

Schema Aware Iterative Knowledge Graph Completion[★]

Kemas Wiharja^{a,b}, Jeff Z. Pan^{a,c}, Martin J. Kollingbaum^d and Yu Deng^e

^aDepartment of Computing Science, University of Aberdeen, United Kingdom

^bSchool of Computing, Telkom University, Indonesia

^cSchool of Informatics, University of Edinburgh, United Kingdom

^dLakeside Labs, Austria

^eIBM Research, United States

ARTICLE INFO

Keywords:

Knowledge Graph Completion
Schema Aware
Knowledge Graph Representation
Knowledge Graph Reasoning
Approximate Reasoning
Consistency Checking
SHACL constraint
Correctness and coverage

ABSTRACT

Recent success of Knowledge Graph has spurred widespread interests in methods for the problem of Knowledge Graph completion. However, efforts to understand the quality of the candidate triples from these methods, in particular from the schema aspect, have been limited. Indeed, most existing Knowledge Graph completion methods do not guarantee that the expanded Knowledge Graphs are consistent with the ontological schema of the initial Knowledge Graph.

In this work, we challenge the silver standard method, by proposing the notion of schema-correctness. A fundamental challenge is how to make use of different types of Knowledge Graph completion methods together to improve the production of schema-correct triples. To address this, we analyse the characteristics of different methods and propose a schema aware iterative approach to Knowledge Graph completion.

Our main findings are: (i) Some popular Knowledge Graph completion methods have surprisingly low schema-correctness ratio; (ii) Different types of Knowledge Graph completion methods can work with each other to help overcome individual limitations; (iii) Some iterative sequential combinations of Knowledge Graph completion methods have significantly better schema-correctness and coverage ratios than other combinations; (iv) All the MapReduce based iterative methods outperform involved single-pass methods significantly over the tested Knowledge Graphs in terms of productivity of schema-correct triples.

Our findings and infrastructure can help further work on evaluating Knowledge Graph completion methods, more fine-grained approaches for schema aware iterative knowledge graph completion, as well as new approximate reasoning approaches based Knowledge Graph completion methods.

1. Introduction

From the point of view of description logics, a Knowledge Graph (KG) can be seen as an ontology with an entity-centric view, consisting of a set of interconnected typed entities and their attributes, as well as some schema axioms for defining the vocabulary (terminology) used in the KG [55, 50]. In other words, a Knowledge Graph consists of a set of triples describing entities and the relationships among these entities. A subset of these triples provides information or descriptions about the types of entities recorded in the Knowledge Graph, summarily regarded as the *schema* of the Knowledge Graph (or the TBox of the ontology represented by the Knowledge Graph). Knowledge Graphs usually grow fast by incorporating new triples. They exhibit a certain “dynamicity”. In order to capture changes in the real world, Knowledge Graphs must be robust in terms of knowledge (triples) being added and deleted. This dynamic nature of Knowledge Graphs raises problems of completeness and correctness that have to be met accordingly. Moreover, as pointed out in [56], Knowledge Graphs usually may not reach “full coverage”, which means that they may not “contain information about each and every entity in the universe”.


However, completeness is an important property when it comes to, for example, supporting query answering.

Completeness and correctness, therefore, are two of the most prominent issues in adding new triples into a Knowledge Graph. Completeness refers to the coverage of the needed information for a given task or application by a target Knowledge Graph. We base our notion of correctness on the notion of consistency from [43]. In addition, we adopt the spirit of SHACL (the W3C Shapes Constraint Language for validating RDF graphs)¹ on considering domain and range also as constraints.

In this paper, we investigate the problem of schema aware Knowledge Graph completion. We refer methods that can produce new triples to complete a Knowledge Graph as *triple producers*. There are two kinds of completion services for Knowledge Graphs, one for adding types of entities, called *type prediction*, while the other for adding relations between entities, called *link prediction* [31]. The latter one is more challenging, according to a recent survey [56]. In this paper, we mainly focus on link prediction related triple producers. Under this category, there are at least two typical types of triple producers, such as Knowledge Graph embedding and Rule learning. In *schema aware Knowledge Graph completion*, schema can be used by type producers to infer correct new triples.

Triple producers based on Knowledge Graph embedding

[★]This document is partially supported by the research project funded by Lembaga Pengelola Dana Pendidikan (LPDP), the Ministry of Finance of Indonesia. See the Acknowledgement section for other supports.

 bagindokemas@telkomuniversity.ac.id (K. Wiharja)

ORCID(s): 0000-0002-3660-8128 (K. Wiharja)

¹<https://www.w3.org/TR/shacl/#core-components-range>

(KGE), such as TransE [7], TransR [38], TransH [75], STransE [48] have seen a recent rise in popularity. The idea is to perform Knowledge Graph completion by learning an embedded representation of entities and relations in Knowledge Graphs. These representations are then used to predict missing entities or relationships. In addition to these KGE triple producers, one can also complete a Knowledge Graph by using rule learning methods. These type of triple producers identify or “learn” rules from an existing set of triples (of a Knowledge Graph) through the identification of relations between triples. These rules are then, again, applied to this set of triples, in order to generate new triples. ([72, 25, 81]).

More recently, there have been some efforts to combine Knowledge Graph embedding triple producers with rule learning triple producers to complete a Knowledge Graph. Such a combination is used to achieve two different objectives, (1) using the rules to enhance a embedding model, and (2) using the embedding to guide the process of generating rules. KALE [22], RUGE [23], and IterE [83] use such a combination to achieve the first objective - to enhance the embedding model by applying rules. IterE, for example, addresses the data sparsity problem of Knowledge Graph embedding triple producers ([59]) by injecting new triples into the original Knowledge Graph.

RUGE and KALE represent triples and rules in a joint framework in order to enhance the capability of a Knowledge Graph embedding triple producer to predict new facts. RuLES [25], in contrast, follows the second objective (mentioned above) and uses the embedding model to evaluate the quality of rules. This eventually increases the quality of rules that are produced through rule learning.

There are also some recent efforts of exploiting schema of a Knowledge Graph in performing a link prediction task. These schema-based triple producers embed the axioms of the schema into a vector space (Embeds [9], TransC [40]), or even infer new triples to enrich the schema of a Knowledge Graph (Cose [13]).

None of these triple producers, however, use the schema of a Knowledge Graph to guide the correctness checking of new facts. Thus, these triple producers do not guarantee that the new triples produced are correct with regards to the schema of a Knowledge Graph. In measuring the performance of Knowledge Graph completion triple producers, the so-called silver standard method [30] is often applied to measure the performance of Knowledge Graph completion approaches. This method assumes that the Knowledge Graph itself is already of reasonable quality. In this paper, we argue that this assumption need to be reviewed. Our experiment shows that only 19% of the triples in the NELL-995 Knowledge Graph are correct with regards to the NELL schema, under the SHACL-enhanced notion of correctness discussed earlier. One drawback of such an assumption is that Knowledge Graph completion approaches might learn incorrect embeddings based on noisy patterns in a Knowledge Graph. Our experiment shows that using TransE for link prediction over the NELL Knowledge Graph, the correctness ratio is less than 1% (cf. Table 4).

We, therefore, regard the ontological schema of a Knowledge Graph to present a unique opportunity to both detect triples (from other triple producers) that do not comply with the schema (we regard these triples to be “erroneous” or “incorrect”) and also to perform Knowledge Graph completion by using a semantic reasoning technique.

Our core objective is to produce “correct” triples as many as possible with respect to the schema of a Knowledge Graph. In the silver standard method, some existing links in the data sub-graph (ABox) are removed for testing if triple producers can help to recover the missing links. Our goals are more ambitious: we not only want to make sure the produced triples are schema-correct, we also want to have as many schema-correct triples as possible. To measure this, we introduce the notion of coverage to measure the productivity of triple producers.

In this paper, we present an iterative approach to applying different types of triple producers for Knowledge Graph completion. Our hypothesis is that such an iterative approach is more effective than existing “single-pass” triple producers. “Single-pass” triple producer here means that we run one type of the triple producers once. There are five different single-pass triple producers that we used, as follows:

1. single-pass Knowledge Graph Embedding triple producers (KB2E-TransE [38]).
2. single-pass rule learning triple producers (RUMIS [72])
3. single-pass triple producer that integrates logic and learning tightly (KGE-HAKE [84])
4. single-pass triple producer that combine rule learning and Knowledge Graph Embedding (RuLES [25])
5. single-pass triple producer using Materialization service (cf. Section 2) from a reasoner (like HermiT [66]).

We call our approach as Schema Aware Iterative Knowledge Graph Completion (SIC). It should be noted that the detection of “incorrect” triples (with respect to a schema) plays an important role in our iterative completion approach. As we apply Knowledge Graph completion methods iteratively, the Knowledge Graph will be expanded with new triples at each iteration. With such an “error detection”, we guarantee that a Knowledge Graph will be expanded / completed only with “schema-correct” triples, as we retain at each iteration only those new triples that comply with the schema. Furthermore, the error detection process can be seen as an effective way to provide feedback to the triple producer in the next iterations, so as to somehow ‘guide’ the triple producers to produce correct triples. Such iterative process is repeated until certain stopping conditions (cf. Section 4.3) are met.

To perform the aforementioned error detection, one option is to use reasoning as described by [36]. Another option is to use some approximate consistency checking method, which is introduced in [77]. As demonstrated in [77], this method is very efficient in detecting errors in a Knowledge Graph containing millions of triples, such as, DBpedia [1]. For this reason, we decide to go with the option of approximate consistency checking. In this paper, we show how

our schema-aware iterative knowledge completion approach can produce more correct triples (with respect to a schema) than the single-pass triple producers, such as, for example, KB2E-TransE. We use NELL [46] and a subset of DBpedia [1] for evaluating our approach.

The contributions of this paper can be summarised as follows:

1. Firstly, we propose the new problems of schema aware Knowledge Graph completion, as well as schema aware iterative Knowledge Graph completion, where the schema plays a dual role of both producing triples (as one but not the only one triple producer) and checking the correctness of produced triples. While there are some approaches using schema to produce new triples, the aspect of correctness checking is largely ignored by existing approaches.
2. We propose to consider both ontological TBox and SHACL constraints as parts of the schema of Knowledge Graphs. Accordingly, we consider both ontological consistency checking and SHACL constraints when we define the notion of correctness. Experimental results suggest that the existing correctness notion based on silver standard method is highly questionable. Existing leading methods on link prediction based on the silver standard method might need to be re-evaluated on their effectiveness.
3. We propose a systematic framework to iteratively exploit existing Knowledge Graph completion methods and schema based logical reasoning, for both checking correctness and producing triples. Such an iterative framework allows to use schema based correctness checking to provide guidance for triple producers.
4. We analyse the characteristics of existing triple producers and design a few iterative completion patterns accordingly, for exploiting such characteristics.
5. Our evaluation confirms the underlying hypothesis of this paper; i.e., an iterative application of a mixture of different types of completion approaches would be more effective than one-pass embedding based completion.

2. Background

The Web Ontology Language OWL, which is based on Description Logics (DLs), is a key standard schema language of Knowledge Graphs. The second version of OWL, or OWL 2, was standardised in 2009. It is based on the *SR_QIQ* DL ([26]). Compared to the first version of OWL, OWL 2 provides more expressive power, such as a stronger support for data types ([51, 47]) and rules ([33]). OWL 2 has three tractable sub-languages, which are OWL 2 EL ([3]), OWL 2 QL ([8]) and OWL 2 RL ([19]).

We define a Knowledge Graph $\mathcal{G} = (\mathcal{T} \cup \mathcal{C}, \mathcal{A})$ as consisting of a set \mathcal{A} (ABox) of interconnected typed entities and their attributes (triples stored in the Knowledge Graph),

Table 1
Some relation axioms

ID	Formal Form	Description
1	$\exists r. \top \sqsubseteq A$	Domain(r) = A
2	$\top \sqsubseteq \forall r. A$	Range(r) = A
3	$r \equiv r^{-}$	Symmetric Relation
4	$\text{Disjoint}(r, r^{-})$	Asymmetric Relation
5	$r1 \sqsubseteq r2$	Relation Inclusion axioms

and a schema $\mathcal{T} \cup \mathcal{C}$, which consists of a set of ontological axioms \mathcal{T} (TBox) that defines the vocabulary used in a Knowledge Graph, as well as a set of SHACL constraints \mathcal{C} . In this paper, when the context is clear, we sometimes abuse the notion and treat \mathcal{G} as $\mathcal{T} \cup \mathcal{C} \cup \mathcal{A}$.

The ABox statements captured in the Knowledge Graph are of the following two forms:

- **Relation assertion** $r(h, t)$, where r is a relation, h and t are the entities in the position of *subject* and *object* (respectively) of the triples of an ABox. In the triple format, it can be written as (h, r, t) . With that, r is a *role* that links h and t . For example, *playInLeague*(*ACMilan*, *ItalianLeague*) is a role assertion.
- **Type assertion** $C(a)$, where a is an entity, while C is a type. In the triple format, it can be written as $(a, \text{rdf:type}, C)$. For example, *FootballClub*(*ACMilan*) is a type assertion.

The schema or TBox of a Knowledge Graph is the set of type and relation axioms. We refer the reader to [4] for detailed introduction of description logics. Here we only give some brief and informal introduction. A type axiom is of the form $C \sqsubseteq D$, where C and D are type descriptions, such as the following one: $\top \mid \perp \mid A \mid \neg C \mid C \sqcap D \mid \exists r. C \mid \{o\}$, where \top is a top type (representing all types), \perp is the bottom type (representing an empty set), A is a named type, r is a relation, and o is an entity. For example, the types of *River* and *City* being disjoint can be represented as $\text{River} \sqsubseteq \neg \text{City}$, or $\text{River} \sqcap \text{City} \sqsubseteq \perp$. Table 1 lists some example relation axioms. Note, that for the first and second axiom schema listed in table 1, we allow multiple domains and ranges.

Given a Knowledge Graph $\mathcal{G} = (\mathcal{T} \cup \mathcal{C}, \mathcal{A})$, we use the following semantic reasoning services: (1) Consistency checking: \mathcal{G} is consistent, if there exists a model that satisfies all statements in \mathcal{T} and \mathcal{A} . (2) Classification: this service computes all the subsumptions among named concepts in \mathcal{T} . (3) Entailment checking: this service checks if an axiom α is a consequence of $\mathcal{T} \cup \mathcal{A}$, or $\mathcal{T} \cup \mathcal{A} \models \alpha$. (4) Materialisation: this service computes all individuals of concepts and roles in \mathcal{G} . (5) Justification: given a reasoning task t over \mathcal{G} and its result r , this service computes the minimal subsets of \mathcal{G} that justifies the result r . Reasoning services can be provided by existing reasoners, such as Hermit [66] and TrOWL [70].

The two-layer architecture of OWL 2 discussed earlier allows the approximation of OWL 2 ontologies with ontolo-

gies expressed in a tractable OWL sub-language, such as approximations with OWL 2 QL ([54]), OWL 2 EL ([52, 64]) and OWL 2 RL ([85]), so as to exploit efficient and scalable reasoners of these sub-languages. The popularity of approximate reasoning methods is mainly due to the following reasons:

1. Some optimal approximation approaches, such as semantic approximation [54], can preserve both soundness and completeness of reasoning for large categories of practical queries. In other words, such queries cannot tell the differences between the original Knowledge Graphs and their approximations. Note that the optimality does not come for free; i.e., optimal approximations are often more expensive to compute than non-optimal approximations.
2. Some non-optimal approximation approaches, such as some faithful syntactic approximations [52, 64], provide a good trade-off between completeness and efficiency. For example, the approximate reasoner TrOWL [70] outperforms some sound and complete reasoners in the Ontology REasoning (ORE) competitions.
3. Real-world knowledge and data are hardly perfect or fully digitised. It is a complex task to build a comprehensive schema for a large Knowledge Graph. That is why Knowledge Graph completion techniques are so popular these days. Many of these methods, as discussed earlier, do not even use schema for producing new triples.

In this paper, two reasoning services are most relevant: on the one hand, materialisation is a powerful tool for completing a Knowledge Graph. On the other hand, consistency checking is helpful for checking if the new triples suggested by embedding-based or rule-based triple producers are consistent with the schema of the target Knowledge Graph.

The Shapes Constraint Language (SHACL) is a recent W3C language for the validation of Knowledge Graphs, which has been adopted by mainstream tools and triple stores. A SHACL schema is a collection of shapes, which are constraints that an RDF graph should satisfy. SHACL cannot replace the ontological consistency checking service mentioned earlier; however, it could introduce additional constraints for target Knowledge Graphs.

In this paper, we also consider a set of SHACL constraints C as part of the schema of Knowledge Graphs. Note that not all existing Knowledge Graphs have SHACL constraints. From a practical perspective, we can reuse some axioms, such as domain and range, in \mathcal{T} as constraints in the spirit of SHACL. In this paper, domain and range statements from \mathcal{T} are also considered as constraints in C as follows. A domain statement $\text{Domain}(r)=A$ can be represented in a SHACL constraint as follows:

```
ex:domainShape
  a sh:NodeShape ;
  sh:property [
    sh:path [ sh:inversePath r ] ;
    sh:class A ;
```

].

A range statement $\text{Range}(r)=A$ can be represented as a SHACL constraint as follows:

```
ex:rangeShape
  a sh:NodeShape ;
  sh:property [
    sh:path r ;
    sh:class A ;
  ] .
```

We finalize this section by briefly explaining the differences between a domain (range) axiom and a domain (range) constraint: given a triple (h, r, t) , a domain (range) axiom $\text{Domain}(r)=A$ ($\text{Range}(r)=A$) infers that $A(h)$ (resp. $A(t)$). However, a domain (range) constraint demands that h (resp. t) must be an instance of A . SHACL constraints can be represented, e.g., as SPARQL ASK queries.²

3. Problem Statement

3.1. Schema Aware Knowledge Graph Completion

Given a Knowledge Graph $\mathcal{G} = (\mathcal{T} \cup \mathcal{C}, \mathcal{A})$, the task of “schema-aware” Knowledge Graph completion is to make use of the schema $\mathcal{T} \cup \mathcal{C}$ for generating a set of schema-correct triples for the data sub-graph \mathcal{A} . Here, the schema $\mathcal{T} \cup \mathcal{C}$ is not only used to produce, together with existing triple producers, such as embedding based or rule based ones, a set of new triples \mathcal{E} through materialisation, but also to detect the schema-correct subset \mathcal{A}' (cf. Def 1) of \mathcal{E} . Some existing work uses schema information for Knowledge Graph completion; however, they do not use the schema to identify whether triples generated during the completion process are schema-correct. For example, Embeds [9] and TransC [40] consider domain and range axioms, or even disjointness axioms in the case of [13]; however, these axioms are *not* used as constraints to check correctness.

We regard triples of a Knowledge Graph as being either *correct*, *incorrect*, or *unknown* in terms of their compliance with the schema of a Knowledge Graph. A triple is *schema-correct*, if it is consistent with the schema of a Knowledge Graph, it is considered as *schema-incorrect* if it is *not consistent* with the TBox \mathcal{T} of a Knowledge Graph, and it is regarded as *schema-unknown*, if it neither schema-correct nor schema-incorrect; in other words, it is consistent with \mathcal{T} but not yet satisfying the constraints in C , due to lack of some type information for h or t .

Definition 1. Given a Knowledge Graph $\mathcal{G} = (\mathcal{T} \cup \mathcal{C}, \mathcal{A})$, a triple (h, r, t) , where h and t are entities and r is an object property in \mathcal{G} , with C_h, C_t being some types of h and t resp., and D_r, R_r being some domain and range of r . We say (h, r, t) is a *schema-correct triple w.r.t. \mathcal{G}* if:

1. the expanded Knowledge Graph $\mathcal{T} \cup \mathcal{A} \cup (h, r, t)$ is consistent, and
2. $C_h \equiv D_r$ and $C_t \equiv R_r$ (domain and range constraints in C).

²<https://www.w3.org/TR/rdf-sparql-query/>

Table 2
The inconsistency justification patterns

ID	TBox subset of the Pattern	ABox subset of the Pattern
1	Domain(r)= D , $D \sqcap A \sqsubseteq \perp$	($e1, r, e2$), ($e1, \text{rdf:type}, A$)
2	Range(r)= R , $R \sqcap A \sqsubseteq \perp$	($e1, r, e2$), ($e2, \text{rdf:type}, A$)
3	Asymmetric(r)	($e1, r, e2$), ($e2, r, e1$)
4	Symmetric($r1$), Asymmetric($r2$), $r1 \sqsubseteq r2$	($e1, r1, e2$)
5	Symmetric($r2$), Asymmetric($r3$), $r1 \sqsubseteq r2, r1 \sqsubseteq r3$	($e1, r1, e2$)
6	Irreflexive(r)	($e1, r, e1$)
7	Asymmetric($r1$), InverseOf($r1, r2$)	($e1, r1, e2$), ($e1, r2, e2$)

Note that in the above definition, domain and range are used as constraints in \mathcal{C} , so as to ensure that the types of the head entity (tail entity) matches some domain (range, resp.) of the relation.

Definition 2. Given a Knowledge Graph $\mathcal{G} = (\mathcal{T} \cup \mathcal{C}, \mathcal{A})$, a triple (h, r, t) , where h and t are entities and r is an object property in \mathcal{G} , with C_h, C_t being some types of h and t resp., and D_r, R_r being the domain and range of r , (h, r, t) is a schema-incorrect triple w.r.t. \mathcal{G} if:

1. $\mathcal{T} \cup \mathcal{A} \models C_h \sqcap D_r \sqsubseteq \perp$, or
2. $\mathcal{T} \cup \mathcal{A} \models C_t \sqcap R_r \sqsubseteq \perp$.

A schema-incorrect triple is not consistent with the schema of a Knowledge Graph.

Definition 3. Given a Knowledge Graph $\mathcal{G} = (\mathcal{T} \cup \mathcal{C}, \mathcal{A})$, a triple (h, r, t) , where h and t are entities and r is an object property in \mathcal{G} , (h, r, t) is a schema-unknown triple w.r.t. \mathcal{G} if it is neither schema-correct nor schema-incorrect w.r.t. \mathcal{G} .

The following examples, taken from a small subset of the NELL-995 Knowledge Graph, illustrate schema-correct, schema-incorrect, and schema-unknown triples. For example, here are some schema axioms/constraints and type assertions:

- Domain(leads_geopolitical_organization) = Person
- Range(leads_geopolitical_organization) = Geopolitical-organization
- Person \sqcap Geopoliticalorganization $\sqsubseteq \perp$
- Person(mugabe), Person(trump), City(summit), Geopoliticalorganization(african_country)

Here are three relation assertions.

1. leads_geopolitical_organization(mugabe, african_country)
2. leads_geopolitical_organization(mugabe, trump)
3. leads_geopolitical_organization(mugabe, summit)

Based on the TBox information, the relation assertion 1 is a schema-correct triple; the relation assertion 2 is a schema-incorrect triple, and the relation assertion 3 is a schema-unknown triple.

3.2. Schema Aware Iterative Knowledge Graph Completion

In this paper, we focus on a sub-problem of schema aware Knowledge Graph completion, namely Schema Aware Iterative Knowledge Graph Completion (SIC), where a consistency checking method with triple producers (a Knowledge Graph embedding method, a rule learning method, and an ontology reasoning method for materialisation) to detect the inconsistencies that either already exist in the Knowledge Graph, or are introduced by triple producers in an iterative process, so that only the schema-correct triples will be used in the next iteration step of completion.

In other words, the Knowledge Graph completion process is not a one-off process. One related question is how to measure the progress in each iteration in terms of correctness and completeness.

3.3. Correctness and Coverage Levels

In fact, correctness and completeness are key measures even for one-step Knowledge Graph completion, although most existing approaches only address completeness by performing link prediction. To properly measure the effectiveness of triple producers, we propose some functions to calculate the correctness and coverage levels.

3.3.1. Correctness Ratio

Based on the notion of schema-correctness in Definition 1, we propose the correctness level function $f_{Correctness}$ to calculate the correctness ratio of a Knowledge Graph completion approach across all iterations:

$$f_{Correctness} = \frac{1}{n} * \sum_{i=1}^n \frac{|\mathcal{A}'_i|}{|\mathcal{E}_i|} \quad (1)$$

where \mathcal{E}_i is the set of new triples produced by a triple producer at iteration i , \mathcal{A}'_i is the set of schema-correct triples contained in \mathcal{E}_i . It is clear that the score of $f_{Correctness}$ is between 0 and 1. In the case of a one-off Knowledge Graph completion, n in (1) is set to 1.

In this paper, we use correctness ratio to measure the performance of different triple producers, in terms of schema-correctness.

3.3.2. Coverage Ratio

The measure of completeness is more debatable. As discussed in the Introduction, the so-called silver standard method

[30] is often applied to measure the performance of Knowledge Graph completion approaches. This method assumes that the Knowledge Graph itself is already of reasonable quality. Our experiment shows that only 19% of the triples in the NELL-995 Knowledge Graph are schema-correct (cf. Definition 1) with regards to the NELL schema. Our recommendation is, therefore, that the schema-correct subset of the target Knowledge Graph, rather than the target Knowledge Graph itself, should be used as the silver standard.

Without knowing the ‘complete’ Knowledge Graph, it is rather hard to define a proper measure for completeness. Instead, in this paper, we use the notion of coverage, to account for the scale of schema-correct triples added into the original Knowledge Graph, thanks to the Knowledge Graph completion process. Accordingly, we propose a function $f_{Coverage}$ to calculate the coverage ratio of a Knowledge Graph:

$$f_{Coverage} = \frac{\sum_{i=1}^n |\mathcal{A}'_i|}{|\mathcal{Y}|} \quad (2)$$

where \mathcal{A}'_i is the set of schema-correct triples in iteration i , and \mathcal{Y} is a subset of the target Knowledge Graph \mathcal{G} that consists of schema-unknown plus schema-correct triples.

These two functions are used to calculate in our iterative completion approach how many schema-correct triples \mathcal{A}' can be collected. We continue calculating the \mathcal{A}' at each iteration, until a triple producer does not produce new triples anymore. We sum the measurements taken for each \mathcal{A}' set from the first to the last iteration and then divide it by the size of \mathcal{Y} , the subset of \mathcal{G} . Larger $f_{Coverage}$ scores are an indication that the used triple producer has produced a high number of new triples.

3.3.3. Harmonic Average Ratio

We propose a third function $f_{Correctness,Coverage}$, which is the harmonic means of $f_{correctness}$ and $f_{coverage}$, in order to balance the correctness and coverage levels. Our ultimate goal is to find out which approach can produce the highest number of new and schema-correct triples.

$$f_{Correctness,Coverage} = \frac{2 * f_{Correctness} * f_{Coverage}}{f_{Correctness} + f_{Coverage}} \quad (3)$$

4. Our Approach

In this section, we present the SIC (Schema-aware Iterative Completion) approach³ to addressing the problem of schema-aware iterative Knowledge Graph completion introduced in Section 3.2. In this approach, the completion of a Knowledge Graph takes place in an iterative fashion. During each micro-iteration, a few triple producers are used sequentially to generate new candidate triples, which are then checked by a correctness checking component to identify the schema-correct triples. The schema-correct triples are then

used in the next iteration of the SIC approach. This iterative process is repeated until some stopping condition is satisfied.

Our SIC approach assumes that a target Knowledge Graph has a schema, which at least contains subsumption hierarchies, as well as domain and range information. However, for Knowledge Graphs that lack a schema, one can use ontology learning tools (cf. [55] to generate a schema for such a Knowledge Graph). For Knowledge Graphs with incomplete schema (for example, lacking any domain and range information), we refer the reader to use the methods described [71, 41, 57] to enrich such a schema.

In this iterative process, the expanded Knowledge Graphs can become a lot larger than the original Knowledge Graph. Due to the scalability issues of sound and complete reasoners, such as FACT++, when being run against the DBpedia Knowledge Graph, we decided to use the approximate consistency checking (ACC) method that was introduced in [77]. This approach allows the detection of logical inconsistencies in the DBpedia Knowledge Graph by using the so-called ‘‘inconsistency justification patterns’’ shown in table 2. It took about 5 hours to detect the inconsistent triples in DBpedia version 2016-04 with ACC. In contrast, a recent experiment described in [58] showed that using the HermiT reasoner [66] for detecting the logical inconsistencies in DBpedia consumed 15 days.

We consider three kinds of triple producers: (i) Knowledge Graph embedding methods (‘‘E-method’’), (ii) rule learning methods (‘‘R-method’’), (iii) materialization services (‘‘M-method’’). The triple producers used in our approach are only loosely coupled, which allows an arbitrary combination of them.

4.1. Characteristics of Different Triple Producers

To better understand the characteristics of the three triple producers, we designed an experiment with two Knowledge Graphs: the NELL-995 Knowledge Graph, and a subset related to political issues from the DBpedia Knowledge Graph (DBped-P). We expanded these Knowledge Graphs by using three triple producers: KB2E-TransE [38] (E-method), RUMIS [72] (R-method), and a materialization service using the HermiT reasoner [66] (M-method).

We modified KB2E-TransE so that only the top 10 predictions are produced. This was done to limit the number of predictions by this embedding algorithm. Otherwise, the ‘‘Approximate Consistency Checking’’ (ACC) algorithm [77] would have to check all the possibilities, which is not necessary and very costly. Our modification of KB2E-TransE inserts a loop before the calculation of the parameter ‘‘HIT’’ to output the triples. HIT is the parameter that is commonly reported by Knowledge Graph embedding methods and refers to how many correct triples are contained in the top N predictions. We used the ACC algorithm to check whether the R-method and E-method produce schema-incorrect triples. Triples produced by the M-method are always correct with respect to the schema. We present the results in Table 3.

Here are some observations regarding the results in Table 3:

³Please see <https://github.com/bagindokemas/SAIKGC> for the implementation

Table 3

The comparison of the performance among R (Rule-based), E (Embedding-based) and M (Materialisation-based) methods. The column \mathcal{E} denotes the number of new triples produced by a triple producer. The column #R denotes the number of relations that exist on the new triples that are produced by a triple producer. The column #IT denotes the number of schema-incorrect triples that are produced by a triple producer. The column %OIT denotes the percentage of schema-incorrect triples from all new triples that are produced by a triple producer.

KGs	R-method			E-method				M-method				
	\mathcal{E}	#R	#IT	%OIT	\mathcal{E}	#R	#IT	%OIT	\mathcal{E}	#R	#IT	%OIT
NELL-995	600,449	6	136,196	22.68	201,370	110	106,319	52.80	42,692	184	0	0
DBped-P	334,537	2	0	0	637,479	142	227,963	35.76	152	13	0	0

1. The R-method and E-method produce more candidate triples than the M-method. This somehow confirms why these two types of triple producers are so popular;
2. The E-method produces a lot more incorrect triples than the R-method;
3. Among the three methods, the E-method produces triples with the highest number of relations. The R-method generates triples with far fewer relations than the other two methods. The relations produced by the M-method for triples are schema-dependent.
4. The R-method has a “cold start problem”. It is not able to produce any new triples if a Knowledge Graph is small (such as less than 30k triples) and has a simple data sub-graph.

The above observations suggest that there is a need to combine these three triple producers to produce more schema-correct triples. In general, the R-method can produce many candidate triples with relatively low schema-incorrectness ratio, but it has the cold start issue. Accordingly, we think there are some potential benefits of combining the individual triple producers together:

- For small Knowledge Graphs, the E-method could be used to produce more triples, mitigating the cold start problem that exists for the R-method; and
- The M-method can help to produce further schema-correct triples for the other two methods to learn from, in particular to the R-method, due to its cold start issue.

4.2. Completion with Multiple Triple Producers

Now that we see some potential benefits of combining triple producers, we design a small-scale experiment to verify the idea, running an M-method, an E-method and an R-method sequentially. We perform one ACC after the E-method and one after the R-method. In the following, we represent such a sequence with this simple form: “M-E-R”.

As shown in Table 4, in terms of $f_{Coverage}$, for the DBped-P Knowledge Graph, the combined approach produces a much better coverage level (2.29) than any of the individual approaches. However, for the NELL-995 Knowledge Graph, the R-method alone achieved higher coverage than combined

completion, because the E-method produces many schema-unknown triples, which lower the coverage level of the combined approach.

As for the $f_{correctness}$, table 4 shows that the $f_{Correctness}$ of the E-method and R-method is consistently lower than that of the iterative approach. The M-method has a very high correctness but its coverage is much lower than the Iterative Completion.

4.3. Iterative Combined Completion

One key advantage of combined approach is that we could execute the completion process in an iterative manner. Running an individual triple producer repeatedly does not produce new completion result. However, this paper shows that combining triple producers can produce new triples in an iterative manner. In our approach, each iteration has a pattern like M-E-R. In what follows, we further investigate which combinations of the three triple producers in an iterative way are most effective.

We investigate two kinds of SIC approaches: (a) schema aware iterative completion based on sequential combinations (or simply, Classic SIC), or and (b) the schema aware iterative completion based on sequential combinations using “MapReduce” (or simply, MapReduce SIC). The MapReduce SIC is needed since after a few iterations, expanded Knowledge Graphs can become rather large. In the classic combination, we choose KB2E-TransE as the Knowledge Graph embedding method (E-method) and RUMIS as the rule learning method (R-method). As for the MapReduce combination, we choose TransC as the Knowledge Graph embedding method (E-method) and RuLES as the rule learning method (R-method). We choose these methods because, in their respective categories, they are the most efficient methods.

4.3.1. Classic SIC

When combining triple producers (e.g. in a sequential fashion), the following observations about the role of each triple producer can be made:

- E-method: In case of a small Knowledge Graph (less than 30k triples), the E-method helps to address the cold-start problem of the R-method.
- R-method: this method acts as the main contributor of

Table 4

The comparison of the $f_{Correctness}$ and $f_{Coverage}$ between individual and combined completion: Cov for coverage and Corr for correctness.

KG	E-method		R-method		M-method		Combined	
	Cov	Corr	Cov	Corr	Cov	Corr	Cov	Corr
NELL-995	0.0053	0.0037	1.55	0.36	0.3	1	0.61	0.39
DBped-P	0.34	0.18	0.69	0.59	0.0004	1	2.29	0.64

good quality new triples as it can produce more triples than the other producers.

- M-method: this method can play two roles: 1) inferring new triples based on the schema of a Knowledge Graph, and 2) increasing the performance of other triple producers by providing schema-correct triples to subsequent producers.

The approximate consistency checking (ACC) acts as a “cleanup” step for each expanded graph as produced by the E-method and R-method.

ACC takes two inputs, a Knowledge Graph that should be tested for inconsistencies (we call it the Initial Knowledge Graph IKG), and a set of Inconsistency Justification Patterns (see table 2). Inconsistency Justification Patterns are derived through a scanning process applied to the TBox part of the Knowledge Graph. Given this input, ACC groups all triples in the Knowledge Graph into three subsets: schema-correct, schema-incorrect, and schema-unknown.

Due to the limits of computing resources used in these experiments, it emerged that after many iterations, our approach expanded the initial Knowledge Graph to a particular size (number of triples) that cannot be handled by an embedding method nor a rule learning method.

4.3.2. MapReduce SIC

In order to meet this challenge, we created a version of our SIC method that uses the MapReduce algorithm. Each iteration is now separated into two main phases, implementing the Map and the Reduce phase respectively. In the Map phase, we partition the initial Knowledge Graph (IKG) into several partitions. A number of p triple producers are applied to these partitions in order to produce new triples. In the subsequent Reduce phase, the following steps are done during an iteration i :

- Merging all the outputs/new facts that are produced by all triple producers. The result set of this merge process is M^i ;
- Remove duplicate triples in M^i , resulting in set DM^i that is the result of this removal process;
- Compare DM^i and the initial Knowledge Graph (IKG) to produce the set RN^i of new triples;
- Process RN^i , and the schema of the initial Knowledge Graph by using the approximate consistency checking ACC, and produce schema-correct triples (SCT), schema-incorrect triples (SIT), and schema-unknown triples (SUT);

- Prepare the input for the Map phase of the next triple producer.

Each iteration consists of different triple producers, which may be either the R-method, E-method, or M-method. Immediately after applying the E-method or R-method, we always run ACC as the cleanup step.

Figure 1 shows a typical order of an iteration in MapReduce SIC. In total, we have 2 Map phases and 2 Reduce phases for each iteration. P1 until P5 are the partitions of the initial Knowledge Graph. RL 1 until RL 5 refer to five Rule Learning methods that run in parallel. NT1 until NT5 are the new triples that are produced by the triple producers. E1 until E5 refer to five embedding methods that run in parallel. As Figure 1 shows, at each iteration, given an initial Knowledge Graph, we will determine the optimal number of triples (n) that can be processed by a triple producer. Based on this number n , we can set the number of triple producers that will work in parallel in the Map phase (we only do this for the rule learning methods and embedding methods).

The order of the triple producers is the same throughout all the iterations. We repeat the iterations until a stopping condition is satisfied. By assuming that the number of correct new triples that are produced at the end of the n th iteration is NCT_N , we define the stopping condition as $\frac{NCT_N}{NCT_{N-1}} \leq 1\%$. We choose 1% as the threshold because, based on our experiments, if we continue to run the triple producers after reaching less than 1%, the triple producers will produce 0 triples. Otherwise, if the stopping condition is not satisfied, a new iteration will start. From three different triple producers, we propose the following 6 types of combinations:

- E-method, ACC, M-method, R-method, ACC (E-M-R)
- E-method, ACC, R-method, ACC, M-method (E-R-M)
- M-method, R-method, ACC, E-method, ACC (M-R-E)
- M-method, E-method, ACC, R-method, ACC (M-E-R)
- R-method, ACC, M-method, E-method, ACC (R-M-E)
- R-method, ACC, E-method, ACC, M-method (R-E-M)

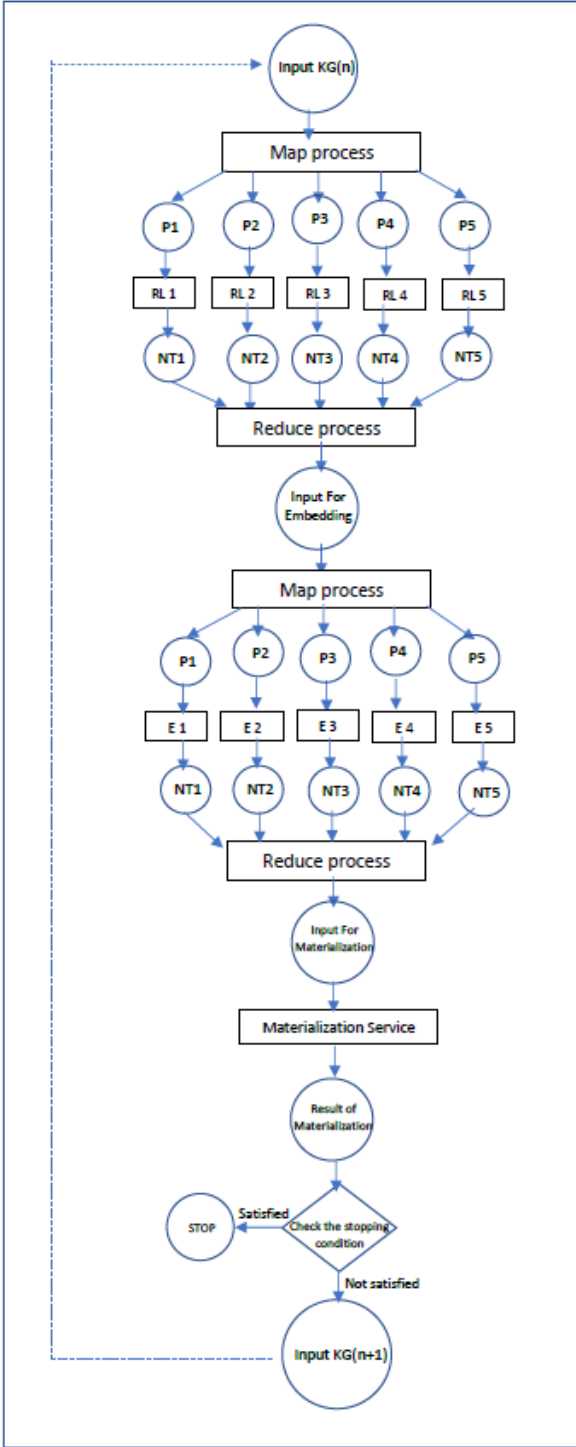


Figure 1: An illustration of one iteration in SIC

The decision, which type(s) of combination(s) to use in our experiments, was based on the following three metrics: (1) the number of schema-correct new triples \mathcal{A}' produced by a combination, (2) the number of relations $\#R$ that we find from the schema-correct triples after a combination has finished, and (3) the percentage of the new triples that are

Table 5

The alternative combinations of Iterative Completion

The combination	\mathcal{A}'	$\#R$	$\%\mathcal{A}'$
E-M-R	18,593	23	8.48%
E-R-M	4,587	2	2.87%
M-R-E	77,532	185	92.31%
M-E-R	78,236	185	49.79%
R-M-E	307,785	181	37.58%
R-E-M	310,160	185	36.33%

schema-correct ($\%\mathcal{A}'$).

We conducted an additional experiment on the NELL-995 Knowledge Graph in order to support this decision process.

Table 5 shows that the highest percentage of new correct triples are achieved by combination M-R-E. This is due to the M-method generating schema-correct triples that are subsequently fed into R-method, resulting in the production of 92.31% schema-correct triples. However, the number of new triples that this combination produces is still far less than the number produced by combinations R-M-E and R-E-M. We also observe that combination E-M-R and combination E-R-M are not promising combinations. Since the primary focus of our research is to increase the coverage of the Knowledge Graph, i.e., to produce as many schema-correct triples as possible, we decided to choose M-R-E, M-E-R, R-M-E, and R-E-M as the main combinations in our detailed empirical studies outlined in section 5. For each combination of triple producers is applied to a Knowledge Graph: algorithm 1 outlines the application of combination R-E-M, algorithm 2 outlines the application of combination R-M-E, algorithm 3 outlines the application of combination M-R-E, and algorithm 4 outlines the application of combination M-E-R.

5. Evaluation

We perform an empirical study, evaluating our approach for completion accuracy. In terms of the experimental setup for evaluating completion accuracy, we compare 5 types of single-pass triple producers (cf. Table 8) with two different combinations of schema-aware iterative completion (SIC), which are classic combination (cf. Table 6) and MapReduce combination (cf. Table 7) in terms of $f_{Correctness}$, $f_{Coverage}$, and $f_{Correctness, Coverage}$.

In Table 9, we illustrate schema-correct, schema-incorrect, and schema-unknown triples that are produced by our approach.

5.1. Datasets

In Knowledge Graph completion, there are several Knowledge Graphs that are considered as the benchmarks [2],[65]: Freebase, Wordnet, DBpedia, and NELL-995. Freebase is an open and collaborative Knowledge Graph publicly launched in 2007 and closed in 2015 [69]. It contains more than 3 billion triples about almost 50 million entities. Freebase does

Algorithm 1 MapReduce for SIC (combination 1)

```

1: function MAPREDUCE(IKG(i), η, ξ)
2:   while TRUE do
3:     determine n
4:     p ← (size of IKG(i))/n
5:     Θ1(i) ← RuleLearning1(IKG1(i))
6:     Θ2(i) ← RuleLearning2(IKG2(i))
7:     .....
8:     Θp(i) ← RuleLearningp(IKGp(i))
9:     MRi ← Merge(Θ1(i) until Θp(i))
10:    DMRi ← RemoveDuplicate(MRi)
11:    RNRi ← Compare(DMRi, IKG(i))
12:    φR(i) ← APR(RNRi, η, ξ)

13:    IFEi ← Merge(IKGi, φR(i))
14:    q ← (size of IFEi)/n
15:    Θ1(i) ← Embedding1(IFE1(i))
16:    Θ2(i) ← Embedding2(IFE2(i))
17:    .....
18:    Θq(i) ← Embeddingq(IFEq(i))
19:    MEi ← Merge(Θ1(i) until Θq(i))
20:    DMEi ← RemoveDuplicate(MEi)
21:    RNEi ← Compare(DMEi, IFE(i))
22:    φE(i) ← APR(RNEi, η, ξ)

23:    IFMi ← Merge(IFEi, φE(i))
24:    φM(i) ← Materialize(IFMi)
25:    if φM(i) == IKGi then
26:      BREAK
27:    else
28:      IKGi+1 ← φM(i)
29:    end if
30:    return (φR(i), φE(i), φM(i))
31: end while
end function

```

not have a rigid hierarchy of types or concepts. It even allows conflicting and contradictory types and properties [6]. Hence, Freebase only has a very simple and poorly designed schema. Wordnet, as stated in [44], “is a large lexical database of English. It contains Nouns, verbs, adjectives and adverbs which are grouped into sets of cognitive synonyms (synsets), each expressing a distinct concept”. Wordnet has 79689 synsets. 90.37% of these synsets are concepts and only 9.63% are entities or instances [45]. The less number of instances make the Wordnet Knowledge Graph does not have any sufficient triples. DBpedia is a large-scale, multilingual Knowledge Graph that is made by the community by extracting structured data from Wikipedia editions in 111 languages. It has 3 billion triples and also a manually created schema that contains 685 concepts, which form a subsumption hierarchy, and 2,795 different properties [35]. NELL-995, short for Never Ending Language Learner, is created through a semi-

Algorithm 2 MapReduce for SIC (combination 2)

```

1: function MAPREDUCE(IKG(i), η, ξ)
2:   while TRUE do
3:     determine n
4:     p ← (size of IKG(i))/n
5:     Θ1(i) ← RuleLearning1(IKG1(i))
6:     Θ2(i) ← RuleLearning2(IKG2(i))
7:     .....
8:     Θp(i) ← RuleLearningp(IKGp(i))
9:     MRi ← Merge(Θ1(i) until Θp(i))
10:    DMRi ← RemoveDuplicate(MRi)
11:    RNRi ← Compare(DMRi, IKG(i))
12:    φR(i) ← APR(RNRi, η, ξ)

13:    IFMi ← Merge(IKGi, φR(i))
14:    φM(i) ← Materialize(IFMi)

15:    IFEi ← Merge(IFMi, φM(i))
16:    q ← (size of IFEi)/n
17:    Θ1(i) ← Embedding1(IFE1(i))
18:    Θ2(i) ← Embedding2(IFE2(i))
19:    .....
20:    Θq(i) ← Embeddingq(IFEq(i))
21:    MEi ← Merge(Θ1(i) until Θq(i))
22:    DMEi ← RemoveDuplicate(MEi)
23:    RNEi ← Compare(DMEi, IFE(i))
24:    φE(i) ← APR(RNEi, η, ξ)

25:    if φE(i) == IKGi then
26:      BREAK
27:    else
28:      IKGi+1 ← φE(i)
29:    end if
30:    return (φR(i), φE(i), φM(i))
31: end while
end function

```

supervised machine-learning algorithm that crawls the web and learns common sense facts about the world without human supervision [46]. The result of this unsupervised learning process is a Knowledge Graph that contains more than 1.2 million entities and 100K relationships [60]. The schema of the NELL-995 Knowledge Graph has 1187 concepts, 894 object properties and many types of object property axioms, such as inverse object properties, functional object property, asymmetric object property, irreflexive object property, and object property domain and range. Our notion of consistency in this paper relies on 2 criteria: (1) high enough number of triples, and (2) a schema that has a rigid hierarchy of concepts and rich object properties. From all the benchmarks in Knowledge Graph Completion, only the NELL-995 and DBpedia Knowledge Graphs fulfill these criteria. Therefore, we decided to evaluate our approach with both the NELL-995

Algorithm 3 MapReduce for SIC (combination 3)

```

1: function MAPREDUCE(IKG(i), η, ξ)
2:   while TRUE do
3:     IFMi ← IKGi
4:     φM(i) ← Materialize(IFMi)

5:     determine n
6:     IFRi ← Merge(IFMi, φM(i))
7:     p ← (size of IFRi)/n
8:     Θ1(i) ← RuleLearning1(IFR1(i))
9:     Θ2(i) ← RuleLearning2(IFR2(i))
10:    .....
11:    Θp(i) ← RuleLearningp(IFRp(i))
12:    MRi ← Merge(Θ1(i) until Θp(i))
13:    DMRi ← RemoveDuplicate(MRi)
14:    RNRi ← Compare(DMRi, IKG(i))
15:    φR(i) ← APR(RNRi, η, ξ)

16:    IFEi ← Merge(IFRi, φR(i))
17:    q ← (size of IFEi)/n
18:    Θ1(i) ← Embedding1(IFE1(i))
19:    Θ2(i) ← Embedding2(IFE2(i))
20:    .....
21:    Θq(i) ← Embeddingq(IFEq(i))
22:    MEi ← Merge(Θ1(i) until Θq(i))
23:    DMEi ← RemoveDuplicate(MEi)
24:    RNEi ← Compare(DMEi, IFE(i))
25:    φE(i) ← APR(RNEi, η, ξ)

26:    if φE(i) == IKGi then
27:      BREAK
28:    else
29:      IKGi+1 ← φE(i)
30:    end if
31:    return (φR(i), φE(i), φM(i))
32:  end while
end function

```

and DBpedia Knowledge Graphs.

The NELL-995 Knowledge Graph is a dataset developed at Carnegie Mellon University and contains 142,065 triples. In terms of the DBpedia Knowledge Graph, we only process a subset of DBpedia because of limited computational resources⁴, which only includes 352,754 triples that are related to a political issues, called DBped-P.

5.2. Results for $f_{Coverage}$

We can see from Tables 6 and 7, that for both the NELL-995 and DBped-P Knowledge Graph, the combinations R-M-E and R-E-M produce best results. Both of them use the R-method as the first triple producer.

In combination R-E-M of classic SIC, the R-method helps

⁴<http://wiki.dbpedia.org/downloads-2016-04>

Algorithm 4 MapReduce for SIC (combination 4)

```

1: function MAPREDUCE(IKG(i), η, ξ)
2:   while TRUE do
3:     IFMi ← IKGi
4:     φM(i) ← Materialize(IFMi)

5:     determine n
6:     IFEi ← Merge(IFMi, φM(i))
7:     q ← (size of IFEi)/n
8:     Θ1(i) ← Embedding1(IFE1(i))
9:     Θ2(i) ← Embedding2(IFE2(i))
10:    .....
11:    Θq(i) ← Embeddingq(IFEq(i))
12:    MEi ← Merge(Θ1(i) until Θq(i))
13:    DMEi ← RemoveDuplicate(MEi)
14:    RNEi ← Compare(DMEi, IFE(i))
15:    φE(i) ← APR(RNEi, η, ξ)

16:    IFRi ← Merge(IFEi, φE(i))
17:    p ← (size of IFRi)/n
18:    Θ1(i) ← RuleLearning1(IFR1(i))
19:    Θ2(i) ← RuleLearning2(IFR2(i))
20:    .....
21:    Θp(i) ← RuleLearningp(IFRp(i))
22:    MRi ← Merge(Θ1(i) until Θp(i))
23:    DMRi ← RemoveDuplicate(MRi)
24:    RNRi ← Compare(DMRi, IKG(i))
25:    φR(i) ← APR(RNRi, η, ξ)

26:    if φR(i) == IKGi then
27:      BREAK
28:    else
29:      IKGi+1 ← φR(i)
30:    end if
31:    return (φR(i), φE(i), φM(i))
32:  end while
end function

```

the E-method by providing more than 200k of new triples for the NELL-995 Knowledge Graph and more than 5 million new triples for the DBped-P Knowledge Graph as an input. Such inputs help the E-method to produce more new triples. With the new and schema-correct triples that are produced by the R-method and E-method, the productivity of the M-method in producing new triples increased 100% (compare that with a situation where the M-method acts as an individual triple producer).

As for the combination R-E-M of the MapReduce SIC approach (using the DBped-P Knowledge Graph), in the early iterations (iteration 1 until 10), every triple producer only processes a small number of triples (< 200k triples). This impacts negatively on the productivity of each triple producer in producing new triples. However, after the initial

Table 6
The result of the completion accuracy for classic SIC

KGs	R-M-E			R-E-M			M-R-E			M-E-R		
	Corr	Cov	Hav	Corr	Cov	Hav	Corr	Cov	Hav	Corr	Cov	Hav
NELL-995	0.14	2.30	0.27	0.13	2.28	0.25	0.47	0.59	0.52	0.39	0.61	0.48
DBped-P	0.70	18.13	1.35	0.70	17.47	1.35	0.76	3.06	1.22	0.64	2.29	0.99

Table 7
The result of the completion accuracy for MapReduce SIC

KGs	R-M-E			R-E-M			M-R-E			M-E-R		
	Corr	Cov	Hav	Corr	Cov	Hav	Corr	Cov	Hav	Corr	Cov	Hav
NELL-995	0.35	26.41	0.69	0.25	23.13	0.49	0.26	13.03	0.50	0.24	16.87	0.48
DBped-P	0.93	4.34	1.54	0.94	39.11	1.84	0.91	5.12	1.54	0.77	9.9	1.42

Knowledge Graph grows to > 4 million triples (this number is reached in iteration 42), each triple producer processes more than 500k triples. This eventually boosts the performance of each triple producer. In iteration 43, the RuLES triple producer produces more than 5 million new triples and the TransC triple producer produces more than 1 million new triples. The combination R-E-M of MapReduce SIC stops at iteration 44, where it successfully scores **39.11** points for the metric $f_{Coverage}$.

As for the combination R-M-E of MapReduce SIC (using the NELL-995 Knowledge Graph), at each iteration, RuLES and TransC produce $\pm 20k$ triples on average. This eventually increases the size (size here refers to the number of triples) of the NELL-995 Knowledge Graph to 26 times larger than its original size.

5.3. Results for $f_{Correctness}$ and $f_{Correctness,Coverage}$

For both the NELL-995 and DBped-P Knowledge Graphs, the best option of the classic SIC is to use combination M-R-E, from which we observe that the M-method provides the R-method with a set of schema-correct triples, and this helps the R-method to produce also a good set of schema-correct triples. The good performance of combination M-R-E is, therefore, not only caused by a good combination of different triple producers, but also through the reasoning process in the M-method that utilizes the schema of a Knowledge Graph to infer new schema-correct triples.

As for the MapReduce SIC, for the DBped-P Knowledge Graph, all combinations consistently score > 0.70 point for the metric $f_{Correctness}$. The highest score (0.94) is achieved by combination R-E-M. There are three reasons why the combination R-E-M (in the MapReduce SIC) performs better in terms of $f_{Correctness}$ compared to the combination M-R-E (in classic SIC):

- At each iteration, both the R-method in MapReduce SIC (RuLES) and the E-method in MapReduce SIC (TransC) produce more schema-correct triples compared to the R-method in classic SIC (RUMIS) and the E-method in classic SIC (TransE).
- At each iteration, on average the classic SIC produces a higher number of schema-correct triples (460,654 triples), compared to MapReduce SIC (320,817 triples).

However, as the number of iterations required for the combination M-R-E of classic SIC (only 3) is far less than the number of iterations required for the combination R-E-M of MapReduce SIC (which is 44), the combination R-E-M produces in total 13,212,697 new schema-correct triples. The reason why the combination R-E-M in MapReduce SIC can reach 44 iterations, is because by applying the MapReduce algorithm to our SIC approach, each triple producer in the Map phase at each iteration will focus on expanding different parts of the initial Knowledge Graph.

As for the NELL-995 Knowledge Graph, our MapReduce SIC approach only achieves 0.35 points for the metric $f_{Correctness}$. This result is not in line with what the MapReduce SIC approach achieved for the DBped-P Knowledge Graph. It can be explained through a closer analysis of the NELL-995 Knowledge Graph itself. In contrast to the DBped-P Knowledge Graph, the NELL-995 Knowledge Graph has a richer TBox. It usually contains descriptions with various object property characteristics, such as *functional*, *transitive*, *symmetric*, *asymmetric*, *reflexive*, and *irreflexive*. For the NELL-995 Knowledge Graph, TransC, as the triple producer in our MapReduce SIC approach, produces less schema-correct triples. On average in every iteration, only 24% of all new triples produced are schema-correct. If we check the schema-incorrect triples that are produced by TransC, we find that the majority of the schema-incorrect is caused by the *functional* and *irreflexive* object property characteristics. Here is an example:

During an iteration, TransC produced these 3 new triples:

1. <n1962_world_series> <Sportsgamewinner> <pirates>
2. <n1962_world_series> <Sportsgamewinner> <seattle_mariners>
3. <pirates> <teamplaysagainstteam> <seattle_mariners>

In the TBox of the NELL-995 Knowledge Graph, we know that *Sportsgamewinner* is a *functional* object property characteristic. We also know that *teamplaysagainstteam* is an *irreflexive* object property characteristic.

Based on the information from the TBox and from the 3 new triples, a reasoner will report 1,2, and 3 as schema-incorrect triples. 1 and 2 are incorrect, because a functional

Table 8
Completion Accuracy for the Single-Pass triple producers

KGs	NELL-995			DBped-P		
	Corr	Cov	Hav	Corr	Cov	Hav
KB2E-TransE	0.0037	0.0053	0.004	0.18	0.34	0.24
RUMIS	0.36	1.55	0.59	0.59	0.69	0.63
KGE-HAKE	0.12	2.84	0.22	0.34	1.51	0.56
RuLES	1	0.098	0.18	0.99	0.0045	0.0089
HermiT	1	0.30	0.46	1	0.0004	0.0009

object property only allows 1 entity as the tail entity. Meanwhile, 3 is also schema-incorrect, as the reasoner perceives that *pirates* and *seattle_mariners* are the same entity, and that the same entity cannot be linked by an irreflexive object property. RuLES completely outperforms TransC terms of producing schema-correct triples. Thanks to the reasoning mechanism of RuLES, in every iteration of each combination of MapReduce SIC for the NELL-995 Knowledge Graph, RuLES can produce more than 90% schema-correct triples (out of all the new triples produced).

In terms of the $f_{Correctness,Coverage}$, we observe that MapReduce SIC consistently outperforms the classic SIC and the single pass triple producers on all Knowledge Graphs (for the NELL-995 Knowledge Graph, the best combination is R-M-E and for the DBped-P Knowledge Graph, the best combination is R-E-M).

5.4. Comparison with Sing-pass Triple Producers

Last but not least, Table 8 presents the results of the individual triple producer in a single-pass. HermiT and RuLES produce very good quality triples; however, their coverage levels are far away from the most competitive triple producers. Among the individual triple producers, KGE-HAKE is the best in terms of coverage, for both the tested Knowledge Graphs. However, the coverage levels from KGE-HAKE are significantly (about 9-25 times) worse than the best iterative approaches discussed earlier.

5.5. Illustration

Table 9 list all subsets of new triples that are produced by our approach. There are three subsets, which are: (a) *schema-correct*, (b) *schema-incorrect*, and (c) *schema-unknown*. As shown in the table, each triple has three components: the head entity, the object property, and the tail entity. All the triples in the *schema-correct* subset meet the following condition: the type of the head or the tail entity is exactly the required domain or range of an object property. For example, the first triple in the first column of table 9 is schema-correct because the type of the head entity is not disjoint with the domain of the object property *party* and the type of the tail entity is exactly the required range of the object property *party*. As for the *schema-incorrect* subset, the type of the head entity or the type of the tail entity is disjoint with the domain or the range of an object property. For example, the first triple in the second column of table 9 is schema-incorrect because the type of the tail entity is disjoint with the required range of the object property *party*.

As for the *schema-unknown* subset, the type of the head

entity or the type of the tail entity is neither disjoint nor the same with the domain or the range of an object property. For example, the first triple in the third column of table 9 is *schema-unknown*, because the type of the head or the type of the tail entity is neither disjoint nor the same with the domain or the range of the object property *party*.

6. Related Work

Knowledge Graph completion is studied under the umbrella of statistical relational learning (SRL) [16] [62]. There are three main categories of statistical relational learning methods according to [31]: weighted rule learning, tensor factorization/Knowledge Graph embedding, and graph random walk.

The idea of a Knowledge Graph embedding method is to represent an entity as a k -dimensional vector \mathbf{h} (or \mathbf{t}) and to define a scoring function $f_r(\mathbf{h}, \mathbf{t})$ in order to measure the plausibility of the triplet (h, r, t) in the embedding space. The representations of entities and relations are obtained by minimizing a global loss function involving all entities and relations. Different Knowledge Graph embedding algorithms often differ in their scoring function, transformation and loss function [38, 75, 7, 49].

A rule learning method only selects specific relations from the ABox of a Knowledge Graph as their point of interest in expanding the graph. This method will choose the relations that are composed of other relations. Based on these relations, this method will produce a pattern of the form $h(X, Y) \leftarrow p(X, Z); q(Z, Y)$. Furthermore, a rule created according to this pattern will be used for producing new triples. Very recently, [25] uses the feedback from a Knowledge Graph embedding method to improve the quality of the learned rules. As a result, [25] has higher quality completion when compared to any Knowledge Graph completion method that only uses a Knowledge Graph embedding model. However, since this method only considers certain relations, the quality of an expanded Knowledge Graph is uneven. Different rule learning algorithms often differ in their type of rule or the mechanism to produce the rules [22], [72], [81]. All methods in statistical relational learning are aiming only for higher performance in link prediction. Our approach, on the other hand, focuses on adding more triples into a Knowledge Graph and dealing with the correctness of these new triples.

In the last few years, combining the Knowledge Graph embedding and rule learning methods as a completion method has been investigated. Guo [20] pioneered this method by

Algorithm 5 Approximate consistency checking algorithm

```

1: function APR(Inputs: IKG,IJP,Outputs:  $\phi_{IKG},\chi_{IKG},\pi_{IKG}$ )
2:   for each pattern in IJP_1 do
3:     relation, domain, list of Disjoint Classes  $\leftarrow$  split IJP_1 lines
4:     for each line in IKG do
5:       headEntity  $\leftarrow$  split IKG lines
6:       if (headEntity in the list of DisjointClasses) then
7:          $\chi_{IKG} \leftarrow$  line of IKG
8:       else
9:          $\pi_{IKG} \leftarrow$  line of IKG
10:      end if
11:    end for
12:  end for
13:  for each pattern in IJP_2 do
14:    relation, range, list of Disjoint Classes  $\leftarrow$  split IJP_2 lines
15:    for each line in IKG do
16:      tailEntity  $\leftarrow$  split IKG lines
17:      if (tailEntity in the list of DisjointClasses) then
18:         $\chi_{IKG} \leftarrow$  line of IKG
19:      else
20:         $\pi_{IKG} \leftarrow$  line of IKG
21:      end if
22:    end for
23:  end for
24:  for each pattern in IJP_3,IJP_4,IJP_6,IJP_7 do
25:    relation  $\leftarrow$  split IJP lines
26:    for each line in IKG do
27:      predicate  $\leftarrow$  split IKG lines
28:      if (predicate == relation from one of the IJP files) then
29:         $\chi_{IKG} \leftarrow$  line of IKG
30:      end if
31:    end for
32:  end for
33:  for each line in IKG do
34:    predicate, head, tail  $\leftarrow$  split IKG line
35:    if (predicate in IJP_5) then
36:      if headEntity==tailEntity then
37:         $\chi_{IKG} \leftarrow$  line of IKG
38:      end if
39:    end if
40:    if (predicate in IJP_8) then
41:      check whether the head and tail appear more than 1 in IKG
42:      if more than 1 then
43:         $\chi_{IKG} \leftarrow$  line of IKG
44:      end if
45:    end if
46:  end for
47:   $\phi_{IKG} \leftarrow$  IKG -  $\chi_{IKG}$  -  $\pi_{IKG}$ 
48:  return  $\phi_{IKG}, \chi_{IKG}, \pi_{IKG}$ 
49: end function

```

generating the rules from a domain knowledge to further refine the inference results given by TransE. The work by [20] is implemented in KALE [22] (a framework for jointly embedding Knowledge Graphs and rules) and RUGE (Rule-Guided Embedding) [23] that are embedding methods where

triples of a Knowledge Graph and rules are embedded in a joint framework. For ITeRE, which is described in ([83]), the authors point out that their iterative combination can address the sparsity problem for embedding learning and also the efficiency problem for rule learning. SoLE [82] improved on

Table 9
Sets of Newly Produced Triples

schema-correct	schema-incorrect	schema-unknown
Terrence_Murphy_(chiropractor) party Freedom_and_Solidarity	Gheorghe_Tatarescu party Kingdom_of_Romania	Tony_Roman party All_India_Majlis-e-Ittehadul_Muslimeen
Yvon_Levesque successor Deepak_Obhai	Sydney_Irving country Georgy_Malenkov	New_Right_(Netherlands) colour Gunnar_Strang
Francesco_Rutelli party Colombian_Liberal_Party	Geneva mayor Wolfern	Rahmon_Nabiyev primeMinister Yen_Chia-kan
Sandra_Kalnite party Peruvian_Nationalist_Party	Jim_Fulghum country Sergey_Menyaylo	For_the_Autonomies award Reform_Party_(Norway,_1974)
Cadwallader_Colden successor Milt_Harradence	Mauritius internationalAffiliation Nazi_Germany	Vasile_Pintea mergedIntoParty New_Generation_Party

the previous work (KALE, RUGE, and ITeRE) by performing forward chaining inference over rules and injecting the logical background knowledge of rules into embeddings. All of these combination of methods have succeeded in achieving good performance in link prediction tasks. However, the metrics that they used for link prediction (HITS and Mean Rank) are designed only for a static Knowledge Graph and are not suitable for measuring the coverage and the correctness metric of an expanding/growing Knowledge Graph.

To enrich the embedding process, many researchers in the field of the Knowledge Graph embedding consider additional information such as entity types ([21]), relation paths ([37]), textual information ([79]), entity attributes ([78]), temporal information ([18]), logical rules ([22]), as well as the schema of the Knowledge Graph ([9],[40],[13]). We highlight here the last group since it is very related to our approach. Embeds ([9]) enrich their embedding model by considering ontological information in the form of schema axioms, such as: *subclassof*, *subproperty*, *isA*, *domain* and *range*. TransC ([40]) is based on a novel method in this schema-oriented embedding by representing the concepts, instances, and relations differently in the vector spaces. Cose ([13]) is the first method in the field of Knowledge Graph embedding that embeds and completes only the schema of a Knowledge Graph. Compared to Embeds, TransC, and Cose, our approach deals with a wider range of schema axioms, such as *symmetric relation*, *asymmetric relation*, *irreflexive*, *domain range*, *disjointness* and *hierarchy*. Our approach also utilizes the schema of a Knowledge Graph to decide whether a new fact is correct or not.

Several more recent methods use external information such as textual information [37] and logical information/rules [10], [75] to improve their performance in the link prediction task. However, none of these methods guarantee that an expanded Knowledge Graph is consistent with the ontological schema of the original Knowledge Graph.

Farber et al. [11] investigates the consistency of Knowledge Graphs. However, unlike our approach, they focus on concept assertion and datatype properties, rather than relation assertion and object properties.

Error detection is another important related topic. Some existing work aims to detect errors in Knowledge Graphs by various means, such as using outlier detection methods

[76], using learned axioms [41], using axioms from an external ontology [57], or cross-checking with other Knowledge Graphs [39]. Lertvittayakumjorn et al. [36] suggest to automatically correct the detected range violations under closed world semantics.

7. Discussions and Future Works

In this paper, we investigated the importance of the problem of schema-aware Knowledge Graph completion and proposed an iterative and schema-aware approach for Knowledge Graph completion. In this approach, semantic reasoners play a dual role: (1) a reasoner is used as a triple producer, applying materialisation to both TBox and ABox, and (2) another reasoner is used for correctness checking, using the TBox as well as SHACL constraints, to check the quality of triples that are produced by triple producers other than the materialisation reasoner.

Interesting, our experimental results show that the existing correctness notion based on the silver standard is highly questionable. Instead, we propose a new notion of correctness that is based on the Knowledge Graph schema. As far as we know, this is the first approach to applying both consistency checking and SHACL constraints as correctness checking for knowledge graph completion. We found out that the schema-correctness of TransE over NELL-995 is less than 1%. In a sense, this is not very surprising, as there are already some theoretical analysis suggesting that the translation based approaches, such as TransE, have limited expressiveness. For example, Kazemi and Poole [32] show that translation based methods are not fully expressive, only able to represent restricted forms of relations. Gutierrez-Basulto and Schockaert [24] show that translation based methods cannot properly capture simple rules. Our experimental results pragmatically confirm such theoretical analysis, at least for the case of TransE.

Another insight is that, despite the issue of expressiveness, embedding based triple producers can still be helpful should we use them properly. After showing the limitations of translation based methods, Gutierrez-Basulto and Schockaert[24] further show that even the bilinear models, such as DisMult [80] and Simple [24], are severely limited when representing subsumption or equivalence between

relations. This indicates that embedding based triple producers might have limited capability in terms of representing schema of Knowledge Graphs. Our analysis of embedding based triple producers (E-methods) shows that they can produce a lot of triples (including many incorrect ones). If used properly, they can help solve the cold start problem of rule learning based triple producers. Indeed, we have shown that different Knowledge Graph completion methods (i.e., E-methods, R-methods, M-methods) can work with each other and help overcome individual limitations. In fact, our experiments show that some iterative sequential combinations might have significantly better correctness and coverage ratios than other combinations.

Our overall goals are ambitious. We not only aim to make sure that the triple producers can produce schema-correct triples, we also aim to produce as many schema-correct triples as possible. To properly measure these two targets, we have proposed the correctness ratio (for schema-correctness) and the coverage ratio (for productivity) as the main measures. We conducted extensive experiments on the Knowledge Graphs of NELL-995 and DBpedia-Politic. We find out that the R-M-E and R-E-M iterative combinations outperform all the other iterative combinations. As for single-pass triple producers, RUMIS (an R-method) has the best overall performance, within the tested triple producers. The best iterative combination methods outperform RUMIS on both tested Knowledge Graphs, demonstrating amazing coverage ratios of 26.41 in the case of NELL-995 and 39.11 in the case of DBpedia-Politic. In other words, these methods produce 20-40 times of schema-correct triples compared to the set of non-incorrect triples within the original input Knowledge Graphs.

There are many potential paths for future work.

- Firstly, it would be interesting to use our proposed schema-aware evaluation framework to evaluate some state of the art Knowledge Graph completion methods, such as AnyBURL [42] and TUCKER [5]. Given that the silver standard approach is highly questionable, it seems necessary to re-evaluate Knowledge Graph completion methods using our proposed correctness and coverage ratios.
- Secondly, it might be an idea to further restrict our notion of correctness. Even though the the quality of schema-correct triples are a lot better than the schema-incorrect ones, it does not mean that all the schema-correct triples are actually correct [12], since schema could be largely incomplete. One straight forward approach is to further include more SHACL constraints, in addition to the domain and range constraints considered in this paper. There might be some room for further theoretical analysis of correctness too.
- Thirdly, we want to extend or adapt our framework to better understand the connection among approximate Knowledge Graph reasoning [52], uncertain reasoning for Knowledge Graphs [68, 61, 67] and Knowledge

Graph embedding based reasoning, such as [15]. Indeed, our iterative methods can be seen as some kind of hybrid approximate Knowledge Graph reasoning methods, by combining logic based, embedding based and rule learning based methods.

- Furthermore, it might be interesting to see how our proposed framework can be applied in some dynamic settings, such as reasoning [63] and learning [34, 29] for stream Knowledge Graphs, incremental Knowledge Graph completion [42], and temporal Knowledge Graph completion [14].
- Last but not least, given the quality of schema-correct triples are better than the schema-incorrect ones, it would be interesting to see if one can apply them to help improve the performance in downstream applications, such as search [53], recommendation [73] and knowledge based NLP tasks, such as entailment graph construction [27] and knowledge based fake news detection [28].

Acknowledgement

This work was supported by Lembaga Pengelola Dana Pendidikan (LPDP), the Ministry of Finance of Indonesia, IBM Faculty Award and the EU Marie Curie K-Drive project (286348). We would like to thank NELL2RDF ([17]) for providing us the tool for transforming NELL-995 Knowledge Graph into OWL format. Furthermore, we want to thank the anonymous reviewers for their valuable comments and suggestions for improving the quality of the paper.

CRedit authorship contribution statement

Kemas Wiharja: Conceptualization of this study, Methodology, Software, Data Curation. **Jeff Z. Pan:** Methodology, Writing. **Martin J. Kollingbaum:** Methodology, Writing. **Yu Deng:** Methodology, Writing.

References

- [1] Auer, S., Bizer, C., Kobilarov, G., Lehmann, J., Cyganiak, R., Ives, Z.G., 2007. Dbpedia: A nucleus for a web of open data, in: Aberer, K., Choi, K., Noy, N.F., Allemang, D., Lee, K., Nixon, L.J.B., Golbeck, J., Mika, P., Maynard, D., Mizoguchi, R., Schreiber, G., Cudré-Mauroux, P. (Eds.), The Semantic Web, 6th International Semantic Web Conference, 2nd Asian Semantic Web Conference, ISWC 2007 + ASWC 2007, Busan, Korea, November 11-15, 2007, Springer. pp. 722–735. URL: https://doi.org/10.1007/978-3-540-76298-0_52, doi:10.1007/978-3-540-76298-0_52.
- [2] Ayala, D., Borrego, A., Hernández, I., Rivero, C.R., Ruiz, D., 2019. AYNEC: all you need for evaluating completion techniques in knowledge graphs, in: Hitzler, P., Fernández, M., Janowicz, K., Zaveri, A., Gray, A.J.G., López, V., Haller, A., Hammar, K. (Eds.), The Semantic Web - 16th International Conference, ESWC 2019, Portorož, Slovenia, June 2-6, 2019, Proceedings, Springer. pp. 397–411. URL: https://doi.org/10.1007/978-3-030-21348-0_26, doi:10.1007/978-3-030-21348-0_26.
- [3] Baader, F., Brandt, S., Lutz, C., 2005. Pushing the EL envelope, in: IJCAI2015.

- [4] Baader, F., Calvanese, D., McGuinness, D.L., Nardi, D., Patel-Schneider, P.F. (Eds.), 2007. *The Description Logic Handbook*. 2 ed., Cambridge University Press. URL: <https://www.cambridge.org/core/books/description-logic-handbook/F050683766E57EE9BB07BC01BB7A7069>.
- [5] Balazevic, I., Allen, C., Hospedales, T.M., 2019. Tucker: Tensor factorization for knowledge graph completion, in: *EMNLP/IJCNLP 2019*, pp. 5184–5193.
- [6] Bollacker, K.D., Cook, R.P., Tufts, P., 2007. Freebase: A shared database of structured general human knowledge, in: *Proceedings of the Twenty-Second AAAI Conference on Artificial Intelligence*, July 22–26, 2007, Vancouver, British Columbia, Canada, AAAI Press. pp. 1962–1963. URL: <http://www.aaai.org/Library/AAAI/2007/aaai07-355.php>.
- [7] Bordes, A., Usunier, N., García-Durán, A., Weston, J., Yakhnenko, O., 2013. Translating embeddings for modeling multi-relational data, in: Burges, C.J.C., Bottou, L., Ghahramani, Z., Weinberger, K.Q. (Eds.), *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013*. Proceedings of a meeting held December 5–8, 2013, Lake Tahoe, Nevada, United States, pp. 2787–2795. URL: <http://papers.nips.cc/paper/5071-translating-embeddings-for-modeling-multi-relational-data>.
- [8] Calvanese, D., Giacomo, G.D., Lembo, D., Lenzerini, M., Rosati, R., 2007. Tractable reasoning and efficient query answering in description logics: The dl-lite family. *J. OF AUTOMATED REASONING* 39, 385–429.
- [9] Diaz, G.I., Fokoue, A., Sadoghi, M., 2018. Embeds: Scalable, ontology-aware graph embeddings, in: Böhlen, M.H., Pichler, R., May, N., Rahm, E., Wu, S., Hose, K. (Eds.), *Proceedings of the 21th International Conference on Extending Database Technology, EDBT 2018, Vienna, Austria, March 26–29, 2018, OpenProceedings.org*. pp. 433–436. URL: <https://doi.org/10.5441/002/edbt.2018.40>, doi:10.5441/002/edbt.2018.40.
- [10] Du, J., Qi, K., Wan, H., Peng, B., Lu, S., Shen, Y., 2017. Enhancing knowledge graph embedding from a logical perspective, in: [74]. pp. 232–247. URL: https://doi.org/10.1007/978-3-319-70682-5_15, doi:10.1007/978-3-319-70682-5_15.
- [11] Färber, M., Bartscherer, F., Menne, C., Rettinger, A., 2018. Linked data quality of dbpedia, freebase, opencyc, wikidata, and YAGO. *Semantic Web* 9, 77–129. URL: <https://doi.org/10.3233/SW-170275>, doi:10.3233/SW-170275.
- [12] Flouris, D., Huang, Z., Pan, J.Z., Plexousakis, D., Wache, H., 2006. Inconsistencies, negations and changes in ontologies, in: *AAAI 2006*, pp. 1295–1300.
- [13] Gao, H., Zheng, X., Li, W., Qi, G., Wang, M., 2019. Cosine-based embedding for completing schematic knowledge, in: Tang, J., Kan, M., Zhao, D., Li, S., Zan, H. (Eds.), *Natural Language Processing and Chinese Computing - 8th CCF International Conference, NLPCC 2019, Dunhuang, China, October 9–14, 2019, Proceedings, Part I*, Springer. pp. 249–261. URL: https://doi.org/10.1007/978-3-030-32233-5_20, doi:10.1007/978-3-030-32233-5_20.
- [14] García-Durán, A., Dumancic, S., Niepert, M., 2018. Learning Sequence Encoders for Temporal Knowledge Graph Completion, in: *EMNLP 2018*, pp. 4816–4821.
- [15] Garg, D., Ikbal, S., Srivastava, S.K., Vishwakarma, H., Karanam, H.P., Subramaniam, L.V., 2019. Quantum embedding of knowledge for reasoning, in: *NeurIPS 2019*, pp. 5595–5605.
- [16] Getoor, L., Taskar, B., 2007. *Introduction to statistical relational learning (adaptive computation and machine learning)*.
- [17] Giménez-García, J.M., Duarte, M.C., Zimmermann, A., Gravier, C., Jr., E.R.H., Maret, P., 2018. NELL2RDF: reading the web, and publishing it as linked data. *CoRR abs/1804.05639*. URL: <http://arxiv.org/abs/1804.05639>, arXiv:1804.05639.
- [18] Goel, R., Kazemi, S.M., Brubaker, M., Poupard, P., 2019. Diachronic embedding for temporal knowledge graph completion. *CoRR abs/1907.03143*. URL: <http://arxiv.org/abs/1907.03143>, arXiv:1907.03143.
- [19] Grosz, B.N., Horrocks, I., Volz, R., Decker, S., 2003. Description logic programs: combining logic programs with description logic, in: *WWW2003*, pp. 48–57.
- [20] Guo, S., Ding, B., Wang, Q., Wang, L., Wang, B., 2016a. Knowledge base completion via rule-enhanced relational learning, in: Chen, H., Ji, H., Sun, L., Wang, H., Qian, T., Ruan, T. (Eds.), *Knowledge Graph and Semantic Computing: Semantic, Knowledge, and Linked Big Data - First China Conference, CCKS 2016, Beijing, China, September 19–22, 2016, Revised Selected Papers*, Springer. pp. 219–227. URL: https://doi.org/10.1007/978-981-10-3168-7_22, doi:10.1007/978-981-10-3168-7_22.
- [21] Guo, S., Wang, Q., Wang, B., Wang, L., Guo, L., 2015. Semantically smooth knowledge graph embedding, in: *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26–31, 2015, Beijing, China, Volume 1: Long Papers, The Association for Computer Linguistics*. pp. 84–94. URL: <https://doi.org/10.3115/v1/p15-1009>, doi:10.3115/v1/p15-1009.
- [22] Guo, S., Wang, Q., Wang, L., Wang, B., Guo, L., 2016b. Jointly embedding knowledge graphs and logical rules, in: Su, J., Carreras, X., Duh, K. (Eds.), *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1–4, 2016, The Association for Computational Linguistics*. pp. 192–202. URL: <https://www.aclweb.org/anthology/D16-1019/>.
- [23] Guo, S., Wang, Q., Wang, L., Wang, B., Guo, L., 2018. Knowledge graph embedding with iterative guidance from soft rules, in: McIlraith, S.A., Weinberger, K.Q. (Eds.), *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2–7, 2018, AAAI Press*. pp. 4816–4823. URL: <https://www.aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/16369>.
- [24] Gutiérrez-Basulto, V., Schockaert, S., 2018. From knowledge graph embedding to ontology embedding? an analysis of the compatibility between vector space representations and rules, in: *KR2018*, pp. 379–388.
- [25] Ho, V.T., Stepanova, D., Gad-Elrab, M.H., Kharlamov, E., Weikum, G., 2018. Rule learning from knowledge graphs guided by embedding models, in: Vrandečić, D., Bontcheva, K., Suárez-Figueroa, M.C., Presutti, V., Celino, I., Sabou, M., Kaffee, L., Simperl, E. (Eds.), *The Semantic Web - ISWC 2018 - 17th International Semantic Web Conference, Monterey, CA, USA, October 8–12, 2018, Proceedings, Part I*, Springer. pp. 72–90. URL: https://doi.org/10.1007/978-3-030-00671-6_5, doi:10.1007/978-3-030-00671-6_5.
- [26] Horrocks, I., Kutz, O., Sattler, U., 2006. The even more irresistible sroiq, in: *KR2006*, pp. 57–67.
- [27] Hosseini, M.J., Chambers, N., Reddy, S., Holt, X.R., Cohen, S.B., Johnson, M., Steedman, M., 2018. Learning typed entailment graphs with global soft constraints. *Transactions of the Association for Computational Linguistics* 6, 703–717. URL: <https://www.aclweb.org/anthology/Q18-1048>, doi:10.1162/tacl_a_00250.
- [28] Jeff Z. Pan, Siyana Pavlova, C.L.N.L.Y.L.J.L., 2018. Content based fake news detection using knowledge graphs, in: *ISWC 2018*, pp. 669–683.
- [29] Jiaoyan Chen, Freddy Lecue, J.Z.P.H.C., . Learning from Ontology Streams with Semantic Concept Drift.
- [30] Kang, N., van Mulligen, E.M., Kors, J.A., 2012. Training text chunkers on a silver standard corpus: can silver replace gold? *BMC Bioinformatics* 13, 17. URL: <https://doi.org/10.1186/1471-2105-13-17>, doi:10.1186/1471-2105-13-17.
- [31] Kazemi, S.M., Poole, D., 2018a. Simple embedding for link prediction in knowledge graphs, in: Bengio, S., Wallach, H.M., Larochelle, H., Grauman, K., Cesa-Bianchi, N., Garnett, R. (Eds.), *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Process-*

- ing Systems 2018, NeurIPS 2018, 3-8 December 2018, Montréal, Canada., pp. 4289–4300. URL: <http://papers.nips.cc/paper/7682-simple-embedding-for-link-prediction-in-knowledge-graphs>.
- [32] Kazemi, S.M., Poole, D., 2018b. Simple embedding for link prediction in knowledge graphs, in: *Advances in Neural Information Processing Systems* 31, pp. 4284–4295.
- [33] Krötzsch, M., Rudolph, S., Hitzler, P., 2008. Description logic rules, in: *ECAI2008*, pp. 80–84.
- [34] Lecue, F., Pan, J.Z., 2013. Predicting Knowledge in An Ontology Stream, in: *Proc. of the 23rd International Joint Conference on Artificial Intelligence (IJCAI 2013)*.
- [35] Lehmann, J., Isle, R., Jakob, M., Jentzsch, A., Kontokostas, D., Mendes, P.N., Hellmann, S., Morsey, M., van Kleef, P., Auer, S., Bizer, C., 2015. Dbpedia - A large-scale, multilingual knowledge base extracted from wikipedia. *Semantic Web* 6, 167–195. URL: <https://doi.org/10.3233/SW-140134>, doi:10.3233/SW-140134.
- [36] Lertvittayakumjorn, P., Kertkeidkachorn, N., Ichise, R., 2017. Resolving range violations in dbpedia, in: [74], pp. 121–137. pp. 121–137. URL: https://doi.org/10.1007/978-3-319-70682-5_8, doi:10.1007/978-3-319-70682-5_8.
- [37] Lin, Y., Liu, Z., Luan, H., Sun, M., Rao, S., Liu, S., 2015a. Modeling relation paths for representation learning of knowledge bases, in: Márquez, L., Callison-Burch, C., Su, J., Pighin, D., Marton, Y. (Eds.), *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015, The Association for Computational Linguistics*. pp. 705–714. URL: <https://www.aclweb.org/anthology/D15-1082/>.
- [38] Lin, Y., Liu, Z., Sun, M., Liu, Y., Zhu, X., 2015b. Learning entity and relation embeddings for knowledge graph completion, in: Bonet, B., Koenig, S. (Eds.), *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, January 25-30, 2015, Austin, Texas, USA, AAAI Press*. pp. 2181–2187. URL: <http://www.aaai.org/ocs/index.php/AAAI/AAAI15/paper/view/9571>.
- [39] Liu, S., d’Aquin, M., Motta, E., 2015. Towards linked data fact validation through measuring consensus, in: Rula, A., Zaveri, A., Knuth, M., Kontokostas, D. (Eds.), *Proceedings of the 2nd Workshop on Linked Data Quality co-located with 12th Extended Semantic Web Conference (ESWC 2015), Portorož, Slovenia, June 1, 2015, CEUR-WS.org*. URL: http://ceur-ws.org/Vol-1376/LDQ2015_paper_04.pdf.
- [40] Lv, X., Hou, L., Li, J., Liu, Z., 2018. Differentiating concepts and instances for knowledge graph embedding, in: Riloff, E., Chiang, D., Hockenmaier, J., Tsujii, J. (Eds.), *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018, Association for Computational Linguistics*. pp. 1971–1979. URL: <https://www.aclweb.org/anthology/D18-1222/>.
- [41] Ma, Y., Gao, H., Wu, T., Qi, G., 2014. Learning disjointness axioms with association rule mining and its application to inconsistency detection of linked data, in: Zhao, D., Du, J., Wang, H., Wang, P., Ji, D., Pan, J.Z. (Eds.), *The Semantic Web and Web Science - 8th Chinese Conference, CSWS 2014, Wuhan, China, August 8-12, 2014, Revised Selected Papers, Springer*. pp. 29–41. URL: https://doi.org/10.1007/978-3-662-45495-4_3, doi:10.1007/978-3-662-45495-4_3.
- [42] Meilicke, C., Chekol, M.W., Ruffinelli, D., Stuckenschmidt, H., 2019. Anytime bottom-up rule learning for knowledge graph completion, in: *IJCAI 2019*, pp. 3137–3143.
- [43] Mendes, P.N., Mühleisen, H., Bizer, C., 2012. Sieve: linked data quality assessment and fusion, in: Srivastava, D., Ari, I. (Eds.), *Proceedings of the 2012 Joint EDBT/ICDT Workshops, Berlin, Germany, March 30, 2012, ACM*. pp. 116–123. URL: <https://doi.org/10.1145/2320765.2320803>, doi:10.1145/2320765.2320803.
- [44] Miller, G.A., 1995. Wordnet: A lexical database for english. *Commun. ACM* 38, 39–41. URL: <http://doi.acm.org/10.1145/219717.219748>, doi:10.1145/219717.219748.
- [45] Miller, G.A., Hristea, F., 2006. Wordnet nouns: Classes and instances. *Comput. Linguistics* 32, 1–3. URL: <https://doi.org/10.1162/coli.2006.32.1.1>, doi:10.1162/coli.2006.32.1.1.
- [46] Mitchell, T.M., Cohen, W.W., Jr., E.R.H., Talukdar, P.P., Yang, B., Betteridge, J., Carlson, A., Mishra, B.D., Gardner, M., Kisiel, B., Krishnamurthy, J., Lao, N., Mazaitis, K., Mohamed, T., Nakashole, N., Platanios, E.A., Ritter, A., Samadi, M., Settles, B., Wang, R.C., Wijaya, D., Gupta, A., Chen, X., Saparov, A., Greaves, M., Welling, J., 2018. Never-ending learning. *Commun. ACM* 61, 103–115. URL: <https://doi.org/10.1145/3191513>, doi:10.1145/3191513.
- [47] Motik, B., Horrocks, I., 2008. Owl datatypes: Design and implementation, in: *ISWC2008*, pp. 307–322.
- [48] Nguyen, D.Q., Sirts, K., Qu, L., Johnson, M., 2016. Stranse: a novel embedding model of entities and relationships in knowledge bases, in: Knight, K., Nenkova, A., Rambow, O. (Eds.), *NAACL HLT 2016, The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego California, USA, June 12-17, 2016, The Association for Computational Linguistics*. pp. 460–466. URL: <https://www.aclweb.org/anthology/N16-1054/>.
- [49] Nickel, M., Tresp, V., Kriegel, H., 2011. A three-way model for collective learning on multi-relational data, in: Getoor, L., Scheffer, T. (Eds.), *Proceedings of the 28th International Conference on Machine Learning, ICMML 2011, Bellevue, Washington, USA, June 28 - July 2, 2011, Omnipress*. pp. 809–816. URL: https://icml.cc/2011/papers/438_icmlpaper.pdf.
- [50] Pan, J.Z., Calvanese, D., Eiter, T., Horrocks, I., Kifer, M., Lin, F., Zhao, Y. (Eds.), 2017a. Reasoning Web: Logical Foundation of Knowledge Graph Construction and Query Answering - 12th International Summer School 2016, Aberdeen, UK, September 5-9, 2016, Tutorial Lectures. volume 9885 of *Lecture Notes in Computer Science*, Springer. URL: <https://doi.org/10.1007/978-3-319-49493-7>, doi:10.1007/978-3-319-49493-7.
- [51] Pan, J.Z., Horrocks, I., 2006. OWL-Eu: Adding Customised Datatypes into OWL. *Journal of Web Semantics* , 29–39.
- [52] Pan, J.Z., Ren, Y., Zhao, Y., 2016. Tractable approximate deduction for OWL. *Artificial Intelligence* , 95–155.
- [53] Pan, J.Z., Taylor, S., Thomas, E., 2009. JReducing Ambiguity in Tagging Systems with Folksonomy Search Expansion, in: the Proc. of the 6th European Semantic Web Conference (ESWC2009).
- [54] Pan, J.Z., Thomas, E., 2007. Approximating OWL-DL Ontologies, in: the Proc. of the 22nd National Conference on Artificial Intelligence (AAAI-07), pp. 1434–1439.
- [55] Pan, J.Z., Vetere, G., Gómez-Pérez, J.M., Wu, H. (Eds.), 2017b. Exploiting Linked Data and Knowledge Graphs in Large Organisations. Springer. URL: <https://doi.org/10.1007/978-3-319-45654-6>, doi:10.1007/978-3-319-45654-6.
- [56] Paulheim, H., 2017. Knowledge graph refinement: A survey of approaches and evaluation methods. *Semantic Web* 8, 489–508. URL: <https://doi.org/10.3233/SW-160218>, doi:10.3233/SW-160218.
- [57] Paulheim, H., Gangemi, A., 2015. Serving dbpedia with DOLCE - more than just adding a cherry on top, in: Arenas, M., Corcho, Ó., Simperl, E., Strohmaier, M., d’Aquin, M., Srinivas, K., Groth, P.T., Dumontier, M., Heflin, J., Thirunarayan, K., Staab, S. (Eds.), *The Semantic Web - ISWC 2015 - 14th International Semantic Web Conference, Bethlehem, PA, USA, October 11-15, 2015, Proceedings, Part I, Springer*. pp. 180–196. URL: https://doi.org/10.1007/978-3-319-25007-6_11, doi:10.1007/978-3-319-25007-6_11.
- [58] Paulheim, H., Stuckenschmidt, H., 2016. Fast approximate a-box consistency checking using machine learning, in: Sack, H., Blomqvist, E., d’Aquin, M., Ghidini, C., Ponzetto, S.P., Lange, C. (Eds.), *The Semantic Web. Latest Advances and New Domains - 13th International Conference, ESWC 2016, Heraklion, Crete, Greece, May 29 - June 2, 2016, Proceedings, Springer*. pp. 135–150. URL: https://doi.org/10.1007/978-3-319-34129-3_9, doi:10.1007/978-3-319-34129-3_9.
- [59] Pujara, J., Augustine, E., Getoor, L., 2017. Sparsity and noise: Where knowledge graph embeddings fall short, in: Palmer, M., Hwa, R., Riedel, S. (Eds.), *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017, Association for Computational Linguistics*. pp. 1751–1756. URL: <https://www.aclweb.org/anthology/D17-1184/>.

- [60] Pujara, J., Miao, H., Getoor, L., Cohen, W.W., 2013. Knowledge graph identification, in: Alani, H., Kagal, L., Fokoue, A., Groth, P.T., Biemann, C., Parreira, J.X., Aroyo, L., Noy, N.F., Welty, C., Janowicz, K. (Eds.), *The Semantic Web - ISWC 2013 - 12th International Semantic Web Conference*, Sydney, NSW, Australia, October 21-25, 2013, Proceedings, Part I, Springer. pp. 542–557. URL: https://doi.org/10.1007/978-3-642-41335-3_34, doi:10.1007/978-3-642-41335-3_34.
- [61] Qi, G., Pan, J.Z., Ji, Q., 2007. A Possibilistic Extension of Description Logics, in: Proc. of 2007 International Workshop on Description Logics (DL2007).
- [62] Raedt, L.D., Kersting, K., Natarajan, S., Poole, D., 2016. *Statistical Relational Artificial Intelligence: Logic, Probability, and Computation. Synthesis Lectures on Artificial Intelligence and Machine Learning*, Morgan & Claypool Publishers. URL: <https://doi.org/10.2200/S00692ED1V01Y201601AIM032>, doi:10.2200/S00692ED1V01Y201601AIM032.
- [63] Ren, Y., Pan, J.Z., 2011. Optimising Ontology Stream Reasoning with Truth Maintenance System, in: Proc. of the ACM Conference on Information and Knowledge Management (CIKM 2011).
- [64] Ren, Y., Pan, J.Z., Zhao, Y., 2010. Soundness Preserving Approximation for TBox Reasoning, in: AAAI2010.
- [65] Röder, M., Sherif, M.A., Saleem, M., Conrads, F., Ngomo, A.N., 2020. Benchmarking knowledge graphs on the web. CoRR abs/2002.06039. URL: <https://arxiv.org/abs/2002.06039>, arXiv:2002.06039.
- [66] Shearer, R., Motik, B., Horrocks, I., 2008. Hermit: A highly-efficient OWL reasoner, in: Dolbear, C., Ruttenberg, A., Sattler, U. (Eds.), *Proceedings of the Fifth OWLED Workshop on OWL: Experiences and Directions*, collocated with the 7th International Semantic Web Conference (ISWC-2008), Karlsruhe, Germany, October 26-27, 2008, CEUR-WS.org. URL: http://ceur-ws.org/Vol-432/owlled2008eu_submission_12.pdf.
- [67] Stoilos, G., Stamou, G., Pan, J.Z., Tzouvaras, V., Horrocks, I., Reasoning with Very Expressive Fuzzy Description Logics. *Journal of Artificial Intelligence Research*, 273–320.
- [68] Stoilos, G., Stamou, G.B., Pan, J.Z., 2006. Handling imprecise knowledge with fuzzy description logic, in: DL 2006.
- [69] Tanon, T.P., Vrandečić, D., Schaffert, S., Steiner, T., Pintscher, L., 2016. From freebase to wikidata: The great migration, in: Bourdeau, J., Hendler, J., Nkambou, R., Horrocks, I., Zhao, B.Y. (Eds.), *Proceedings of the 25th International Conference on World Wide Web, WWW 2016, Montreal, Canada, April 11 - 15, 2016*, ACM. pp. 1419–1428. URL: <https://doi.org/10.1145/2872427.2874809>, doi:10.1145/2872427.2874809.
- [70] Thomas, E., Pan, J.Z., Ren, Y., 2010. TrOWL: Tractable OWL 2 reasoning infrastructure, in: ESWC. URL: https://link.springer.com/chapter/10.1007/978-3-642-13489-0_38.
- [71] Töpper, G., Knuth, M., Sack, H., 2012. Dbpedia ontology enrichment for inconsistency detection, in: Presutti, V., Pinto, H.S. (Eds.), *I-SEMANTICS 2012 - 8th International Conference on Semantic Systems*, @proceedingsDBLP:conf/i-semantics/2012, editor = Valentina Presutti and Helena Sofia Pinto, title = I-SEMANTICS 2012 - 8th International Conference on Semantic Systems, I-SEMANTICS '12, Graz, Austria, September 5-7, 2012, publisher = ACM, year = 2012, url = <http://dl.acm.org/citation.cfm?id=2362499>, isbn = 978-1-4503-1112-0, timestamp = Fri, 07 Sep 2012 13:34:38 +0200, biburl = <https://dblp.org/rec/conf/i-semantics/2012.bib>, bibsource = dblp computer science bibliography, <https://dblp.org> I-SEMANTICS '12, Graz, Austria, September 5-7, 2012, ACM. pp. 33–40. URL: <https://doi.org/10.1145/2362499.2362505>, doi:10.1145/2362499.2362505.
- [72] Tran, H.D., Stepanova, D., Gad-Elrab, M.H., Lisi, F.A., Weikum, G., 2016. Towards nonmonotonic relational learning from knowledge graphs, in: Cussens, J., Russo, A. (Eds.), *Inductive Logic Programming - 26th International Conference, ILP 2016, London, UK, September 4-6, 2016, Revised Selected Papers*, Springer. pp. 94–107. URL: https://doi.org/10.1007/978-3-319-63342-8_8, doi:10.1007/978-3-319-63342-8_8.
- [73] Wang, X., He, X., Cao, Y., Liu, M., Chua, T.S., 2019. KGAT: Knowledge Graph Attention Network for Recommendation, in: KDD 2019.
- [74] Wang, Z., Turhan, A., Wang, K., Zhang, X. (Eds.), 2017. *Semantic Technology - 7th Joint International Conference, JIST 2017, Gold Coast, QLD, Australia, November 10-12, 2017*, Proceedings. volume 10675 of *Lecture Notes in Computer Science*, Springer. URL: <https://doi.org/10.1007/978-3-319-70682-5>, doi:10.1007/978-3-319-70682-5.
- [75] Wang, Z., Zhang, J., Feng, J., Chen, Z., 2014. Knowledge graph embedding by translating on hyperplanes, in: Brodley, C.E., Stone, P. (Eds.), *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*, July 27 -31, 2014, Québec City, Québec, Canada, AAAI Press. pp. 1112–1119. URL: <http://www.aaai.org/ocs/index.php/AAAI/AAAI14/paper/view/8531>.
- [76] Wienand, D., Paulheim, H., 2014. Detecting incorrect numerical data in dbpedia, in: Presutti, V., d'Amato, C., Gandon, F., d'Aquin, M., Staab, S., Tordai, A. (Eds.), *The Semantic Web: Trends and Challenges - 11th International Conference, ESWC 2014, Anissaras, Crete, Greece, May 25-29, 2014*, Proceedings, Springer. pp. 504–518. URL: https://doi.org/10.1007/978-3-319-07443-6_34, doi:10.1007/978-3-319-07443-6_34.
- [77] Wiharja, K., Pan, J.Z., Kollingbaum, M.J., Deng, Y., 2018. More is better: Sequential combinations of knowledge graph embedding approaches, in: Ichise, R., Lécué, F., Kawamura, T., Zhao, D., Muggleton, S., Kozaki, K. (Eds.), *Semantic Technology - 8th Joint International Conference, JIST 2018, Awaji, Japan, November 26-28, 2018*, Proceedings, Springer. pp. 19–35. URL: https://doi.org/10.1007/978-3-030-04284-4_2, doi:10.1007/978-3-030-04284-4_2.
- [78] Wu, Y., Wang, Z., 2018. Knowledge graph embedding with numeric attributes of entities, in: Augenstein, I., Cao, K., He, H., Hill, F., Gella, S., Kiro, J., Mei, H., Misra, D. (Eds.), *Proceedings of The Third Workshop on Representation Learning for NLP, Rep4NLP@ACL 2018, Melbourne, Australia, July 20, 2018*, Association for Computational Linguistics. pp. 132–136. URL: <https://doi.org/10.18653/v1/w18-3017>, doi:10.18653/v1/w18-3017.
- [79] Xie, R., Liu, Z., Jia, J., Luan, H., Sun, M., 2016. Representation learning of knowledge graphs with entity descriptions, in: Schuurmans, D., Wellman, M.P. (Eds.), *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, February 12-17, 2016, Phoenix, Arizona, USA, AAAI Press. pp. 2659–2665. URL: <http://www.aaai.org/ocs/index.php/AAAI/AAAI16/paper/view/12216>.
- [80] Yang, B., tau Yih, W., He, X., Gao, J., Deng, L., 2015. Embedding entities and relations for learning and inference in knowledge bases, in: ICLR2015.
- [81] Yang, F., Yang, Z., Cohen, W.W., 2017. Differentiable learning of logical rules for knowledge base reasoning, in: Guyon, I., von Luxburg, U., Bengio, S., Wallach, H.M., Fergus, R., Vishwanathan, S.V.N., Garnett, R. (Eds.), *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pp. 2319–2328. URL: <http://papers.nips.cc/paper/6826-differentiable-learning-of-logical-rules-for-knowledge-base-reasoning>.
- [82] Zhang, J., Li, J., 2019. Enhanced knowledge graph embedding by jointly learning soft rules and facts. *Algorithms* 12, 265. URL: <https://doi.org/10.3390/a12120265>, doi:10.3390/a12120265.
- [83] Zhang, W., Paudel, B., Wang, L., Chen, J., Zhu, H., Zhang, W., Bernstein, A., Chen, H., 2019. Iteratively learning embeddings and rules for knowledge graph reasoning, in: Liu, L., White, R.W., Mantrach, A., Silvestri, F., McAuley, J.J., Baeza-Yates, R., Zia, L. (Eds.), *The World Wide Web Conference, WWW 2019, San Francisco, CA, USA, May 13-17, 2019*, ACM. pp. 2366–2377. URL: <https://doi.org/10.1145/3308558.3313612>, doi:10.1145/3308558.3313612.
- [84] Zhang, Z., Cai, J., Zhang, Y., Wang, J., 2020. Learning hierarchy-aware knowledge graph embeddings for link prediction, in: *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, AAAI Press. pp. 3065–3072. URL:

<https://aaai.org/ojs/index.php/AAAI/article/view/5701>.

- [85] Zhou, Y., Grau, B.C., Horrocks, I., Wu, Z., Banerjee, J., 2013. Making the most of your triple store: query answering in OWL 2 using an RL reasoner, in: WWW2013, pp. 1569–1580.

Kemas Wiharja is currently a Ph.D. student at the University of Aberdeen and a Lecturer at the School of Computing, Telkom University, Indonesia. His research interests include Knowledge Graph completion, Knowledge Graph reasoning, knowledge representation, and data science.

Jeff Z. Pan is a Reader in Knowledge Graphs in the School of Informatics at The University of Edinburgh. He received his Ph.D. in computer science from The University of Manchester. His research focuses primarily on knowledge representation and artificial intelligence, in particular on Knowledge Graph based learning and reasoning, and knowledge based natural language understanding and generations, as well as their applications. He is an Associate Editor of the Journal of Web Semantics (JWS).

Martin J. Kollingbaum holds a Ph.D. in Computing Science (University of Aberdeen). He is a Researcher at Lakeside Labs, Austria. His research interests are Multiagent Systems, with a focus on normative reasoning, and applications in Transport and Logistics.

Yu Deng is a Research Staff Member at IBM T.J. Watson Research Center. Her research interests are in information extraction, question answering, Knowledge Graph and semantic analysis. Yu is an IBM Academy of Technology member and IBM Master Inventor. She has received her Ph.D. in Computer Science from the University of Maryland, College Park, and is an IEEE Senior Member.