Scalable Instance Reconstruction in Knowledge Bases via Relatedness Affiliated Embedding

Richong Zhang
BDBC and SKLSDE,
Beihang University
zhangrc@act.buaa.edu.cn

Jiajie Mei
BDBC and SKLSDE,
Beihang University
meijj@act.buaa.edu.cn

Junpeng Li
BDBC and SKLSDE,
Beihang University
lijp12@act.buaa.edu.cn

Yongyi Mao
School of Electrical Engineering and Computer Science,
University of Ottawa
ymao@uottawa.ca

ABSTRACT

The knowledge base (KB) completion problem is usually formulated as a link prediction problem. Such formulation is incapable of capturing certain application scenarios when the KB contains multi-fold relations. In this paper, we present a new formulation of KB completion, called instance reconstruction. Unlike its link-prediction counterpart, which has linear complexity in the size of the KB, this problem has its complexity behave as a high-degree polynomial. This presents a significant challenge in developing scalable instance reconstruction algorithms. In this paper, we present a novel knowledge embedding model (RAE) and build on it an instance reconstruction algorithm (SIR). The SIR algorithm utilizes schema-based filtering as well as “relatedness” filtering for complexity reduction. Here relatedness refers to the likelihood that two entities co-participate in a common instance, and the relatedness metric is learned from the RAE model. We show experimentally that SIR significantly reduces computation complexity without sacrificing reconstruction performance. The complexity reduction corresponds to reducing the KB size by 100 to 1000 folds.

CCS CONCEPTS

• **Computing methodologies** → **Reasoning about belief and knowledge**; • **Mathematics of computing** → *Graph algorithms*;
• **Information systems** → *Data extraction and integration*;

KEYWORDS

Link Prediction, Knowledge Base, Multi-fold Relation

ACM Reference Format:

Richong Zhang, Junpeng Li, Jiajie Mei, and Yongyi Mao. 2018. Scalable Instance Reconstruction in Knowledge Bases via Relatedness Affiliated Embedding. In *WWW 2018: The 2018 Web Conference, April 23–27, 2018, Lyon, France*. ACM, New York, NY, USA, 10 pages. <https://doi.org/10.1145/3178876.3186017>

This paper is published under the Creative Commons Attribution 4.0 International (CC BY 4.0) license. Authors reserve their rights to disseminate the work on their personal and corporate Web sites with the appropriate attribution.

WWW 2018, April 23–27, 2018, Lyon, France

© 2018 IW3C2 (International World Wide Web Conference Committee), published under Creative Commons CC BY 4.0 License.

ACM ISBN 978-1-4503-5639-8/18/04.

<https://doi.org/10.1145/3178876.3186017>

(a) Link Prediction (b) Instance Reconstruction
Figure 1: Link prediction vs. instance reconstruction

1 INTRODUCTION

Knowledge bases (KBs), such as YAGO [12], DBpedia[1] and Freebase [2], have been created for easy and rapid information retrieval and knowledge discovery. The enormous amount of linked knowledge therein enables the development of various applications [9, 17, 18]. Notably this linked data is usually extracted from human-edited knowledge systems, such as Wikipedia. As a result, the consistency and the quality of the data vary significantly and the completeness of the extracted knowledge graph is hardly guaranteed. This makes KB completion an important direction of research.

In the literature, the problem of KB completion is usually formulated as a *link prediction* problem [3–5, 7, 8, 11, 14, 16]. Although such a problem is often formulated for binary relations, it can be generalized to multi-fold or n -ary relations [15]. In a KB containing multi-fold relations, the link prediction problem can be formulated as solving for the entity missing from a given role of a given instance. Figure 1 (a) describes an example of the link prediction problem, where one wishes to answer the question “where did Kobe Bryant got married with Venessa Bryant”. Despite its wide adoption in the research literature, this formulation is limited in its scope of capturing practical scenarios, particularly for KB containing multi-fold relations. For example, as shown in Figure 1 (b), one may wish to answer the question “where did Kobe Bryant got married and with whom”. That is, one may wish to reconstruct an instance in which entities are missing from all but one roles. We call such a problem the *instance reconstruction* problem.

It is obvious that for binary relational data, the instance reconstruction problem reduces to the link prediction problem. However, for multi-fold relations, the two problems differ significantly. It is remarkable that the significant difference between the two problems has little to do with their mathematical nature. Rather this

difference is fundamentally reflected by their computational complexities. For example, for a m -fold relation in a KB containing N entities, the complexity of a link prediction task is linear in N , whereas the complexity of instance reconstruction is in the order of N^{m-1} . For KBs containing multi-fold relations, this complexity can be prohibitive when the number of entities is large and when some of the relations are non-binary. In fact, for a given set of instance reconstruction tasks, even when the fraction of non-binary tasks does not dominate, the complexity of the non-binary tasks can still dominate the overall complexity. For example, in the set of instance reconstruction tasks, even when there are a small fraction of tasks that are 6-ary, their complexity order of N^5 may dominate all tasks with lower arities. As a consequence, even for a modestly-sized set of instance-reconstruction tasks, the overall complexity can be prohibitive. As such the primary challenge in instance reconstruction is to develop an efficient reconstruction algorithm that scales well with the size of the KB.

Since the instance reconstruction problem has a similar mathematical nature to the link prediction problem, our approach to this problem is to build into a link prediction model a complexity reduction mechanism. In the recent literature, various models have been developed for the link prediction purpose [4–6, 8, 13, 14, 16], among which the KB embedding models have demonstrated to be the state of the art [4, 8, 14, 16]. We then select a KB embedding model and augment its loss function with an additional loss measuring the likelihood of two entities being “related”. Here two entities are said to be *related* if there is an instance involving both the entities. After this overall model is learned, it is then possible to first use the relatedness loss to filter all possible candidate entity pairs and then use the embedding loss to evaluate whether a combination of the remaining pairs may be spliced to form a solution instance. We note that the complexity of the filtering step is $O(N^2)$ and the complexity of splicing and evaluation is still $O(N^{m-1})$. However, since filtering can significantly reduce the qualified entities to be paired, the overall complexity is in fact $(\epsilon N)^{m-1}$ for a small constant ϵ . This makes the reconstruction algorithm much more scalable. Our experiments on a multi-fold relational dataset, JF17K [15], show that the constant ϵ ranges from 0.001 to 0.0077. This corresponds to an effective reduction of the KB size by 100 to 1000 folds.

2 PROBLEM STATEMENT

Let $\mathcal{N}$ denote the set of all entities in a given knowledge base (KB). We will use $\mathcal{R}$ to index the set of all relations therein, namely, for each index $r \in \mathcal{R}$, there is a relation R_r . Each relation R_r is associated with a set $\mathcal{M}(r)$ of ordered role tuples $\{\rho_1, \rho_2, \dots, \rho_{|\mathcal{M}(r)|}\}$, and the cardinality $|\mathcal{M}(r)|$ of $\mathcal{M}(r)$ is the arity of R_r . Each instance in the relation R_r is a vector $(x_1, x_2, \dots, x_{|\mathcal{M}(r)|})$ of entities, in which entity x_i corresponds to role $\rho_i \in \mathcal{M}(r)$. The relation R_r is then understood as the set of all such vectors. We note that each relation R_r is allowed to have an arbitrary arity.

In practice, KBs are far from being complete, namely that R_r is not completely given. Instead it is revealed via an incomplete collection of instances in R_r , which we will denote by $\mathcal{T}_r$. The KB is then specified by $G = (\mathcal{N}, \mathcal{R}, \{\mathcal{T}_r : r \in \mathcal{R}\})$. We will use G^0 to denote the underlying complete KB. i.e. $G^0 := (\mathcal{N}, \mathcal{R}, \{\mathcal{R}_r : r \in \mathcal{R}\})$.

Broadly speaking, a KB completion problem deals with inferring unobserved instances in G^0 based on observed instances in G . Such

a problem may be set up in at least two ways, *link prediction* as is standard, or *instance reconstruction* as we introduce in this paper.

The link prediction problem can be formulated as follows. Let $\rho^\bullet \in \mathcal{M}(r)$ for a relation R_r , and suppose that t^* is a subvector of an instance $t = (x_1, x_2, \dots, x_{|\mathcal{M}(r)|}) \in R_r$, where t^* contains all elements of t except for the one corresponding to role $\rho^\bullet$. Then the link prediction problem is to recover t from t^* (Figure 1 (a)), and t^* will be referred to as the *key* for the problem. The role $\rho^\bullet$ will be referred to as the *query role*.

Due to the limited application scope of the link prediction problem, we now formulate another KB completion problem, the *instance reconstruction* problem. Let $\rho^* \in \mathcal{M}(r)$ for a relation R_r , and suppose that x^* is an entity of an instance $t = (x_1, x_2, \dots, x_{|\mathcal{M}(r)|}) \in R_r$ corresponding to ρ^* . Then the instance reconstruction problem is to recover t from x^* (Figure 1 (b)), and, likewise, x^* will be referred to as the *key* for the problem.

Apparently, an instance reconstruction problem for a given primary entity x^* can be resolved by repeatedly solving many link prediction problems. More precisely, this involves 1) going through each relation R_r 2) and for each R_r , testing $\binom{|\mathcal{N}|-1}{|\mathcal{M}(r)|-1}$ entity combinations to see if they may form an instance of R_r . This entails a complexity $\sum_{R_r \in \mathcal{R}} \binom{|\mathcal{N}|-1}{|\mathcal{M}(r)|-1}$, which is prohibitive for large KB containing many relations. Thus the main challenge in instance reconstruction is dealing with its complexity.

3 JOINT MODELLING AND LEARNING OF RELATEDNESS AND EMBEDDING

In this section, we will first introduce a new modelling framework, which we call *Relatedness Affiliated Embedding*, or *RAE*, which augments any KB embedding model with an additional loss that models the relatedness of entities. We will also describe a concrete example of an RAE model. We then describe the training algorithm of the proposed RAE model.

3.1 Relatedness Affiliated Embedding (RAE)

The RAE model is specified by a loss function which consists of two components, a “global embedding loss” and “relatedness loss”.

3.1.1 global embedding loss. The *global embedding loss* function may be the optimization objective function in any parametrized KB embedding model. Thus this subsection serves two purposes. On one hand, we give a concise review of the KB embedding framework, and on the other hand, we will explain our embedding loss function.

Given a KB $G = (\mathcal{N}, \mathcal{R}, \{\mathcal{T}_r : r \in \mathcal{R}\})$, the KB embedding problem is to represent $\mathcal{N}$ and each R_r as quantities in a K -dimensional Euclidean space $U := \mathbb{R}^K$ such that the structure of the KB is preserved. Let function $\phi : \mathcal{N} \rightarrow \mathbb{R}^K$ be the desired embedding map which represents every entity $x \in \mathcal{N}$ as a vector $\phi(x) \in U$. For any instance $t := (x_1, x_2, \dots, x_{|\mathcal{M}(r)|}) \in R_r$, its image $\phi(t)$ under the embedding map is then $(\phi(x_1), \phi(x_2), \dots, \phi(x_{|\mathcal{M}(r)|}))$. To model the structure of a relation R_r , one usually associates R_r with a loss function f_r on $U^{|\mathcal{M}(r)|}$ and desires $f_r(t)$ to be near 0 for $t \in R_r$ and to be large for $t \notin R_r$.

When the set $\mathcal{T} := \bigcup_{r \in \mathcal{R}} \mathcal{T}_r$ is given, one may take a learning approach to decide what these functions $\{f_r : r \in \mathcal{R}\}$ and the embedding map ϕ should be by examining how the examples in

$\mathcal{T}$ look like. To fit the problem into a supervised learning regime, one may also create a set of negative examples $\mathcal{T}^-$. This can be done by taking the instances in each $\mathcal{T}_r$ and randomly replacing an entity therein by another entity. This creates a set $\mathcal{T}_r^-$, in which each element is a “non-instance” or negative example with respect to relation R_r with high probability. Then $\mathcal{T}^-$ may be compiled as the union $\bigcup_{r \in \mathcal{R}} \mathcal{T}_r^-$.

The key of designing an embedding model is designing the functional form of f_r . When this is done, we can set up a global embedding loss function E_I as

$$E_I(\Theta_I, \phi) := \sum_{r \in \mathcal{R}} \left(\sum_{t \in \mathcal{T}_r} f_r(t) + \sum_{t \in \mathcal{T}_r^-} \text{ReLU}(C - f_r(t)) \right) \quad (1)$$

where Θ_I denotes $\{\theta_r : r \in \mathcal{R}\}$, C is a pre-specified threshold and $\text{ReLU}(\cdot)$ is the rectified linear unit.

Existing KB embedding models use various forms of f_r [4, 8, 14, 16]. A majority of these embedding models assume that each relation is binary or two-fold, so that f_r has two variables. These models include TransE [4], TransH [14], TransR [8], etc. For example, in TransH, each f_r is parameterized by a hyperplane with normal vector $\mathbf{n}_r$ together with the displacement vector $\mathbf{b}_r$ on the plane:

$$f_r(\mathbf{x}, \mathbf{y}) := \|\mathbb{P}_{\mathbf{n}_r}(\mathbf{x}) + \mathbf{b}_r - \mathbb{P}_{\mathbf{n}_r}(\mathbf{y})\|^2 \quad (2)$$

where $\mathbb{P}_{\mathbf{n}_r}$ is the projection operator on the hyperplane with normal vector $\mathbf{n}_r$.

Recognizing the limitation of these binary models, the authors of [15] introduced the mTransH model by extending TransH to multi-fold relations. More specifically, the global embedding loss function f_r in mTransH is defined as:

$$f_r(\mathbf{x}_{\rho_1}, \mathbf{x}_{\rho_2}, \dots, \mathbf{x}_{\rho_{|\mathcal{M}(r)|}}) := \left\| \sum_{\rho \in \mathcal{M}(r)} \mathbf{a}_\rho \mathbb{P}_{\mathbf{n}_r}(\mathbf{x}_\rho) + \mathbf{b}_r \right\|^2, \quad (3)$$

where $\mathbf{n}_r$ is a unit vector and $\mathbf{b}_r$ is a vector on the hyperplane with normal vector $\mathbf{n}_r$. In our RAE model, we will use the loss function in TransH (2) or mTransH (3) as our embedding loss function f_r , depending on whether the KB we deal with is binary or multi-fold. We note however that the RAE model is not restricted to these embedding loss functions, and virtually any multi-fold embedding model can be used.

3.1.2 relatedness loss. The *relatedness loss* function can be the optimization objective in any binary classification model. Here entities x and y are said to be *related*, denoted by $x \sim y$, if there is a relation R_r and an instance $t \in R_r$ such that x and y are two components of vector t . And we write $p(x \sim y)$ as the probability for this event (under our to-be-specified relatedness model).

Let $\mathcal{B}^0$ denote the set of all related entity pairs in G^0 , and $\mathcal{B}$ denote the subset of $\mathcal{B}^0$ that is observed in G . The relatedness loss for every pair (x, y) of entities measures whether x and y are related, or if $(x, y) \in \mathcal{B}^0$. Similar to the way that one constructs the negative examples set $\mathcal{T}^-$, we may create a set $\mathcal{B}^-$ of unrelated entity pairs. This will allow us to use $\mathcal{B}$ and $\mathcal{B}^-$ to train the relatedness model.

In this work, we choose the relatedness loss to be the loss function of multi-layer perceptron (MLP) neural network, which we now specify.

Let w_1 be a $2K \times L$ matrix and w_2 an $L \times 2$ matrix. Let $\text{softmax} : \mathbb{R}^2 \rightarrow \mathbb{R}^2$ be the softmax function on $\mathbb{R}^2$. That is, for every vector $c = [c_1, c_2]^T \in \mathbb{R}^2$,

$$\text{softmax}(c)_i := \frac{\exp(c_i)}{\exp(c_1) + \exp(c_2)}, i = 1, 2.$$

Let $b_1 \in \mathbb{R}^L$ and $b_2 \in \mathbb{R}^2$. Our MLP model is defined by the following sequence of function compositions. For any vector $v \in \mathbb{R}^{2K}$,

$$g := w_1 v + b_1; h := \text{ReLU}(g); c := w_2 h + b_2; z := \text{softmax}(c).$$

For any input $v \in \mathbb{R}^{2K}$, let $\text{MLP}(v)$ denote the output of the MLP. For any pair of entities (x, y) , let

$$p(x \sim y) := \text{MLP}(\phi(x), \phi(y)). \quad (4)$$

That is, if we supply the MLP with the concatenation of the embedding vectors for x and y , the output of the MLP models the likelihood that x and y are related. The parameter of the MLP (w_1, w_2, b_1, b_2) will be denoted by Θ_C . The overall relatedness loss function on the training sets $\mathcal{B}$ and $\mathcal{B}^-$ can then be naturally defined as:

$$E_C(\Theta_C, \phi) := - \sum_{(x, y) \in \mathcal{B}} \log p(x \sim y) - \sum_{(x', y') \in \mathcal{B}^-} \log(1 - p(x' \sim y')) \quad (5)$$

3.1.3 Overall RAE Model. When the model for $p(x \sim y)$ is chosen, the overall loss function of the RAE model is defined as

$$E(\Theta_I, \Theta_C, \phi) := E_I(\Theta_I, \phi) + \lambda E_C(\Theta_C, \phi), \quad (6)$$

where λ is a hyper-parameter governing the relative weighting of relatedness loss and embedding loss.

In the RAE framework, one is free to choose a binary-classification model for $p(x \sim y)$ for related and unrelated (x, y) , as long as $p(x \sim y)$ is made dependent of the embedding map ϕ in some sensible way.

3.2 RAE Training Algorithm

The training of the RAE model can be formulated as solving the minimization problem with objective function $E(\Theta_I, \Theta_C, \phi)$. The minimizing configuration then gives us both an embedding model, which we may use to evaluate if a $|\mathcal{M}(r)|$ -tuple qualifies for being an instance of R_r , and a classifier, which we may use to determine the probability that two entities are related.

The summation form in the definition of the overall loss E and in E_I and E_C allows us to solve the problem via stochastic gradient descent (SGD). In this work, we developed a training algorithm, which is a combination of a three-way alternating minimization and SGD. More specifically, the training of the model consists of three elementary steps:

Step-1: Update (Θ_I, ϕ) following the gradient of E_I with respect to Θ_I . This can be done by SGD on the cost function E_I .

Step-2: Update Θ_C following the gradient of E_C with respect to Θ_C . This can be done by standard back propagation training of the neural network (Figure 2, Top).

Step-3: Update ϕ following the gradient of E_C with respect to ϕ . This can also be done by back propagation, where instead of propagating error back to Θ_C , the error is propagated back to $v = (\phi(x), \phi(y))$ (Figure 2, Bottom). This is essentially an SGD approach and is a modest innovation of this work.

Figure 2: (Top) Back propagation update of Θ_C ; (Bottom) Back propagation update of v

4 SCALABLE INSTANCE RECONSTRUCTION

Algorithm 1: SIR Algorithm

```

for every role  $\rho \in \mathcal{M}(r) \setminus \{\rho^*\}$  do
   $\mathcal{N}(x^*; \rho) := \text{SCH-FILTER}(x^*, r, \rho)$ ;
end
 $\mathcal{N}(x^*; \rho^*) := \{x^*\}$ ;
for every two roles  $\rho_i, \rho_j \in \mathcal{M}(r)$  with  $i < j$  do
   $\mathcal{P}(x^*; \rho_i, \rho_j) := \text{REL-FILTER}(\mathcal{N}(x^*; \rho_i), \mathcal{N}(x^*; \rho_j))$ ;
end
 $\mathcal{I}(x^*) := \text{SPLICE}(\{\mathcal{P}(x^*; \rho_i, \rho_j)\}, x^*)$ ;
 $\mathcal{L}(x^*) := \text{RANK}(\mathcal{I}(x^*))$ ;

```

As discussed earlier, the instance reconstruction problem entails significant and sometimes prohibitive complexity. A natural strategy to fix this problem is to use a “filtering” approach to drastically reduce the number of candidate tuples that will be evaluated under the embedding model. This gives rise to our proposed Scalable Instance Reconstruction (SIR) algorithm.

For a given instance reconstruction task with key x^* (and in relation R_r), the overall structure of the SIR algorithm is given in the pseudo code in Algorithm 1. The SIR algorithm contains three parts: filtering, splicing, and ranking. In this section, we first walk through this algorithm.

We note that for notational convenience, in the description of Algorithm 1, each role in $\mathcal{M}(r)$ is associated with an index $i \in \{1, 2, \dots, |\mathcal{M}(r)|\}$ and the key role ρ^* is ρ_1 , namely, the role with the smallest index.

4.1 Filtering

In this algorithm, we consider two types of filtering: schema-based filtering (SCH-FILTER) and Relatedness filtering (REL-FILTER).

4.1.1 Schema-Based Filtering: SCH-FILTER. Schema-based filtering, which is denoted by subroutine SCH-FILTER, aims to reduce the number of entities to be considered in forming candidate instances. It leverages the type requirements on the entities dictated by the schema of a relation. More specifically, for the relation R_r of the to-be-reconstructed instance and a non-key role $\rho \in \mathcal{M}(r) \setminus \{\rho^*\}$,

Figure 3: EECG Example

the subroutine SCH-FILTER finds the entity types allowed at the role ρ by the schema of R_r , and returns a set of entities by excluding from $\mathcal{N}$ the entities whose types are not compatible with allowed types.

4.1.2 Relatedness filtering: REL-FILTER. Relatedness filtering is treated as a subroutine REL-FILTER in Algorithm 1. The RAE model includes a relatedness model, in which any two entities participate in the same instance in the KB are considered related. When the RAE model is learned, the parameter Θ_C for relatedness model $p(x \sim y)$ is determined. A natural approach is to compare $p(x \sim y)$ with a threshold τ to predict if the two entities are related. More specifically, given an entity x , an arbitrary set $\mathcal{U}$ of entities, the learned parameter Θ_C and an acceptance threshold τ , the function REL-FILTER returns the set of all entity pairs (x, y) such that $y \in \mathcal{U}$ and $p(x \sim y)$ exceeds τ .

4.2 Splicing: SPLICE

The objective of pair splicing is to construct the set $\mathcal{I}(x^*)$ of all *candidate tuples*. It turns out that the problem of constructing candidate tuples can be formulated in an elegant graph-theoretic framework, which we now explain.

The Maximum Color-Matched Clique Problem An *edge-end-coloured graph*, or EECG, $\mathcal{G}$ of colour set $\mathcal{H}$ is an undirected graph in which the two end points of each edge are coloured by two distinct colours in $\mathcal{H}$. For any given EECG $\mathcal{G}$, we will use $V(\mathcal{G})$ to denote the set of vertices of $\mathcal{G}$. If e is an edge in an EECG whose two end points are coloured by colours h and h' respectively, we use $v(e; h)$ (resp. $v(e; h')$) to denote the vertex connected by edge e through its end-point colour h (resp. h'). For any vertex u in an EECG, we will use $\mathcal{H}(u)$ to denote the set of all colours on the edge endpoints attached to u , and refer to $\mathcal{H}(u)$ as the colour set of vertex u . Figure 3 shows an example of an EECG, where $v(e_1; o) = 1$, $v(e_1; r) = 2$, $\mathcal{H}(2) = \{r\}$ and $\mathcal{H}(1) = \{r, o\}$.

Given an EECG $\mathcal{G}$ of colour set $\mathcal{H}$, a *colour-matched clique*, or CMC, is a sub-EECG $\mathcal{G}'$ of $\mathcal{G}$ (namely $\mathcal{G}'$ contains a subset of the edges in $\mathcal{G}$) satisfying the following conditions. (1) $\mathcal{G}'$ is complete. (2) The colour set $\mathcal{H}(u)$ of every vertex u in $\mathcal{G}'$ is a singleton set; we will denote the single colour contained in $\mathcal{H}(u)$ by $h(u)$ and call it the *colour* of u . (3) For every two different vertices u and u' in $\mathcal{G}'$, $h(u) \neq h(u')$, or every two vertices have different colours.

The size of a CMC refers to the number of vertices it contains. In the EECG in Figure 3, the sub-EECG consisting of edges $\{e_2, e_4, e_5\}$ is a CMC of size 3, and every edge is a CMC of size 2. Note that by definition, in a CMC C of size m , the m vertices have m different colours, and we will use $\mathcal{H}(C)$ to denote this set of m colours. A

Algorithm 2: SPLICE Algorithm**Data:** an EECG $\mathcal{G}$ with seed CMC $\{x\}$ **Result:** the set $\mathcal{S}$ of MCMC's of $\mathcal{G}$ **Algorithm $\mathcal{S} := \text{SPLICE}(\mathcal{G}, x)$** $\mathcal{S}_1 := \{\{x\}\};$ **for** $h = 2 \dots |\mathcal{H}|$ **do** $\mathcal{S}_h := \text{GROW-BATCH}(\mathcal{G}, \mathcal{S}_{h-1}, h-1);$ **end****return** $\mathcal{S} := \bigcup \mathcal{S}_h;$ **Procedure** $\text{GROW-BATCH}(\mathcal{G}, \mathcal{A}, h)$ **Data:** an EECG $\mathcal{G}$ of colour set $\mathcal{H}$, a set $\mathcal{A}$ of CMC's C of $\mathcal{G}$ in which each C has $\mathcal{H}(C) = \mathcal{H}'$, a colour $h \in \mathcal{H} \setminus \mathcal{H}'$ **Result:** the set $\mathcal{S}$ of all CMCs C' of $\mathcal{G}$ such that C' contains some $C \in \mathcal{A}$ and $\mathcal{H}(C') = \mathcal{H}' \cup \{h\}$ **for each** $C \in \mathcal{A}$ **do** find $\mathcal{S}_C := \text{GROW}(\mathcal{G}, C, h);$ **end** $\mathcal{S} := \bigcup_{C \in \mathcal{A}} \mathcal{S}_C;$ **return** $\mathcal{S};$ **Procedure** $\text{GROW}(\mathcal{G}, C, h)$ **Data:** an EECG $\mathcal{G}$ of colour set $\mathcal{H}$, a CMC C of $\mathcal{G}$, a colour $h \in \mathcal{H} \setminus \mathcal{H}(C)$ **Result:** the set $\mathcal{S}$ of all CMCs C' of $\mathcal{G}$ such that C' contains C and $\mathcal{H}(C') = \mathcal{H}(C) \cup \{h\}$ **for each** $u \in V(C)$ **do** find $\mathcal{V}(u; h);$ **end** $W := \bigcap_{u \in V(C)} \mathcal{V}(u; h);$ $\mathcal{S} := \{C + w : w \in W\};$ **return** $\mathcal{S};$

CMC C in the EECG $\mathcal{G}$ of colour set $\mathcal{H}$ will be called a *maximum CMC*, or *MCMC*, if $\mathcal{H}(C) = \mathcal{H}$. The following results can be proved.

LEMMA 1. *For a given reconstruction task with key x^* , interpret the set $\{\mathcal{P}(x^*; \rho_i, \rho_j) : \rho_i, \rho_j \in \mathcal{M}(r) : i < j\}$ (Algorithm 1) as an EECG $\mathcal{G}(x^*)$. Every related entity pair in G^0 can be interpreted as an endpoint-coloured edge. Under such interpretations, if the set of all related entities are contained in $\mathcal{G}(x^*)$, then each instance in G^0 involving x^* can be interpreted as an MCMC containing x^* .*

Under this lemma, the pair splicing problem can be equated with finding the set of all MCMC's of $\mathcal{G}(x^*)$.

A MCMC Finding Algorithm On an EECG $\mathcal{G}$ of colour set $\mathcal{H}$, for any given vertex u and any given colour $h \in \mathcal{H}$, let $\mathcal{V}(u; h)$ be the set of all vertices u' in $\mathcal{G}$ such that there is an edge, say e , connecting u and u' and $v(e; h) = u'$. If $\mathcal{G}'$ is a sub-EECG of $\mathcal{G}$ and u is vertex in $\mathcal{G}$ but not in $\mathcal{G}'$, then we use $\mathcal{G}' + u$ to denote the sub-EECG of $\mathcal{G}$ which consists of all edges and vertices in $\mathcal{G}'$, vertex u , and the set of all edges connecting to u in $\mathcal{G}$.

Using these notations, we present an algorithm SPLICE (Algorithm 2), which finds the set of all MCMC's for a given EECG $\mathcal{G}$ with seed CMC $\{x\}$, where the colour set $\mathcal{H}$ of $\mathcal{G}$ is expressed as $\{1, 2, \dots, |\mathcal{H}|\}$ and the colour of the seed CMC is taken as 1.

LEMMA 2. *The algorithm SPLICE returns the set of all MCMC's of $\mathcal{G}(x^*)$.*

The proof of the lemma can be established via induction.

4.3 Ranking: RANK

Upon obtaining the set $\mathcal{I}(x^*)$ of candidate tuples, Algorithm 1 finally executes the RANK subroutine, which performs the following computation in order.

- (1) Evaluating the embedding loss for each candidate tuple $t = (x_1, x_2, \dots, x_{|\mathcal{M}(r)|}) \in \mathcal{I}(x^*)$ using the learned embedding loss function f_r , namely, evaluating the quantity $f_r(\phi(x_1), \phi(x_2), \dots, \phi(x_{|\mathcal{M}(r)|}))$.
- (2) Ranking the tuples in $\mathcal{I}(x^*)$ according to the their embedding loss, from low to high (low loss having a better ranking). The resulting ranked list is denoted by $L(x^*)$.

5 EXPERIMENTAL STUDY**5.1 Datasets**

Since the objective of this paper is to develop a scalable model and algorithm for instance reconstruction, which is particularly challenging for multi-fold relational data, the primary dataset in our experiments is the JF17K dataset. Having complexity reduction as our main goal, it is also desirable to investigate the reconstruction performance of our approach and hope that this reduction in complexity entails no loss of performance, comparing with the standard embedding models. Such a comparison is however infeasible on the JF17K dataset, since using any traditional model even on a modestly sized KB with multi-fold relations would entail significant complexity. For this reason, we also experiment our approach and a traditional embedding model on the FB15K dataset, as a reference for performance comparison.

5.1.1 FB15K and JF17K. FB15K, developed by [4] and filtered from Freebase, contains binary relational data. Note that Freebase originally contains multi-fold relations. As discussed in [4] and [15], in FB15K, these data are converted to binary in a way much similar to how we define the relatedness relation $\sim$. JF17K, developed in [15], is also filtered from Freebase, while having multi-fold relational structure preserved. JF17K dataset can be found at <http://github.com/lijp12/SIR>.

Both datasets are further pre-processed using the schema of Freebase. The Freebase schema lists the set of all included relations and defines the expected entity types for the roles of each relation. Although FB15K converts multi-fold relations to binary relations, its schema can be retrieved from Freebase. The most common type in Freebase schema is "common.topic"; in FB15K and JF17K, more than 99% of the entities have this type label. This type, containing virtually no information, is removed from the schema. In addition, relations not included in the schema are also deleted. In total, 97 relations from FB15K and 5 relations from JF17K are removed, together with the entities only involved in these relations. Each resulting dataset is partitioned into the training set $\mathcal{T}$ and the testing set $\mathcal{T}_{\text{test}}$. Table 1 shows the statistics of FB15K and JF17K datasets.

5.1.2 Derived Datasets. Several datasets are derived from FB15K and JF17K.

Table 1: Statistics of Datasets

Dataset	$ \mathcal{N} $	$ \mathcal{R} $	$ \mathcal{T} $	$ \mathcal{B} $	$ \mathcal{T}_{\text{test}} $	$ \mathcal{B}_{\text{test}} $
FB15K	14921	1243	455101	455101	55721	55721
JF17K	28645	322	76379	171559	24568	66615

Table 2: Breakdown of testing keys by relation arity

Dataset	$ \mathcal{K}(2) $	$ \mathcal{K}(3) $	$ \mathcal{K}(4) $	$ \mathcal{K}(5) $	$ \mathcal{K}(6) $	$ \mathcal{K} $
FB15k	43180	0	0	0	0	43180
JF17K	5199	3277	776	87	7	9346

Negative Training Sets ($\mathcal{T}^-$ and $\mathcal{B}^-$) Negative training examples $\mathcal{T}^-$ and $\mathcal{B}^-$ are constructed as follows. For each positive training example $t \in \mathcal{T}$ (and resp. $b \in \mathcal{B}$), we build a negative example in $\mathcal{T}^-$ (and resp. $\mathcal{B}^-$), by substituting one entity in t (resp. in b) with another random entity.

Relatedness Prediction Testing Sets ($\mathcal{B}_{\text{test}}$ and $\mathcal{B}_{\text{test}}^-$) For the purpose of evaluating the relatedness prediction performance under the relatedness model, we constructed two classification testing sets for FB15K and JF17K. The positive testing set $\mathcal{B}_{\text{test}}$ is constructed from $\mathcal{T}_{\text{test}}$. For FB15K, $\mathcal{B}_{\text{test}}$ is constructed by simply removing the relation label for each triple in $\mathcal{T}_{\text{test}}$, which turns the triples into pairs. That is, $|\mathcal{T}_{\text{test}}| = |\mathcal{B}_{\text{test}}|$. For JF17K, we turn each m -fold instance in $t \in \mathcal{T}_{\text{test}}$ into $\binom{m}{2}$ testing pairs by pairing every two possible entities involved in t . The negative testing set $\mathcal{B}_{\text{test}}^-$ is constructed by randomly replacing one entity for each pair in $\mathcal{B}_{\text{test}}$.

Testing Set of Keys ($\mathcal{K}$) For each dataset (FB15K and JF17K), we extract a testing set $\mathcal{K}$ of keys. Given a relation R_r and any of its role $\rho \in \mathcal{M}(r)$, let $\gamma(\rho; r)$ be the number of distinct entities that participate in an instance of $\mathcal{T}_r$ as role ρ . We define the role ρ^* that maximizes $\gamma(\rho; r)$ as the *primary role* for relation R_r . Table 2 shows the statistics about testing keys by relation arity. Note that FB15K dataset only contains binary relations, so its relation arity is just 2. For a given instance $t \in R_r$, let x^* be the entity in t that is corresponding to the primary role of R_r . The entity x^* is then taken as the key for reconstructing t . For example, in Figure 1, the primary role for relation PeopleMarriage is PERSON, since the number of distinct people participating in this relation as role PERSON is larger than the number of distinct entities participating the relation in each of the other roles. As such, KobeBryant is chosen as the key. Using this procedure, we extract distinct keys from $\mathcal{T}_{\text{test}}$ to form the set $\mathcal{K}$ of testing keys.

5.2 General Setup and Model Training

We train the proposed RAE model on both JF17K and FB15K while training a baseline TransH model only on FB15K. The dimension of the embedding space for all models is chosen as 100.

5.2.1 Training of RAE Model on JF17K and FB15K. The embedding loss f_r in the objective function of the RAE model is chosen as the mTransH loss for JF17K and TransH loss for FB15K. In the MLP network of the relatedness model, the input layer and the hidden layer contain 200 and 400 neurons respectively.

In the training of the model, the algorithm loops over the three steps for 10 outer iterations. In each outer iteration:

- The Step-1 update is run for 100 epochs. In each epoch, (Θ_I, ϕ) is updated using mini-batch stochastic gradient descent. Each mini-batch contains 1000 pairs of positive and

negative examples (in $\mathcal{T}$ and $\mathcal{T}^-$). The margin C for both TransH and mTransH is set to 0.5.

- The Step-2 update is run for 10 epochs. In each epoch, Θ_C is updated by going through the entire positive and negative training examples (in $\mathcal{B}$ and $\mathcal{B}^-$) once using mini-batched Back Propagation [10]. Each mini-batch contains 1000 pairs of positive and negative training examples. The learning rate is set as 0.005.
- The Step-3 update is run for 5 epochs. Each epoch is structured as mini-batches in the same way as those in the Step-2 updates. For each mini-batch, only the embedding vectors of the involved entities are updated. The learning rate is set as 0.001.

5.2.2 Training of TransH Model on FB15K. The training of the TransH model on FB15K is structured similarly as the Step-1 update in training the RAE model on FB15K, except that in total 1000 epochs are run.

5.3 Relatedness Prediction

Applying the learned parameter Θ_C of RAE to the relatedness model $p(x \sim y)$, we investigate the model's relatedness-prediction performance on the testing sets $\mathcal{B}_{\text{test}}$ and $\mathcal{B}_{\text{test}}^-$, over the entire range of the acceptance threshold value τ . For a given value of acceptance threshold τ , the true positive rate $\text{TP}(\tau)$ and false positive rate $\text{FP}(\tau)$ are used as the performance criteria.

The behaviour of $\text{TP}(\tau)$ and $\text{FP}(\tau)$ on FB15K and JF17K is plotted as the Receiver Operation Characteristics (ROC) curves (namely, the trajectory of point $(\text{TP}(\tau), \text{FP}(\tau))$ as τ increases in $[0, 1]$) in Figure 4. The small area above the ROC curves suggests an excellent prediction performance.

Superimposed on the ROC plots in Figure 4, the value of τ is also plotted against its corresponding $\text{FP}(\tau)$. Observing that the TP and FP values at $\tau = 0.5$ (Figure 4), we conclude that $\tau = 0.5$ is a good choice and adopt it in the rest of the experiments.

Figure 4: Relatedness prediction performances. Left: FB15K, $\text{TP}(0.5)=90.38\%$, $\text{FP}(0.5)=4.37\%$. Right: JF17K, $\text{TP}(0.5)=95.53\%$, $\text{FP}(0.5)=5.44\%$

5.4 Link Prediction

In order to verify whether the RAE model affects the embedding performance of its component embedding model, we perform a set of link prediction tasks. The experiments are performed on both JF17K and FB15K. For each instance $t \in \mathcal{T}_{\text{test}}$ and each ρ of the instance, we extract a key t^* by making ρ the query role. This gives us a set of keys for link prediction tasks. We then train separately a

standard embedding model (TransH for FB15K and mTransH for JF17K), and use the model to obtain the link prediction performance. We also take the embedding model from the trained RAE model and obtain its link prediction performance. The standard Mean Rank and HIT@10 metrics are used as performance measures.

From Table 3, it can be observed that the incorporation of the relatedness model does not degrade the traditional TransH/mTransH performance. In some cases, the HIT@10 and Mean Rank are even improved.

Table 3: Link Prediction Performance

	FB15K		JK17K	
	TransH	RAE	mTransH	RAE
HIT@10	44.30%	44.27%	49.74%	50.35%
Mean Rank	171.24	171.03	237.82	229.27

5.5 Instance Reconstruction

In our instance-reconstruction experiments, we are interested in investigating the complexity reduction attained by the SIR algorithm in knowledge base containing multi-fold relations. This makes JF17K our primary test bench. Additionally we wish to validate the performance of the SIR algorithm by comparing it with the baseline reconstruction algorithm that solely relies on KB embedding (mTransH). The latter is however infeasible since the baseline algorithm entails prohibitive complexity when applied to JF17K data. We then use FB15K for this purpose, on which the baseline algorithm is much less costly (note that although the FB15K dataset is binary in nature, the instance reconstruction problem is still well defined, except that it reduces to the link prediction problem). So we perform three instance-reconstruction experiments: SIR reconstruction on JF17K, SIR reconstruction on FB15K, and TransH-based reconstruction on FB15K. Note that in the TransH-based reconstruction for each key entity x^* , the candidate triples are evaluated using the TransH loss function.

For any dataset with the set $\mathcal{K}$ of reconstruction keys, each algorithm under comparison outputs a ranked list $L(x^*)$ of candidate tuples for each key $x^* \in \mathcal{K}$. For SIR, $L(x^*)$ is defined as in Algorithm 1, which may be denoted by $L^{\text{SIR}}(x^*)$, whenever required, to distinguish from the list $L^{\text{mTransH}}(x^*)$ or $L^{\text{TransH}}(x^*)$ produced by the baseline algorithm. Note that $L^{\text{mTransH}}(x^*)$ and $L^{\text{TransH}}(x^*)$ contain all length- $|\mathcal{M}(r)|$ tuples in which the entity at role ρ^* is x^* .

It is remarkable that the output lists $L(x^*)$ from the compared algorithms differ significantly in length and in content and that the ranking is done within their own respective lists. Thus one must be cautious about the performance metric used to compare the lists. Using usual metrics such as HIT@K or Mean Rank to directly compare the lists, which in fact favours the shorter list (namely $L^{\text{SIR}}(x^*)$), fails to give a fair comparison.

5.6 Instance Reconstruction Performance

5.6.1 Performance Metrics. For each key x^* , the output of each reconstruction algorithm is a ranked list $L(x^*)$. We now introduce a generalized notion of rank for key x^* , then explain what it means when it takes a particular form.

Let $\mathcal{W}(x^*)$ be a subset of $\mathcal{T}_{\text{test}}$. For each key $x^* \in \mathcal{K}$, its average rank with respect to $\mathcal{W}(x^*)$ is naturally defined as

$$\text{RNK}(x^*; \mathcal{W}(x^*)) = 1 + \frac{1}{|\mathcal{W}(x^*)|} \sum_{t \in \mathcal{W}(x^*)} l(t) \quad (7)$$

where $l(t)$ is the number of instances in $L(x^*)$ which have a rank value smaller (i.e., better) than the correct answer t and which are neither in $\mathcal{T}$ nor in $\mathcal{T}_{\text{test}}$. Note that the tuples that are neither in $\mathcal{T}$ nor in $\mathcal{T}_{\text{test}}$ are incorrect answers if we assume that $\mathcal{T}$ and $\mathcal{T}_{\text{test}}$ contain all correct answers (in the sense of reflecting the ground truth) for the reconstruction tasks. Then $\text{RNK}(x^*; \mathcal{W}(x^*)) - 1$ measures on average how many incorrect answers are put ahead of the “correct answers” in list $L(x^*)$.

In this work, we will take $\mathcal{W}(x^*)$ in the following three forms.

- **“SIR-Covered”** form, namely, $\mathcal{W}(x^*) = \mathcal{T}_{\text{test}} \cap L^{\text{SIR}}(x^*)$. In this case, only the testing instances that are included in SIR’s list are regarded as “correct answers”.
- **“SIR-Missed”** form, namely, $\mathcal{W}(x^*) = \mathcal{T}_{\text{test}} \setminus L^{\text{SIR}}(x^*)$. In this case, only the testing instances that are excluded in SIR’s list are regarded as “correct answers”.
- **“All”** form, namely, $\mathcal{W}(x^*) = \mathcal{T}_{\text{test}}$. In this case, all testing instances are regarded as “correct answers”.

Further suppose that $\mathcal{W} = \{\mathcal{W}(x^*) : x^* \in \mathcal{K}\}$. Then we define the mean rank $\overline{\text{RNK}}(\mathcal{W})$ with respect to $\mathcal{W}$ as

$$\overline{\text{RNK}}(\mathcal{W}) = \frac{1}{|\mathcal{K}|} \sum_{x^* \in \mathcal{K}} \text{RNK}(x^*; \mathcal{W}(x^*)) \quad (8)$$

where $\mathcal{W}(x^*)$ can take any of the above three forms.

The **TopK**($\mathcal{W}$) performance with respect to $\mathcal{W}$ is defined as the fraction of keys $x^* \in \mathcal{K}$ with $\text{RNK}(x^*; \mathcal{W}(x^*)) \leq K$. We will primarily use the **TopK** and $\overline{\text{RNK}}$ metrics to measure and compare the performance of SIR and its baseline algorithms. We note that these metrics are only well defined for an algorithm when $\mathcal{W}(x^*)$ is a subset of the algorithm’s output list $L(x^*)$.

To measure to what extent the candidate tuple lists produced by SIR cover the testing set $\mathcal{T}_{\text{test}}$, we also define a notion *SIR Coverage*, or **Coverage**, as

$$\text{Coverage} := \frac{|\mathcal{T}_{\text{test}} \cap (\bigcup_{x^* \in \mathcal{K}} L^{\text{SIR}}(x^*))|}{|\mathcal{T}_{\text{test}}|}.$$

5.6.2 Instance Reconstruction Performance Result. The performance of SIR algorithm (overall and breakdown according to relation arity) on JF17K is shown in Table 4. The $\overline{\text{RNK}}$ performance might have appeared poorer than usually those seen in link prediction over binary relations. This is merely a consequence of much larger number of tuples existing as possible candidates for reconstruction in multi-fold relations. Notably, however, the **TopK** performance appear quite impressive. For example, reconstruction from 19.24% of the keys ranks the correct instances at the very top of its list; reconstruction from 41.29% of the keys ranks the correct instances within top 10 of its list. Additionally, the candidate list produced by the SIR algorithm covers 88.54% of the correct answers (i.e., testing instances).

Nonetheless, from Table 4 alone, no conclusion can be drawn whether SIR algorithm suffers from a loss of performance relative to more naive baseline. Such a conclusion can however be drawn from experiments on the FB15k dataset, as seen in Table 5.

In Table 5, the reconstruction performance of SIR is compared with its baseline TransH-based algorithm on FB15K. With respect to the SIR-Covered “correct answers”, SIR improves upon the baseline by a large margin in both **TopK** and **RNK** metrics. Specifically, SIR reconstruction from over 50% of the keys ranks the “correct answers” at the very top of its list, contrasting the 0.38% **Top1** performance of

Table 4: Instance Reconstruction Performance of SIR on JF17K. $\mathcal{W}(x^*)$ in SIR-Covered form. Coverage = 88.54%

arity	Top1	Top10	Top20	Top50	RNK
2	25.07%	52.68%	60.45%	74.63%	47.63
3	12.42%	27.29%	32.52%	41.21%	610.09
4	11.56%	29.48%	34.67%	41.88%	34844.11
5	2.78%	8.33%	9.72%	13.89%	10229.41
6	20.00%	60.00%	80.00%	100.00%	10.80
overall	19.24%	41.29%	47.89%	59.45%	2981.81

Table 5: Instance reconstruction(FB15K): performance of SIR and its baseline. Coverage=88.73%

	$\mathcal{W}(x^*)$	TransH	SIR
Top10	SIR-Covered	83.37%	92.23%
	SIR-Missed	42.00%	-
	All	78.09%	-
Top1	SIR-Covered	0.38%	50.45%
	SIR-Missed	3.78%	-
	All	0.85%	-
RNK	SIR-Covered	10.96	4.15
	SIR-Missed	252.55	-
	All	41.13	-

TransH. In addition, SIR reconstruction ranks the “correct answers” at top 4.15 on average, whereas the TransH baseline ranks them at top 10.96 on average. Of course, SIR misses some “correct answers”. It is interesting to note however that with respect to these SIR-Missed answers, the TransH baseline also ranks them much more unfavourably than it does with the SIR-Covered answers. For example, it ranks SIR-Missed “correct answers” at top 252.55 in its list on average, significantly contrasting its RNK performance of 10.96 on the SIR-Covered answers. That is, even though SIR misses some answers, which is a necessary tradeoff due to filtering, these answers are also unlikely to be selected by the baseline algorithm. On the other hand, for the answers that favoured by the baseline algorithm, SIR further promotes them on average.

Although reconstruction performance advantage of SIR over TransH demonstrated in Table 5 is observed on the FB15K dataset, one may expect that it extrapolates reasonably well to JF17K, since both datasets are extracted from the same knowledge base, Freebase.

From these results, we conclude that instance reconstruction by the SIR algorithm does not appear to entail any performance loss. But it offers significant complexity advantage as we will show next.

5.7 Instance Reconstruction Complexity: Analysis and Experimental Results

In this work, we are not only interested in characterizing the complexity of the proposed SIR algorithm, but we would also like to compare it against other baseline methods. Noting that SIR contains two key filtering mechanisms for complexity reduction, the schema-based filtering and relatedness filtering, we then choose the following baseline for complexity comparison: reconstruction with no filtering (NF), reconstruction with schema-based filtering only (SCH), and reconstruction with relatedness filtering only (REL).

5.7.1 Complexity Metrics for Instance Reconstruction. For large knowledge bases, the computational complexity of instance reconstruction, both for the SIR algorithm and for the base embedding

model is nearly exclusively dominated by splicing the candidate pairs into candidate instances and by evaluating the embedding loss for these tuples. This is because forming the candidate pairs requires a complexity of order no more than N^2 , whereas for splicing and loss evaluation, the complexity order is a high-degree polynomial in N . For example, in our experiments on JF17K dataset, the SIR algorithm (Algorithm 1) spends more than 99% of its total computation time on executing the SPLICE and RANK subroutines. Thus, the primary complexity measures are those concerning pair splicing and loss evaluation.

As such, we define two fundamental notions of complexity measure: the splicing complexity (**CmpSp**) and the Evaluation Complexity (**CmpEv**). In reconstructing an instance in relation R_r from a given key x^* , the splicing complexity **CmpSp**(x^*) is defined as the total number of $|\mathcal{M}(r)|$ -tuples that need to be *examined* in order to produce the candidate instances set; the evaluation complexity **CmpEv**(x^*) is defined as the total number of candidate instances that need to be evaluated under the embedding loss function.

The two respective notions of complexity can be extended to various levels, which we now define:

$$\text{CmpX}(m) := \sum_{x^* \in \mathcal{K}(m)} \text{CmpX}(x^*) \quad (9)$$

$$\text{CmpX} := \sum_m \text{CmpX}(m) \quad (10)$$

where “X” can be either “Sp” or “Ev”. Obviously **CmpX**(x^*), **CmpX**(m), and **CmpX** measure respectively the complexities for a particular reconstruction task x^* , for all reconstruction tasks in m -fold relations, and for the entire set of reconstruction tasks.

By definition, it is obvious that the various notions of Splicing Complexity are upper bounds of the respective notions of Evaluation Complexity.

5.7.2 Instance Reconstruction: Complexity Analysis. Using above notions of complexity, we now analyze the reconstruction complexities for NF (no filtering), SCH (schema-based filtering only), REL (relatedness filtering only) and SIR.

LEMMA 3. *For NF, the Splicing Complexity and Evaluation Complexity coincide, which will be both denoted by **CmpX**^{NF}. Then*

$$\text{CmpX}^{\text{NF}}(x^*) = N^{m-1}$$

Let $\mathcal{N}(x^*; \rho)$ be that defined in Algorithm 1, namely, $\mathcal{N}(x^*; \rho)$ is the set of all entities left for role ρ after schema-based filtering. Then the complexities of SCH can be expressed using this term.

LEMMA 4. *For SCH, the Splicing Complexity and Evaluation Complexity coincide, which will be both denoted by **CmpX**^{SCH}. Then*

$$\text{CmpX}^{\text{SCH}}(x^*) = \prod_{\rho \in \mathcal{M}(r) \setminus \{\rho^*\}} |\mathcal{N}(x^*; \rho)|$$

To determine the complexity for REL, for each non-key role $\rho \in \mathcal{M}(r) \setminus \{\rho^*\}$, let $\hat{\mathcal{N}}(x^*)$ be the set of all entities that are pre-dicted to be related to x^* after applying the relatedness filtering. Let $\hat{\mathcal{P}}(x^*) := \text{REL-FILTER}(\hat{\mathcal{N}}(x^*), \hat{\mathcal{N}}(x^*))$. That is, $\hat{\mathcal{P}}(x^*)$ is the set of all related entity pairs (a, b) where both a and b are related to x^* . Interpreting $\hat{\mathcal{P}}(x^*)$ as an undirected graph where each pair $(a, b) \in \hat{\mathcal{P}}(x^*)$ is interpreted as an edge connecting entities (vertex) a and b , we will use $\mathcal{J}(x^*, k; \hat{\mathcal{P}}(x^*))$ to denote the set of all cliques in $\hat{\mathcal{P}}(x^*)$ having k vertices.

LEMMA 5. For REL,

$$\begin{aligned}\text{CmpSp}^{\text{REL}}(x^*) &= \prod_{\rho \in \mathcal{M}(r) \setminus \{\rho^*\}} |\hat{\mathcal{N}}(x^*)| \\ \text{CmpEv}^{\text{REL}}(x^*) &= \left| \mathcal{J}(x^*, |\mathcal{M}(r)| - 1; \hat{\mathcal{P}}(x^*)) \right|\end{aligned}$$

Let $\mathcal{I}(x^*)$ be that defined in Algorithm 1, namely, $\mathcal{I}(x^*)$ is the set of all candidate tuples resulted from SPLICE in the SIR algorithm.

LEMMA 6. For SIR,

$$\begin{aligned}\text{CmpSp}^{\text{SIR}}(x^*) &= \prod_{\rho \in \mathcal{M}(r) \setminus \{\rho^*\}} |\tilde{\mathcal{N}}(x^*; \rho)| \\ \text{CmpEv}^{\text{SIR}}(x^*) &= |\mathcal{I}(x^*)|\end{aligned}$$

Since we have determined $\text{CmpX}(x^*)$ through the above lemmas, we can further get $\text{CmpX}(m)$ and CmpX by definition (9) and (10). The computation of the complexities of these algorithms usually requires counting the filtering results (except for the case of NF). As such, we will investigate the complexity comparison of these algorithms in our experiments.

Among the four algorithms, NF is clearly the most naive baseline, comparing with which all other three algorithms have reduced complexity. It is important to note however that the complexity reduction attained by other algorithms, or by any algorithm is not via reducing the order of the polynomial complexity of NF. The complexity reduction is via significantly reducing the coefficients of the polynomial, or alternatively put, via “effectively” reducing the size of the KB. We now elaborate on this perspective.

Now we can re-express CmpSp^A in the form of

$$\text{CmpSp}^A = \sum_m |\mathcal{K}(m)| (\epsilon_m^A N)^{m-1} \quad (11)$$

to match the polynomial expression $\sum_m |\mathcal{K}(m)| N^{m-1}$ of NF complexity in Lemma 3. The term ϵ_m^A in the equation above has an interesting interpretation: *as far as m -fold relations are concerned, the KB size is effectively scaled down by a factor of ϵ_m^A .*

LEMMA 7. Let m be an arbitrary arity value. If in each instance reconstruction task from key x^* , each $|\mathcal{N}^A(x^*; \rho)|$ is distributed independently and the distribution is independent of the relation R_r and ρ , then

$$\tilde{\epsilon}_m^A := \frac{1}{|\mathcal{K}(m)|} \sum_{x^* \in \mathcal{K}(m)} \left(\prod_{\rho \in \mathcal{M}(r) \setminus \{\rho^*\}} \frac{N^A(x^*; \rho)}{N} \right)^{\frac{1}{m-1}} \quad (12)$$

is an unbiased estimate of ϵ_m^A .

Under this lemma, we may use $\tilde{\epsilon}_m^A$ to estimate by how much the algorithm A effectively reduces the KB size.

5.7.3 Instance Reconstruction: Complexity Results (on JF17K). In this subsection, we compare the complexity of SIR on JF17K with baselines that utilize schema-based filtering only (SCH), relatedness filtering only (REL), or no filtering (NF). The computation of the complexity metrics is based on the analytic tools developed above and data collected from experiments. In particular, complexity results for SIR and REL are obtained using the trained RAE model, whereas the results for NF and SCH are obtained using the base embedding model, mTransH.

Tables 6 and 7 report the Evaluation Complexity and Splicing Complexity for the compared schemes. Compared with the naive NF

scheme, SCH, REL, and SIR reduce Evaluation Complexity by about 10^{11} , 10^4 , and 10^{13} folds respectively. Similar orders of reduction are seen in Splicing Complexity. This suggests that schema-based filtering plays a more important role than relatedness filtering. Nonetheless both filtering schemes are required to make instance reconstruction complexity reduce to an affordable level.

For NF, the complexity is dominated by highest-arity (arity-6) keys, although there are only 7 such keys (Table 2). However when filtering schemes are applied, this dominance reduces. In fact, when both filtering schemes are applied, namely, in SIR, arity-4 keys become dominant, noting that there are 776 arity-4 keys, significantly more abundant than the arity-5 and arity-6 keys while also significantly scarcer than the arity-2 and arity-3 keys (Table 2).

The reduction factors $\tilde{\epsilon}_m^{\text{SIR}}$ shown in Table 7 suggest that with respect to relations of various arities, SIR effectively reduces the KB size by a scaling factor of 10^{-3} to 10^{-2} . But since arity-6 keys dominates the NF complexity, the overall reduction factor can be justified as close to 1.31×10^{-3} .

Table 6: Evaluation Complexity (JF17K)

arity	NF	SCH	REL	SIR
2	1.49e+8	2.91e+6	1.96e+7	1.14e+6
3	2.69e+12	5.10e+8	3.19e+10	3.14e+7
4	1.82e+16	4.01e+11	$\leq 4.76e+13$	9.26e+9
5	5.86e+19	1.08e+11	$\leq 2.18e+17$	8.99e+7
6	1.35e+23	2.96e+12	$\leq 3.27e+19$	2.28e+6
overall	1.35e+23	3.47e+12	$\leq 3.29e+19$	9.38e+9

Table 7: Splicing Complexity (JF17K)

m	NF	SCH	REL	SIR	$\tilde{\epsilon}_m^{\text{SIR}}$
2	1.49e+8	2.91e+6	1.96e+7	1.14e+6	0.00766
3	2.69e+12	5.10e+8	4.61e+10	7.25e+7	0.00377
4	1.82e+16	4.01e+11	2.65e+14	6.65e+10	0.01122
5	5.86e+19	1.08e+11	2.74e+17	4.33e+9	0.00152
6	1.35e+23	2.96e+12	3.21e+20	2.13e+9	0.00131
overall	1.35e+23	3.47e+12	3.21e+20	7.31e+10	-

6 CONCLUSIONS

Relations in the real world are often multi-fold. Storing instances from multi-fold relations in their original (multi-fold) format has the advantage of better preserving their information content. We observe that knowledge bases (KB) containing multi-fold relations are however challenged by a new class of KB completion problem, the instance reconstruction problem. This paper formally introduces this problem and explains its complexity challenge. We also develop a new KB embedding model, called Relatedness Affiliated Embedding (RAE), and build on it a Scalable Instance Reconstruction (SIR) algorithm. We demonstrate experimentally that the SIR algorithm has significantly reduced complexity with virtually no loss of reconstruction performance.

7 ACKNOWLEDGMENTS

This work is supported partly by China 973 program (No. 2014CB340305, 2015CB358700), by the National Natural Science Foundation of China (No. 61772059, 61421003) and by the Beijing S&T Committee (Z161100000216127). This paper is also supported by the State Key Laboratory of Software Development Environment of China and Beijing Advanced Innovation Center for Big Data and Brain Computing.

REFERENCES

- [1] Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. 2007. *Dbpedia: A nucleus for a web of open data*. Springer.
- [2] Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*. ACM.
- [3] Antoine Bordes, Xavier Glorot, Jason Weston, and Yoshua Bengio. 2014. A semantic matching energy function for learning with multi-relational data. *Machine Learning* 94, 2 (2014), 233–259.
- [4] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Advances in Neural Information Processing Systems*. 2787–2795.
- [5] Antoine Bordes, Jason Weston, Ronan Collobert, and Yoshua Bengio. 2011. Learning structured embeddings of knowledge bases. In *Conference on Artificial Intelligence*.
- [6] David Liben-Nowell and Jon M. Kleinberg. 2007. The link-prediction problem for social networks. *JASIST* 58, 7 (2007), 1019–1031.
- [7] Yankai Lin, Zhiyuan Liu, Huanbo Luan, Maosong Sun, Siwei Rao, and Song Liu. 2015. Modeling Relation Paths for Representation Learning of Knowledge Bases. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Lisbon, Portugal.
- [8] Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. 2015. Learning Entity and Relation Embeddings for Knowledge Graph Completion. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence (AAAI'15)*.
- [9] Alex Marin, Roman Holenstein, Ruhi Sarikaya, and Mari Ostendorf. 2014. Learning phrase patterns for text classification using a knowledge graph and unlabeled data. In *Fifteenth Annual Conference of the International Speech Communication Association*.
- [10] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. 1988. Learning representations by back-propagating errors. *Cognitive modeling* 5, 3 (1988), 1.
- [11] Richard Socher, Danqi Chen, Christopher D Manning, and Andrew Ng. 2013. Reasoning with neural tensor networks for knowledge base completion. In *Advances in Neural Information Processing Systems*. 926–934.
- [12] Fabian M Suchanek, Gjergji Kasneci, and Gerhard Weikum. 2007. Yago: a core of semantic knowledge. In *Proceedings of the 16th international conference on World Wide Web*. ACM, 697–706.
- [13] Yi Tay, Anh Tuan Luu, Siu Cheung Hui, and Falk Brauer. 2017. Random Semantic Tensor Ensemble for Scalable Knowledge Graph Link Prediction. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining, WSDM 2017, Cambridge, United Kingdom, February 6-10, 2017*. 751–760.
- [14] Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014. Knowledge graph embedding by translating on hyperplanes. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*. 1112–1119.
- [15] Jianfeng Wen, Jianxin Li, Yongyi Mao, Shini Chen, and Richong Zhang. 2016. On the Representation and Embedding of Knowledge Bases beyond Binary Relations. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016, New York, NY, USA, 9-15 July 2016*. 1300–1307.
- [16] Han Xiao, Minlie Huang, and Xiaoyan Zhu. 2016. TransG : A Generative Model for Knowledge Graph Embedding. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*.
- [17] Chenyan Xiong and Jamie Callan. 2015. EsdRank: Connecting Query and Documents Through External Semi-Structured Data. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management (CIKM '15)*. ACM, New York, NY, USA, 951–960.
- [18] Chenyan Xiong and Jamie Callan. 2015. Query Expansion with Freebase. In *Proceedings of the Fifth ACM International Conference on the Theory of Information Retrieval*. ACM.