental type discovery for large RDF data sources. In: 33rd International Conference on Scientific and Statistical Database Management, pp. 97–108 (2021)
26. Bollacker, K., Evans, C., Paritosh, P., Sturge, T., Taylor, J.: Freebase: a collaboratively created graph database for structuring human knowledge. In: Proceedings of the International Conference on Management of Data (SIGMOD), pp. 1247–1250 (2008)

27. Suchanek, F.M., Kasneci, G., Weikum, G.: Yago: a core of semantic knowledge. In: Proceedings of the World Wide Web Conference (WWW), pp. 697–706 (2007)

Affiliations

Zepeng Li[1] · Rikui Huang[1] · Minyu Zhai[1] · Zhenwen Zhang[1] · Bin Hu[1,2,3]

Zepeng Li
lizp@lzu.edu.cn

Rikui Huang
huangrk19@lzu.edu.cn

Minyu Zhai
zhaimy21@lzu.edu.cn

Zhenwen Zhang
zhangzhw20@lzu.edu.cn

1 Gansu Provincial Key Laboratory of Wearable Computing, School of Information Science and Engineering, Lanzhou University, Lanzhou, 730000, Gansu, China

2 School of Medical Technology, Beijing Institute of Technology, Beijing, 100081, Beijing, China

3 CAS Center for Excellence in Brain Science and Intelligence Technology, Shanghai Institutes for Biological Sciences, Chinese Academy of Sciences, Shanghai, 200000, Shanghai, China