# RETA: A Schema-Aware, End-to-End Solution for Instance Completion in Knowledge Graphs

Paolo Rosso
paolo.rosso@unifr.ch
University of Fribourg
Switzerland

Dingqi Yang*
dingqiyang@um.edu.mo
State Key Laboratory of Internet of Things for Smart City and Department of Computer and Information Science,
University of Macau
Macao SAR, China

Natalia Ostapuk
natalia.ostapuk@unifr.ch
University of Fribourg
Switzerland

Philippe Cudré-Mauroux
pcm@unifr.ch
University of Fribourg
Switzerland

## ABSTRACT

Knowledge Graph (KG) completion has been widely studied to tackle the incompleteness issue (i.e., missing facts) in modern KGs. A fact in a KG is represented as a triplet $(h, r, t)$ linking two entities $h$ and $t$ via a relation $r$. Existing work mostly consider *link prediction* to solve this problem, i.e., given two elements of a triplet predicting the missing one, such as $(h, r, ?)$. This task has, however, a strong 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*, ?). In addition, the KG completion problem has also been formulated as a *relation prediction* task, i.e., when predicting relations $r$ for a given entity $h$. Without predicting $t$, this task is however a step away from the ultimate goal of KG completion. Against this background, this paper studies an instance completion task suggesting $r$-$t$ pairs for a given $h$, i.e., $(h, ?, ?)$. We propose an end-to-end solution called RETA (as it suggests the Relation and Tail for a given head entity) consisting of two components: a RETA-Filter and RETA-Grader. More precisely, our RETA-Filter first generates candidate $r$-$t$ pairs for a given $h$ by extracting and leveraging the schema of a KG; our RETA-Grader then evaluates and ranks the candidate $r$-$t$ pairs considering the plausibility of both the candidate triplet and its corresponding schema using a newly-designed KG embedding model. We evaluate our methods against a sizable collection of state-of-the-art techniques on three real-world KG datasets. Results show that our RETA-Filter generates of high-quality candidate $r$-$t$ pairs, outperforming the best baseline techniques while reducing by 10.61%-84.75% the candidate size under the same candidate quality guarantees. Moreover, our RETA-Grader also significantly outperforms state-of-the-art link prediction techniques on the instance completion task by 16.25%-65.92% across different datasets.

*Corresponding author


*WWW '21, April 19–23, 2021, Ljubljana, Slovenia*

ACM ISBN 978-1-4503-8312-7/21/04.
https://doi.org/10.1145/3442381.3449883

## CCS CONCEPTS

• **Computing methodologies** → **Knowledge representation and reasoning**.

## KEYWORDS

Knowledge graph embedding, Entity Types, Instance completion

**ACM Reference Format:**
Paolo Rosso, Dingqi Yang, Natalia Ostapuk, and Philippe Cudré-Mauroux. 2021. RETA: A Schema-Aware, End-to-End Solution for Instance Completion in Knowledge Graphs. In *Proceedings of the Web Conference 2021 (WWW '21), April 19–23, 2021, Ljubljana, Slovenia.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3442381.3449883

## 1 INTRODUCTION

Knowledge Graphs (KGs), such as Freebase [5], Wikidata[1] or Google's Knowledge Graph[2], have become a key resource powering a broad spectrum of Web applications, such as semantic search [48], question-answering [51], or recommender systems [54]. A typical KG represents a large collection of entities (real-world objects or abstract concepts) interconnected via relations, where entities contribute to the description of other entities via relations. Using a triplet representation scheme, 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 entity as head in Wikidata. Despite the fact that 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* [31].

In this context, Knowledge Graph completion problems have been widely studied. In the current literature, these problems are mostly formulated as a link prediction task [31, 43], i.e., to predict missing links in a KG. More precisely, given two elements of a triplet, the task is to predict the missing one, such as $(h, r, ?)$,

[1] http://wikidata.org/
[2] https://googleblog.blogspot.com/2012/05/introducing-knowledge-graph-things-not.html

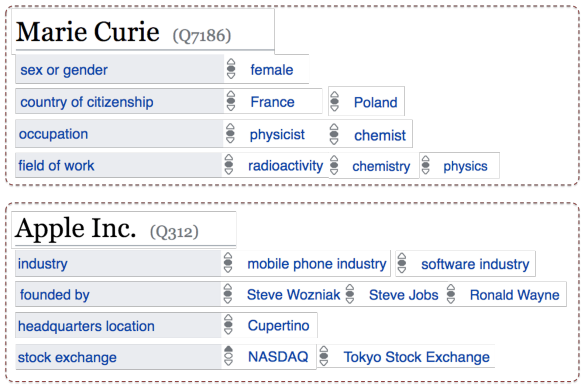


**Figure 1: Examples of entities *Marie Curie* and *Apple Inc.* from Wikidata. For each entity, we present a subset of triplets having that entity as head from Wikidata.**

$(h, ?, t)$ 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. In fact, the two known elements in a triplet must 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*, ?) do not make any sense, 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 many real-world use cases of KG completion where we are not given two correlated elements in a triplet. Another thread of work on KG completion focuses on the relation prediction task[3], suggesting missing relations to a given entity by collaboratively using the information about other entities that are similar to the given entity, for example [2]. However, these techniques only suggest relations for a given head, without predicting the tails and hence are only part of the solution.

In this paper and toward the ultimate goal of KG completion, we tackle 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 a challenging task, 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 the large number of candidate triplets to consider. For example, such an approach takes 350.7 hours (CPU time) even with an efficient link prediction technique TransE on our JF17K dataset (see our experiments for more detail), resulting in poor performance of 0.0097 for recall@5.

A straightforward solution to this problem is to combine the relation prediction and link prediction tasks [7]; for a given $h$, we first predict a set of relations using a relation prediction technique, and then predict $(h, r, ?)$ using a link prediction technique for each predicted relation $r$. Subsequently, with a small set of predicted relations, the number of candidate *r-t* pairs fed to the link prediction technique can be significantly reduced. However, such an approach still shows subpar performance, as it fails to fully consider the triplewise correlation of the three elements in a triplet, in particular the schema information encoded in the entity-typed triplet $(h\_type, r, t\_type)$. Taking real-world examples from Wikidata, a relation prediction technique may suggest two relations, *headquarters location* and *industry*, to the head entity *Apple Inc.*; a link prediction technique then predicts on *(Apple Inc., headquarters location, ?)* and *(Apple Inc., industry, ?)*, resulting in a ranked list of candidate *r-t* pairs. Although such a scheme reduces the number of candidate *r-t* pairs by predicting relevant relations, it still fails to filter out many noisy *r-t* pairs, such as *headquarters location-software industry* or *industry-Mountain View*. In Wikidata, the entities *Apple Inc.*, *software industry* and *Mountain View* are all associated with types (*enterprise*, *economic branch* and *city*, respectively). If we have the schema information represented as entity-typed triplets $(h\_type, r, t\_type)$ —*(enterprise, headquarters location, city)* and *(enterprise, industry, economic branch)*, we could easily filter out such noisy *r-t* pairs that do not match the schema of the KG (to the given $h$).

Against this background and to effectively solve our instance completion problem over KGs $(h, ?, ?)$, we propose an end-to-end solution fully leveraging schema information encoded in triplets. Specifically, our solution consists of two components. First, to effectively filter out noisy *r-t* pairs for a given $h$, we design RETA-Filter, a Relation-Tail pair filter based on entity types. 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 [2]. Therefore, by extracting entity-typed triplets from an input KG, we build a *head_type-relation-tail_type* tensor, which encodes the schema information of the KG. Based on this tensor, we can efficiently search for candidate *r-t* pairs schematically matching a given $h$ via tensor operations. In essence, RETA-Filter outputs a set of coarse-grained candidate *r-t* pairs fitting the schema of the KG. Second, to predict the most promising *r-t* pairs from the filtered candidates, we design RETA-Grader, a Relation-Tail pair grader leveraging a newly designed KG embedding model scoring each candidate *r-t* pair with a particular consideration on the plausibility of both the input triplet and its corresponding schema. To achieve this goal, we design two neural network pipelines. On one hand, we learn from a triplet $(h, r, t)$, generating a triplet relatedness feature vector. On the other hand, we learn from the corresponding entity-typed triplet(s) $(h\_type, r, t\_type)$, generating one or multiple (when $h$ or $t$ has multiple types) relatedness feature vector(s), based on which we then compute a schema relatedness feature vector by taking the minimum value along each feature dimension over these relatedness feature vectors. The basic assumption behind this operation is that for a given valid triplet $(h, r, t)$, the relatedness of

[3]This task is also known as property prediction or recommendation by [7, 24, 52], as these works refer to a triplet by (*entity,property,value*), which is equivalent to our notion of triplet as (*head, relation, tail*).

all its entity-typed triplets ($h\_type, r, t\_type$) should be high; subsequently, the minimum relatedness along each feature dimension is expected to be high. Afterward, we fed the concatenated triplet and schema relatedness feature vectors to a fully connected projection to output the final predicted score of the input triplet $(h, r, t)$.

Our contributions can be summarized as follows:

- We revisit the impractical settings of existing approaches to KG completion, and propose to study a novel instance completion problem with more realistic settings, i.e., predicting relation-tail pairs given a head $(h, ?, ?)$.
- We propose an end-to-end solution to our instance completion problem. We first propose RETA-Filter, a $r$-$t$ pair filter based on entity-typed triplets, generating a set of coarse-grained candidate $r$-$t$ pairs matching the schema of the KG (to the given $h$). We then propose RETA-Grader, a $r$-$t$ pair grader leveraging a newly designed KG embedding model scoring each candidate $r$-$t$ pair with a particular consideration on the plausibility of both the input triplet and its corresponding schema.
- 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 our RETA-Filter is able to generate a set of high-quality candidate $r$-$t$ pairs for a given $h$, outperforming the best baseline techniques with 10.61%-84.75% reduction on the candidate set size, under the same candidate quality guarantees. Moreover, our RETA-Grader also significantly outperforms state-of-the-art link prediction techniques on the instance completion task by 16.25%-65.92% across different datasets.

## 2 RELATED WORK

Knowledge Graph completion predicts missing facts in a KG. In the following, we briefly discuss existing work implementing one of the three types of tasks for KG completion: 1) link prediction tasks suggesting one missing element in a triplet; 2) relation prediction tasks predicting missing relations to a given entity; and 3) instance completion task suggesting relation-tail pairs to a given head entity.

### 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, i.e., $(h, r, ?)$, $(h, ?, t)$ or $(?, r, t)$. To solve this problem, early work resorted to rule/feature-based relational learning, such as association rules [11] or hand-crafted features [25, 33] (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 that exploit distance-based scoring functions to create the embeddings. One representative model of this family is TransE [6], which learns from triplets $(h, r, t)$ such that the relation between the head and tail are preserved as $h + r \approx t$. Several works further improve TransE to capture richer KG structures, such as involving relation-specific hyperplanes [45] or spaces [9, 17, 26], for example. Second, semantic matching models that exploit similarity-based scoring functions. One typical model in that context is RESCAL [32], which represents each entity as a vector and each relation as a matrix, and then uses a bilinear function to model the relation between two entities. Several works extend RESCAL by reducing the complexity of the models [38, 50] or of the training processes [53], by improving the model expressiveness [4], by capturing asymmetric relations [42], by considering unbalanced data in KGs [18], or by modeling non-linear relations using neural networks [3, 8, 30, 37, 39].

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 [12, 22], images [28] or types [16, 34, 47] 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) [36, 41, 44, 49].

However, we argue in this paper that such link prediction tasks are impractical in the sense that 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 [1, 2, 7, 10, 24, 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. [52] built a "property suggester" for Wikidata, suggesting candidate properties based on association rules learnt from existing triplets in Wikidata. Lajus et al. [24] 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. [7] 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 [2] 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 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, 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 [7], which suggests relation-tail pairs to a long-tail 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. To the best of our knowledge, this is the only work in the current literature considering the instance completion task, i.e., predicting relation-tail pairs to a given head. 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 [55] for n-ary (or multi-fold) 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 [46], 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. Therefore, it fundamentally differs from our instance completion task, where we do not assume any information about the relations.

## 3 SCHEMA-AWARE INSTANCE COMPLETION

To solve our instance completion problem on KGs, we propose an end-to-end solution leveraging the explicit and implicit schema information encoded through the triplets. More precisely, we first design RETA-Filter, a $r$-$t$ pair filter based on entity-typed triplets, generating a set of coarse-grained candidate $r$-$t$ pairs matching the schema of the KG (to the given $h$). We then propose RETA-Grader, a $r$-$t$ pair grader leveraging a newly designed KG embedding model scoring and ranking these candidate $r$-$t$ pairs with a particular consideration on the plausibility of the triplets and their corresponding schema.

### 3.1 RETA-Filter

Our RETA-Filter is designed to generate a set of coarse-grained $r$-$t$ pairs by extracting and leveraging the schema of an input 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 $h\_type_1, h\_type_2, ... \in T$, 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 characterized by a set of entity-typed 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 guideline

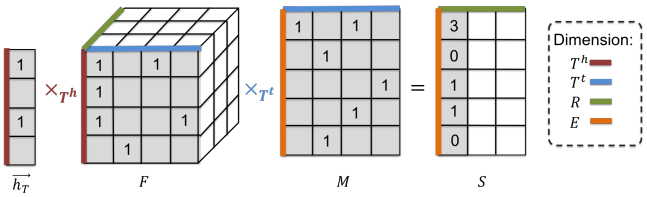


**Figure 2: A toy example of our filter, where we highlight an example of computation for one relation (in gray). Given an head entity $h$, RETA-Filter fetches its type vector $\overrightarrow{h_T}$ and multiply it with the tensor $\mathbf{F}$. The resulting matrix is then multiplied with the type matrix $\mathbf{M}_T$ in order to compute the candidate $r$-$t$ pairs for $h$.**

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

To implement our 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 $t$ for 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 $\mathbf{F}$, where the first and third dimensions of $\mathbf{F}$ refer to head and tail types, respectively. Finally, by representing $h$ using its type vector $\overrightarrow{h_T} \in \mathbb{B}^{|T_h|}$ 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:

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

where $\times_n$ denotes the mode-$n$ tensor product [21]. 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 2 shows a toy example of our filter. To implement this filter, we have to take into account the following two practical considerations:

- **Frequency of entity-typed triplets.** Entity-typed triplets $(h\_type, r, t\_type)$ encode the schema of the corresponding KG. When converting the triplets of a 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 $\alpha$) to build $\mathbf{F}$. However, a too high value of $\alpha$ could also remove useful entity-typed triplets, resulting in an incomplete schema captured by $\mathbf{F}$. In other words, $\alpha$ 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 $\alpha$ could include noisy entity-typed triplets in $\mathbf{F}$, resulting in a large candidate set with noisy and redundant $r$-$t$

pairs. On the other hand, a too high value of $\alpha$ 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 $\alpha$), 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 (frequent) 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 2, 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 $\beta$ as candidate *r-t* pairs. Obviously, a lower value of $\beta$ selects more candidate *r-t* pairs and thus leads to a higher coverage of true *r-t* pairs being included in the candidate set, and vice versa. In essence, $\beta$ directly balances the tradeoff between the size of the candidate (*r-t* pair) set and its coverage.

In summary, our RETA-Filter is designed to efficiently generate a set of coarse-grained *r-t* pairs by matching with the KG schema extracted from the entity-typed triplets of a KG. We use two tunable parameters $\alpha$ and $\beta$ balancing the tradeoff between the size of the candidate (*r-t* pair) set and its coverage. By varying $\alpha$ and $\beta$, our RETA-Filter is able to 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).

## 3.2 RETA-Grader

Based on the set of candidate *r-t* pairs provided by our RETA-Filter, our 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 3 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.

*3.2.1 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 [8, 29]. As shown in Figure 3, 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 $3 \times 3$. We set the filter size to $3 \times 3$ to capture the triple-wise 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.

*3.2.2 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. To capture such schema information, 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 $\{(h\_type_i, r, t\_type_j) | 1 \le i \le m, 1 \le j \le n\}$. Afterward, we learn from each of these entity-typed triplet $(h\_type_i, r, t\_type_j)$ to generate its relatedness feature vector, using a similar method as for learning from triplets. Specifically, as shown in Figure 3, 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 $3\times3$ to capture the triple-wise relatedness between $\overrightarrow{h\_type_i}$, $\vec{r}$, and $\overrightarrow{t\_type_j}$; 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 $mn$ entity-typed 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* $\vec{\psi}$. 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 [13].

*3.2.3 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 $\sigma$ from the overall relatedness feature vector.

*3.2.4 RETA-Grader Training Process.* To train our RETA-Grader model, we minimize a softplus loss, which is defined as the negative

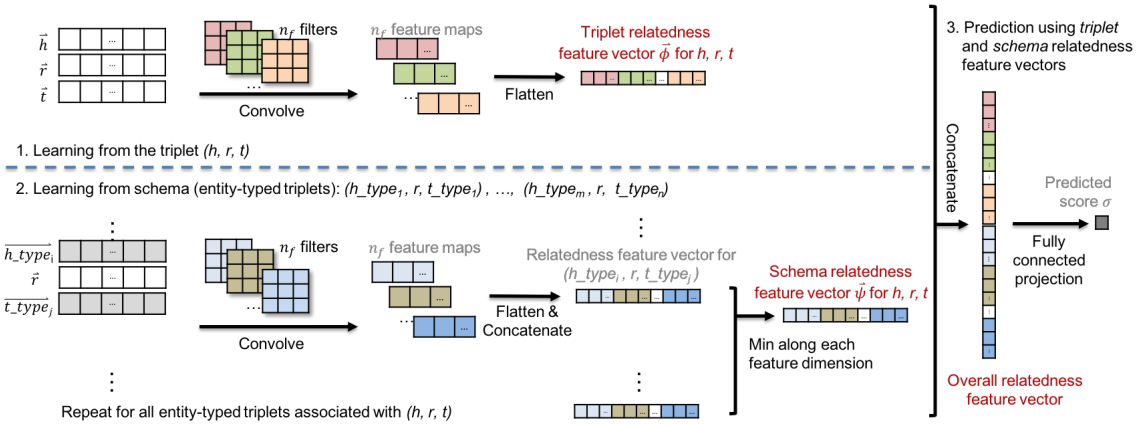


**Figure 3: Architecture of RETA-Grader**

log-likelihood of the logistic model:

$$\sum_{\omega \in \Omega} log(1 + e^{-\sigma(\omega)}) + log(1 + e^{\sigma(\omega')}) \quad (2)$$

where $\Omega$ represents the set of training triplets. For each triplet $\omega = (h, r, t)$, one negative sample $\omega'$ is generated by randomly corrupting one element in the triplet $h$, $r$ or $t$. $\sigma(\omega)$ and $\sigma(\omega')$ denote the predicted score of our RETA-Grader model for the true fact $\omega$ and the negative fact $\omega'$, respectively. The loss function in Eq. 2 is minimized using the Adam optimizer [20], and the model parameters are learnt via back propagation. We use rectified linear units (ReLU) as the activation function [23] and batch normalization [15] after the two CNN layers for fast training.

*3.2.5 Practical considerations on design choices.* To implement our RETA-Grader, we have to take into account the following two practical considerations:

- **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 $m$ and $n$ types, respectively, the second module in Figure 3 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. We empirically show in our experiments below that a small value of $k$ is able to achieve a good performance on our instance completion task.
- **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 in our instance completion task, 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, $\vec{h_T}$ 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 $\mathbf{M}$ 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, $\vec{h_T}$ and the corresponding row of $\mathbf{M}$ 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 our RETA-Grader learn from entities without types, we assign an "unknown" type to such entities, and then keep the same processing 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 $\{(unknown\_type, r, t\_type_j) | 1 \leq j \leq n\}$. 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 3 is able to automatically learn more from the triplet relatedness feature vector to make predictions.

The implementation of RETA and used datasets are available here[4].

## 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 [46] and [6], 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)[5]. As our instance completion task

[4]https://github.com/eXascaleInfolab/RETA_code/
[5]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"

**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 on both $h$ and $t$ | 19,015 | 144,117 | 87,150 |
| #Fact w/ types on either $h$ or $t$ | 322 | 10,776 | 21,049 |
| #Fact w/o types | 5 | 23 | 0 |

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 our proposed method against a sizable collection of state-of-the-art techniques from the following three categories.

- *Relation (property) prediction techniques.* **BPR** [35] 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**) [52] recommends relations to an entity using association rules learnt from existing triplets in Wikidata; this technique is an API[6] on Wikidata, and thus can only be applied to our HumanWiki dataset. **Recoin** [2] 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** [7] predicts relations associated with an entity using an attention-based graph neural network model[7]. 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 $N$ actually 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 relations 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 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 [55] for n-ary relational facts where an n-ary relation links multiple entities $\{e_1, e_2, e_3, ..., e_n\}$); 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 [55] 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 [55]. First, a relatedness affiliated embedding (**RAE**) model, which learns a multi-layer perceptron neural network to predict a relatedness score between two entities, and considers them to be relevant if the score is higher than a threshold $\gamma$. Tuning $\gamma$ balances the tradeoff between the size and the coverage of the resulting candidate ($r$-$t$ pair) set. Second, a schema based predictor (**Sch**), which 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.* Based on a set of candidate $r$-$t$ pairs for a given $h$, we apply the following link prediction techniques. First, we consider classical link prediction techniques evaluating the plausibility of a triplet. The translational distance models we consider include: **TransE** [6], which learns to preserve the relation between two entities as $h + r \approx t$; **TransH** [45], which extends TransE to better capture multi-mapping relations using relation-specific hyperplanes; **TransR** [26], which introduces relation-specific projections to also better capture multi-mapping relations; **TransD** [17], which further extends 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** [32], 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** [50], which simplifies Rescal by representing each relation embedding as a diagonal matrix; **ComplEx** [42], which further extends DistMult in the complex space in order to better model both symmetric and asymmetric relations; **Analogy** [27], which explicitly models analogical structures in multi-relational KG embeddings; **SimplE** [19], which is an expressive and interpretable KG embedding techniques based on canonical polyadic tensor decomposition; **RotatE** [40], which defines a relation as a rotation from a head to a tail in the complex vector space, capturing richer relation patterns. For each of these link prediction technique, we use the hyperparameters as recommended by [14]. We also consider a variation of RETA-Grader as an additional baseline, where we learn from triplets only, without learning from the schema (entity-typed triplets); we refer to this baseline as **RETA-Grader (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

instances on Wikidata, using prior knowledge for better performance on relation prediction.

[6] https://www.wikidata.org/w/api.php?action=help&modules=wbsgetsuggestions

[7] 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).

both the input triplet and its corresponding entity-typed triplets. **TypeDM** and **TypeComplex** [16] are extended from DistMult and ComplEx, respectively, by explicitly modeling entity type compatibility.

For our RETA-Grader, we set the number of types learnt per entity $k = 2$, 1 and 4, and the number of Filters $n_f = 50$, 200 and 100 for JF17K, FB15K and HumanWiki, respectively. More details on these parameter sensitivity study will be present later.

*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 generated 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 higher 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 technique* (**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 $\gamma$ for RAE when applicable), we balances the tradeoff between the size and the coverage of the resulting candidate (*r-t* pair) set. For our method, we tune $\alpha$ and $\beta$ to balance this tradeoff.

Second, by fixing the set of candidate *r-t* pairs ensuring 95% coverage using our RETA-Filter, we evaluate the performance of different (link prediction) techniques and our RETA-Grader when ranking these candidate *r-t* pairs. We report Recall (Rec), Mean Average Precision (MAP), and Normalized Discounted Cumulative Gain (NDCG).

## 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. Figure 4 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 indeed 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 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.

**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 |

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 a 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 *r-t* 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 best-performing 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.

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

Based on the candidate *r-t* pair set generated by our RETA-Filter, we then evaluate the performance of our RETA-Grader on our instance completion task. Table 3 shows the results. We observe that our RETA-Grader consistently outperforms all baseline techniques on our instance completion task in general. Taking MAP as an example, it yields an improvement of 16.25%, 65.92% and 35.58% over the best performing baselines on JF17K, FB15K and HumanWiki, respectively. Moreover, compared to RETA-Grader (no type), which is the

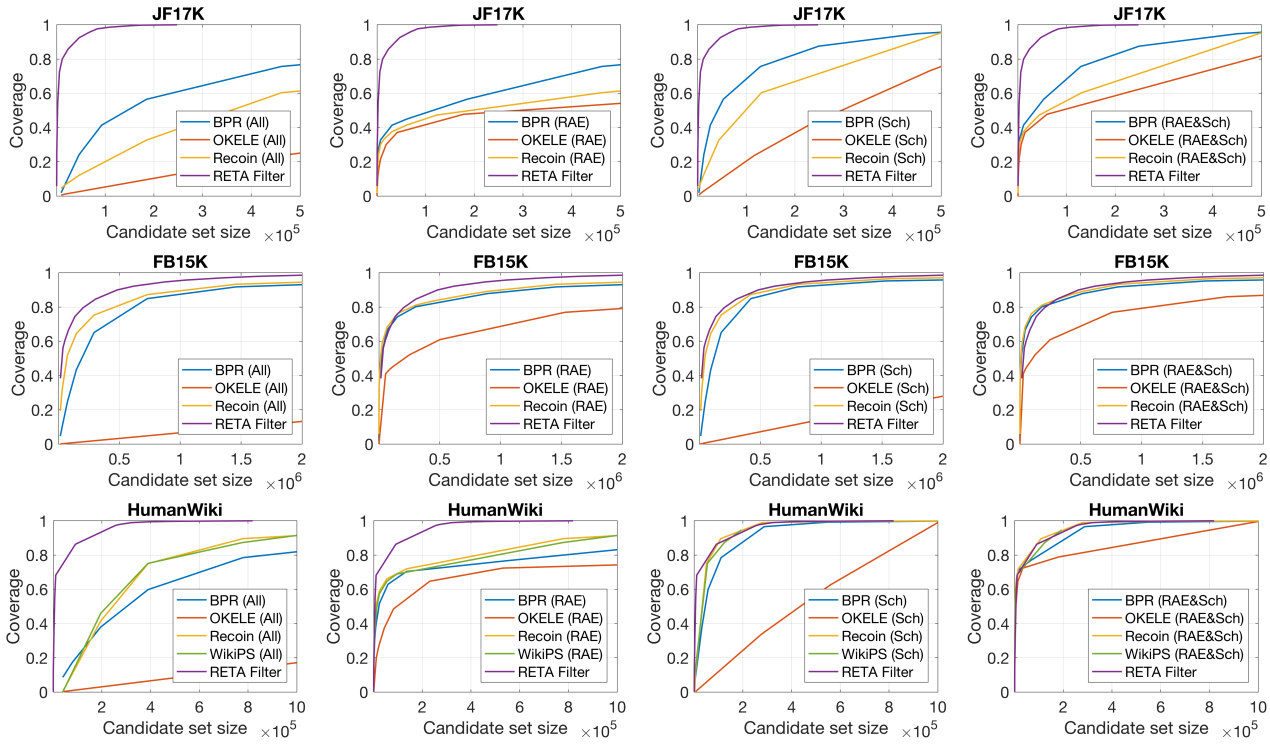


**Figure 4: Tradeoff between the size and the coverage of the resulting candidate ($r$-$t$ pair) set from different filtering techniques. Note that our RETA-Filter has a unique tradeoff line on each dataset, as it does not use any tail candidate refinement techniques.**

**Table 3: Performance of different methods on our instance completion task. We highlight the top-2 best-performing techniques for each metric.**

| Method | JF17K | | | | FB15K | | | | HumanWiki | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 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 |
| RETA-Grader (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 |
| 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-Grader | **0.1916** | **0.1153** | **0.0615** | **0.1855** | **0.2104** | **0.1288** | **0.1037** | **0.2658** | **0.2049** | **0.1545** | **0.1166** | **0.2332** |

variant of our proposed model without learning from the schema, our RETA-Grader learning from both triplets and their corresponding schema significantly achieves better performance; this shows that learning from entity-typed triplets is indeed helpful for the instance completion task. Compared to link prediction techniques using additional information about entity types, our RETA-Grader yields better performance by evaluating the plausibility of both the input triplet and its corresponding schema using a subtly designed KG embedding model. Note that TypeComplex, which further considers entity types on top of the ComplEx model, underperforms ComplEx; opposite results are reported in [16]. This is due to the different implementations of the ComplEx model. More precisely,

**Table 4: Performance of RETA-Grader on different test facts.**

| Test Facts | JF17K | | FB15K | | HumanWiki | |
|---|---|---|---|---|---|---|
| | MAP | NDCG | MAP | NDCG | MAP | NDCG |
| *known types* | 0.0614 | 0.1857 | 0.1075 | 0.2780 | 0.1440 | 0.2912 |
| *unknown type* | 0.0651 | 0.1577 | 0.0790 | 0.1846 | 0.0608 | 0.1154 |

we tested two different implementation of ComplEx from [16] and [14], respectively, and report the results from the best-performing implementation from [14] in this paper.

Moreover, we further investigate the performance of our RETA-Grader 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 4 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 discriminative for prediction.

We note that the results of *known type* in Table 4 are very close to the results on all test facts from Table 3 (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).

## 4.4 Parameter Sensitivity Study

We also study the impact of two key parameters in our RETA-Grader, i.e., the number of types learnt per entity $k$ and the number of filters $n_f$, on both instance completion performance and evaluation time. 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. In addition, these test facts also dominate the set of all test facts.

*4.4.1 Number of types learnt per entity by RETA-Grader.* 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 entities with many types, the size of this entity-typed 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 evaluation time. Figure 5 shows the results.

On one hand, we observe that a small $k$ is able to achieve a good performance on our 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 evaluation time 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 5). Note that on HumanWiki, both instance completion performance and evaluation time 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 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 $k = 2$, 1 and 4 for JF17K, FB15K and HumanWiki, respectively, for all other experiments.

*4.4.2 Number of Filters.* We also investigate the impact of the number of filters $n_f$ used by RETA-Grader. Figure 6 shows the results. We observe that the optimal $n_f$ varies across datasets, while the evaluation time monotonically increases with $n_f$. Therefore, in all other experiments, we selected the best performing $n_f = 50$, 200 and 100 for JF17K, FB15K and HumanWiki, respectively.

## 5 CONCLUSION

By revisiting the KG completion task, 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, we propose an end-to-end solution consisting of two components: the RETA-Filter and RETA-Grader. More precisely, our RETA-Filter first generates candidate $r$-$t$ pairs for a given $h$ by extracting and leveraging schema information from the KG; our RETA-Grader then evaluates and ranks these candidate $r$-$t$ pairs considering the plausibility of both the candidate triplet and its corresponding schema using a newly designed KG embedding model. We conduct a thorough evaluation of our proposed techniques compared to a sizable collection of state-of-the-art techniques on three real-world KG datasets. Results show that our RETA-Filter is able to generate a set of high-quality candidate $r$-$t$ pairs for a given $h$, outperforming the best baseline techniques with 10.61%-84.75% reduction on the candidate set size under the same candidate quality guarantee. Moreover, our RETA-Grader also significantly outperforms state-of-the-art link prediction techniques on the instance completion task by 16.25%-65.92% across different datasets. In future work, improving the generation of negative samples by corrupting $r$-$t$ pairs may further improve the performance of RETA-Grader.

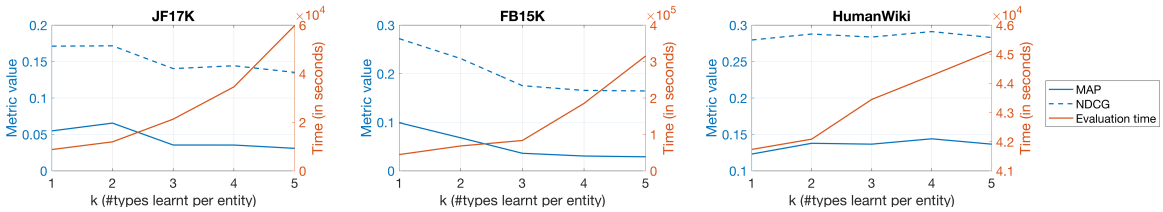


**Figure 5: Impact of the number of types learnt per entity $k$ on instance completion performance and evaluation time**

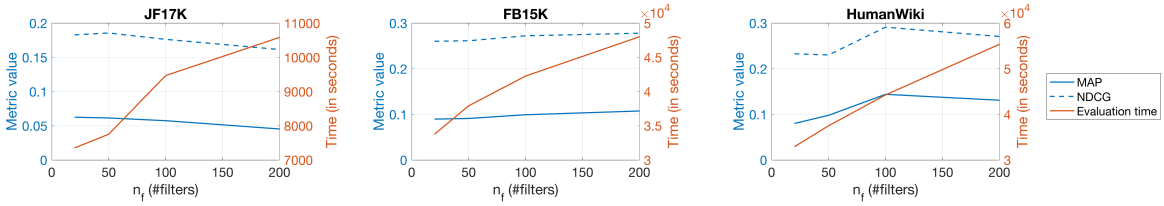


**Figure 6: Impact of the number of filters $n_f$ on instance completion performance and evaluation time**

## ACKNOWLEDGMENTS

This project has received funding from the European Research Council (ERC, grant agreement 683253/GraphInt), and from the Science and Technology Development Fund, Macau SAR (SKL-IOTSC-2021-2023).

## REFERENCES

[1] Alessio Palmero Aprosio, Claudio Giuliano, and Alberto Lavelli. 2013. Extending the Coverage of DBpedia Properties using Distant Supervision over Wikipedia.. In *NLP-DBPEDIA@ ISWC*. CEUR-WS.org.
[2] Vevake Balaraman, Simon Razniewski, and Werner Nutt. 2018. Recoin: relative completeness in Wikidata. In *WWW Companion*. 1787–1792.
[3] Ivana Balazevic, Carl Allen, and Timothy M Hospedales. 2018. Hypernetwork Knowledge Graph Embeddings. *arXiv preprint arXiv:1808.07018* 11731 (2018).
[4] Ivana Balažević, Carl Allen, and Timothy M Hospedales. 2019. Tucker: Tensor factorization for knowledge graph completion. In *EMNLP-IJCNLP)*. 5185––5194.
[5] Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *ACM SIGMOD/PODS*. ACM, ACM, 1247–1250.
[6] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Advances in neural information processing systems*. 2787–2795.
[7] Ermei Cao, Difeng Wang, Jiacheng Huang, and Wei Hu. 2020. Open Knowledge Enrichment for Long-tail Entities. In *WWW*. 384–394.
[8] Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. 2017. Convolutional 2d knowledge graph embeddings. In *AAAI*. 1811–1818.
[9] Takuma Ebisu and Ryutaro Ichise. 2018. Toruse: Knowledge graph embedding on a lie group. In *AAAI*.
[10] Luis Galárraga, Simon Razniewski, Antoine Amarilli, and Fabian M Suchanek. 2017. Predicting completeness in knowledge bases. In *WSDM*. 375–383.
[11] Luis Antonio Galárraga, Christina Teflioudi, Katja Hose, and Fabian Suchanek. 2013. AMIE: association rule mining under incomplete evidence in ontological knowledge bases. In *WWW*. 413–422.
[12] Alberto Garcia-Duran and Mathias Niepert. 2017. Kblrn: End-to-end learning of knowledge base representations with latent, relational, and numerical features. *arXiv preprint arXiv:1709.04676* (2017).
[13] Saiping Guan, Xiaolong Jin, Yuanzhuo Wang, and Xueqi Cheng. 2019. Link Prediction on N-ary Relational Data. In *The World Wide Web Conference*. ACM, 583–593.
[14] Xu Han, Shulin Cao, Xin Lv, Yankai Lin, Zhiyuan Liu, Maosong Sun, and Juanzi Li. 2018. Openke: An open toolkit for knowledge embedding. In *EMNLP*. 139–144.
[15] Sergey Ioffe and Christian Szegedy. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167* (2015).
[16] Prachi Jain, Pankaj Kumar, Soumen Chakrabarti, et al. 2018. 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)*. 75–80.
[17] Guoliang Ji, Shizhu He, Liheng Xu, Kang Liu, and Jun Zhao. 2015. Knowledge graph embedding via dynamic mapping matrix. In *ACL and IJCNLP*, Vol. 1. 687–696.
[18] Guoliang Ji, Kang Liu, Shizhu He, and Jun Zhao. 2016. Knowledge Graph Completion with Adaptive Sparse Transfer Matrix.. In *AAAI*, Vol. 16. 985–991.
[19] Seyed Mehran Kazemi and David Poole. 2018. SimplE Embedding for Link Prediction in Knowledge Graphs. , 4284–4295 pages.
[20] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).
[21] Tamara G Kolda and Brett W Bader. 2009. Tensor decompositions and applications. *SIAM review* 51, 3 (2009), 455–500.
[22] Agustinus Kristiadi, Mohammad Asif Khan, Denis Lukovnikov, Jens Lehmann, and Asja Fischer. 2019. Incorporating literals into knowledge graph embeddings. In *ISWC*. Springer, 347–363.
[23] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *NIPS*. 1097–1105.
[24] Jonathan Lajus and Fabian M Suchanek. 2018. Are all people married? Determining obligatory attributes in knowledge bases. In *WWW*. 1115–1124.
[25] Ni Lao, Tom Mitchell, and William Cohen. 2011. Random walk inference and learning in a large scale knowledge base. In *EMNLP*. 529–539.
[26] Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. 2015. Learning entity and relation embeddings for knowledge graph completion.. In *AAAI*, Vol. 15. 2181–2187.
[27] Hanxiao Liu, Yuexin Wu, and Yiming Yang. 2017. Analogical inference for multi-relational embeddings. In *ICML*. 2168–2178.
[28] Ye Liu, Hui Li, Alberto Garcia-Duran, Mathias Niepert, Daniel Onoro-Rubio, and David S Rosenblum. 2019. MMKG: Multi-modal Knowledge Graphs. In *European Semantic Web Conference*. Springer, 459–474.
[29] Dai Quoc Nguyen, Tu Dinh Nguyen, Dat Quoc Nguyen, and Dinh Phung. 2018. A novel embedding model for knowledge base completion based on convolutional neural network. In *NAACL*. 327––333.
[30] Dai Quoc Nguyen, Thanh Vu, Tu Dinh Nguyen, Dat Quoc Nguyen, and Dinh Phung. 2018. A Capsule Network-based Embedding Model for Knowledge Graph Completion and Search Personalization. *arXiv preprint arXiv:1808.04122* (2018).
[31] Maximilian Nickel, Kevin Murphy, Volker Tresp, and Evgeniy Gabrilovich. 2015. A review of relational machine learning for knowledge graphs. *Proc. IEEE* 104, 1 (2015), 11–33.
[32] Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. 2011. A Three-Way Model for Collective Learning on Multi-Relational Data.. In *ICML*, Vol. 11. 809–816.
[33] Mathias Niepert. 2016. Discriminative gaifman models. In *NIPS*. 3405–3413.
[34] Guanglin Niu, Bo Li, Yongfei Zhang, Shiliang Pu, and Jingyang Li. 2020. AutoETER: Automated Entity Type Representation for Knowledge Graph Embedding. *arXiv preprint arXiv:2009.12030* (2020).
[35] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2012. BPR: Bayesian personalized ranking from implicit feedback. *arXiv preprint arXiv:1205.2618* (2012).

[36] Paolo Rosso, Dingqi Yang, and Philippe Cudre-Mauroux. 2019. Revisiting Text and Knowledge Graph Joint Embeddings: The Amount of Shared Information Matters!. In *BigData*. 2465–2473.
[37] Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne Van Den Berg, Ivan Titov, and Max Welling. 2018. Modeling relational data with graph convolutional networks. In *European Semantic Web Conference*. Springer, 593–607.
[38] Baoxu Shi and Tim Weninger. 2017. ProjE: Embedding projection for knowledge graph completion. In *AAAI*. 1236–1242.
[39] Richard Socher, Danqi Chen, Christopher D Manning, and Andrew Ng. 2013. Reasoning with neural tensor networks for knowledge base completion. In *NIPS*. 926–934.
[40] Zhiqing Sun, Zhi-Hong Deng, Jian-Yun Nie, and Jian Tang. 2019. Rotate: Knowledge graph embedding by relational rotation in complex space.
[41] Kristina Toutanova, Danqi Chen, Patrick Pantel, Hoifung Poon, Pallavi Choudhury, and Michael Gamon. 2015. Representing text for joint embedding of text and knowledge bases. In *EMNLP*. 1499–1509.
[42] Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. 2016. Complex embeddings for simple link prediction. In *ICML*. 2071–2080.
[43] Quan Wang, Zhendong Mao, Bin Wang, and Li Guo. 2017. Knowledge graph embedding: A survey of approaches and applications. *TKDE* 29, 12 (2017), 2724–2743.
[44] Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014. Knowledge graph and text jointly embedding. In *EMNLP*. 1591–1601.
[45] Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014. Knowledge Graph Embedding by Translating on Hyperplanes.. In *AAAI*, Vol. 14. 1112–1119.
[46] Jianfeng Wen, Jianxin Li, Yongyi Mao, Shini Chen, and Richong Zhang. 2016. On the representation and embedding of knowledge bases beyond binary relations. In *IJCAI*. AAAI Press, 1300–1307.
[47] Ruobing Xie, Zhiyuan Liu, and Maosong Sun. 2016. Representation Learning of Knowledge Graphs with Hierarchical Types.. In *IJCAI*. 2965–2971.
[48] Chenyan Xiong, Russell Power, and Jamie Callan. 2017. Explicit semantic ranking for academic search via knowledge graph embedding. In *WWW*. 1271–1279.
[49] Ikuya Yamada, Hiroyuki Shindo, Hideaki Takeda, and Yoshiyasu Takefuji. 2016. Joint learning of the embedding of words and entities for named entity disambiguation. In *CoNLL*. 250–259.
[50] Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. 2015. Embedding entities and relations for learning and inference in knowledge bases. In *ICLR*.
[51] Scott Wen-tau Yih, Ming-Wei Chang, Xiaodong He, and Jianfeng Gao. 2015. Semantic parsing via staged query graph generation: Question answering with knowledge base. In *ACL and IJCNLP*. 1321–1331.
[52] Eva Zangerle, Wolfgang Gassler, Martin Pichl, Stefan Steinhauser, and Günther Specht. 2016. An empirical evaluation of property recommender systems for Wikidata and collaborative knowledge bases. In *OpenSym*. 1–8.
[53] Chuxu Zhang, Huaxiu Yao, Chao Huang, Meng Jiang, Zhenhui Li, and Nitesh V Chawla. 2020. Few-Shot Knowledge Graph Completion. In *AAAI*. 3041–3048.
[54] Fuzheng Zhang, Nicholas Jing Yuan, Defu Lian, Xing Xie, and Wei-Ying Ma. 2016. Collaborative knowledge base embedding for recommender systems. In *KDD*. ACM, 353–362.
[55] Richong Zhang, Junpeng Li, Jiajie Mei, and Yongyi Mao. 2018. Scalable instance reconstruction in knowledge bases via relatedness affiliated embedding. In *Proceedings of the 2018 World Wide Web Conference*. International World Wide Web Conferences Steering Committee, 1185–1194.