Fast and Slow Thinking: A Two-Step Schema-Aware Approach for Instance Completion in Knowledge Graphs

Dinggi Yang, Bingging Qu, Paolo Rosso, and Philippe Cudre-Mauroux

Abstract—Modern Knowledge Graphs (KG) often suffer from an incompleteness issue (i.e., missing facts). By representing a fact as a triplet (h, r, t) linking two entities h and t via a relation r, existing KG completion approaches mostly consider a link prediction task to solve this problem, i.e., given two elements of a triplet predicting the missing one, such as (h, r, ?). However, this task implicitly has a strong yet impractical assumption on the two given elements in a triplet, which have to be correlated, resulting otherwise in meaningless predictions, such as (Marie Curie, headquarters location, ?). Against this background, this paper studies an instance completion task suggesting r-t pairs for a given h, i.e., (h,?,?). Inspired by the human psychological principle "fast-and-slow thinking", we propose a two-step schema-aware approach RETA++ to efficiently solve our instance completion problem. It consists of two components: a fast RETA-Filter efficiently filtering candidate r-t pairs schematically matching the given h, and a deliberate RETA-Grader leveraging a KG embedding model scoring each candidate r-t pair considering the plausibility of both the input triplet and its corresponding schema. RETA++ systematically integrates them by training RETA-Grader on the reduced solution space output by RETA-Filter via a customized negative sampling process, so as to fully benefit from the efficiency of RETA-Filter in solution space reduction and the deliberation of RETA-Grader in scoring candidate triplets. We evaluate our approach against a sizable collection of state-of-the-art techniques on three real-world KG datasets. Results show that RETA-Filter can efficiently reduce the solution space for the instance completion task, outperforming best baseline techniques by 10.61%-84.75% on the reduced solution space size, while also being 1.7x-29.6x faster than these techniques. Moreover, RETA-Grader trained on the reduced solution space also significantly outperforms the best state-of-the-art techniques on the instance completion task by 31.90%-105.02%.

Index Terms—Knowledge graph embedding, Entity types, Instance completion, Fast and slow thinking

1 INTRODUCTION

K Nowledge Graphs (KGs), such as Wikidata [1], Free-base [2], or Google's Knowledge Graph [3], have become a key resource enabling a wide range of Web applications, including semantic search [4], question-answering [5], and recommender systems [6], etc. A typical KG contains a large collection of entities (representing real-world objects or abstract concepts) interconnected via relations, where entities contribute to the description of other entities via relations. Using a widely adopted triplet representation scheme [7], a KG consists of a set of triplets, (head, relation, *tail*), or (h, r, t) for short, each of which encoding a relation connecting a head entity to a tail entity, such as Marie Curie (head) sex or gender (relation) female (tail). Figure 1 shows two real-world examples of entities Marie Curie and Apple Inc. from Wikidata; each example is associated with a set of triplets having the entities as head in Wikidata. Despite modern KGs contain high-quality structured data, they are also known to suffer from an incompleteness issue, i.e., missing facts. For example, 71% of all people from Freebase have no place of birth [8], even though this is a mandatory property of the schema [9].

Manuscript received xxx; revised xxx.



Fig. 1. Examples of entities Marie Curie and Apple Inc. from Wikidata. For each entity, we present a subset of triplets having the entity as head.

To overcome this issue, Knowledge Graph completion problems have been widely studied by the current literature. These problems are mostly formulated as a link prediction task [7], [8], i.e., predicting missing links in a KG. More precisely, given two elements of a triplet, the task is to predict the missing one, such as (h, r, ?), (h, ?, t), or (?, r, t), where the question mark represents the missing entity/relation. However, despite the wide adoption of such link prediction tasks by existing work for KG completion, this task is often impractical due to its strong assumption on knowing two elements in a triplet; the two known elements in a triplet must

Dinggi Yang is with the State Key Laboratory of Internet of Things for Smart City and Department of Computer and Information Science, University of Macau, Macao SAR, China, E-mail: dingqiyang@um.edu.mo. Bingging Qu is with the BNU-HKBU United International College, China, E-mail: bingqingqu@uic.edu.cn. Paolo Rosso and Philippe Cudre-Mauroux are with the University of Fribourg, Switzerland, E-mail: paolo@exascale.info, philippe.cudre-mauroux@unifr.ch.

somehow be correlated in order to make a meaningful prediction, otherwise the task will result in meaningless results. For example, the predictions for (Marie Curie, headquarters *location*, ?) or (*Apple Inc., sex or gender*, ?) are nonsense, while they would both be considered as valid input for the link prediction task. Although existing work on link prediction implement this task by taking out one element from a true triplet in a KG and then making predictions on the triplet (which implicitly ensures the correlation between the two remaining elements in the triplet), such an experimental setting indeed departs from real-world use cases where we are not given two correlated elements in a triplet. Another thread of work on KG completion focuses on the relation prediction task¹, suggesting missing relations to a given entity using the information about other similar entities [13]. However, these techniques only suggest relations for a given head, without predicting the tails and hence are only part of the solution.

Toward the ultimate goal of KG completion, we tackle in this paper a more complex *instance completion* problem. Specifically, considering a (head) entity as an instance, we complete its descriptions by suggesting relation-tail pairs. In other words, we make predictions on (h, ?, ?), suggesting *r*-*t* pairs to a given *h*. Different from the link prediction task, our instance completion task has a more realistic setting without assuming knowing two correlated elements in a triplet. However, such an instance completion problem on KGs is more challenging, due to a large number of potential *r*-*t* pairs for a given *h*. Directly evaluating and ranking all combinations of r and t not only incurs a significant computation cost, but also results in poor performance due to a large number of candidate triplets to consider (a large solution space). For example, such an approach takes 350.7 hours (CPU time) even with an efficient link prediction technique TransE on our JF17K dataset, resulting in poor performance of 0.0097 for recall@5 [14]. An alternative solution to this problem is to combine the relation prediction and link prediction tasks [10]; for a given *h*, it first predicts a set of relations using a relation prediction technique, and then predicts (h, r, ?) using a link prediction technique for each predicted relation r. Although this approach can reduce the number of candidate triplets to improve the prediction efficiency, it does not fully leverage the schema information encoded by the triplets (e.g., head_type-relation-tail_type), thus resulting in a suboptimal performance as we showed in our previous work [14].

Against this background, inspired by the human psychological principle "fast-and-slow thinking", we propose a two-step schema-aware approach RETA++ to efficiently solve our instance completion problem over KGs (h, ?, ?). Specifically, fast-and-slow thinking is proposed by Nobelist Daniel Kahneman [15] in psychology and behavioral science, suggesting a dual-process model of human thinking, where fast thinking is emotional, stereotypic, and unconscious, and can process a large amount of information effortlessly, while slow thinking is effortful, logical, and conscious, and can process a small amount of information with deliberate reasoning [16]. Inspired by this two-step process, in our context, we design two distinct components mimicking the respective fast and slow thinking processes, and then systematically combine them for solving the instance completion problem over KGs. First, inspired by the fast thinking process, we design RETA-Filter, a fast Relation-Tail pair filter using the schema information of a KG to efficiently filter candidate r-t pairs schematically matching the given *h* via tensor product, and thus significantly reduce the solution space. Second, inspired by the slow thinking process, we design RETA-Grader, a deliberate Relation-Tail pair grader leveraging a KG embedding model scoring each candidate r-t pair according to the plausibility of both the input triplet and its corresponding schema. Finally, toward the ultimate goal of instance completion over KGs, we systematically integrate these two components by training RETA-Grader on the reduced solution space output by RETA-Filter via a customized negative sampling process, so as to fully benefit from the efficiency of RETA-Filter in solution space reduction and the deliberation of RETA-Grader in scoring candidate triplets. Our contributions can be summarized as follows:

- By revisiting the existing approaches to KG completion, we identify the impractical settings of the widely studied link prediction problems, and thus propose to study a novel instance completion problem with more realistic settings, i.e., predicting relation-tail pairs given a head (*h*, ?, ?).
- Inspired by the human psychological principle "fastand-slow thinking", we propose a two-step schemaaware approach RETA++ to efficiently solve our instance completion problem. It tightly combines, via a customized negative sampling process, a *fast* RETA-Filter to efficiently reduce the solution space and a *deliberate* RETA-Grader to evaluate the plausibility of the candidate *r*-*t* pairs in the reduced solution space.
- We conduct a thorough evaluation of our proposed techniques compared to a sizable collection of baselines on three real-world KG datasets. Results show that RETA-Filter is able to efficiently filter candidate *r*-*t* pairs for the instance completion task, outperforming the best baseline techniques by 10.61%-84.75% on the reduced solution space size under the same quality guarantees, while also being 1.7x-29.6x faster than these techniques. Moreover, RETA-Grader trained on the reduced solution space also significantly outperforms the best state-of-the-art link prediction techniques on the instance completion task by 31.90%-105.02%. In particular, compared to our previous work RETA, RETA++ achieves 16.9% higher performance while requiring 35% fewer training epochs at the same time.

2 RELATED WORK

Knowledge Graph completion predicts missing facts in a KG. In the following, we first briefly discuss existing work implementing one of the three tasks (i.e., link prediction, relation prediction, and instance completion) for KG completion, followed by related work on knowledge graph schema.

^{1.} This task is also known as property prediction/recommendation by [10], [11], [12], as these works refer to a triplet by (*entity,property,value*), which is equivalent to our notion of triplet as (*head, relation, tail*).

2.1 Link Prediction Task

In the current literature, KG completion has been mostly formulated as a link prediction task, i.e., predicting one missing element in a triplet, such as (h, r, ?), (h, ?, t) or (?, r, t). Early solutions often resorted to rule/feature-based relational learning, such as association rules [17] or handcrafted features [18], [19] (e.g., paths linking entities) for link prediction. Recently, KG embedding techniques have been proposed to learn latent representations of entities/relations in a KG, which can then be effectively used for link prediction tasks over the KG. These techniques can be classified into two broad categories. First, translational distance models exploit distance-based scoring functions to create embeddings. One representative model of this family is TransE [20], which learns from triplets (h, r, t) such that the relation between the head and tail is preserved as $h + r \approx t$. Several works further improve TransE to capture richer KG structures, such as involving relation-specific hyperplanes [21] or spaces [22], [23], [24], for example. Second, semantic matching models exploit similarity-based scoring functions. One typical model in that context is RESCAL [25], which represents each entity as a vector and each relation as a matrix, and uses a bilinear function to model the relation between two entities. Several works extend RESCAL by reducing the complexity of the models [26], [27] or of the training processes [28], by improving the model expressiveness [29], by capturing asymmetric relations [30], by considering unbalanced data in KGs [31], or by modeling non-linear relations using neural networks [32], [33], [34], [35], [36].

In addition, there are also a few works on link prediction combining triplets with other data. According to the sources of such data, these works can be classified into two categories, i.e., data in the KG and third-party data. On one hand, besides triplets linking entities via relations, literals [37], [38], images [39] or types [14], [40], [41], [42], [43], [44], [45], [46] associated with entities in the KG can be combined with triplets to improve the performance on link prediction. On the other hand, some other techniques learn entity/relation embeddings from triplets in a KG jointly with third-party data sources, in particular with text (e.g., Wikipedia articles) [47], [48], [49], [50].

However, we argue in this paper that such link prediction tasks are impractical as they assume knowing two of the elements in a triplet, which is often not the case in practice. Those two elements must be correlated to make meaningful predictions, which otherwise result in meaningless outputs such as (*Marie Curie, headquarters location, ?*). Therefore, we study an instance completion problem in this paper, suggesting relation-tail pairs for a given (head) entity.

2.2 Relation Prediction Task

KG completion problems have also been studied as a relation prediction task, suggesting missing relations to a given entity [10], [11], [12], [13], [51], [52]. Specifically, for a given entity, the objective is to suggest a list of relations (so-called properties by some of these works) which are relevant to the entity. For example, Zangerle et al. [12] built a "property suggester" for Wikidata, suggesting candidate properties based on association rules learnt from existing triplets in Wikidata. Lajus et al. [11] studied the problem of determining obligatory relations for a given entity in a KG by extracting and using the class hierarchy (of entities) in the KG. Cao et al. [10] designed a relation prediction technique by applying an attention-based graph neural network model to a bipartite entity-relation graph built from a KG. Recoin [13] suggests properties to an entity on Wikidata by collaboratively using the information about other entities that are similar to that entity; to ensure the high quality of the suggested properties, it sometimes involves much prior knowledge when defining the similarity between entities on Wikidata. For example, for entities of type human, a (Boolean) similarity is manually defined as whether the two entities have the same *occupation* or not.

Our work differs from these techniques by considering a more complex problem, that of predicting relation-tail pairs for a given (head) entity. Although these relation prediction techniques can be combined with link prediction techniques to perform our instance completion task, such an approach yields subpar performance [14], as it does not fully leverage the schema information encoded by the triplets.

2.3 Instance Completion Task

A closely related work to our instance completion problem is OKELE [10], which suggests relation-tail pairs to a longtail head (i.e., a less popular/frequent entity) in a KG, using open Web data. Specifically, for a given head h, OKELE first implements a relation prediction technique by applying an attention-based graph neural network model to a bipartite entity-relation graph built from a KG, predicting a list of relations; for each suggested relation r, it then extracts and verifies potential tails from open Web data including semi-structured vertical websites, unstructured plain text in Web content and structured HTML tables. However, OKELE differs from our work by extensively using open Web data. In contrast, our proposed solution only requires triplets and entity types from a KG, without the need of involving any extra data sources.

We also note that the so-called instance reconstruction task on KGs has been defined by [53] for n-ary (or multifold) relational facts, where an n-ary relation consisting of a set of relations $\{r_1, r_2, ..., r_n\}$ links multiple entities $\{e_1, e_2, ..., e_n\}$, respectively. For example, the n-ary relation "PeopleMariage" containing the following three relations person, spouse, location, links three entities Kobe Bryant, Venessa Bryant and California. The authors define the instance reconstruction task as follows: given an n-ary relation $\{r_1, r_2, ..., r_n\}$ and a subset of the entities linked by this relation, predict the rest of the entities, such as $\{e_1, ?, ..., ?\}$. With the help of relation paths [54], this task boils down to a link prediction task, i.e., predicting $(e_1, r_1r_2, ?)$, $(e_1, r_1r_3, ?)$ and so on. Alternatively, this task can also be formulated as a hyper(-relational) graph link prediction problem [55], [56], [57]. Therefore, this task fundamentally differs from our instance completion task, where we do not assume any information about the relations.

Compared to our previous work [14] in which we proposed RETA-Filter and RETA-Grader, in this paper, we further design a systematic integration of the two components to fully benefit from the efficiency of RETA-Filter in solution space reduction and the deliberation of RETA-Grader in scoring candidate triplets. Specifically, different from our previous work where RETA-Grader is trained independently of RETA-Filter, we now tightly couple the training process of RETA-Grader with RETA-Filter by training RETA-Grader to identify correct r-t pairs for a given h from the reduced solution space output by RETA-Filter using a customized negative sampling process; subsequently, RETA-Grader can better focus on discriminating correct answers from the schematically correct candidate r-t pairs (the reduced solution space) rather than the schematically incorrect r-t pairs which have been filtered out by RETA-Filter. The new experiments we present show that compared to our previous work, our newly proposed RETA++ achieves significantly better performance (with an improvement of 16.9% on average), while requiring 35% fewer training epochs at the same time.

2.4 Knowledge Graph Schema

KG Schema defines a high-level structure and semantics that a KG follows or should follow [58]. For domain-specific KGs, their schema is usually manually defined with simple structures and usually has a small scale, such as the medical KG in [59] with 9 entity types and 9 relations for modeling electronic medical records, or the biomedical KG [60] with 11 entity types and 13 relations for modeling drug-target interactions. Note that in these KGs, a relation is manually defined to connect two specific entity types only. However, open-domain KGs (e.g., Wikidata [1] and Freebase [2]) usually do not have a unified and fixed schema [13]; relevant and irrelevant entity types can sometimes even evolve over time [61]. Even though some effort such as Schema.org has been made to unify schemas for structured data on the Web, such schemas still have a low coverage on the Web, and thus have not been widely adopted by modern KGs (with about 10 million Web sites adopting the schema vocabulary [62], compared to about 1.98 billion sites in total on the Web [63]). Subsequently, KG schema extraction (a.k.a., ontology learning, taxonomy extraction, or ontology population) has been studied, for purposes including refining an existing ontology [64], adapting a given ontology to a new KG [65], or building an new ontology [66], from either a given KG [67] or text corpora [68]. In this paper, we focus on the KG completion tasks with schema awareness.

In the current literature, schema information has been shown to be significantly helpful for KG completion. Existing approaches use such schema information in different stages of link prediction tasks: 1) as a pre-processing filter to reduce the solution space before link prediction techniques score the plausibility of a fact [14]; 2) as an additional input together with a fact fed to link prediction techniques when scoring the plausibility of the fact [40], [41], [42], [45], [46], [67], [69], [70]; 3) as a post-processing step to verify the schema correctness of the predicted facts generated by link prediction techniques [43]. Our work differs from these existing techniques in two aspects. First, we focus on the instance completion task with a more realistic setting rather than the link prediction task. Second, we use the schema information in two stages of the task, i.e., not only as a preprocessing filter for solution space reduction, but also as an

additional input for scoring candidate facts over the reduced solution space.

3 RETA++: Two-Step Schema-Aware Instance Completion

In this section, we present RETA++, a two-step schemaaware instance completion approach inspired by the human psychology principle "fast-and-slow thinking". We first present the overview of our proposed approach, followed by the detail of the two steps and the integration of them for instance completion tasks.

3.1 RETA++ Overview

RETA++ is designed for efficiently solving the instance completion problem (h,?,?). Inspired by the "fast-andslow thinking" principle, we follow a two-step approach as shown in Figure 2. First, inspired by the fast thinking process, the first step designs RETA-Filter, leveraging the schema of a KG (represented as entity-typed triplets) to fast filter candidate *r*-*t* pairs schematically matching the given h, so as to efficiently reduce the solution space. Second, inspired by the slow thinking process, the second step designs a KG embedding model RETA-Grader scoring candidate r-t pairs considering the plausibility of both the triplet and its corresponding schema. Figure 2 shows an example of the input/output of these two steps. For the instance completion problem (*China*,?,?) where $(?,?) \in \{all \ combinations \}$ of relations and entities in the input KG, the first step uses RETA-Filter to filter r-t pairs schematically matching the entity China (of type Country) by leveraging the entitytyped triplets, resulting in a reduced solution space where $(?,?) \in \{(hasCapital, entity of type City), (hasCurrency, entity$ type Currency), ... }. Afterward, the second step uses RETA-Grader to score each candidate r-t pair using a KG embedding model learnt from the triplets and its corresponding schema from the input KG, resulting in a ranked list of r-t pairs where the top ones are the most plausible r-t pairs for the head entity *China*, such as {(*hasCapital*, *Beijing*), (hasCurrency, CNY), ...}. In the following, we present the detailed design of each step.

3.2 RETA-Filter

Our RETA-Filter is designed to filter candidate r-t pairs schematically matching the given h. More precisely, although the schema of a KG implies valuable information about the structure of the KG, open-domain KGs such as Wikidata do not have a fixed schema [13]. Therefore, we resort to entity-typed triplets from an input KG, which encodes the schema information of the KG. Specifically, a KG contains a set of triplets (h, r, t), where $h, t \in E$ and $r \in R$; E and R are the sets of all entities and relations in the KG, respectively. Each entity h (or t) can have one or multiple types $h_type_1, h_type_2, \dots \in T$, where T is the set of all entity types in the KG (we will discuss later the case where an entity does not have any type). Subsequently, the schema of a KG can be characterized by a set of entitytyped triplets (h_type, r, t_type) , indicating that an entity of type h_type could be linked to an entity of type t_type via a relation r. Such information can serve as an important



Fig. 2. Overview of our two-step approach RETA++ consisting of RETA-Filter and RETA-Grader.



Fig. 3. A toy example of RETA-Filter, where we highlight an example of computation for one relation (in gray). Given an head entity h, RETA-Filter fetches its type vector $\vec{h_T}$ and multiply it with the tensor **F**. The resulting matrix is then multiplied with the type matrix \mathbf{M}_T in order to compute the candidate r-t pairs for h.

guideline to identify eligible r-t pairs for a given h, thus avoiding many meaningless predictions.

To implement RETA-Filter, we first extract entity-typed triplets (h_type, r, t_type) from the triplets (h, r, t) in a KG, by considering all the combinations of the types of h and tfor each triplet (h, r, t). For example, if h and t have m and *n* types, respectively, we extract $m \times n$ entity-typed triplets. Afterward, we represent all these entity-typed triplets as a *head type-relation-tail type* tensor $\mathbf{F} \in \mathbb{B}^{|T^h| \times |R| \times |T^t|}$, where T^h and T^t are the sets of all head and tail types, respectively, while R is the set of all relations in the KG. Note that T^h and T^t are the same as T in practice, as they are both the set of all entity types. We use different notations for them to distinguish the semantic meaning of the corresponding dimensions of **F**, where the first and third dimensions of F refer to head and tail types, respectively. Finally, by representing h using its type vector $\vec{h_T} \in \mathbb{B}^{|T_h|}$ and all tails using a type matrix $\mathbf{M}_T \in \mathbb{B}^{|E| \times |T|}$, we can efficiently compute the candidate r-t pairs for h via a tensor product:

$$\mathbf{S} = h_T \times_{T_h} \mathbf{F} \times_{T_t} \mathbf{M} \tag{1}$$

where \times_n denotes the mode-*n* tensor product [71]. The resulting matrix $\mathbf{S} \in \mathbb{N}^{|R| \times |E|}$ encodes the number of matches between *h* and each *r*-*t* pair, under the extracted schema. Figure 3 shows a toy example. To implement this filter, we have to take into account the following two practical issues:

• **Frequency of entity-typed triplets.** Entity-typed triplets (*h_type*, *r*, *t_type*) encode the schema of a KG. When con-

verting the triplets of the KG into entity-typed triplets, we also obtain the frequency of each entity-typed triplets, indicating how many times it appears in the KG. Intuitively, a low frequency indicates a small contribution of the corresponding entity-typed triplet to the schema of the KG, which could also be considered as noise. Subsequently, we could select frequent entity-typed triplets (whose frequencies are higher than a threshold α) to build **F**. However, a too high value of α could also remove useful entity-typed triplets, resulting in an incomplete schema captured by **F**. In other words, α controls the quality of the extracted schema, and subsequently balances the tradeoff between the size of the candidate (r-t pair) set and its coverage of true r-t pairs being included in the candidate set. On one hand, a too low value of α could include noisy entitytyped triplets in **F**, resulting in a large candidate set with noisy and redundant r-t pairs. On the other hand, a too high value of α captures incomplete schema information when building **F**, resulting in a small but low-coverage candidate set (i.e., missing true *r*-*t* pairs).

We note that once we select frequent entity-typed triplets (whose frequencies are higher than α), we build **F** as a Boolean tensor without considering the absolute frequency of each selected entity-typed triplet, since the absolute frequencies of these frequent entity-typed triplets are not useful when representing the schema of the KG. For example, an entity-typed triplet (*human, occupation, profession*) with a frequency of 10,000 does not necessarily mean that it is (ten times) more important than an entity-typed triplet (*enterprise, headquarters location, city*) with a frequency of 1,000 when representing the structure of the KG; such a frequency difference may just be caused by the varying popularity of different entity types in a KG. Therefore, we do not distinguish the selected entity-typed triplets by their frequencies when building **F**.

Number of matches between *r*-*t* pairs and *h*. The resulting matrix S encodes the number of matches between *h* and each *r*-*t* pair under the extracted schema. Intuitively, a higher number of matches indicates a higher plausibility of the corresponding *r*-*t* pair being a candidate for *h*. Following the example in Figure 3, the number of matches

between *h* and the first *r*-*t* pair is $\mathbf{S}_{0,0} = 3$. Subsequently, instead of taking all *r*-*t* pairs with non-zero matches (in **S**) as the candidate *r*-*t* pairs, we could further select higher-quality *r*-*t* pairs whose numbers of matches are higher than a threshold β as candidate *r*-*t* pairs. Obviously, a lower value of β selects more candidate *r*-*t* pairs and thus leads to higher coverage of true *r*-*t* pairs being included in the candidate set, and vice versa. In essence, β directly balances the tradeoff between the size of the candidate (*r*-*t* pair) set and its coverage.

In summary, RETA-Filter is designed to efficiently filter candidate *r*-*t* pairs schematically matching with the KG schema extracted from the entity-typed triplets. We use two tunable parameters α and β balancing the tradeoff between the size of the candidate (*r*-*t* pair) set and its coverage. By varying α and β , RETA-Filter can achieve the best Pareto frontier when trading off the size of the candidate (*r*-*t* pair) set and its coverage, compared to a sizable collection of baselines (see our experiments for more detail).

3.3 RETA-Grader

Based on the set of filtered candidate r-t pairs provided by RETA-Filter, RETA-Grader further evaluates and ranks these candidate *r*-*t* pairs considering the plausibility of both the triplet and its corresponding schema using a subtly designed KG embedding model. Figure 4 shows our embedding model consisting of three parts. Specifically, for each fact (h, r, t), it 1) learns to capture the structural information of the triplet (h, r, t), generating a triplet relatedness feature vector, and 2) learns to capture the corresponding schema information encoded by the triplet, generating a set of relatedness feature vectors, one for each entity-typed triplet (h_type, r, t_type) , and then merges them into a unique schema relatedness feature vector. Finally, it concatenates the triplet and schema relatedness feature vectors into an overall relatedness feature vector to output a final prediction score, measuring the plausibility of both the input triplet and its corresponding schema. In the following, we discuss the detailed design of these three modules.

3.3.1 Learning from triplets

To learn from a triplet (h, r, t), we use a Convolutional Neural Network (CNN) to model the interaction between the three elements in the triplet, i.e., head h, relation r and tail t. We adopt a CNN here, as it has been successfully used to learn from triplets in KGs by previous work [33], [72]. As shown in Figure 4, we start by concatenating the three embedding vectors $\vec{h}, \vec{r}, \vec{t} \in \mathbb{R}^K$ (K is the embedding dimension), resulting in a matrix $I \in \mathbb{R}^{3 \times K}$. The matrix I is fed to a 2D convolutional layer with n_f filters of size 3×3 . We set the filter size to 3×3 to capture the triplewise relatedness between the embeddings of h, r and t. This convolutional layer returns n_f feature maps of size K - 2 each, which are then flattened into a *triplet relatedness feature vector* $\vec{\phi} \in \mathbb{R}^{1 \times n_f(K-2)}$. This relatedness vector $\vec{\phi}$ characterizes the plausibility of a fact (h, r, t) of being true.

3.3.2 Learning from schema (entity-typed triplets)

The schema information encoded by the triplet (h, r, t) is also an important predictor for the plausibility of the triplet.

Therefore, we extract and learn from entity-typed triplets considering all the combinations of the types of h and t. When h and t have m and n types, respectively, we extract a set of mn entity-typed triplets $\{(h_type_i, r, t_type_i)|1 \leq$ $i \leq m, 1 \leq j \leq n$. Afterward, we learn from each of these entity-typed triplet (h_type_i, r, t_type_i) to generate its relatedness feature vector, using a similar method as for learning from triplets. Specifically, as shown in Figure 4, by concatenating three embedding vectors for head type $\overrightarrow{h_type_i}$, relation \vec{r} , tail type $\overrightarrow{t_type_j}$, we also resort to a 2D convolutional layer with n_f filters of size 3×3 to capture the triple-wise relatedness between h_type_i , \vec{r} , and $\overline{t_t type'_i}$; the resulted feature maps are then flattened into a relatedness vector of size $1 \times n_f(K-2)$. Note that the filters in this module are different from the filters in the first module. We repeat this process for all the mn entitytyped triplets associated with the input triplet. Finally, we concatenate these relatedness feature vectors into a matrix of size $mn \times n_f(K-2)$, and then take the minimum value along each feature dimension, resulting in a unique schema relatedness feature vector ψ . The basic assumption behind this min operation is that for a true triplet, the relatedness of the three elements (*h_type*, *r*, *t_type*) of *any* entity-typed triplets should be high; subsequently, the minimum relatedness along each feature dimension is expected to be high. Similar ideas have also been successfully applied by previous works to merge relatedness scores in a neural network [55], [73].

3.3.3 Prediction using triplet and schema relatedness feature vectors

In the two previous modules, for each triplet, we obtain a triplet and a schema relatedness feature vectors ($\vec{\phi}$ and $\vec{\psi}$, respectively). We then concatenate $\vec{\phi}$ and $\vec{\psi}$ into an *overall* relatedness feature vector of size $2n_f(K-2)$. Finally, we use a fully connected layer to output the predicted score σ from the overall relatedness feature vector.

3.4 Integration of RETA-Filter and RETA-Grader

To fully benefit from the efficiency of RETA-Filter in solution space reduction and the deliberation of RETA-Grader in scoring candidate triplets, we systematically integrate these two components by training RETA-Grader on the reduced solution space output by RETA-Filter via a customized negative sampling process. Specifically, we train our RETA-Grader model by minimizing a softplus loss, which is defined as the negative log-likelihood of the logistic model:

$$\sum_{\omega \in \Omega} \log(1 + e^{-\sigma(\omega)}) + \lambda \mathbb{E}_{\omega'} \log(1 + e^{\sigma(\omega')})$$
 (2)

where Ω represents the set of training triplets. For each positive triplet $\omega = (h, r, t)$, a set of λ negative samples (each denoted as ω') are generated; $\sigma(\omega)$ and $\sigma(\omega')$ denote the predicted score of our RETA-Grader model for the true fact ω and the negative fact ω' , respectively. The expectation term implies ω' is randomly generated/sampled according to a certain strategy/distribution, which we discuss below.

3.4.1 Negative sampling strategies

Negative sampling strategies have been shown to play an important role in learning (knowledge) graph embedding



Fig. 4. Architecture of RETA-Grader

[74], [75] for resolving link prediction tasks. Traditional negative sampling methods for link prediction tasks [7] usually randomly corrupt one element in a positive triplet h, r or t (e.g., replacing t by a randomly sampled entity from the entity set E), which we denote as **Rand_Link**; this scheme is also adopted by our previous work RETA [14]. For our instance completion task where we predict r-t pairs for a given h, we now propose an alternative setting to randomly corrupt r-t pair for a given h (i.e., replacing r and t by randomly sampled relation and entity from the relation set R and the entity set E, respectively), so as to align with the goal the instance completion task, which we denote as **Rand_Inst**. In the following, we discuss the noise distribution from which the negative samples are generated.

3.4.2 Noise distributions

Due to its simplicity, uniform sampling (Uni) is widely used in KG embeddings [7]. However, the uniformly random corruption scheme has been shown to suffer from an inefficiency issue since many negative samples are less informative during the training process [74]. To alleviate this issue, existing solutions resort to customized noise distributions guided by heuristics [76] or dynamically adapted over the learning process [77], [78], [79]. Specifically, a Structure Aware Negative Sampling (SANS) [76] method is proposed to generate negative entities for the link prediction task based on a heuristic that neighboring entities without direct relation to a target entity are good candidates for negative samples. NSCaching [78] keeps negative entities with high plausibility in head and tail caches for each positive triplet during the training process, and then generates negative samples directly from those caches. Note these two methods are explicitly designed for the link prediction task (i.e., Rand_Link strategy) for generating negative entities only.

As an alternative, self-adversarial (**Self_Adv**) negative sampling technique [79] also favorites the negative samples with higher plausibility in the training process; it proposes to sample a negative sample ω' according to a probability computed by applying a softmax function on its predicted score $p_{w'} = \frac{\exp \sigma(\omega')}{\sum_{w''} \exp \sigma(\omega'')}$. However, in practice, it is computationally expensive to compute the denominator over all possible negative samples ω'' to generate one negative sample. The self-adversarial negative sampling thus uniformly samples a set of negative samples, and then reweighs each negative sample w' using the corresponding $p_{w'}$ as the weight [79]. Subsequently, it can be used for both Rand_Corr and Rand_Inst sampling strategies.

3.4.3 RETA-Filter supervised negative sampling

Motivated by the idea of using customized noise distribution, we propose to directly take negative samples from the reduced solution space output by RETA-Filter, where our negative samples are all schematically correct (Sche Corr) facts under the Rand_Inst strategy. Subsequently, RETA-Grader is learnt to focus on discriminating correct answers from the schematically correct candidate r-t pairs (from the reduced solution space only) rather than from those schematically incorrect *r*-*t* pairs which have been filtered out by RETA-Filter. This customized negative sampling process is more efficient as RETA-Grader is now trained in the reduced solution space instead of the whole solution space, under the supervision of RETA-Filter. Note that selfadversarial sampling (Self_Adv) can be further combined with our approach by reweighing a Sche_Corr negative sample. In our experiments later, we will systematically compare these negative sampling techniques, and show that Sche_Corr yields the best performance on the instance completion task.

In summary, Algorithm 1 shows the integrated training process of RETA-Grader under the supervision of RETA-Filter. For each training iteration, we first randomly sample a batch of positive triplets S^+ from the training dataset Ω (Line 2), and initialize an empty batch of negative triplets S^- (Line 3). For each positive triplet w = (h, r, t), we randomly sample one relation-tail pair r'-t' from the filtered results output by RETA-Filter for the given head h, and then include the negative triplet w' = (h, r', t') in S^- (Line 4-8). Afterward, RETA-Grader is trained on the batch including both positive and negative triplets $S = S^+ \cup S^-$, to minimize the loss in Eq. 2 (Line 9-10).

3.5 Practical considerations on design choices

To implement our RETA++, we take into account the following two practical considerations:

Algorithm 1 Integrated training process of RETA-Grader under the supervision of RETA-Filter

	1
Inp	ut: Training dataset Ω , RETA-Filter
1:	while not converge do
2:	Sample a batch of positive triplets S^+ from Ω randomly
3:	Initialize an empty batch of negative triplets S^-
4:	for each positive triplet $w = (h, r, t) \in S^+$ do
5:	Get the filtered result N_h from RETA-Filter
6:	Sample one pair r' - t' from N_h randomly
7:	Insert a negative triplet $w' = (h, r', t')$ into S^-
8:	end for
9:	Get one batch of training triplets $S = S^+ \cup S^-$
10:	Train RETA-Grader on \tilde{S} to minimize the loss in Eq. 2
11:	end while

- Number of types learnt per entity. Our RETA-Grader evaluates a triplet considering the plausibility of both the input triplet and its corresponding schema, where the schema information is represented by a set of entity-typed triplets. For a given triplet (h, r, t) where h and t have mand n types, respectively, the second module in Figure 4 is repeated mn times when evaluating this triplet; this could incur a large computation overhead for large values of m and n. To overcome this issue, we choose to learn at most top-k types (according to the frequency of types in a KG dataset) for each entity, resulting in min (mk, nk, k^2) repetitions of the second module, rather than mn times. In practice, a small value of k is able to achieve a good performance on our instance completion task, which has been shown in our previous work [14].
- Entities without types. Although most entities in modern KGs are associated with one or multiple types, there is still a small portion of entities without types. To accommodate these entities, we make the following adaptation to our proposed solution.

First, to generate a set of candidate r-t pairs for a head *h*, if any entity (*h* or *t*) does not have a type, we assume the entity could be associated with any type, such that we do not miss any potential candidates. Specifically, we have the following three cases: 1) when h has no type but t has types, h_T becomes an all-one vector, where our RETA-Filter still considers the match between r and t; 2) when t has no type but h has types, the corresponding row of M becomes an all-one vector, where our RETA-Filter still considers the match between h and r; 3) when both h and t have no type, h_T and the corresponding row of M become all-one vectors, where our RETA-Filter generates a full matrix S (without zero entry); in the last case in particular, if we set the threshold beta to 1, our RETA-Filter indeed degrades to keep all combinations of *r*-*t* pairs as candidates.

Second, to let RETA-Grader learn from entities without types, we assign an "unknown" type to these entities, and then keep the same pipeline for score prediction. For example, for a triplet (h, r, t) where h has no type and t has n types, the set of entity-typed triplets becomes $\{(unknown_type, r, t_type_j)|1 \le j \le n\}$. From a schema point of view, such an "unknown" type could be linked to any type in a KG via a relation, which makes the schema relatedness feature vector less discriminative for prediction. Subsequently, the fully projected layer in the third module in Figure 4 will automatically learn more from

TABLE 1 Statistics of the datasets

Dataset	JF17k	FB15k	HumanWiki
#Entity	9,233	14,579	38,949
#Entity w/ types	9,174	14,417	34,470
#Entity w/o types	59	162	4,479
#Type	511	588	388
#Type per entity	6.45	10.02	1.08
#Relation	326	1,208	221
#Fact	19,342	154,916	108,199
#Fact w/ types	19,015	144,117	87,150
#Fact w/o types	327	10,799	21,049

the triplet relatedness feature vector to make predictions.

4 EXPERIMENTS

We conduct an extensive evaluation on our instance completion task. In the following, we start by presenting our experimental setup, followed by our results and discussions.

4.1 Experimental Setup

4.1.1 Datasets

We use three popular KG datasets *JF17K*, *FB15K* and *HumanWiki* in our experiments. More precisely, the JF17K and FB15K datasets are extracted from Freebase by [54] and [20], respectively. We extract the HumanWiki dataset from Wikidata by extracting all triplets involving a head entity of type human (i.e., class Q5 human on Wikidata)². As our instance completion task suggests relation-tail pairs for a given head, for each unique head, we randomly split its *r*-*t* pairs in the datasets into 80% training and 20% test datasets. Table 1 shows the main statistics of the three datasets.

4.1.2 Baselines

We compare RETA++ against a sizable collection of state-ofthe-art techniques in the following three categories.

• Relation (property) prediction techniques. BPR [80] is a recommendation technique generating an entity-specific ranked list of relations (where we consider the relation prediction task as recommending relations to entities). Property suggester (WikiPS) [12] recommends relations to an entity using association rules learnt from existing triplets in Wikidata; this technique is provided as an online API³ on Wikidata, and thus can only be applied to the HumanWiki dataset. Recoin [13] suggests relations to an entity by collaboratively using the information about other similar entities, where the similarity is manually defined using a Boolean similarity function that considers two entities as similar if they share at least one type. In particular, it has a special setting for entities of type *human* using heuristics and prior knowledge, where the Boolean similarity is defined as whether two humans have the same occupation or not, i.e., whether the two head entities of type human have the same tail entity linked via the relation occupation or not. OKELE [10] predicts relations

^{2.} We choose to extract the HumanWiki dataset because one of the state-of-the-art relation prediction techniques, Recoin (see below), is specifically designed for "human" instances on Wikidata, using prior knowledge for better performance on relation prediction.

^{3.} https://www.wikidata.org/w/api.php?action=help&modules= wbsgetsuggestions

associated with an entity using an attention-based graph neural network model⁴. As these relation prediction techniques suggest a ranking list of relations to a given head, we take the top-N relations from the list as the predicted relations; tuning N balances the tradeoff between the size and the coverage of the resulting candidate (r-t pair) set.

- *Tail candidate refinement techniques.* Based on the predicted list of relations returned by the above techniques, a straightforward approach to form a set of candidate r-t pairs is to combine each predicted relation with all entities (All). One possible improvement to this step is to have a filtered list of relevant t rather than using all entities, i.e., generating a subset of potential t for the given h and r. Note that this differs from the link prediction task, as we generate a subset of t rather than ranking r-t pairs. In the current literature, an entity relatedness prediction task has been introduced by [53] for n-ary relational facts where an n-ary relation links multiple entities $\{e_1, e_2, e_3, \dots, e_n\}$; this task predicts the relatedness between entities in such an n-ary relational fact, in order to perform an instance reconstruction task $\{e_1, e_2, ?, ..., ?\}$ (see more detail in our Related Work section). The proposed techniques by [53] evaluate the relatedness between two entities, i.e., whether two entities should be linked by a relation or not, which can thus be adopted for our tail candidate refinement. Specifically, two techniques have been proposed by [53]. First, a relatedness affiliated embedding (**RAE**) model, which learns a neural network to predict a relatedness score between two entities, and considers them to be relevant if the score is higher than a threshold γ . Tuning γ balances the tradeoff between the size and the coverage of the resulting candidate (*r*-*t* pair) set. Second, a schema-based predictor (Sch), leverages the type requirements on the entities dictated by the schema of a relation, generating a set of (tail) entities schematically matching a given relation. Subsequently, one can also combine the refined sets of candidate tails from RAE and Sch by taking their intersection (RAE&Sch).
- *Link prediction techniques.* The instance completion task can only be conducted using link prediction techniques that predict a score for a given triplet. We thus exclude those techniques that predict a distribution of (tail) entities for a given head and a given relation (e.g., ConvE [33]), as they cannot be adopted to our instance completion task. Therefore, we consider the following link prediction techniques.

First, we consider classical link prediction techniques evaluating the plausibility of a triplet only. The translational distance models we consider include: **TransE** [20], which learns to preserve the relation between two entities as $h + r \approx t$; **TransH** [21], which extends TransE to better capture multi-mapping relations using relation-specific hyperplanes; **TransR** [22], which introduces relation-specific projections to also better capture multi-mapping relations; **TransD** [23], which further ex-

tends TransR by decomposing the projection matrix into a product of two vectors for an improved efficiency. The semantic matching models we consider are as follows: **Rescal** [25], which represents each entity as a vector and each relation as a matrix, and uses a bilinear function to model the relation between a pair of entities; DistMult [26], which simplifies Rescal by representing each relation embedding as a diagonal matrix; **ComplEx** [30], which further extends DistMult in the complex space to better model both symmetric and asymmetric relations; Analogy [81], which explicitly models analogical structures in multi-relational KG embeddings; SimplE [82], which is an expressive and interpretable KG embedding techniques based on canonical polyadic tensor decomposition; **RotatE** [79], which defines a relation as a rotation from a head to a tail in the complex space, capturing richer relation patterns; ConvKB [72] which uses convolutional neural networks to learn KG embeddings for link prediction; HolE [83] which learns compositional embeddings using circular correlation; QuatE [84] which learns hypercomplex-valued embeddings with three imaginary components (quaternion embeddings); OctonionE [84] which further extends QuatE to the hypercomplex number space with one real part and seven imaginary components (octonion embeddings). For each of these link prediction technique, we use the hyperparameters set by [85] and [86]. We also consider a variation of RETA-Grader as a baseline, where we learn from triplets only, without learning from the schema (entity-typed triplets); we refer to it as **RETA** (no type).

Second, we also consider link prediction techniques using additional information about entity types as baselines, as our RETA-Grader also evaluates a triplet considering the plausibility of both the input triplet and its corresponding entity-typed triplets. TypeTransE and TypeRescal [69] are extended from TransE and Rescal, respectively, by imposing type constraints in the objective functions of the two respective techniques. TypeDM and TypeComplex [41] are extended from DistMult and ComplEx, respectively, by explicitly modeling entity type compatibility. RETA refers to the approach proposed by our previous work [14], where the RETA-Grader is the trained with the negative triplets sampled from the whole solution space by randomly corrupting one element in the positive triplet *h*, r or t; in other words, the training process is independent of RETA-Filter. RETA++ is the current extension of RETA, systematically integrating RETA-Filter and RETA-Grader, where RETA-Grader is trained under the supervision of RETA-Filter via a customized negative sampling process.

For our RETA-Grader, we use the negative sampling technique Sche_Corr, and set the number of types learnt per entity k = 1, 1, and 4, the number of Filters $n_f = 50, 200$, and 50, the number of negative samples $\lambda = 2, 1$, and 1 for JF17K, FB15K, and HumanWiki, respectively. More details on these parameter sensitivity study will be present later. The implementation of RETA++ and used datasets are available here⁵.

^{4.} Note that we consider only the relation prediction technique from OKELE as a baseline technique in this paper, as its link prediction technique extensively uses open Web data, which fundamentally differs from our instance completion task requiring only triplets and entity types from a KG, without the need of involving any extra data sources (see more detail in our Related Work section).

4.1.3 Evaluation Protocol

To implement our instance completion task, for a test h, we first generate a set of candidate r-t pairs, and then score and rank them. We evaluate this task in two steps.

First, we evaluate the quality of the filtered candidate r-t pair sets. We consider two metrics: 1) the coverage of the candidate set (i.e., the percentage of the ground truth r-t covered by the candidate set), and 2) the size of the candidate set. Intuitively, a good candidate set should have high coverage and a small size at the same time. Due to the intrinsic tradeoff between the *coverage* and *size* of the candidate set, we plot and compare the Pareto frontier of different techniques. Specifically, to generate a set of candidate r-t pairs using our baseline techniques, we can use any *relation* prediction technique (BPR, WikiPS, Recoin, Recoin_Human or OKELE) combined with any tail candidate refinement tech*nique* (All, RAE, Sch or RAE&Sch). We first take the top N relations generated by a relation prediction technique and then use one tail candidate refinement technique to generate a set of candidate *r*-*t* pairs. By tuning N (and also γ for RAE when applicable), we balance the tradeoff between the size and the coverage of the resulting candidate (r-t pair) set. For our method, we tune α and β to balance this tradeoff.

Second, we evaluate the performance of different (link prediction) techniques in ranking these candidate r-t pairs. Specifically, after training a technique on an input KG, we use it to score each candidate r-t pair (together with the test h), and thus generate a ranking list of candidate r-t pairs. We then evaluate this ranking list against the ground truth r-t pairs, and report Recall@k (Rec@k), Mean Average Precision (MAP), and Normalized Discounted Cumulative Gain (NDCG). Note that the quality of the filtered candidate r-t pair sets generated by different r-t pair filtering techniques in the previous step will impact the performance in this step. For a fair comparison, we discount this impact by fixing the set of candidate r-t pairs ensuring 95% coverage using our RETA-Filter for all techniques.

4.2 Performance on Filtering *r*-*t* Pairs

In this section, we evaluate the first step of our instance completion task by investigating the performance of filtering r-t pairs using different techniques.

4.2.1 Tradeoff between the size and coverage of the filtered results

Figure 5 shows the Pareto frontier when trading off the size and the coverage of the candidate r-t pair set using each baseline technique and our RETA-Filter. We observe that our RETA-Filter outperforms state-of-the-art techniques in general by achieving a better tradeoff in most cases, i.e., the resulting Pareto frontier is closer to the upper-left corner of the plot. Moreover, we find that adding tail candidate refinement techniques to relation prediction techniques does help improve the quality of the resulting r-t pair sets, where combining RAE and Sch (RAE&Sch) shows the best performance.

Furthermore, under the tail candidate refinement technique RAE&Sch, we find that Recoin (RAE&Sch) is the most competitive baseline, which is able to achieve comparable results to our RETA-Filter on FB15K and HumanWiki. More

TABLE 2

Size of candidate set with at least 95% coverage. *WikiPS works only on HumanWiki and fails to reach 95% coverage, due to the fact that its suggested relations (returned from its API online) do not include all relations in our HumanWiki dataset.

Method	JF17k	FB15k	HumanWiki
BPR (RAE&Sch)	450,933	1,526,529	287,247
OKELE (RAE&Sch)	699,890	7,734,614	1,011,185
Recoin (RAE&Sch)	514,930	1,567,367	278,253
WikiPS (RAE&Sch)*	N/A	N/A	_*
RETA-Filter	68,745	1,048,053	248,721

precisely, RETA-Filter is better than Recoin (RAE&Sch) by achieving a slightly smaller candidate set under the same coverage when the coverage is higher than 82% on FB15K (95% on HumanWiki), while we observe the opposite result when the coverage is lower than 82% on FB15K (95% on HumanWiki). However, Recoin achieves such results by using heuristics and prior knowledge on the structure of a KG (i.e., manually defined similarity between entities), in particular on HumanWiki where the similarity between entities is defined as whether two humans have the same occupation or not, i.e., whether the two head entities of type human have the same tail entity linked via the relation occupation or not. In addition, WikiPS (RAE&Sch) also performs well on HumanWiki (the second best baseline). However, it cannot reach high coverage, due to the API limitations, as the returned relations do not include all relations in our HumanWiki dataset.

4.2.2 Solution space reduction under the same coverage

In practice, as the first step of our instance completion task, the *r*-*t* pair filtering step should generate a set of candidate *r*-*t* pairs with high coverage (95% or even higher), in order to let the following link prediction techniques or our RETA-Grader identify the true *r*-*t* pairs by scoring and ranking the candidate *r*-*t* pairs. Otherwise, the prediction on a candidate set with a low coverage will certainly lead to low performance on our instance completion task, as a candidate set with a low coverage excludes many ground-truth *r*-*t* pairs, which can never be correctly predicted. In other words, the coverage of the candidate set in this step is indeed the upper bound of recall@N when ranking the candidate rt pairs in the following prediction step. Therefore, we set the coverage to 95% and compare the size of the candidate sets generated by different methods using the bestperforming tail candidate refinement technique RAE&Sch. Table 2 shows the results. We see that our RETA-Filter consistently outperforms the baseline techniques, and reduces the size of the candidate set by 84.75%, 31.34%, and 10.61% compared to the best-performing baselines on JF17k, FB15k, and HumanWiki, respectively. In the following experiments, we use our RETA-Filter to generate the set of candidate r-t pairs with 95% coverage.

4.2.3 Runtime performance

We also evaluate the runtime performance of different techniques in both pre-processing/training (if applicable) and filtering processes. Table 3 shows the results on our benchmark hardware (Intel Xeon6248@2.50GHz, 128GB RAM@2666Hz, NVIDIA Tesla V100 16GB, Ubuntu 18.04). We observe that RETA-Filter achieves the best runtime



Fig. 5. Tradeoff between the size and the coverage of the resulting candidate (*r*-*t* pair) set from different filtering techniques. Note that our RETA-Filter has a unique tradeoff line on each dataset, as it does not use any tail candidate refinement techniques.

TABLE 3 Runtime performance of different techniques. *Note that WikiPS is only applicable to HumanWiki dataset; its pre-processing/training time is unknown as it is provided as an online API by Wikidata, and its filtering time is measured by the querying time of the API.

Mathad	(Offline) Pre-proce	ssing/Training	(Online) Filtering			
Method	JF17k	FB15k	Wiki	JF17k	FB15k	Wiki	
BPR (RAE&Sch)	27.37s	57.37s	33.4s	21.78s	110.88s	47.79s	
OKELE (RAE&Sch)	1h2m	34h40m	4h43m	18m18s	3h1m	41m32s	
Recoin (RAE&Sch)	16.16s	82.10s	54.82s	9.46s	37.2s	34.86s	
WikiPS* (RAE&Sch)	N/A	N/A	N/A	N/A	N/A	39m52s	
RETA-Filter	0.99s	7.15s	0.90s	8.54s	29.14s	13.19s	

performance in both the (offline) pre-processing/training process and (online) filtering process. Specifically, BPR and OKELE both require a model training process (a matrix factorization model and a graph neural network, respectively), resulting in long training and filtering time. WikiPS is provided as an online API by Wikidata, where we have only its filtering time measured by the querying time of the API. Recoin and our RETA-Filter do not need an actual training process, but require pre-processing the input dataset to extract schema-related information, which makes them more efficient than other techniques. Moreover, RETA-Filter benefiting from the efficiency of tensor operations on the GPU-enabled hardware further outperforms Recoin, showing an average speedup of 29.6x and 1.7x in the offline pre-processing and online filtering processes, respectively.

4.3 Performance on Ranking *r*-*t* Pairs

In this section, we evaluate the second step of our instance completion task. For a fair comparison, based on the candidate r-t pair set generated by our RETA-Filter, we evaluate the performance of different link prediction techniques.

4.3.1 Performance comparison

Table 4 shows the results. We observe that RETA/RETA++ consistently outperforms all baseline techniques on our instance completion task in general. Taking MAP as an example, it yields an improvement of 31.90%, 105.02%, and 45.38% over the best-performing baselines on JF17K, FB15K, and HumanWiki, respectively. We discuss the results in detail as follows.

First, schema-aware techniques using entity types achieve better performance in general. In particular, compared to RETA (no type), which is the variant of our proposed model without learning from the schema, RETA/RETA++ learning from both triplets and their corresponding schema significantly achieve better performance; this shows that learning from entity-typed triplets is indeed helpful for the instance completion task. Note that TypeComplex, which further considers entity types on top of the ComplEx model, underperforms ComplEx; opposite results are reported in [41]. This is due to the different implementations of the ComplEx model. More precisely, we tested two different implementation of ComplEx from [41] and [85], respectively, and report the results from the bestTABLE 4

Performance of different methods on our instance completion task. We highlight the best-performing one of RETA/RETA++ and also the best-performing baseline techniques for each metric.

Mathad		JF1	7K			FB1	5K			Huma	nWiki	
Method	Rec@10	Rec@5	MAP	NDCG	Rec@10	Rec@5	MAP	NDCG	Rec@10	Rec@5	MAP	NDCG
TransE	0.0682	0.0321	0.0230	0.0233	0.0411	0.0162	0.0242	0.1654	0.0098	0.0008	0.0147	0.1140
TransH	0.0410	0.0188	0.0173	0.1248	0.0216	0.0069	0.0175	0.1505	0.0110	0.0007	0.0119	0.1086
TransR	0.0657	0.0316	0.0229	0.1343	0.0441	0.0124	0.0240	0.1648	0.0052	0.0006	0.0124	0.1118
TransD	0.0465	0.0238	0.0179	0.1253	0.0253	0.0086	0.0196	0.1566	0.0050	0.0005	0.0108	0.1061
Rescal	0.0074	0.0057	0.0048	0.0791	0.0009	0.0004	0.0002	0.0566	0.0000	0.0000	0.0000	0.0474
Distmult	0.0892	0.0499	0.0367	0.1392	0.0596	0.0260	0.0245	0.1559	0.1400	0.1035	0.0767	0.1747
ComplEx	0.0841	0.0523	0.0317	0.1377	0.1235	0.0683	0.0597	0.1994	0.0986	0.0586	0.0416	0.1365
Analogy	0.1129	0.0679	0.0414	0.1424	0.1496	0.0841	0.0625	0.2017	0.0136	0.0064	0.0077	0.0891
SimplE	0.0881	0.0398	0.0290	0.1336	0.0483	0.0198	0.0245	0.1536	0.1151	0.0812	0.0573	0.1520
RotatE	0.1745	0.0996	0.0529	0.1694	0.0583	0.0341	0.0359	0.1805	0.0429	0.0064	0.0171	0.1172
ConvKB	0.1264	0.0756	0.0422	0.1487	0.0782	0.0202	0.0356	0.1758	0.1256	0.0902	0.0498	0.1586
HolE	0.1795	0.0976	0.0581	0.1698	0.1406	0.0798	0.0587	0.1943	0.1348	0.0988	0.0586	0.1689
QuatE	0.1791	0.0989	0.0556	0.1710	0.1540	0.0958	0.0687	0.2079	0.1356	0.1072	0.0634	0.1674
OctonionE	0.1732	0.0977	0.0554	0.1703	0.1498	0.0934	0.0673	0.2043	0.1366	0.1062	0.0627	0.1651
RETA (no type)	0.1414	0.0976	0.0528	0.1600	0.1262	0.0653	0.0538	0.2114	0.1606	0.1109	0.0860	0.2038
TypeTransE	0.0692	0.0379	0.0244	0.0240	0.0435	0.0174	0.0258	0.1767	0.0098	0.0009	0.0150	0.1215
TypeRescal	0.0083	0.0060	0.0049	0.0796	0.0009	0.0005	0.0002	0.0560	0.0000	0.0000	0.0000	0.0456
TypeDM	0.1481	0.0524	0.0452	0.1651	0.1274	0.0693	0.0576	0.1999	0.1285	0.1143	0.0789	0.2079
TypeComplex	0.0665	0.0425	0.0203	0.1204	0.0985	0.0552	0.0439	0.1755	0.0581	0.0015	0.0165	0.1082
RETA	0.1916	0.1153	0.0615	0.1855	0.2104	0.1288	0.1037	0.2658	0.2049	0.1545	0.1166	0.2332
RETA++	0.2153	0.1418	0.0766	0.1986	0.2560	0.1770	0.1409	0.3233	0.2191	0.1535	0.1250	0.2463

TABLE 5 Performance of RETA and RETA++ on different test facts.

Mathad	Tost Facts	JF	17K	FB	15K	HumanWiki		
wiethou	lest racis	MAP	NDCG	MAP	NDCG	MAP	NDCG	
DETA	known types	0.0614	0.1857	0.1075	0.2780	0.1440	0.2912	
KEIA	unknown type	0.0651	0.1577	0.0792	0.1875	0.0608	0.1154	
DETALL	known types	0.0766	0.1987	0.1282	0.3248	0.1530	0.3076	
KEIA++	unknown type	0.0786	0.1609	0.2224	0.3103	0.0683	0.1216	

performing implementation from [85] in this paper.

Second, compared to link prediction techniques using additional information about entity types, our RETA/RETA++ yield better performance by evaluating the plausibility of both the input triplet and its corresponding schema using a subtly designed KG embedding model. Moreover, RETA++ further outperforms RETA by 16.9% on average, which verifies the effectiveness of our newly proposed integration scheme of RETA-Filter and RETA-Grader. In other words, training RETA-Grader under the supervision of RETA-Filter can better benefit from the reduced solution space output by RETA-Filter, where RETA-Grader is learnt to discriminate correct answers from the schematically correct candidate r-t pairs (from the reduced solution space only) rather than those schematically incorrect *r*-*t* pairs which have been filtered out by RETA-Filter (as our previous work RETA does).

4.3.2 Performance on facts with and without types

We further investigate the performance of RETA and RETA++ when handling entities without types, where we assign an artificially created "unknown" type to such entities. To this end, we divide all test facts into two sets depending on whether a fact involves an "unknown" type or not: 1) test facts with types on both h and t (denoted as *known type*); and 2) test facts involving "unknown" type (denoted as *unknown type*), including both facts with types on either h or t and facts without types at all. We then compare the performance of our RETA-Grader on these two

sets of test facts. Table 5 shows the results. We observe that the performance on the facts with *known type* is generally better than the facts with *unknown type*. On one hand, for the facts with *known type*, our RETA-Grader is able to fully leverage the corresponding entity-typed triplets to evaluate the plausibility of a triplet from a schema perspective, resulting in better performance. On the other hand, for the facts with *unknown type*, our RETA-Grader can only evaluate the schematic plausibility based on the assumption that an "unknown" type could be linked to any type in a KG via a relation, which makes the schema relatedness feature vector less informative for prediction.

Moreover, compared to our previous work RETA, RETA++ consistently shows significant improvement of 13.27% and 47.79% on facts with known and unknown types, respectively. The improvement difference here is probably due to the following reason. For the facts with unknown type, where the schema relatedness feature vectors are less informative for prediction, training RETA-Grader on the reduced solution space is more beneficial for identifying the correct answers from the reduced solution space. In contrast, for the fact with known type, where the schema relatedness feature vectors are more informative for prediction, the benefit of training RETA-Grader on the reduced solution space is thus relatively smaller compared to the case of the facts with unknown type.

We note that the results of *known type* in Table 5 are very close to the results on all test facts from Table 4 (MAP and NDCG) on JF17K and FB15K, but not on HumanWiki. This

difference can be explained by the fact that facts with *known type* dominate the set of all test facts on JF17K and FB15K, representing 98.3% and 93.0% of all test facts on JF17K and FB15K, respectively, while this statistic on HumanWiki is 80.5% (see Table 1 for more detail).

4.3.3 Impact of Different Negative Sampling Techniques

In this experiment, we study the impact of different negative sampling techniques using RETA-Grader on our instance completion task. As discussed in Section 3.4, we compare two sampling strategies (**Rand_Link** and **Rand_Inst**) configured with different noise distributions (**Uni**, **SANS**, **NSCaching**, **Self_Adv**, **Sche_Corr**) with applicable. Table 6 shows the results.

First, we observe that the sampling strategy Rand_Inst generally performs slightly better than Rand_Link under the same noise distribution, as Rand_Inst matches better the problem setting of instance completion predicting r-t pairs. For example, with Uni and Self_Adv distributions, Rand_Inst outperforms Rand_Link by 1.17% and 0.93%, respectively.

Second, we find that although SANS and NSCaching have been shown to be useful for link prediction tasks, they are not helpful for the instance completion task, which is due to the fact that they are explicitly designed for the link prediction task to only generate negative entities. In addition, we see that Self_Adv is generally helpful by reweighing uniformly generated negative samples, showing consistent improvement over Uni under both Rand_Inst and Rand_Link sampling strategies, with 3.75% and 3.51% improvement, respectively.

Finally, our Sche_Corr achieves the best performance compared to other negative sampling techniques, yielding an improvement of 11.44% over the best-performing baseline (Self_Adv under the Rand_Inst strategy). This shows the advantage of integrating RETA-Grader with RETA-Filter for the instance completion task. Interestingly, when further combined with Self_Adv, Sche_Corr (Self_Adv) shows comparable results to Sche_Corr, which implies that Self_Adv is not helpful on top of Sche_Corr. This is due to the fact that the reduced solution space from RETA-Filter is sufficiently small and meanwhile has high quality; reweighing the negative samples there thus shows marginal improvement.

4.3.4 Runtime Performance

We also study the runtime performance of RETA and RETA++ by investigating the number of training epochs for convergence (achieving the previously reported performance), under the same set of training hyperparameters. Table 7 shows the results. We observe that RETA++ training RETA-Grader on the reduced solution space converges faster than RETA, requiring 35% fewer training epochs, while still achieving higher performance than RETA on instance completion tasks with an improvement of 16.9% (see Table 4). We note that the evaluation time of RETA and RETA++ are the same, as they both use the RETA-Grader for scoring and ranking r-t Pairs.

4.3.5 Parameter Sensitivity Study

We study the impact of three parameters of RETA-Grader in RETA and RETA++, i.e., the number of types learnt per



Fig. 6. Parameter sensitivity study on RETA and RETA++. The impact of 1) the number of types learnt per entity k, 2) the number of filters n_f , and 3) the number of negative samples λ , are shown in three rows from the top to bottom, respectively. The runtime performance is measured by the training time per epoch.

entity k, the number of filters n_f , and the number of negative samples λ on both instance completion performance and training time per epoch. In this section, we focus on test facts with known type only, i.e., test facts with known types on both h and t, as these test facts can be fully evaluated by our RETA-Grader considering the plausibility of both the input triplet and its corresponding schema.

• Number of types learnt per entity. RETA-Grader evaluates a triplet considering the plausibility of both the input triplet and its corresponding schema, where the schema information is represented by a set of entity-typed triplets. For entities with many types, the size of this entitytyped triplet set is large, incurring a large computation overhead. To solve this issue, our RETA-Grader considers at most top-k types for each entity. In this experiment, we investigate the impact of k on instance completion performance and training time per epoch. Figure 6 shows the results. On one hand, we observe that a small k can achieve a good performance on the instance completion task. Although considering entity types could improve the performance on instance completion, learning from a too large set of entity-typed triplets indeed makes it hard to capture the key schematic structure of the KG, due to the noise included in the large set of entity-typed triplets, resulting in degraded performance. On the other hand, the training time per epoch consistently increases with k, as our RETA-Grader repeats its second module $\min(mk, nk, k^2)$ times for a triplet where h and t have m and n types, respectively. We observe an exponentially increasing time on JF17K (and FB15K), as for most triplets in the dataset we have $\min(mk, nk, k^2) = k^2$ (the average number of types per entity is 6.45 on JF17K (and 10.02 on FB15K), which is larger than k in Figure 6). Note that on HumanWiki, both instance completion performance and training time per epoch have a small variation when increasing k, due to the small number of types per entity in the dataset. In essence, as the average number of types

TABLE 6

Impact of different negative sampling techniques. Note that SANS and NSCaching can only be used with the Rank_Link strategy, as they are designed to only sample negative entities; Sche_Corr is our proposed solution taking negative samples from the results of RETA-Filter, which intrinsically follows the Rand_Inst strategy.

Sampling	Noisy		JF1	7K			FB1	5K			Huma	nWiki	
Strategy	Distribution	Rec@10	Rec@5	MAP	NDCG	Rec@10	Rec@5	MAP	NDCG	Rec@10	Rec@5	MAP	NDCG
	Uni	0.1916	0.1153	0.0615	0.1855	0.2104	0.1288	0.1037	0.2658	0.2049	0.1545	0.1166	0.2332
Pand Link	SANS	0.1895	0.1147	0.0603	0.1906	0.2074	0.1276	0.1029	0.2700	0.2035	0.1543	0.1136	0.2310
Kanu_Link	NSCaching	0.1863	0.1131	0.0611	0.1825	0.2037	0.1255	0.1035	0.2606	0.2011	0.1447	0.1123	0.2307
	Self_Adv	0.1999	0.1206	0.0673	0.1928	0.2174	0.1355	0.1119	0.2806	0.2080	0.1556	0.1182	0.2370
	Uni	0.1966	0.1188	0.0615	0.1847	0.2150	0.1286	0.1090	0.2668	0.2049	0.1546	0.1163	0.2364
Pand Inst	Self_Adv	0.1999	0.1191	0.0676	0.1857	0.2197	0.1360	0.1190	0.2860	0.2093	0.1547	0.1188	0.2388
Kanu_mst	Sche_Corr	0.2153	0.1418	0.0766	0.1986	0.2560	0.1770	0.1409	0.3233	0.2191	0.1535	0.1250	0.2463
	Sche_Corr(Self_Adv)	0.2155	0.1417	0.0762	0.1989	0.2563	0.1779	0.1406	0.3235	0.2188	0.1533	0.1255	0.2469

TABLE 7 Number of training epochs for convergence

Method	JF17K	FB15K	HumanWiki
RETA	3000	400	3000
RETA++	2300	300	1300

per entity is only 1.08 on HumanWiki, increasing k affects only very few entities which have more than k types. In summary, we select the best-performing k = 1, 1, and 4 for JF17K, FB15K, and HumanWiki, respectively, for all other experiments.

- Number of filters n_f . The impact of the number of filters n_f used by RETA-Grader is shown in Figure 6. We observe that the optimal n_f varies across datasets, while the training time per epoch monotonically increases with n_f . Therefore, in all other experiments, we selected the best-performing $n_f = 50, 200$, and 50 for JF17K, FB15K, and HumanWiki, respectively.
- Number of negative samples λ . The impact of the number of negative samples λ is shown in Figure 6. We see that the number of negative samples has a marginal impact on the instance completion performance, while the training time per epoch linearly increases with λ , as the number of trained negative facts linearly increases. Therefore, in all other experiments, we selected the best-performing $\lambda = 2$, 1, and 1 for JF17K, FB15K, and HumanWiki, respectively.

Finally, compared to RETA, RETA++ consistently achieves better performance over different hyperparameters settings. On the other hand, although RETA++ incurs a small expense on the computational overhead per epoch compared to RETA, it is trained on a reduced solution space, and thus requires fewer training epochs (as evidenced above), which still yields an improved runtime performance.

5 CONCLUSION

This paper studies an instance completion problem over KGs, where we predict relation-tail r-t pairs for a given head h. To this end, inspired by the human psychological principle "fast-and-slow thinking", we propose a two-step schema-aware approach RETA++ consisting of two components: a *fast* RETA-Filter efficiently filtering candidate r-t pairs schematically matching the given h, and a *deliberate* RETA-Grader leveraging a KG embedding model scoring each candidate r-t pair considering the plausibility of both

the input triplet and its corresponding schema. RETA++ systematically integrates these two components by training RETA-Grader on the reduced solution space output by RETA-Filter via a customized negative sampling process, benefiting from the efficiency of RETA-Filter in solution space reduction and the deliberation of RETA-Grader in scoring candidate triplets. We evaluate our approach against a sizable collection of state-of-the-art techniques on three real-world KG datasets. Results show that our RETA-Filter can efficiently reduce the solution space for the instance completion task, outperforming best baseline techniques by 10.61%-84.75% on the reduced solution space size, while also being 1.7x-29.6x faster than these techniques. Moreover, RETA-Grader trained on the reduced solution space also significantly outperforms the best state-of-the-art link prediction techniques on the instance completion task by 31.90%-105.02%. In particular, compared to our previous work RETA where RETA-Grader is trained independently of RETA-Filter, RETA++ integrates these two components by training RETA-Grader on the reduced solution space output by RETA-Filter, now achieving 16.9% higher performance while requiring 35% fewer training epochs at the same time.

In future work, we plan to investigate more efficient solutions for the instance completion problem using graph neural networks.

6 ACKNOWLEDGEMENT

This project has received funding from University of Macau (SRG2021-00002-IOTSC), The Science and Technology Development Fund, Macau SAR (0038/2021/AGJ, 0047/2022/A1, SKL-IOTSC(UM)-2021-2023), UIC Research Grant (UICR0700021-22), the European Research Council (ERC, grant agreement 683253/GraphInt).

REFERENCES

- [1] Wikidata, http://wikidata.org/, 2012.
- [2] K. Bollacker, C. Evans, P. Paritosh, T. Sturge, and J. Taylor, "Freebase: a collaboratively created graph database for structuring human knowledge," in ACM SIGMOD/PODS, ACM. ACM, 2008, pp. 1247–1250.
- [3] Google, "Introducing the knowledge graph: things, not strings," https://blog.google/products/search/ introducing-knowledge-graph-things-not/, 2012.
- [4] C. Xiong, R. Power, and J. Callan, "Explicit semantic ranking for academic search via knowledge graph embedding," in WWW, 2017, pp. 1271–1279.

- [5] S. W.-t. Yih, M.-W. Chang, X. He, and J. Gao, "Semantic parsing via staged query graph generation: Question answering with knowledge base," in ACL and IJCNLP, 2015, pp. 1321–1331.
- F. Zhang, N. J. Yuan, D. Lian, X. Xie, and W.-Y. Ma, "Collaborative [6] knowledge base embedding for recommender systems," in KDD. ACM, 2016, pp. 353-362.
- Q. Wang, Z. Mao, B. Wang, and L. Guo, "Knowledge graph [7] embedding: A survey of approaches and applications," TKDE, vol. 29, no. 12, pp. 2724-2743, 2017.
- [8] M. Nickel, K. Murphy, V. Tresp, and E. Gabrilovich, "A review of relational machine learning for knowledge graphs," Proceedings of the IEEE, vol. 104, no. 1, pp. 11-33, 2015.
- R. West, E. Gabrilovich, K. Murphy, S. Sun, R. Gupta, and D. Lin, [9] "Knowledge base completion via search-based question answering," in WWW. ACM, 2014, pp. 515-526.
- [10] E. Cao, D. Wang, J. Huang, and W. Hu, "Open knowledge enrichment for long-tail entities," in WWW, 2020, pp. 384-394.
- [11] J. Lajus and F. M. Suchanek, "Are all people married? determining obligatory attributes in knowledge bases," in WWW, 2018, pp. 1115-1124
- [12] E. Zangerle, W. Gassler, M. Pichl, S. Steinhauser, and G. Specht, "An empirical evaluation of property recommender systems for wikidata and collaborative knowledge bases," in OpenSym, 2016, pp. 1-8.
- V. Balaraman, S. Razniewski, and W. Nutt, "Recoin: relative com-[13] pleteness in wikidata," in WWW Companion, 2018, pp. 1787-1792.
- [14] P. Rosso, D. Yang, N. Ostapuk, and P. Cudré-Mauroux, "Reta: A schema-aware, end-to-end solution for instance completion in knowledge graphs," in WWW, 2021, pp. 845-856.
- [15] D. Kahneman, Thinking, fast and slow. Macmillan, 2011.
- [16] J.-F. Bonnefon and I. Rahwan, "Machine thinking, fast and slow," Trends in Cognitive Sciences, vol. 24, no. 12, pp. 1019–1027, 2020.
- [17] L. A. Galárraga, C. Teflioudi, K. Hose, and F. Suchanek, "Amie: association rule mining under incomplete evidence in ontological knowledge bases," in WWW, 2013, pp. 413-422.
- [18] N. Lao, T. Mitchell, and W. Cohen, "Random walk inference and learning in a large scale knowledge base," in EMNLP, 2011, pp. 529-539
- [19] M. Niepert, "Discriminative gaifman models," in NIPS, 2016, pp. 3405-3413.
- [20] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, and O. Yakhnenko, "Translating embeddings for modeling multi-relational data," in Advances in neural information processing systems, 2013, pp. 2787-2795.
- [21] Z. Wang, J. Zhang, J. Feng, and Z. Chen, "Knowledge graph embedding by translating on hyperplanes." in AAAI, vol. 14, 2014, pp. 1112-1119.
- [22] Y. Lin, Z. Liu, M. Sun, Y. Liu, and X. Zhu, "Learning entity and relation embeddings for knowledge graph completion." in AAAI, vol. 15, 2015, pp. 2181-2187.
- [23] G. Ji, S. He, L. Xu, K. Liu, and J. Zhao, "Knowledge graph embedding via dynamic mapping matrix," in ACL and IJCNLP, vol. 1, 2015, pp. 687-696.
- [24] T. Ebisu and R. Ichise, "Toruse: Knowledge graph embedding on a lie group," in AAAI, 2018.
- [25] M. Nickel, V. Tresp, and H.-P. Kriegel, "A three-way model for collective learning on multi-relational data." in ICML, vol. 11, 2011, pp. 809-816.
- [26] B. Yang, W.-t. Yih, X. He, J. Gao, and L. Deng, "Embedding entities and relations for learning and inference in knowledge bases," in ICLR, 2015.
- [27] B. Shi and T. Weninger, "Proje: Embedding projection for knowledge graph completion," in AAAI, 2017, pp. 1236-1242.
- [28] C. Zhang, H. Yao, C. Huang, M. Jiang, Z. Li, and N. V. Chawla, "Few-shot knowledge graph completion," in AAAI, 2020, pp. 3041-3048.
- [29] I. Balažević, C. Allen, and T. M. Hospedales, "Tucker: Tensor factorization for knowledge graph completion," in EMNLP-IJCNLP), 2019, pp. 5185-5194.
- [30] T. Trouillon, J. Welbl, S. Riedel, É. Gaussier, and G. Bouchard, "Complex embeddings for simple link prediction," in ICML, 2016, pp. 2071-2080.
- [31] G. Ji, K. Liu, S. He, and J. Zhao, "Knowledge graph completion with adaptive sparse transfer matrix." in AAAI, vol. 16, 2016, pp. 985-991.

- [32] R. Socher, D. Chen, C. D. Manning, and A. Ng, "Reasoning with neural tensor networks for knowledge base completion," in NIPS, 2013, pp. 926–934.
- [33] T. Dettmers, P. Minervini, P. Stenetorp, and S. Riedel, "Convolutional 2d knowledge graph embeddings," in AAAI, 2017, pp. 1811-1818
- [34] M. Schlichtkrull, T. N. Kipf, P. Bloem, R. Van Den Berg, I. Titov, and M. Welling, "Modeling relational data with graph convolutional networks," in European Semantic Web Conference. Springer, 2018, pp. 593-607.
- [35] I. Balazevic, C. Allen, and T. M. Hospedales, "Hypernetwork knowledge graph embeddings," arXiv preprint arXiv:1808.07018, vol. 11731, 2018.
- [36] D. Q. Nguyen, T. Vu, T. D. Nguyen, D. Q. Nguyen, and D. Phung, "A capsule network-based embedding model for knowledge graph completion and search personalization," arXiv preprint arXiv:1808.04122, 2018.
- [37] A. Garcia-Duran and M. Niepert, "Kblrn: End-to-end learning of knowledge base representations with latent, relational, and numerical features," arXiv preprint arXiv:1709.04676, 2017.
- [38] A. Kristiadi, M. A. Khan, D. Lukovnikov, J. Lehmann, and A. Fischer, "Incorporating literals into knowledge graph embeddings," in ISWC. Springer, 2019, pp. 347–363.
- [39] Y. Liu, H. Li, A. Garcia-Duran, M. Niepert, D. Onoro-Rubio, and D. S. Rosenblum, "Mmkg: Multi-modal knowledge graphs," in European Semantic Web Conference. Springer, 2019, pp. 459–474.
- G. Niu, B. Li, Y. Zhang, S. Pu, and J. Li, "Autoeter: Automated en-[40] tity type representation for knowledge graph embedding," arXiv preprint arXiv:2009.12030, 2020.
- [41] P. Jain, P. Kumar, S. Chakrabarti et al., "Type-sensitive knowledge base inference without explicit type supervision," in Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), 2018, pp. 75–80.
- [42] R. Xie, Z. Liu, and M. Sun, "Representation learning of knowledge graphs with hierarchical types." in IJCAI, 2016, pp. 2965–2971.
- [43] K. Wiharja, J. Z. Pan, M. J. Kollingbaum, and Y. Deng, "Schema aware iterative knowledge graph completion," Journal of Web Semantics, vol. 65, p. 100616, 2020.
- [44] Y. Xue, J. Jin, A. Song, Y. Zhang, Y. Liu, and K. Wang, "Relationbased multi-type aware knowledge graph embedding," Neurocomputing, vol. 456, pp. 11-22, 2021.
- [45] S. Guan, X. Jin, J. Guo, Y. none Wang, and X. Cheng, "Link prediction on n-ary relational data based on relatedness evaluation," IEEE Transactions on Knowledge and Data Engineering, 2021
- [46] Y. Liu, Q. Yao, and Y. Li, "Role-aware modeling for n-ary relational knowledge bases," in WWW, 2021, pp. 2660–2671. [47] Z. Wang, J. Zhang, J. Feng, and Z. Chen, "Knowledge graph and
- text jointly embedding," in EMNLP, 2014, pp. 1591-1601.
- [48] I. Yamada, H. Shindo, H. Takeda, and Y. Takefuji, "Joint learning of the embedding of words and entities for named entity disambiguation," in CoNLL, 2016, pp. 250-259.
- [49] K. Toutanova, D. Chen, P. Pantel, H. Poon, P. Choudhury, and M. Gamon, "Representing text for joint embedding of text and knowledge bases," in EMNLP, 2015, pp. 1499-1509.
- [50] P. Rosso, D. Yang, and P. Cudre-Mauroux, "Revisiting text and knowledge graph joint embeddings: The amount of shared information matters!" in BigData, 2019, pp. 2465-2473.
- [51] A. P. Aprosio, C. Giuliano, and A. Lavelli, "Extending the coverage of dbpedia properties using distant supervision over wikipedia. in NLP-DBPEDIA@ ISWC. CEUR-WS.org, 2013.
- [52] L. Galárraga, S. Razniewski, A. Amarilli, and F. M. Suchanek, "Predicting completeness in knowledge bases," in WSDM, 2017, pp. 375-383.
- [53] R. Zhang, J. Li, J. Mei, and Y. Mao, "Scalable instance reconstruction in knowledge bases via relatedness affiliated embedding," in WWW, 2018, pp. 1185-1194.
- [54] J. Wen, J. Li, Y. Mao, S. Chen, and R. Zhang, "On the representation and embedding of knowledge bases beyond binary relations," in IJCAI, 2016, pp. 1300-1307.
- [55] P. Rosso, D. Yang, and P. Cudré-Mauroux, "Beyond triplets: hyperrelational knowledge graph embedding for link prediction," in WWW, 2020, pp. 1885-1896.
- [56] D. Yang, B. Qu, J. Yang, and P. Cudre-Mauroux, "Revisiting user mobility and social relationships in lbsns: a hypergraph embedding approach," in WWW, 2019, pp. 2147–2157. D. Yang, B. Qu, J. Yang, and P. Cudré-Mauroux, "Lbsn2vec++:
- [57] Heterogeneous hypergraph embedding for location-based social

networks," IEEE Transactions on Knowledge and Data Engineering, vol. 34, no. 4, pp. 1843–1855, 2020.

- [58] A. Hogan, E. Blomqvist, M. Cochez, C. d'Amato, G. de Melo, C. Gutiérrez, S. Kirrane, J. E. Labra Gayo, R. Navigli, S. Neumaier, A.-C. Ngonga Ngomo, A. Polleres, S. M. Rashid, A. Rula, L. Schmelzeisen, J. F. Sequeda, S. Staab, and A. Zimmermann, *Knowledge Graphs*, ser. Synthesis Lectures on Data, Semantics, and Knowledge. Morgan & Claypool, 2021, no. 22. [Online]. Available: https://kgbook.org/
- [59] L. Li, P. Wang, J. Yan, Y. Wang, S. Li, J. Jiang, Z. Sun, B. Tang, T.-H. Chang, S. Wang *et al.*, "Real-world data medical knowledge graph: construction and applications," *Artificial intelligence in medicine*, vol. 103, p. 101817, 2020.
- [60] S. K. Mohamed, V. Nováček, and A. Nounu, "Discovering protein drug targets using knowledge graph embeddings," *Bioinformatics*, vol. 36, no. 2, pp. 603–610, 2020.
- [61] A. Zouaq and F. Martel, "What is the schema of your knowledge graph? leveraging knowledge graph embeddings and clustering for expressive taxonomy learning," in *Proceedings of the international workshop on semantic big data*, 2020, pp. 1–6.
- [62] Schema.org, https://schema.org/, 2022.
- [63] N. Huss, "How many websites are there in the world?" https: //siteefy.com/how-many-websites-are-there/, 2022.
- [64] F. Wu and D. S. Weld, "Automatically refining the wikipedia infobox ontology," in WWW, 2008, pp. 635–644.
- [65] S. Faralli, A. Panchenko, C. Biemann, and S. P. Ponzetto, "The contrastmedium algorithm: Taxonomy induction from noisy knowledge graphs with just a few links," in ACL, 2017.
- [66] M. Pietrasik and M. Reformat, "A simple method for inducing class taxonomies in knowledge graphs," in *European Semantic Web Conference*. Springer, 2020, pp. 53–68.
- [67] J. Hao, M. Chen, W. Yu, Y. Sun, and W. Wang, "Universal representation learning of knowledge bases by jointly embedding instances and ontological concepts," in *KDD*, 2019, pp. 1709–1719.
- [68] P. Cimiano, Ontology learning and population from text: algorithms, evaluation and applications. Springer Science & Business Media, 2006, vol. 27.
- [69] D. Krompaß, S. Baier, and V. Tresp, "Type-constrained representation learning in knowledge graphs," in *ISWC*. Springer, 2015, pp. 640–655.
- [70] S. Wang, K. Fu, X. Sun, Z. Zhang, S. Li, and S. Yan, "Representation learning of knowledge graphs with the interaction between entity types and relations," *Neurocomputing*, vol. 508, pp. 305–314, 2022.
- [71] T. G. Kolda and B. W. Bader, "Tensor decompositions and applications," SIAM review, vol. 51, no. 3, pp. 455–500, 2009.
- [72] D. Q. Nguyen, T. D. Nguyen, D. Q. Nguyen, and D. Phung, "A novel embedding model for knowledge base completion based on convolutional neural network," in *NAACL*, 2018, pp. 327–333.
- [73] S. Guan, X. Jin, Y. Wang, and X. Cheng, "Link prediction on n-ary relational data," in WWW. ACM, 2019, pp. 583–593.
- [74] H. Kamigaito and K. Hayashi, "Comprehensive analysis of negative sampling in knowledge graph representation learning," in *ICML*, 2022, pp. 10 661–10 675.
- [75] D. Yang, B. Qu, R. Hussein, P. Rosso, P. Cudré-Mauroux, and J. Liu, "Revisiting embedding based graph analyses: Hyperparameters matter!" *IEEE Transactions on Knowledge and Data Engineering*, 2022.
- [76] K. Ahrabian, A. Feizi, Y. Salehi, W. L. Hamilton, and A. J. Bose, "Structure aware negative sampling in knowledge graphs," in *EMNLP*, 2020, pp. 6093–6101.
- [77] L. Cai and W. Y. Wang, "Kbgan: Adversarial learning for knowledge graph embeddings," in NAACL, 2018, pp. 1470–1480.
- [78] Y. Zhang, Q. Yao, Y. Shao, and L. Chen, "Nscaching: simple and efficient negative sampling for knowledge graph embedding," in *ICDE*. IEEE, 2019, pp. 614–625.
- [79] Z. Sun, Z.-H. Deng, J.-Y. Nie, and J. Tang, "Rotate: Knowledge graph embedding by relational rotation in complex space," 2019.
- [80] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme, "Bpr: Bayesian personalized ranking from implicit feedback," arXiv preprint arXiv:1205.2618, 2012.
- [81] H. Liu, Y. Wu, and Y. Yang, "Analogical inference for multirelational embeddings," in *ICML*, 2017, pp. 2168–2178.
- [82] S. M. Kazemi and D. Poole, "Simple embedding for link prediction in knowledge graphs," pp. 4284–4295, 2018.
- [83] M. Nickel, L. Rosasco, and T. Poggio, "Holographic embeddings of knowledge graphs," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 30, no. 1, 2016.

- [84] S. Zhang, Y. Tay, L. Yao, and Q. Liu, "Quaternion knowledge graph embeddings," *NIPS*, vol. 32, 2019.
- [85] X. Han, S. Cao, X. Lv, Y. Lin, Z. Liu, M. Sun, and J. Li, "Openke: An open toolkit for knowledge embedding," in *EMNLP*, 2018, pp. 139–144.
- [86] M. Ali, M. Berrendorf, C. T. Hoyt, L. Vermue, M. Galkin, S. Sharifzadeh, A. Fischer, V. Tresp, and J. Lehmann, "Bringing light into the dark: A large-scale evaluation of knowledge graph embedding models under a unified framework," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 1–1, 2021.



Dingqi Yang is an Associate Professor with the State Key Laboratory of Internet of Things for Smart City and the Department of Computer and Information Science, University of Macau. He received his Ph.D. degree in Computer Science from Pierre and Marie Curie University and Institut Mines-TELECOM/TELECOM Sud-Paris in France, where he won both the CNRS SAMOVAR Doctorate Award and the Institut Mines-TELECOM Press Mention in 2015. Before joining the University of Macau, he worked as

a senior researcher at the University of Fribourg in Switzerland. His research interests include big social media data analytics, ubiquitous computing, and smart city applications.



Bingqing Qu is an Assistant Professor at the BNU-HKBU United International College. She received her Ph.D. in Computer Science from the University of Rennes 1 in 2016. Before joining the BNU-HKBU United International College, she worked on multimedia data analytics for the Swiss Federal Institute of Technology (EPFL) and the University of Fribourg, Switzerland. Her research interests include multimedia data mining, social media data analytics, and computer vision.



Paolo Rosso received his Ph.D. degree in Computer Science from the University of Fribourg in Switzerland where he worked on Knowledge Graph embeddings. Before starting his Ph.D., he did the Erasmus traineeship at High Performance Computing Center in Stuttgart (Germany) where he developed a platform to benchmark different architectures for High Performance Computing. His research interests are Semantic Web and Machine Learning.



Philippe Cudre-Mauroux is a Full Professor and the director of the eXascale Infolab at the University of Fribourg in Switzerland. He received his Ph.D. from the Swiss Federal Institute of Technology EPFL, where he won both the Doctorate Award and the EPFL Press Mention. Before joining the University of Fribourg he worked on information management infrastructures for IBM Watson Research, Microsoft Research Asia, and MIT. His research interests are in next-generation, Big Data management

infrastructures for non-relational data. Webpage: http://exascale.info/phil