More is Better: Sequential Combinations of Knowledge Graph Embedding Approaches

Kemas Wiharja[1], Jeff Z. Pan[1], and Martin Kollingbaum[1] and Yu Deng[2]

[1] Department of Computing Science, University of Aberdeen, UK
[2] IBM Research, US

Abstract. Constructing and maintaining large-scale good quality knowledge graphs present many challenges. Knowledge graph completion has been regarded a promising direction in the knowledge graph community. The majority of current work for knowledge graph completion approaches do not take the schema of a target knowledge graph as input. As a result, the triples generated by these approaches are not necessarily consistent with the schema of the target knowledge graph. This paper proposes to improve the correctness of knowledge graph completion based on Schema Aware Triple Classification (SATC), which enables sequential combinations of knowledge graph embedding approaches. Extensive experiments show that our proposed approaches can significantly improve the correctness of the new triples produced by knowledge graph embedding methods.

Keywords: Knowledge Graph · Embedding · Schema Aware Triple Classification · Knowledge Representation and Reasoning · Approximate Reasoning · Artificial Intelligence.

1 Introduction

The idea of representing knowledge as graphs dates back to early proposals such as semantic networks in the research of Knowledge Representation and Reasoning, which is an important branch of Artificial Intelligence. It has become popular again since Google coined the term "Knowledge Graph" and used it to improve its search engine in 2012. In general, a knowledge graph 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 knowledge graph [1, 2]. It is often assumed that, in a knowledge graph, the size of the data (or *ABox* in description logic terminology) is much bigger than the size of the schema (or *TBox* in description logic terminology) [3, 2].

Constructing and maintaining large-scale good quality knowledge graphs presents many challenges. At the core of them is the well-known trade-off between completeness and correctness. Most existing knowledge graph refinement approaches either focus on adding missing knowledge to the graph, i.e., completion, or on identifying wrong information in the graph, i.e., error detection.

Knowledge graph completion is sometimes also regarded as a kind of knowledge graph reasoning, or similarity reasoning [4], to be more precise, which is different from logic-based knowledge graph reasoning (cf. Sec 2.1). Several research directions have emerged to complete a knowledge graph, such as knowledge graph embedding and rule learning. Most of them focus on multi-relational data [5], while ignoring the TBox schema and the type assertions in ABox. Link Prediction (LP) [11] is perhaps one of the most studied notions of knowledge graph completion. Given an entity and a relation, as well as the ABox of a knowledge graph, the task is to predict the missing entity in this relation. Knowledge graph embedding approaches, such as TransE [5], TransR [8], TransH [9], STransE [10] (and many others), complete an input knowledge graph ABox by representing the entities and their relations in a vector space, so as to learn some embedded representations of entities and relations in a knowledge graph for predicting missing entities or relationships. Rule learning approaches, such as [29, 30], complete an input knowledge graph ABox by learning rules based on patterns in the ABox. The learnt rules can then be used to produce new relation assertions.

Correctness checking in knowledge graph completion often relies on the task of triple classification [12], which computes the likelihood a given triple is correct or not w.r.t. a given knowledge graph. The risk, however, is that the current notion of task classification does not usually take the schema of the target ontology into account. Therefore, the triples regarded as *correct* under this notion of triple classification might *not* be consistent with the schema of the target knowledge graph.

In this paper, we propose to improve the correctness of knowledge graph completion based on a Schema Aware Triple Classification (SATC). In general, our approach can be applied to both knowledge graph embedding and rule learning approaches mentioned above. In this paper, we will focus on the knowledge graph embedding (KGE) approaches and leave the rule learning approaches for future work. We envision two types of SATC: black-box SATC and white-box SATC approaches. The latter approaches require a tight integration of logical reasoning into KG embedding, while the former approaches do not. In this paper, we will focus on black-box SATC. The advantage of the black-box SATC approaches is that they can be applied to any KGE approaches with any KG reasoners, without having to revise the KGE algorithms. The key contributions of the paper can be summarized as follows:

1. To the best of our knowledge, this work presents the first black-box SATC approach. In order to evaluate the feasibility of this approach, we conducted several experiments with some knowledge graphs, including NELL-995, DBpedia-Politics and a knowledge graph from IBM, automatically generated based on IBM Storwize Knowledge Center articles. Our investigation shows that applying some KGE approaches, such as TransE, on NELL-995, the percentage of correct triples is less than 1% under SATC. This confirms the importance of using schema for quality assurance.
2. In our SATC approach, we propose to use approximate reasoning to help improve the efficiency of consistency checking w.r.t. the schema.

3. We propose to use our SATC approach to support sequential combinations of the KGE approaches, by applying the KGE approaches over the union of the original graph and the correct triples from a previous iteration of KG completion. Our experiment confirms that, with the help of SATC, the KGE approaches can be run sequentially to get higher percentages of correct triples. Extensive experiments show that our proposed approaches can significantly improve the correctness of the new triples produced by the knowledge graph embedding methods; e.g., in the case of NELL-995, the improvement is over 42%.

2 Background

2.1 Knowledge Graph

More formally, we define a knowledge graph $\mathcal{G} = \mathcal{T} \cup \mathcal{A}$ consisting of two parts, $\mathcal{T}$ and $\mathcal{A}$, where $\mathcal{A}$ is the data sub-graph (or ABox) and $\mathcal{T}$ is the schema sub-graph (or TBox) [1, 2]. The size of $\mathcal{T}$ is often much smaller than that of $\mathcal{A}$. Facts in the ABox are represented as triples of the following two forms:

– *Relation assertion* (h,r,t), where h is the head entity, and r the relation and t the tail entity; e.g., (ACMilan, playInLeague, ItalianLeague) is a relation assertion.
– *Type assertion* (e, rdf:type, C), where eis an entity, rdf:type is the instance-of relation from the standard W3C RDF specification and C is a type; e.g., (ACMilan, rdf:type, FootballClub) is a type assertion.

A TBox includes Type Inclusion axioms C $\sqsubseteq$ D, where C and D are type descriptions, such as the following ones: $\top \mid \bot \mid$ A $\mid \neg$C$\mid$ C $\sqcap$ D $\mid \exists$r.C $\mid \{o\}$, where $\top$ is the top type (representing all entities), $\bot$ is the bottom type (representing an empty set), A is a named type, r is a relation, and o is an entity. For example, given the two types River and City, the disjointness of these types can be represented as River $\sqsubseteq \neg$City, or River $\sqcap$ City $\sqsubseteq \bot$.

A TBox can also include *Relation Domain axioms* Domain(r) = A, *Relation Range axioms* Range (r) = A, *Symmetric Relation axioms* Symmetric(r), *Asymmetric Relation axioms* Asymmetric(r), *Relation Inclusion axioms* r1$\sqsubseteq$r2 and *Relation Chain axioms* r1 $\circ$ r2 $\sqsubseteq$ r3, where r1, r2 and r3 are properties. Note that we allow multiple domains (and ranges), and the resulting domain (range) is the intersection of the individual domains (ranges).

2.2 Knowledge Graph Completion

Given a knowledge graph $\mathcal{G} = \mathcal{T} \cup \mathcal{A}$, the task of knowledge graph completion (KGC) is to produce an extension ABox $\mathcal{A}$', of which the triples use only types and relations from $\mathcal{T}$. The task of link prediction is a special form of KGC, in that triples in $\mathcal{A}$' only are relation assertions only. More precisely, the task of link prediction is to predict the missing head h or the missing tail t of a triple (h,r,t). Link prediction methods often output a list of ranked entities instead of providing the best entity [14].

In this paper, we focus on knowledge graph embedding (KGE) approaches. The idea of embedding is to represent an entity as a k-dimensional vector **h** (or **t**) and defines a scoring function $f_r(\mathbf{h},\mathbf{t})$ to measure the plausibility of the triple (h,r,t) in the embedding space. The representations of entities and relations are obtained by minimising a global loss function involving all entities and relations. Different KGE algorithms often differ in their scoring functions, transformations, and loss functions.

Triple classification (TC) is a KGC related task of searching for a relation-specific threshold σ_r to identify whether a triple (h,r,t) is plausible [12]. For doing TC, one needs to construct three datasets which are train, test and development datasets. The development dataset is considered as the golden standard. In reality, golden standards take too many efforts. Thus silver standards are often applied, which assume that the input knowledge graph itself is already of reasonable quality. This assumption is often not satisfied in real world knowledge graphs. For example, even for DBpedia, a recent version has over 25% of incorrect triples.

3 Problem Statement

We define the problem of schema aware triple classification (SATC) in this section. Firstly, we introduce the notions of schema-consistent triples.

Definition 1. *Given a knowledge graph $\mathcal{G} = \mathcal{T} \cup \mathcal{A}$, a triple (h,r,t), where h and t are entities and r is a relation in $\mathcal{G}$, (h,r,t) is a schema-consistent triple w.r.t. $\mathcal{G}$ if the extended knowledge graph $\mathcal{G} \cup$ (h,r,t) is consistent.*

From the perspective of data engineering, we further introduce the notions of correct, incorrect and unknown triples.

Definition 2. *Given a knowledge graph $\mathcal{G} = \mathcal{T} \cup \mathcal{A}$, a triple (h,r,t), where h and t are entities and r is a relation 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 correct triple w.r.t. $\mathcal{G}$ if:*

1. *the extended knowledge graph $\mathcal{G} \cup$ (h,r,t) is consistent, and*
2. *$C_h \equiv D_r$ and $C_t \equiv R_r$.*

Definition 3. *Given a knowledge graph $\mathcal{G} = \mathcal{T} \cup \mathcal{A}$, a triple (h,r,t), where h and t are entities and r is a relation 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 an incorrect triple w.r.t. $\mathcal{G}$ if:*

1. *$\mathcal{G} \models C_h \sqcap D_r \sqsubseteq \bot$, or*
2. *$\mathcal{G} \models C_t \sqcap R_r \sqsubseteq \bot$.*

It is straight forward to see that an incorrect triple is not schema-consistent.

Lemma 1. *Given a knowledge graph $\mathcal{G} = \mathcal{T} \cup \mathcal{A}$, a triple (h,r,t), where h and t are entities and r is a relation in $\mathcal{G}$. If (h,r,t) is incorrect w.r.t. $\mathcal{G}$, then (h,r,t) is not schema-consistent w.r.t. $\mathcal{G}$.*

Definition 4. *Given a knowledge graph $\mathcal{G} = \mathcal{T} \cup \mathcal{A}$, a triple (h,r,t), where h and t are entities and r is a relation in $\mathcal{G}$, (h,r,t) is an unknown triple w.r.t. $\mathcal{G}$ if it is neither correct nor incorrect w.r.t. $\mathcal{G}$.*

Definition 5. *Let $\mathcal{G}$ be a knowledge graph, and $\mathcal{E}$ the new triples that are produced by a KGE method, the task of schema aware triple classification (SATC) is to identify the subset ζ of correct triples within $\mathcal{E}$; the percentage of correctness (PC) is defined as follows:*

$$PC = \frac{|\zeta|}{|\mathcal{E}|} * 100\% \tag{1}$$

Given an input knowledge graph $\mathcal{G}$, our research aim is to identify the most suitable KGE methods, so as to maximise the PC for SATC.

4 Our Approach

4.1 Approximate Consistency Checking for Schema Aware Triple Classification

In this section, we will present an approach for schema aware triple classification (SATC), with the help of an approximate reasoning to address the concern of high computation complexity in ontology reasoning.

The idea behind an approximate reasoning [6, 7] is to identify minimal inconsistent sub-graphs (justifications) with the help of *inconsistency justification patterns* (or IJ patterns). Obviously, it could be expensive to compute all possible justifications. Since some recent study [18] suggests type assertions are most often more correct in knowledge graphs than relation assertions, in this paper, we propose to focus on some simple inconsistency justification patterns related to relation axioms.

There are three requirements for these inconsistency justification patterns:

(R1) TBox reasoning with schema sub-graphs should be done offline.
(R2) No online reasoning is needed for dealing with data sub-graphs.
(R3) IJ patterns should help not just detect inconsistencies but also repair them.

(R1) is feasible, since knowledge graph completion algorithms only produce ABox assertions, so that the TBox parts of the IJ patterns can be computed in advance. In Table 1, IJ patterns 1-4 are about interactions among domain axioms, range axioms, relation inclusion axioms, and class disjoint axioms. IJ patterns 5-7 are about interactions among asymmetric relation axioms, symmetric relation axioms, and relation inclusion axioms. IJ pattern 8 is about irreflexive axioms. Thus, we need to run the following TBox reasoning services offline: (i)

Table 1. 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, rdf:type, A)
2	Range(r)=R, R $\sqcap$ A $\sqsubseteq \perp$	(e1, r, e2), (e2, rdf:type, A)
3	Domain(r1)=D1, Domain(r2)=D2, r1 $\sqsubseteq$ r2, D1 $\sqcap$ D2 $\sqsubseteq \perp$	(e1, r1, e2)
4	Range(r1)=R1, Range(r2)=R2, r1 $\sqsubseteq$ r2, R1 $\sqcap$ R2 $\sqsubseteq \perp$	(e1, r1, e2)
5	Asymmetric(r)	(e1, r, e2), (e2, r, e1)
6	Symmetric(r1), Asymmetric(r2), r1 $\sqsubseteq$ r2	(e1, r1, e2)
7	Symmetric(r2), Asymmetric(r3), r1 $\sqsubseteq$ r2, r1 $\sqsubseteq$ r3	(e1, r1, e2)
8	Irreflexive(r)	(e1, r, e1)

computing relation subsumptions and (ii) compute disjoint named types. Since the ABox subsets of the IJ patterns contain at most two data triples, we need to scan through the data sub-graph of a knowledge graph at most twice and no reasoning is needed during runtime. Hence, (R2) is addressed.

Once an IJ pattern is detected within a target knowledge graph, one can repair it by removing the relation assertions in the pattern, i.e., the assertions in the third column in Table 1. Therefore, IJ patterns not only help us to detect logical inconsistencies but also help us to repair logical inconsistencies (R3).

4.2 Sequential Combination of Knowledge Graph Embedding Algorithms for Schema Aware Triple Classification

In this section, we will use the SATC approach to support the sequential combinations of KGE approaches, by applying a KGE over the union of the original graph and the correct triples from a previous iteration of KG completion. We incorporate the sequential combinations with an approximate reasoning for increasing the correctness of the new facts that a knowledge graph embedding approach produces. As shown in Figure 1, the components of our approach are two knowledge graph embedding algorithms (could be the same or different) and an approximate reasoning [3].

Given an input knowledge graph, our approach works as follows. Each iteration has two steps. The first step is to run Knowledge Graph Embedding algorithm (or simply KGE) to produce new triples. In the second step, we only take the correct triples detected by the approximate reasoning and merge them with the initial KG. Unless some stopping condition (e.g., no new correct triples are produced in the current iteration) is satisfied, we will start a new iteration. Once some stopping condition is satisfied, we count the percentage of correctness (PC) of the sequential combination by using the number of correct triples and the number of new triples that we got from the executed iterations 3.

More precisely, at the first step, given an input knowledge graph G, and a link prediction system LP, we produce the output knowledge graph G' for that iteration as follows:

[3] The implementation of our approach: http://github.com/bagindokemas/meOnJIST2018

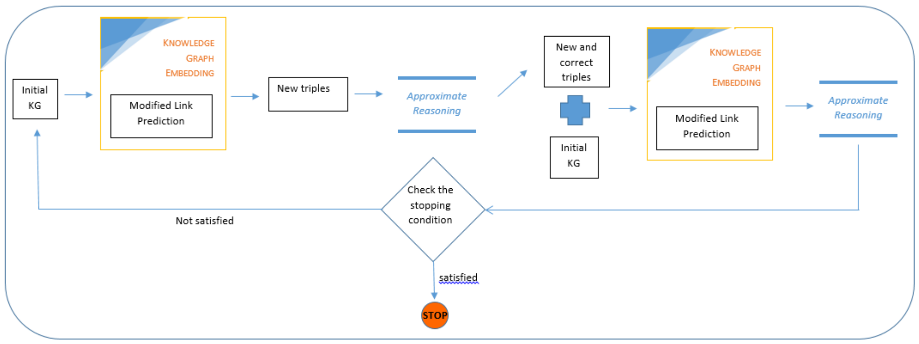

Fig. 1. Sequential combination of knowledge graph embedding (SCE) algorithms

1. Compute the set of recommended triples RT from LP that are above the threshold t;
2. Compute the set of new triples NT $\doteq$ RT $\setminus$ G;
3. G' $\doteq$ G $\cup$ NT.

We will then pass G' to the approximate reasoning module. At the second step, given the set of incorrect triples IT detected by IJ patterns, the remaining new triples are NT' $\doteq$ NT $\setminus$ IT. Accordingly, we can define a stopping condition as $|$ NT'$| \leq$ s, where the stop threshold s is a non-negative integer. If s is set to be 0, it means that the iterations will stop when no new triples are produced in the current iteration.

5 Evaluation

We empirically study and evaluate our approach on three tasks: (1) Comparing the existing knowledge graph embedding approaches in terms of producing correct triples according to the schema, (2) Investigating the sequential combinations of knowledge graph embedding approaches and (3) Comparing our approximate reasoning with related justification services which was provided by some existing off-the-shelf reasoner. The first two tasks are aiming to validate the sequential combinations of knowledge graph embedding approaches meanwhile the last task is aiming to validate the approximate reasoning.

For the first and the second task, since our approach needs a KG that has a TBox and a ABox, we use these three knowledge graphs: (i)NELL-995 which is a dataset from Carnegie Mellon university containing 142,065 triples, (ii) DBpedia-Politic which is a subset of DBpedia version 2016-04 that only include triples that are related to political issue containing 352,754 triples and (iii) IBM-KG which is a knowledge graph automatically generated based on IBM Storwize Knowledge Center articles[4] dataset containing 28,982 triples.

[4] https://www.ibm.com/support/knowledgecenter/en/ST3FR7

For the third task, we are only able to test our approximate reasoning on two knowledge graphs as follows: (i) DBpedia version 2016-04 which is a dataset from DBpedia containing 17,678,218 triples, and (ii) IBM-KG.

Our approximate reasoning module consists of two components, such as the TBox scanner and the ABox scanner. We implement the TBox scanner using Java and OWLAPI [27] library. This scanner process the schema of a KG to find all the IJ patterns (as mentioned in table 1). The ABox scanner parses all the triples on the ABox based on the IJ patterns to find the correct, incorrect, and the unknown triples.

5.1 Comparing Existing Knowledge Graph Embedding Approaches

Experimental Setup. We use this task to check which knowledge graph embedding approaches (KGE) is the best in terms of the percentage of correctness.

KGE Approaches. We used NELL-995, DBpedia-Politic, and IBM-KG as the knowledge graphs/dataset. For the KGE Approaches, we choose KB2E-TransE, STranSE, OpenKE-DistMult, and OpenKE-Complex. We chose these methods because, in their respective categories, they are the most efficient methods. We consider the top ten results from these systems for each link prediction task.

Results. Firstly, we expand a KG using a KGE approach. After that, we collect all the triples that are produced by a KGE approach and then running our schema-based inconsistency checking against the new triples. The last step of the experiment is counting the percentage of the correctness.

We can see from Table 2 that the order of the best KGE approaches in terms of the percentage of correctness is as follows: STransE, DistMult, Complex and TransE.

The reason for the very good performance of STransE is in STransE, the head and tail entities are associated with their own project matrices, rather than using the same matrix for both, as in other embedding approaches.

Table 2. The comparison result of the percentage of correctness

Knowledge Graph	KB2E_TransE	STranSE	OpenKE-DistMult	OpenKe-Complex
IBM-KG	1.52%	**61.50%**	21.96%	21.59%
NELL-995	0.37%	**20.36%**	0.72%	0.67%
DBpedia-Politic	18.06%	**43.45%**	18.90%	18.90%

5.2 Investigating Sequential Combinations of KGE

Experimental Setup. Our objectives in doing this task are as follows:

– (P1). we would like to know whether running the same KGE twice sequentially will outperform than running it once (compares KGE-KGE with KGE).

- (P2). we would like to know which option is better (in terms of producing a higher percentage of correctness): running the schema-based inconsistency checking right after each run of a KGE approach (KGE-IC-KGE-IC) or running it only once in the end of a sequential combination (KGE-KGE-IC).
- (P3). we would like to know whether we can get a higher percentage of correctness (compared with all the scenarios from P1 until P2) if we combine different KGE approaches with the schema-based inconsistency checking (KGE1-IC-KGE2 or KGE2-IC-KGE1).

For the first purpose (P1), we make four scenarios as follows:

- comparing STransE-IC-STransE-IC (SE-IC-SE-IC) with STransE-IC (SE-IC)
- comparing KB2E-IC-KB2E-IC (KE-IC-KE-IC) with KB2E-IC (KE-IC)
- comparing DistMult-IC-DistMult-IC (DM-IC-DM-IC) with DistMult-IC (DM-IC)
- comparing Complex-IC-Complex-IC (CE-IC-CE-IC) with Complex-IC (CE-IC)

For the second purpose(P2), we also make four scenarios as follows:

- comparing STransE-IC-STransE-IC (SE-IC-SE-IC) with STransE-STransE-IC (SE-SE-IC)
- comparing KB2E-IC-KB2E-IC (KE-IC-KE-IC) with KB2E-KB2E-IC (KE-KE-IC)
- comparing DistMult-IC-DistMult-IC (DM-IC-DM-IC) with DistMult-DistMult-IC (DM-DM-IC)
- comparing Complex-IC-Complex-IC (CE-IC-CE-IC) with Complex-Complex-IC (CE-CE-IC)

For the third purpose(P3), we make twelve scenarios as follows:

- KB2E-IC-STransE-IC (KE-IC-SE-IC)
- STransE-IC-KB2E-IC (SE-IC-KE-IC)
- KB2E-IC-DistMult-IC (KE-IC-DM-IC)
- DistMult-IC-KB2E-IC (DM-IC-KE-IC)
- KB2E-IC-Complex-IC (KE-IC-CE-IC)
- Complex-IC-KB2E-IC (CE-IC-KE-IC)
- STransE-IC-DistMult-IC (SE-IC-DM-IC)
- DistMult-IC-STransE-IC (DM-IC-SE-IC)
- STransE-IC-Complex-IC (SE-IC-CE-IC)
- Complex-IC-STransE-IC (CE-IC-SE-IC)
- DistMult-IC-Complex-IC (DM-IC-CE-IC)
- Complex-IC-DistMult-IC (CE-IC-DM-IC)

KGE Approaches. We used NELL-995, DBpedia-Politic, and IBM-KG as the knowledge graphs/dataset. For the KGE Approaches, we chose KB2E-TransE, STranSE, OpenKE-DistMult, and OpenKE-Complex.

Table 3. The result for P1

ID	The PC for running KGE twice				The PC for running KGE once			
	Flow	IBM	NELL	DBP	Flow	IBM	NELL	DBP
1	SE-IC-SE-IC	**69%**	**29%**	**55.51%**	SE-IC	**61.50%**	**20.36%**	**43.45%**
2	KE-IC-KE-IC	0.86%	0.17%	17.16%	KE-IC	1.52%	0.37%	18.06%
3	DM-IC-DM-IC	22.42%	0.53%	19.25%	DM-IC	21.96%	0.72%	18.90%
4	CE-IC-CE-IC	20.95%	0.55%	19.24%	CE-IC	21.59%	0.67%	18.90%

Results. Table 3 records all the results that are related with the first purpose/P1. we could see that from all KGE approaches that we try, only STransE that have higher PC (for all KGs) if we run it twice sequentially rather than running it once. If we analyze this result deeper, we see that in the second run, all KGE approaches (except STransE), produce less correct triples compare with the first run. We continue the experiment P1 on STransE (using NELL KG) by running it for three runs sequentially (STransE-IC-StransE-IC-STransE-IC). And we got interesting fact that the percentage of correctness that we got is even higher than running STransE twice (38% against 29%). However, the number of new triples that STransE produce is decreasing on each run. In the first run, the number of new triples is 373,995. In the second run, the number of new triples is 178,654 and in the third run, the number of new triples becomes 107,587.

Table 4. The result for P2

ID	The PC for running IC after each run				The PC for running IC only once			
	Flow	IBM	NELL	DBP	Flow	IBM	NELL	DBP
1	SE-IC-SE-IC	**69%**	**29%**	**55.51%**	SE-SE-IC	**66.44%**	**16.25%**	**43.90%**
2	KE-IC-KE-IC	0.86%	0.17%	17.16%	KE-KE-IC	1.18%	0.28%	17.31%
3	DM-IC-DM-IC	22.42%	0.53%	19.25%	DM-DM-IC	12.28%	0.22%	16.11%
4	CE-IC-CE-IC	20.95%	0.55%	19.24%	CE-CE-IC	12.38%	0.21%	16.07%

For the second purpose/P2, we record all the results in table 4.

From this table we learn that for STransE, DistMult and Complex, running IC after each run produce higher PC than running IC only once because in each run, the number of correct triples that these approaches can feed into the next run is already high (for example in STransE, more than 20% of new triples in each run is correct). Consequently, this will increase the number of correct triples that these approaches produce in the next run.

Meanwhile for KB2E-TransE, running IC only once produce higher PC than running IC after each run, because, in each run, the number of correct triples that KB2E-TransE can feed into the next run is low (less than 1% of new triples in each run is correct). Consequently, this will decrease the number of correct triples that KB2E-TransE produces in the next run.

Table 5. The result for P3

Scenarios	The Percentage of Correctness		
	IBM	NELL	DBP
KE-IC-SE-IC	40.72%	11.85%	30.95%
SE-IC-KE-IC	49.65%	17.34%	45.75%
KE-IC-DM-IC	11.44%	0.51%	17.93%
DM-IC-KE-IC	9.45%	0.98%	19.01%
KE-IC-CE-IC	11.40%	0.41%	17.91%
CE-IC-KE-IC	7.75%	0.87%	19.01%
SE-IC-DM-IC	55.60%	14.14%	46.29%
DM-IC-SE-IC	63.12%	16.92%	48.45%
SE-IC-CE-IC	55.65%	14.14%	46.21%
CE-IC-SE-IC	62.06%	17.15%	47.68%
DM-IC-CE-IC	22.36%	0.56%	19.20%
CE-IC-DM-IC	21.15%	0.51%	19.26%
SE-IC-SE-IC	**69%**	**29%**	**55.51%**
KE-IC	1.52%	0.37%	18.06%
DM-IC	21.96%	0.72%	18.90%
CE-IC	21.59%	0.67%	18.90%
KE-KE-IC	1.18%	0.28%	17.31%
DM-IC-DM-IC	22.42%	0.53%	19.25%
CE-IC-CE-IC	20.95%	0.55%	19.24%

Table 5 records the result of the experiments for the third purpose/P3. We divide the table into three groups as follows: (1). All twelve scenarios from the third purpose/P3, (2). All the best scenarios from the first purpose / P1, (3). All the best scenarios from the second purpose/P2. Following are the lessons that we can learn from the results in table 5:

- Different combination of KGEs (for an example: KGE1-IC-KG2-IC) with IC will produce higher PC when compared to the PC that is produced by the weakest KGE in a scenario (for an example: the weakest KGE between KGE1 and KGE2).
- The best scenario/flow from all three groups aforementioned above is SE-IC-SE-IC

5.3 Comparing our Approximate Reasoning with Sound-and-Complete Reasoner

We compare our Approximate Reasoning approach with existing reasoners in terms of

- how many incorrect triples (in percentage) that are identified by an existing reasoner, can also be detected by our approach. For this comparison item, we collect all the justifications that are generated by a reasoner for a given

knowledge graph. Then, we check whether our approach can detect all the incorrect triples that are stated in each of these explanations
- how long Approximate Reasoning takes compared to reasoning time of an existing reasoner. For this second comparison item, we need to explain first what are the processing stages for each system. In doing reasoning, every existing reasoner has two steps which are the consistency checking and generating justification. Generating justification is often most costly since it needs to calculate the minimal subsets of a knowledge graph. Hence, the total of processing time for a reasoner is the consistency checking plus generating justification. Meanwhile, our approach also has two steps, which are scanning the inconsistency justification patterns (IJPs) from the TBox and then based on the IJPs, scanning the triples that conform to these IJPs in the ABox. Hence, the total processing time for our approach is the TBox scanning plus the ABox scanning. Please see Figure 2 for the comparison of the processing stages between a reasoner and our approach.

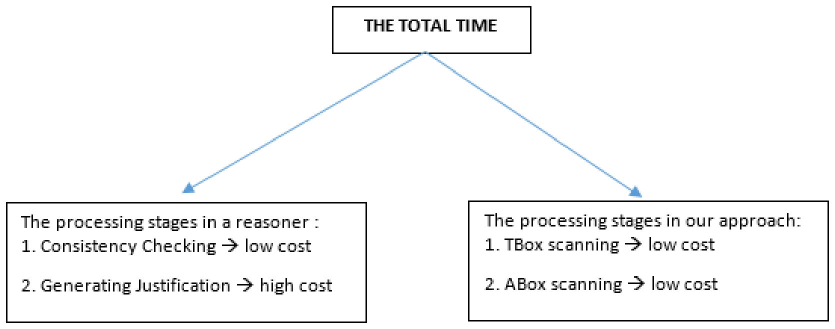
THE TOTAL TIME
The processing stages in a reasoner :
1. Consistency Checking → low cost
2. Generating Justification → high cost
The processing stages in our approach:
1. TBox scanning → low cost
2. ABox scanning → low cost

Fig. 2. the comparison of the processing stages between a reasoner and our approach.

We show the effectiveness of our approach against the consistency checking and related justification services that are provided by FACT++ 1.6.5 (run with Protege). **Dataset.** We used two datasets: (i) DBpedia version 2016-04, and (ii) IBM-KG.

Results. The comparison of the performance of the FACT++ reasoner and our Approximate Reasoning approach can be seen in table 6.

#Exp stands for the number of explanations that the justifications service from FACT++ generated. Each explanation consists of one or several incorrect triples. If FACT++ cannot generate the explanations (either because a KG is consistent or because we cannot run the consistency checking and justification services on a KG), we will give NA (Not Available) as the score. For DBpedia,

Table 6. Comparison between pattern-based reasoning and reasoner

Dataset	Reasoner			Our Approach		
	#Exp	CCT	GJT	CE	TST	AST
IBM-KG	46	1.577	4,620	100%	12.98	296.68
DBpedia	NA	NA	NA	NA	9,605	6,834.76

we got an "insufficient memory" error when we tried to merge the TBox and the ABox of the DBpedia. This happened due to the sheer size of the ABox of DBpedia. Since merging the TBox and the ABox is the prerequisite of running consistency checking and justification services, hence we can not get the #Exp score for the whole DBpedia KG. CCT stands for the time that the reasoner needs to do consistency checking (in second). There are only two types of a score for CCT: the number of times and NA (Not Available). GJT stands for the time that the reasoner needs to generate the justifications (in seconds).

CE stands for coverage of the Explanations. It refers to how many of the explanations that are generated by the justification service of the reasoner, in percentage, are detected by our approach. We get the CE score by dividing the number of explanations that are detected by our approach with the number of the explanations that are generated by the reasoner. Since we do not have a number of explanation scores for the DBpedia data set, we can not count the CE score. TST stands for the time that is needed to scan the TBox part of a knowledge graph (in second). AST stands for the time that is needed to scan the ABox part of a knowledge graph (in seconds).

We observe from table 6, that for IBM-KG, our approach is much more efficient than the consistency checking and justification service provided by the FACT++ reasoner. Meanwhile for DBpedia, we could scan the problematic triples less than five hours while FACT++ reasoner failed.

6 Related Work

6.1 Knowledge Graph Embedding

The idea of knowledge graph embedding is to represent an entity as a k-dimensional vector $\mathbf{h}$ (or $\mathbf{t}$) and defines a scoring function $f_r(\mathbf{h}, \mathbf{t})$ 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 KGE algorithms often differ in their scoring function, transformation and loss function [8], [9], [5], [16], [17], [15], [12], [28].

The above work has improved the performance of knowledge graph embedding in terms of the link prediction task but does not take into account the correctness of the facts that they produce. Our work differs from previous work on knowledge graph embedding as we focus on whether the new triples that are produced by a knowledge graph embedding approach are correct respect to a schema of a knowledge graph.

The most recent work that is very related to our work is [23]. They enhance the existing embedding-based methods by encoding the logical consistency into the learnt distributed representation for the knowledge graph. Their approach enforces the new triples to be consistent in regards to the Horn theory. However, their approach does not compute the truth degrees/the correctness level of new triples, since according to them this technique is time-consuming.

6.2 Correctness of a knowledge graph

There is no specific metric to evaluate the correctness of a knowledge graph. The closest measurements to do this are the accuracy (which is used in [24, 21]) and the percentage of consistent ABoxes ([25]). Following are our explanation about these measurements:

– The first measurement tries to validate the RDF triples in a knowledge graph by collecting consensus of matched triples from other knowledge graphs. Our work differs with this work as the latter using external information/other knowledge graphs to validate the triples.
– The second measurement determines the consistency of ABox regarded to a TBox. We could see that this work has limitation since it cannot guarantee soundness for inconsistency checking and justifications.

6.3 Approximate concistency checking

There are related work on approximate consistency checking. Meilicke et al. [24] addresses this issue but only with the DL-Lite ontology language, which is not enough for our purpose. There are also machine learning based approaches [25, 26]. However, they cannot guarantee soundness for inconsistency checking and justifications.

7 Conclusion and Future Works

In this paper, we have introduced the black-box Schema Aware Triple Classification (SATC). Extensive experiments on IBM, NELL, and DBpedia-Politic show that our approach (SE-IC-SE-IC on Table 5) can increase the level of correctness when compared to the application of single embedding-based approach (SE-IC on table 2). We further show that from all scenarios/combinations that we can make from different knowledge graph embedding approaches, we can achieve the highest percentage of correctness by running the KGE approach that produces most correct triples followed by the approximate reasoning in a sequential manner.

The potential path for future work includes increasing the scalability of our approach and applying the sequential combinations to other approaches in the knowledge graph completion (such as the rule learning).

Acknowledgement

This work was supported by IBM Faculty Award and the EU Marie Currie K-Drive project (286348). Kemas Wiharja was also supported by the Lembaga Pengelola Dana Pendidikan (LPDP), the Ministry of Finance of Indonesia.

References

1. Jeff Z. Pan, Guido Vetere, Jose Manuel Gomez-Perez, Honghan Wu. (Eds.). Exploiting Linked Data and Knowledge Graphs for Large Organisations. ISBN 978-3-319-45652-2, Springer. 2017.
2. Jeff Z. Pan, Diego Calvanese, Thomas Eiter, Ian Horrocks, Michael Kifer, Fangzhen Lin, Yuting Zhao (Ed.). Reasoning Web: Logical Foundation of Knowledge Graph Construction and Query Answering, Spriner. 2017
3. Heiko Paulheim. Knowledge Graph Refinement: A Survey of Approaches and Evaluation Methods. In Semantic Web Journal. (2016).
4. Wenhan Xiong, Thien Hoang, and William Yang Wang: DeepPath: A Reinforcement Learning Method for Knowledge Graph Reasoning. In EMNLP (2017).
5. Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, Oksana Yakhnenko. Translating Embeddings for Modeling Multi-relational Data. Advances in Neural Information Processing Systems 26. Curran Associates, Inc., 2013, pp. 27872795.
6. J. Z. Pan and E. Thomas. Approximating OWL-DL Ontologies. In AAAI-07). pp.1434-1439.
7. J. Z. Pan, Y. Ren and Y. Zhao. Tractable approximate deduction for OWL. In Artificial Intelligence, volume 235: p95-155, 2016.
8. Y. Lin, Z. Liu, M. Sun, Y. Liu, and X. Zhu. Learning entity and relation embeddings for knowledge graph completion. in Proc. 29th AAAI Conf. Artif. Intell., 2015, pp. 21812187
9. Z. Wang, J. Zhang, J. Feng, and Z. Chen. Knowledge graph embedding by translating on hyperplanes. in Proc. 28th AAAI Conf. Artif. Intell., 2014, pp. 11121119.
10. D. Q. Nguyen, K. Sirts, L. Qu, and M. Johnson. STransE: A novel embedding model of entities and relationships in knowledge bases. in Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics: Human Language Technol., 2016, pp. 460466.
11. Xiao, H., Huang, M. and Zhu, X., 2016. TransG: A generative model for knowledge graph embedding. In Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers) (Vol. 1, pp. 2316-2325).
12. R. Socher, D. Chen, C. D. Manning, and A. Y. Ng. Reasoning with neural tensor networks for knowledge base completion. in Proc. Adv. Neural Inf. Process. Syst., 2013, pp. 926934.
13. Shu Guo, Boyang Ding, Quan Wang, Lihong Wang, and Bin Wang. Knowledge base completion via rule-enhanced relational learning. In China Conference on Knowledge Graph and Semantic Computing (pp. 219-227). Springer, Singapore.
14. Du, J., Qi, K., Wan, H., Peng, B., Lu, S. and Shen, Y. Enhancing Knowledge Graph Embedding from a Logical Perspective. In Joint International Semantic Technology Conference (pp. 232-247). Springer, Cham, 2017.
15. M. Nickel, V. Tresp, and H.-P. Kriegel. A three-way model for collective learning on multi-relational data. in Proc. 28th Int.Conf. Mach. Learn., 2011, pp. 809816.

16. R. Jenatton, N. L. Roux, A. Bordes, and G. R. Obozinski. A latent factor model for highly multi-relational data. in Proc. Adv. Neural Inf. Process. Syst., 2012, pp. 31673175.
17. Xavier Glorot, Antoine Bordes, Jason Weston, Yoshua Bengio. A semantic matching energy function for learning with multi-relational data. Mach. Learn., vol. 94, no. 2, pp. 233259, 2014.
18. Heiko Paulheim, Christian Bizer. Improving the Quality of Linked Data Using Statistical Distributions. International Journal on Semantic Web and Information Systems (IJSWIS), 10(2):6386, 2014.
19. Zhen Wang, Jianwen Zhang, Jianlin Feng and Zheng Chen. Knowledge Graph Embedding by Translating on Hyperplanes. In AAAI Conference on Artificial Intelligence, 2014.
20. Tom Mitchell, W. Cohen, E. Hruschka, P. Talukdar, J. Betteridge, A. Carlson, B.Dalvi, M. Gardner, B. Kisiel, J. Krishnamurthy, N. Lao, K. Mazaitis, T. Mohamed, N. Nakashole, E. Platanios, A. Ritter, M. Samadi, B. Settles, R. Wang, D.Wijaya, A. Gupta, X.Chen, A. Saparov, M. Greaves, J. Welling. Never Ending Learning. In AAAI Conference on Artificial Intelligence, 2015, (pp. 2302-2310).
21. Liu, S., dAquin, M. and Motta, E., 2017, June. Measuring accuracy of triples in knowledge graphs. In International Conference on Language, Data and Knowledge (pp. 343-357). Springer, Cham.
22. Shi, B. and Weninger, T., 2017. Open-World Knowledge Graph Completion. arXiv preprint arXiv:1711.03438.
23. Jianfeng Du, Kunxun Qi, and Yuming Shen. 2018. Knowledge Graph Embedding with Logical Consistency. Unpublished paper, Guangdong University of Foreign Studies.
24. Meilicke, C., Ruffinelli, D., Nolle, A., Paulheim, H. and Stuckenschmidt, H., 2017, July. Fast ABox consistency checking using incomplete reasoning and caching. In International Joint Conference on Rules and Reasoning (pp. 168-183). Springer, Cham.
25. Paulheim, H. and Stuckenschmidt, H., 2016, May. Fast approximate a-box consistency checking using machine learning. In International Semantic Web Conference (pp. 135-150). Springer, Cham.
26. Ruffinelli, D., 2017. Towards scalable ontological reasoning using machine learning. In CEUR workshop proceedings (Vol. 1875, pp. Paper-5). RWTH.
27. Horridge, M. and Bechhofer, S., 2009, October. The OWL API: a Java API for working with OWL 2 ontologies. In Proceedings of the 6th International Conference on OWL: Experiences and Directions-Volume 529 (pp. 49-58). CEUR-WS. org.
28. Ringsquandl, M., Kharlamov, E., Stepanova, D., Hildebrandt, M., Lamparter, S., Lepratti, R., Horrocks, I. and Krger, P., 2018, June. Event-Enhanced Learning for KG Completion. In European Semantic Web Conference (pp. 541-559). Springer, Cham.
29. Hai Dang Tran, Daria Stepanova, Mohamed H Gad-Elrab, Francesca A. Lisi, and Gerard Weikum. Towards nonmonotonic relational learning from knowledge graphs. In Proceedings of the 26th International Conference on Inductive Logic Programming (ILP 2016). LNCS (Vol. 10326).
30. Ho, V.T., Stepanova, D., Gad-Elrab, M.H. and Kharlamov, E., 2018. Rule learning from knowledge graphs guided by embedding models. In The 17th International Semantic Web Conference.