Generating Referring Expressions from RDF Knowledge Graphs for Data Linking

Armita Khajeh Nassiri[1](✉), Nathalie Pernelle[1,2](✉), Fatiha Saïs[1](✉), and Gianluca Quercini[1](✉)

[1] LRI, CNRS 8623, Paris Saclay University, 91405 Orsay, France
{armita.khajeh_nassiri,nathalie.pernelle,fatiha.sais, gianluca.quercini}@lri.fr

[2] LIPN, CNRS (UMR 7030), University Sorbonne Paris Nord, Villetaneuse, France

Abstract. The generation of referring expressions is one of the most extensively explored tasks in natural language generation, where a description that uniquely identifies an instance is to be provided. Some recent approaches aim to discover referring expressions in knowledge graphs. To limit the search space, existing approaches define quality measures based on the intuitiveness and simplicity of the discovered expressions. In this paper, we focus on referring expressions of interest for data linking task and present *RE-miner*, an algorithm tailored to automatically discover minimal and diverse referring expressions for all instances of a class in a knowledge graph. We experimentally demonstrate on several benchmark datasets that, compared to existing data linking tools, referring expressions for data linking substantially improve the results, especially the recall without decreasing the precision. We also show that the RE-miner algorithm can scale to datasets containing millions of facts.

Keywords: Knowledge graphs · Referring expressions · Data linking

1 Introduction

A *referring expression* (*RE*) is a description in natural language or a logical formula that can uniquely identify an entity. For instance, the statement "president of the United States who was born in Hawaii" is a referring expression that unambiguously characterizes Barack Obama. There may potentially exist many logical expressions for uniquely identifying an entity. Referring expressions find applications in disambiguation, data anonymization, query answering, and data linking. The generation of referring expressions is a well-studied task in natural language generation [22], and various algorithms with different objectives have been proposed to discover REs automatically. These approaches vary depending on the expressivity of the logical formulas they can generate. For instance, in [8,21], REs are created as conjunctions of atoms, while [27] presents an approach that discovers more complex REs represented in description logics that can involve the universal quantifier. To ensure efficiency and reduce the search space, some of these methods focus on the minimality of the expression they discover and others on predicate preferences [16]. However, most of these methods are neither able to scale to large knowledge graphs such as YAGO or DBpedia with millions of instances nor are suited for the data linking task.

This paper embarks on automatically discovering REs for each entity within a class of a knowledge graph. These REs are conjunctions of atoms, e.g., $isPresident(x) \wedge isPoliticianOf(x, \texttt{\#USA}) \wedge bornIn(x, \texttt{\#Hawaii})$[1], and can also contain existentially quantified variables, e.g., $isPresident(x) \wedge marriedTo(x, y) \wedge hasName(y, \text{“}Michelle\text{”})$. Such conjunctions of atoms are not all relevant when they are exploited in a data linking task. As knowledge graphs are built independently and autonomously, individual IRIs are rarely reused in different knowledge graphs. This is the reason why a referring expression that involves a specified IRI may not be useful for the task of linking instances. Additionally, since data are usually incomplete and generally several referring expressions can be associated with an individual, to foster the utility of REs, it is preferable to diversify the sets of properties that are involved in the referring expressions of a given individual.

In order to reduce the enormous search space of referring expressions, our approach relies on defining types of graph patterns and quality measures that focus on REs that are more suitable in a data linking task. Moreover, we direct our attention to REs that cannot be found by instantiating the keys. As a reminder, the keys of a class are sets of properties whose values can uniquely identify one entity of that class. Hence, if the properties for the keys are instantiated, they can each be considered as a referring expression. For instance, take the class “book” and imagine that ISBN is key to this class. If we instantiate the books with their corresponding ISBNs, we can be sure to find them each uniquely. Recent approaches in the literature can efficiently discover keys in knowledge graphs [28,30,31], some of which do so by first finding the maximal non-keys [30,31]. Hence, our proposed RE-miner algorithm is based on the search space defined by these non-keys. Furthermore, we use the discovered REs in a data linking task and evaluate our approach on three benchmark datasets.

More precisely, our contributions are as follows:

- Defining graph patterns and several quality criteria that set forth REs, potentially relevant for data linking, discovered by our algorithm.
- Proposing an efficient algorithm, RE-miner, that computes complementary REs with regards to those REs that correspond to instantiated OWL2 keys.
- An extensive set of experiments showing that: (i) the approach scales to datasets consisting of millions of facts; (ii) the discovered REs, when used in a data linking task, can significantly increase the recall when compared to other approaches.

The remainder of this paper is organized as follows. We discuss the related work in Sect. 2. Section 3 details the formal problem statement. Section 4

[1] `#USA` and `#Hawaii` are IRIs (Internationalized Resource Identifier) that refer to the country USA and to the state of Hawaii, respectively.

describes the RE-miner algorithm, and Sect. 5 outlines how REs can be used for the data linking task. Section 6 shows our quantitative and data linking experiments, and finally, Sect. 7 concludes the paper.

2 Related Work

Both the generation of referring expressions and link discovery has been an area of active research in past years. The task of generating referring expressions, also known as REG, finds applications in fundamental fields such as natural language generation, text summarization, and query generation. Link discovery is rooted in record linkage, deduplication, and data integration. Existing methods for link discovery alleviate this task by matching the schema, instances, or both. In this paper, we focus on rule-based instance matching.

Referring Expression Generation. Robert Dale is first to frame REG as the problem of determining the properties that must be used to identify an entity [7]. The *Full Brevity* algorithm, outputs the shortest possible description by incrementally testing all combinations of properties to find the RE for the target. Later, acknowledging that finding the shortest RE is NP-hard, Dale approximated *Full Brevity* into a greedy algorithm that generates a RE by iteratively adding to an empty expression the property with the most discriminative power [8]. This algorithm does not necessarily produce the shortest RE but is much more efficient than *Full Brevity*.

As much as the length of a RE is important (people tend to prefer short ones), at times, a slightly longer and more informative RE is preferable. The *Incremental Algorithm* adds properties to an expression based on a preference order and does not necessarily produce the shortest RE [9]. Although logic optimization techniques are used to shorten the resulting REs, in some cases this algorithm might produce overly lengthy REs; a problem addressed in further research [17,19]. Incremental algorithms do not lend themselves well to generating *relational REs* [22], that identify a target through a relation to another entity (e.g., "the dog near the house"). Relational REs are best modelled with a graph (the *scene graph*), where relations between entities are represented as edges that link the corresponding nodes; the generation of referring expressions reduces to searching a subgraph (the *description graph*) that uniquely identifies the target [21]. Croitoru and van Deemter take this graph approach a step further and propose the use of *conceptual graphs*, a logic-based knowledge representation model that enriches the *factual knowledge* with *ontological knowledge* (i.e., background knowledge, e.g., "a cup is a vessel") [6]. The ontological knowledge is particularly useful to perform automatic inference. Other approaches turn to description logics as an alternative knowledge representation model [2,26]. Similar to conceptual graphs, description logics can model background knowledge and apply reasoning.

Nevertheless, the approaches discussed earlier have difficulties scaling to today's knowledge graphs. Galárraga et al. introduced an algorithm named

REMI that mines intuitive REs from a knowledge base [16]. The intuitiveness of a RE is computed with the Kolmogorov distance as a trade-off between its length and the use of properties which people are familiar with (e.g., Paris is better described as the capital of France than the birth place of Voltaire). REMI represents expressions that describe a target entity x in a tree that represents conjunctions of atoms (e.g., cityIn(x,y) $\wedge$ officialLanguage(y, z) $\wedge$ langFamily(z, Romance)) and identifies the RE that has the least cost in terms of intuitiveness through a depth first search with backtracking. Our proposed *RE-miner* algorithm differs from the previous ones by addressing some of the challenges in the field. It discovers not just one, but all referring expressions complementary to those that can be obtained by instantiating key properties. Moreover, to the best of our knowledge, this is the first work that exerts REs that are useful for linking instances of two knowledge graphs.

Link Discovery. Schema matching is the task of deriving alignments between classes and relation in two different knowledge graphs [11]. Instance matching, or data linking, is the ability to determine – with a certain degree of confidence – that two individuals refer to the same real-world object [15]. Many different approaches for schema matching and instance matching have been proposed [11, 25]. These link discovery approaches can fall into 3 different categories regarding how they include schema and instance matching in their workflow.

Some systems only focus on instance matching. Among these systems, some depend on declared linkage rules that can be used to logically infer identity links [1,12] while others compute a similarity score thanks to complex rules that can involve simple similarity measures and aggregation functions, like LIMES or Silk [24,34]. Such rules are generally based on sets of discriminative properties or more complex graph patterns, such as OWL2 keys, and schema mappings. Since such properties are not so easy to specify, some approaches aim to discover discriminative properties using one or several knowledge graphs, assuming that the mappings are known [30–32], or allowing to discover keys that involve property mappings [4]. Other approaches can efficiently discover linkage points which are related property paths sharing values between heterogeneous data sources. Indeed, it has been shown that such properties can enhance the performance of linkage algorithms [18]. However, the efficiency of such rule-based approaches is strongly related to the quality of the discriminative properties and to the proportion of instances that can be covered by them. Some approaches have defined the quality of a key with respect to a specified linking task. In [32], keys obtained in two datasets can be merged to generate valid keys on both datasets, while in [13], keys that involve syntactically similar literals in both datasets are chosen. Nevertheless, no approach has been defined to discover and exploit linkage rules that are based on REs. Other systems only focus on schema matching. These systems usually exploit terminological similarities, structural similarities, similarity of instances, external resources, or logical axioms to discover more or less complex schema mappings (e.g., [5,10]).

Finally, some systems do instance and schema matching in their ontology matching processes (e.g., [14,20,29,33]). Some of these systems like PARIS and ILIADS perform interleaved schema matching and instance matching in iterations where the mappings from one task help refine those of the other task [29,33].

In this paper, we focus on the instance matching task, and more precisely, we aim to discover REs that can be useful for data linking. We assume that a link discovery approach has provided a subset of class and property mappings. It's worth mentioning that in data linking approaches where schema matching is to be known, the mappings do not necessarily have to be the complete set of mappings between properties and classes, and very simple approaches can be used. For instance, in the Knowledge Graph track of OAEI 2019[2], the baseline solution adopting a terminological approach, and using only schema labels, has achieved an F-measure of 0.79 for mapping properties.

3 Problem Statement

Knowledge Graph. A knowledge graph $\mathcal{G}$ is defined by a couple $(\mathcal{O}, \mathcal{F})$ where: the ontology $\mathcal{O} = (\mathcal{C}, \mathcal{DP}, \mathcal{OP}, \mathcal{A})$ is defined by a set of classes $\mathcal{C}$, a set of *owl:DataTypeProperty* $\mathcal{DP}$, a set of *owl:ObjectProperty* $\mathcal{OP}$, and a set of axioms $\mathcal{A}$; $\mathcal{F}$ is a collection of triples (subject, property, object) $\in (\mathcal{I} \cup \mathcal{B}) \times (\mathcal{OP} \cup \mathcal{DP}) \times (\mathcal{I} \cup \mathcal{C} \cup \mathcal{L} \cup \mathcal{B})$[3] where $\mathcal{I}$ is a set of individuals, $\mathcal{B}$ is a set of blank nodes, and $\mathcal{L}$ is a set of Literals.

In this work, we consider referring expressions that are valid for an individual u of a class $\mathcal{C}$ in a knowledge graph G, that are defined as follows:

Definition 1 ***(Referring Expression).*** *A referring expression, denoted by $RE_k(u)$ for the k^{th} RE of a given individual u, of a class C can be expressed by the following first order logic formula:*

$$C(x) \bigwedge_{p_i \in \mathcal{OP} \cup \mathcal{DP}} p_i(w, y)$$

such that the formula, existentially closed, is restricted to those conjunctions of atoms that form a connected graph pattern rooted at x with the leaves being either an individual in $\mathcal{I}$ or a literal in $\mathcal{L}$, and the other nodes being variables.

Definition 2 ***(Referring Expression Validity).*** *A referring expression $RE_k(u)$ is valid in a dataset D if it holds when x is instantiated by u and does not hold for any other individual $v \neq u$ of $\mathcal{C}$ in D.*

[2] http://ceur-ws.org/Vol-2536/oaei19_paper0.pdf.
[3] We do not consider blank nodes in this work.

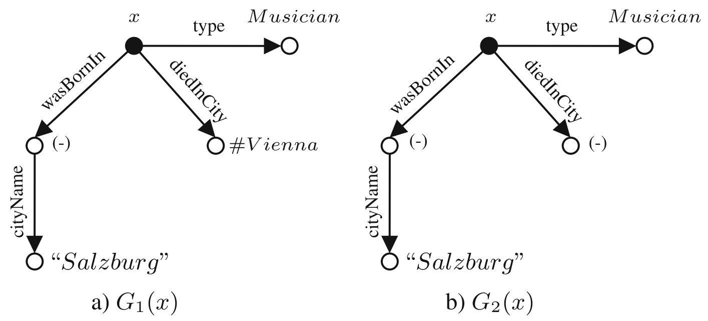


Fig. 1. Two graph patterns. $G_1(x)$ is compliant with Definition 1 and $G_2(x)$ is not.

Example. Two graph patterns $G_1(x)$ and $G_2(x)$ are shown in Fig. 1 where $(-)$ indicates a variable. $G_1(x)$ is compliant with Definition 1 and is a valid referring expression for Mozart: a **musician** who was born in a city named Salzburg and who died in Vienna. $Musician(x) \wedge wasBornIn(x, c) \wedge cityName(c, \text{“}Salzburg\text{”}) \wedge diedInCity(x, \#Vienna)$ Nevertheless, $G_2(x)$ is not compliant with Definition 1, since in this work we do not consider referring expressions that include variables appearing in the leaves of the graph pattern; hence, the following cannot be discovered as a RE.

$$Musician(x) \wedge wasBornIn(x, c) \wedge cityName(c, \text{“}Salzburg\text{”}) \wedge diedInCity(x, z)$$

Propelled by data linking, we aim to discover **minimal** referring expressions; that are the simplest graph patterns allowing to distinguish one individual from all the others.

Definition 3 ***(Referring Expression Minimality).*** *A referring expression $RE_k(u)$ is minimal iff:*

$$\nexists\ RE_j(u)\ s.t.\ (RE_k(u) \cup \mathcal{F} \cup \mathcal{A}) \models RE_j(u)$$

To focus on REs that are of interest to link data when datasets are incomplete, we exploit various properties while limiting the number of REs and their complexity. Hence, we do not construct REs that involve different instantiations of the same property, and we only consider **diversified** REs, simply meaning that when a valued property appears in a RE for an individual, it cannot reappear in another RE for the same individual having more atoms. This should not be confused with the notion of minimality.

Example. Take the valid $RE_1(u)$ for the film *Ocean's Eleven*: $Film(x) \wedge hasActor(x, \#George\_Clooney) \wedge wasCreatedOnYear(x, \text{“2001”})$. Then the following $RE_2(u)$, although valid for this movie and minimal, will not be discovered. $Film(x) \wedge hasActor(x, \#Julia\_Roberts) \wedge wasCreatedOnYear(x, \text{“2001”}) \wedge editedBy(x, \#Stephen\_Mirrione)$. Because $RE_2(u)$ is not diversified; since it has more atoms than $RE_1(u)$ while sharing the subgraph pattern $wasCreatedOnYear(x, \text{“2001”})$ with it.

Definition 4 ***(Diversified Referring Expression).*** *A referring expression* $RE_i(u)$ *is* **diversified** *if there is no* $RE_j(u)$ *with fewer number of atoms that contains a subgraph* $p1(x, t_1) \wedge \ldots \wedge p_i(t_{i-1}, t_i) \wedge p_m(t_{m-1}, v_m)$ *of* $RE_i(u)$*, where* $v_m \in \mathcal{L} \cup \mathcal{I}$*.*

Additionally, in the data linking task, one might argue that graph patterns that involve mostly IRIs of individuals are not relevant. Indeed, individuals that are described in two knowledge graphs are rarely represented with the same IRI. This is why we also consider Expanded REs that are not minimal but where the individuals' IRIs in a RE are replaced by a description constructed from instantiated key properties.

Definition 5 ***(Referring Expression Expansion).*** *The expansion* $exp(RE_k(u))$ *of a referring expression* $RE_k(u)$ *is a set of referring expressions in which, each leaf node* n_i *of* $RE_k(u)$ *that represents an individual i is replaced by an existential variable* x_j*. These variables are recursively expanded by a subgraph G rooted by* n_i *representing one possible instantiation of a key K for the class typing i, such that* $exp(RE_k(u))$ *leads to a graph pattern whose leaves are only literals.*

Example. Consider the following referring expression for Marie Curie: $Scientist(x) \wedge wasBornOnYear(x, \text{“1867”}) \wedge isCitizenOf(x, \#Poland)$. We observe that Poland is a leaf node representing an individual, thus we can expand it by creating keys for the class country (range of the property $isCitizenOf$). Suppose it has two sets of keys, namely $\{hasName\}$ and $\{hasArea, isLocatedIn\}$. We obtain the following RE when $\#Poland$ is replaced by an instantiation of the first key set: $Scientist(x) \wedge wasBornOnYear(x, \text{“1867”}) \wedge isCitizenOf(x, y) \wedge hasName(y, \text{“}Poland\text{”})$. And the following RE, when using the second set of key for country, and subsequently expanding $\#europe$ by considering $\{hasName\}$ as a key for the class location (range of the property $isLocatedIn$): $Scientist(x) \wedge wasBornOnYear(x, \text{“1867”}) \wedge isCitizenOf(x, y) \wedge hasArea(y, \text{“}312,696 < km2 > \text{”}) \wedge isLocatedIn(y, z) \wedge hasName(z, \text{“}europe\text{”})$.

4 Referring Expression Generation Approach

In this section, we present an approach to automatically discover minimal and diversified REs for each instance within a class of a knowledge graph.

Our generation approach is composed of two successive steps. We first generate the set of minimal and diversified REs for each instance using the algorithm *RE-miner*. Since recent approaches have been developed to discover keys, we focus on complementary REs that do not represent an instantiated key. In a second step, we can generate the expansion of each RE.

4.1 Keys, Non-keys and Complementary REs

In a knowledge graph, an OWL2 key can be defined as follows:

Definition 6 ***(Key).*** *A key $\{p_1, \ldots, p_n\}$ for a class C expresses that:*

$$\forall x \forall y \forall z_1 ... z_n (C(x) \wedge C(y) \wedge \bigwedge_{i=1}^{n} (p_i(x, z_i) \wedge p_i(y, z_i)) \rightarrow \; x = y)$$

By definition, each instantiation of the key properties for a class $\mathcal{C}$ will uniquely identify an individual present in $\mathcal{C}$. This instantiation will potentially yield many REs. Nevertheless, this does not represent the complete set of possible minimal REs that can be discovered. We are thus interested in enriching this set with those REs that only involve non-key properties. To this end, we will only exploit sets of properties that are included in one of the maximal non-keys of class $\mathcal{C}$ to construct complementary REs.

Definition 7 ***(Maximal Non-Key).*** *A maximal non-key for a class $\mathcal{C}$ in a knowledge graph $\mathcal{G}$ is a set of properties P such that P is not a key, but the addition of any property to P makes it a key for that class.*

4.2 RE-miner Algorithm

We outline the procedure of mining the complete set of complementary, minimal, and diversified REs in Algorithm 1. To retain a reasonable search space and to prevent the REs from becoming too complex for data linking, the depth of the aimed REs is restricted to 2. Nevertheless, this restriction can be dropped by applying a recursive adaptation of the function $existentialRE(\mathcal{G}, RE_{new})$ (see line 10 of Algorithm 1). The algorithm takes as input a knowledge graph $\mathcal{G}$, a class $\mathcal{C}$, and a Boolean E which is set to True if we aim to mine REs at depth 2 (i.e., REs that contain at least an existential quantifier). If the E is False, the REs will not contain any existential quantifiers.

To generate the set of REs for a given class $\mathcal{C}$ of knowledge graph $\mathcal{G}$, we first create the dataset for that class. This dataset will serve as the search space SS (line 1), and is created by keeping all the facts (s, p, o) in $\mathcal{G}$ whose subjects s belong to $\mathcal{C}$. Then using SAKey [30], a key discovery approach, we create the maximal non-key sets NK of the dataset $\mathcal{SS}$ (line 2). We build the powerset, excluding the empty set, of each set in NK, and group them based on their cardinality (line 3). For instance, imagine that NK $= \{\{p_1, p_2\}, \{p_3, p_4, p_5\}\}$; then level 1, includes subsets of cardinality 1, composed of $\{\{p_1\}, \{p_2\}, \{p_3\}, \{p_4\}, \{p_5\}\}$, level 2 composed of $\{\{p_1, p_2\}, \{p_3, p_4\}, \{p_3, p_5\}, \{p_4, p_5\}\}$, and level 3 containing $\{\{p_3, p_4, p_5\}\}$.

Since we desire to find minimal REs, the algorithm proceeds level by level (line 4), starting from level 1, which results in REs containing only one atom. To mine REs at level l, we take one set of properties P within that level at a time (line 6). The algorithm generates subgraph patterns from the search space with instantiated properties P (e.g. $p_1(x, v)$ for level 1 and $p_1(x, y) \wedge p_2(x, z)$ for level 2) as candidate expressions (line 7). We keep the valid REs, among the candidates in RE_{new} (line 8); these expressions are compliant with Definition 2 and hence uniquely identify an individual of the class $\mathcal{C}$. If we also aim to find REs of depth 2, i.e., if E is True, the *existentialRE* algorithm detailed in 2 is called on the knowledge graph $\mathcal{G}$ and REs found at level l with properties P (line 10). We add all the recently discovered REs composed of properties P, to RE_{level} (line 12), and reiterate until all sets of properties at this level are covered. Then RE_{level} is added to the resulting REs (line 14), and all the facts (s, p, o) involved in these referring expressions are removed from the search space; to ensure both minimality and diversity (line 14).

Algorithm 1: RE-miner

```
   Input: A knowledge graph: $\mathcal{G}$, a class: $\mathcal{C}$, a Boolean:$E$
   Output: The set of minimal REs for instances of type $\mathcal{C}$: $RE_{set}$
1  $\mathcal{SS} \leftarrow$ createData ($\mathcal{G}$, $\mathcal{C}$) // serves as the search space
2  NK $\leftarrow$ generateNK($\mathcal{SS}$)
3  createRankedPowerset(NK) // Dictionary $level$ to $props$
4  for level = 1 to $|longestNonKey|$ do
5      $RE_{level} = \emptyset$
6      foreach P $\in$ props.level do
7          $RE_{candidates} \leftarrow$ constructSubgraphs (SS, P)
8          $RE_{new}$ = validSubgraphs($RE_{candidates}$)
9          if E = True then
10             $RE_{new}$.add(existentialRE($\mathcal{G}$), $RE_{new}$)
11         end
12         $RE_{level}$.add($RE_{new}$)
13     end
14     $RE_{set}$.add($RE_{level}$)
15     SS $\leftarrow$ suppressFacts (SS, $RE_{level}$) // reduce the search space to
       preserve minimality and diversity
16 end
17 return $RE_{set}$
```

As stated in Algorithm 1, we can mine more complex REs containing the existential quantifier where the depth of the subgraph patterns will be 2. To do so, all the REs at a level l composed of properties P, are passed to the *existentialRE* algorithm. The output of this algorithm will be a set of referring expressions, each containing one or more existential quantifiers. The details are sketched in Algorithm 2.

This algorithm starts by keeping a copy of RE_{new} in the RE existential candidate set (line 1). This candidate set will grow as the algorithm proceeds. We iterate over each property p in P (line 2) and get its range using $\mathcal{G}$'s schema (line 3). Let the range of p be the class $\mathcal{C}'$. If not all the instances of the class $\mathcal{C}'$ are literals, we can expand the leaf node with a subgraph pattern; else, we move on to the next property (line 4). These subgraph patterns should be chosen such that when replaced, the whole pattern remains a valid RE. To this end, using RE-miner, Algorithm 1, we construct REs of depth 1, by setting E to False, for the class $\mathcal{C}'$ (line 5). It should be noted that to have the complete and minimal resulting $RE_{existential}$, when creating the dataset for $\mathcal{C}'$ (line 1 of Algorithm 1), we only keep those instances $o \in \mathcal{C}'$ that are involved in such facts $(s, p, o)|s \in \mathcal{C}$. The resulting REs from RE-miner are kept in *inducedSubgraphs* (line 5). Then for each referring expression in the candidate set (line 6), we replace the applicable node, based on p, with the appropriate subgraph in *inducedSubgraphs* (line 7). In the end, we remove the REs of depth one RE_{new} we had added initially to the existential candidate set (line 10), and return the unique subgraphs, which are valid referring expressions having depth 2 (line 11).

Algorithm 2: existentialRE

```
Input: A knowledge graph: $\mathcal{G}$, a set of REs having properties $P$: $RE_{new}$
Output: The set of REs with existential variable : $RE_{existential}$
1  $RE_{existentialCands} = RE_{new}$.copy()
2  foreach p ∈ P do
3      $\mathcal{C}' \leftarrow$ getRange($\mathcal{G}$, p)
4      if exists an instance of $\mathcal{C}' \in \mathcal{I}$ then
5          inducedSubgraphs ← RE-miner($\mathcal{G}$, $\mathcal{C}'$, E= False)
6          foreach RE ∈ $RE_{existentialCands}$ do
7              $RE_{existentialCands}$.add(replaceNodeSubgraph(RE, p,
               inducedSubgraphs))
8          end
9      end
10     $RE_{existentialCands}$.remove($RE_{new}$)
11     $RE_{existential}$ = validSubgraphs($RE_{existentialCands}$)
12 end
13 return $RE_{existential}$
```

4.3 RE Expansion

We have developed a post-processing step to obtain the expansion of referring expressions discovered by *RE-miner*, as defined in Definition 5. To do so, given a *RE*, for every IRI u appearing in the leaves of *RE*, we exploit the set of minimal keys of the class u belongs to and expand the *RE* by instantiating properties of every minimal key. If the keys involve object properties, this step is re-performed recursively on the generated IRIs until either reaches a maximum depth d specified beforehand, or the leaves only correspond to literal values. The sets of minimal keys are generated each time a new class is considered, and are stored on the disk so that there's no need to regenerate them to expand another RE. The graphs resulting from the expansion of one IRI are then combined to other IRI expansions to finally construct expanded REs.

5 Data Linking with REs

Each $RE(u)$ declared for an individual u in the source dataset D_1 can be expressed by a linking rule as follows:

$$\forall x RE(x) \rightarrow sameAs(x, u)$$

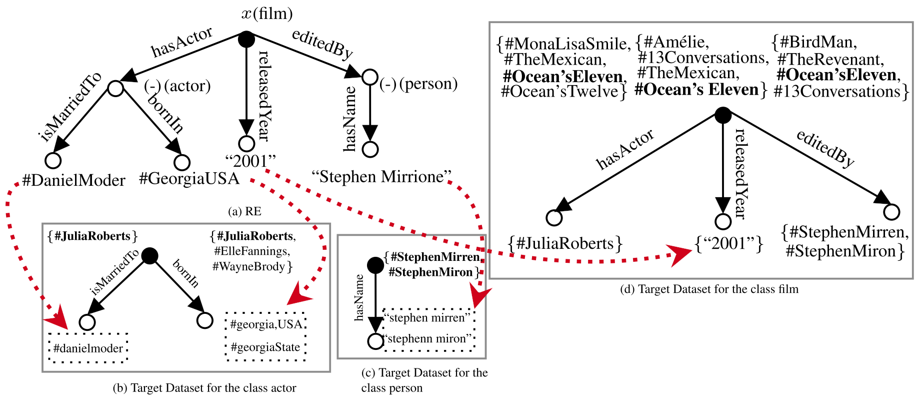
where $RE(x)$ can be rewritten using the classes and properties of a target dataset D_2 at the linking step. These rules can be represented in SWRL[4]. Hence, to discover identity links between individuals described in two given datasets, we focus on referring expressions that only involve mapped properties, and individuals belonging to classes that have been aligned. Such mappings can be obtained using existing schema matching techniques discussed in Sect. 2.

The linkage rules introduced above can be used either logically to deduce identity links, or by linking tools where simple similarity measures and aggregation functions can be introduced. Since available existing linking tools like [23,24,34] do not consider such intricate graph patterns (i.e., not just paths of properties), we have developed a simple bottom-up approach explained in Fig. 2, where normalizations or classical similarity measures can be declared and applied to datatype properties.

We consider a data linking problem between a source dataset in which the REs are discovered and other target datasets that have a non-empty set of properties mapped to the source dataset (that can be obtained using ontology alignment tools [11]). The linking process is comprised of exploiting for every individual u in the source dataset, the set of distinct $RE(u)$s to find all the individuals x in the target datasets that check $RE(u)$. When a RE is discovered in the source dataset, it cannot necessarily be assumed valid for other target datasets. Indeed, even if the source dataset is voluminous, several distinct individuals that can instantiate a RE may exist in the other dataset. Theoretically, when the unique name assumption (UNA) is fulfilled, only one $sameAs(u, x)$ link can be found for a given $RE(u)$ in the target dataset. If this is the case, the quality of $RE(u)$ has to be weakened. Therefore, we assign to every $RE(u)$ a confidence degree inverse proportional to the number of distinct links the $RE(u)$ finds.

To pick the best identity link(s), we adopt a voting strategy that assigns a weight to each link. This weight is the sum of the RE confidence degrees that can be instantiated to generate the link. Eventually, the instance(s) associated with the link(s) having the highest score is selected. The RE confidence degrees can then be stored and updated when another data linking task is performed on the source dataset.

[4] https://www.w3.org/Submission/SWRL/.



x(film)
hasActor
editedBy
releasedYear
(-) (actor)
(-) (person)
isMarriedTo
bornIn
hasName
#DanielModer
#GeorgiaUSA
"2001"
"Stephen Mirrione"
(a) RE
{#JuliaRoberts}
{#JuliaRoberts, #ElleFannings, #WayneBrody}
#danielmoder
#georgia,USA
#georgiaState
(b) Target Dataset for the class actor
{#StephenMirren, #StephenMiron}
"stephen mirren"
"stephenn miron"
(c) Target Dataset for the class person
{#MonaLisaSmile, #TheMexican, #Ocean'sEleven, #Ocean'sTwelve}
{#Amélie, #13Conversations, #TheMexican, #Ocean's Eleven}
{#BirdMan, #TheRevenant, #Ocean'sEleven, #13Conversations}
{#JuliaRoberts}
{"2001"}
{#StephenMirren, #StephenMiron}
(d) Target Dataset for the class film

Fig. 2. Bottom-up linking with $RE_k(u)$. (a) is a valid RE for the film *Ocean's Eleven* in the source dataset. We adopt a bottom-up approach where the traversal begins from leaf nodes. (b,c) For each of the RE's leaf nodes n_i in (x, p, n_i), verifies the matches (s, p, o) in the target dataset for the class of x; such that o has a high similarity with n_i. (d) The matches are intersected at the internal nodes, until those intersections at the root are reported as the links.

6 Experimental Evaluation

To evaluate *RE-miner*, we conducted two series of experiments. The first is quantitative that is dedicated to studying the scalability of the proposed algorithm. In the second series of experiments, we explored how REs can contribute to the data linking task.

All experiments are run on a single machine with processor 2.7 GHz, 8 cores, and 16 GB of RAM that runs Mac OS X 10.13. The source code of our approach is publicly available[5].

6.1 Datasets

We summarize the characteristics of the 3 datasets on which we did our experiments.

DBpedia-YAGO.[6] We use 10 different classes of YAGO and DBpedia knowledge graphs. The data for these 10 classes are the same data used in VICKEY [31], where the properties of the two knowledge graphs have been aligned manually. Moreover, the properties of YAGO have been rewritten using their DBpedia counterparts. This dataset contains 206,736 ground truth entity pairs.

[5] https://github.com/iswc2020/REGeneratipnAndLinking.
[6] https://github.com/lgalarra/vickey.

IM@OAEI2019.[7] We use the Sandbox SPIMBENCH dataset of the instance matching track at OAEI 2019, its gold standard is available and consists of 300 entity pairs. This dataset is composed of a Tbox and an Abox for each of the source and target ontologies. The goal of the task is to match instances describing the same *Creative Work*, which can be a news item, blog post, or a program.

IM@OAEI2011.[8] We use IIMB (ISLab Instance Matching Benchmark) dataset which consists of a set of interlinking tasks used by the instance matching track of OAEI 2011. The source dataset (File 000) describes movies, locations, actors, etc. Files 001 to 080 are generated by applying several transformations to the source dataset. For each of these files, a gold standard containing around 12.3k identity links has been provided.

6.2 Quantitative Results

Here, we study the scalability of our approach and will report the number of REs found on average for each individual in the considered knowledge graph, as well as the average number of nodes in the graphs representing the discovered REs; first at depth 1 and then at depth 2.

Initially, we run *RE-miner* on the 10 classes of YAGO at depth 1, i.e., without allowing for any existential variables. Table 1 details the characteristics of each of these 10 classes. Furthermore, this table shows the number of discovered referring expressions for each class as well as their run time. We can observe that the process takes less than 2 min for all classes except for organization, which took more than 3 hours to complete[9]. On average, there are less than 7 REs per individual for all classes, except for the most voluminous class organization with almost 158 REs for each individual. Without having limited the number of discovered REs' atoms, these expressions do not tend to be complex; the maximum number of atoms among all 10 classes is 4 and on average, each RE has 2 atoms or less[10].

Example. The following examples translated to natural language, have been chosen among the REs of depth 1: (i) Yellow Submarine is an album created by the Beatles on date 1966-05-26. (ii) MIT university's motto is *mind and hand.* (iii) Charles Louis Alphonse Laveran is a scientist who was born on year 1845 in Paris, graduated from university of Strasbourg and has won the Nobel prize in Physiology or Medicine.

Similar results have been obtained on OAEI2011 and OAEI2019 datasets with 1.19 and 3.75 average atoms and a maximum of 4 and 8 atoms, respectively (see Table 4).

[7] https://project-hobbit.eu/challenges/om2019/.
[8] http://oaei.ontologymatching.org/2011/instance/index.html.
[9] Note that the non-key sets had been computed beforehand in all experiments.
[10] Note that the rdf: type properties are not being counted in number of atoms.

Table 1. Class statistics, number of non-keys (#NKs), number of discovered REs at depth 1 (#REs), runtime, and size of the REs

Class	#triples	#instances	#properties	#NKs	#REs	runtime	max#atoms	avg#atoms
Museum	81.6k	21.1k	7	5	53.5k	2.6 s	3	1.23
Mountain	116.7k	32.9k	6	4	59.2k	1.4 s	3	1.28
Book	123.6k	41.8k	7	6	66.3k	3.5 s	3	1.27
University	131.8k	23.3k	9	9	161.8k	17.7 s	3	1.62
Scientist	335.6k	93.1k	18	92	309.9k	64.0 s	4	1.58
Album	381.1k	137.1k	5	2	212.1k	14.7 s	3	1.30
Actor	514.7k	108.4k	16	69	725.6k	95.1 s	3	1.74
Film	533.5k	123.9k	9	7	690.9k	102.3 s	4	1.77
City	1.1M	83.5k	17	29	1.2M	109.7 s	3	1.23
Organization	2.2M	430.3k	17	43	68.3M	3.48 h	4	2.05

To show how many more REs can be found at depth 2 (i.e., REs that contain at least an existential quantifier), we run *RE-miner* with the Boolean E set to True, on the 3 classes of YAGO having the least number of referring expressions at depth 1. As described in Sect. 4.2, the algorithm should create the dataset for the class the variable belongs to. To this end, we use instances of this class and all its sub-classes in the non-saturated (i.e., no OWL2 entailment rule has been applied) YAGO version 3.1[11] to ensure having a dataset with at least 1000 instances whenever possible.

Table 2 shows that as expected, many additional REs can be generated at depth 2. However, the proportion of REs that have only literals values at leaf nodes is rather small, and we can use those REs for data linking. On average, these referring expressions have 2 atoms more than REs of depth 1.

Table 2. Number of additional REs detected at depth 2 (#REs), runtime, percentage of REs having only literals in leaf nodes, maximum and average number of atoms.

Class	#REs	runtime	%AllLiterals	max#atoms	avg#atoms
Mountain	150.8k	3006 s	13.4%	6	3.03
Museum	1.4M	3143 s	5.2%	5	3.57
Book	1.3M	1.2 h	2.50%	6	3.47

6.3 Data Linking

Here, we evaluate data linking on the 3 datasets, each time comparing the results with the previous works in the literature that used the same datasets. We study the advantage of using REs of depth 1 and 2, REs plus keys (i.e., the complete set of referring expressions), and expanded REs. For each of the datasets, we

[11] http://resources.mpi-inf.mpg.de/yago-naga/yago3.1/yago3.1_entire_tsv.7z.

compare the results with a baseline approach that picks random subgraph patterns (i.e., random expressions) and uses them for linking just as it is done with referring expressions. To be fair when comparing the random baseline results to that of REs for data linking, for each dataset, a) the number of generated random expressions is the same as that of discovered REs. b) the size of the random baseline's subgraphs comes from the same distribution as the discovered referring expressions. In other words, the same number of random subgraphs and referring expressions with n atoms exist. c) the results for the random baseline are averaged over three runs.

DBpedia-YAGO. We report our linking results on those 8 classes of this dataset for which other approaches had previously published results. We have used all REs of depth 1 whose statistics were delineated in Table 1, with strict string equality. The quality of linking results is reported in terms of precision, recall and F-measure, and is compared to the results of linking with keys (Ks), keys and conditional keys (Ks+CKs) reported in [31], ontological graph keys (OGK) reported in [23], and the random baseline (RBL). A conditional key is a valid key for a specified part of a class's instances [31]. Ontological graph keys defined in [23] are a variant of keys defined by a graph pattern extended by ontological pattern matching. Table 3 shows that RE-miner outperforms the other approaches in terms of recall and F-measure on all classes except book. More precisely, only using REs of depth 1, we can detect much more correct links without having a significant change to the precision. We also observe that the baseline solution – taking thousands to million of random subgraphs, depending on the dataset, and using them for linking – results in much lower scores than REs; showcasing that using the referring expressions discovered through RE-miner are indeed effective for linking.

Table 3. Linking results with keys (Ks), conditional keys and keys (Ks+CKs), ontological graph keys (OGK), random baseline (RBL), and REs of depth 1.

Class	Recall					Precision					F1				
	Ks	Ks+CKs	OGK	RBL	REs	Ks	Ks+CKs	OGK	RBL	REs	Ks	Ks+CKs	OGK	RBL	REs
Actor	0.27	0.60	0.66	0.19	**0.69**	0.99	0.99	1.00	0.37	0.99	0.43	0.75	0.79	0.25	**0.81**
Album	0.00	0.15	–	0.35	**0.65**	1.00	0.99	–	0.22	0.98	0.00	0.26	–	0.27	**0.78**
Book	0.03	0.13	**0.85**	0.12	0.80	1.00	0.99	0.97	0.38	0.98	0.06	0.23	**0.90**	0.18	0.88
Film	0.04	0.39	–	0.30	**0.73**	0.99	0.98	–	0.73	0.94	0.08	0.55	–	0.43	**0.82**
Mountain	0.00	0.29	–	0.05	**0.78**	1.00	0.99	–	0.08	0.99	0.00	0.45	–	0.06	**0.87**
Museum	0.00	0.29	0.42	0.20	**0.85**	1.00	0.99	0.99	0.34	0.99	0.00	0.45	0.58	0.25	**0.91**
Scientist	0.00	0.29	0.67	0.24	**0.70**	1.00	0.99	0.99	0.14	0.99	0.00	0.45	0.80	0.18	**0.82**
University	0.09	0.25	0.50	0.29	**0.68**	0.99	0.99	0.96	0.64	0.98	0.16	0.40	0.66	0.40	**0.80**

IM@OAEI2011. We first evaluate our data linking results on the entire IIMB dataset. IIMB is made of 13 different classes (e.g., person, actor, location, etc.); 5 of which are at the top of the ontology according to the schema. We create the saturated dataset for these 5 classes and discover all minimal and diversified REs of depth 1 and 2 on the source file 000. We use these REs to find identity links in each of the files 001 to 010. The accuracy measures reported in Table 4 are averaged over these 10 files to compare to the results of the Combinatorial Optimization for Data Integration (CODI) system [20], which reformulates the alignment problem as a maximum-a-posteriori optimization problem. We can observe that *RE-miner* outperforms CODI by a large margin of 12% to 7% in recall and F-measure. Also the baseline solution, which generates 1.3 M random expressions regardless of being RE or not, exhibits poor results. Similar to the previous datasets, this performance reassures us that RE-miner algorithm is indeed beneficial for linking. We also investigated the effects of using expanded RE and observed that it helps increase the recall by 5.1% on average, over REs of depth 1.

Moreover, we compare data linking results using the discovered REs, on the class Film, against data linking with keys reported in [3]. We obtained a high F-measure of 99% and gained about 70% increase in the recall.

Table 4. Class statistics and linking results of REs+keys, random baseline (RBL), and other systems compared to results with REs of depth 1 and 2 on IM@OAEI2011 dataset and the class Film of IM@OAEI2011, and with REs of depth 1 on IM@OAEI2019 dataset.

Dataset	#classes	#triples	#properties	#NKs	#REs	System	Precision	Recall	F-measure
IIMB OAEI 2011	13	87.3k	23	17	1.3M	REs	0.92	0.87	0.90
						REs+Ks	0.93	**0.88**	**0.91**
						CODI	**0.94**	0.76	0.84
						RBL	0.69	0.29	0.41
Film OAEI 2011	1	11.8k	13	4	1.2M	REs	0.99	**0.98**	**0.99**
						Ks	**1.00**	0.27	0.43
						REs+Ks	0.99	**0.98**	**0.99**
						RBL	0.89	0.03	0.06
SPIMBENCH OAEI 2019	1	6.2k	18	3	1.6k	REs	0.98	0.84	0.91
						REs+Ks	**0.99**	0.99	**0.99**
						Lily	0.84	**1.00**	0.91
						AML	0.83	0.89	0.86
						FTRLIM	0.85	**1.00**	0.92
						RBL	0.78	0.87	0.82

IM@OAEI2019. We report the linking results using the discovered REs of depth 1 for the *Creative Work* class on the source dataset of SPIMBENCH; as the datasets were not saturated, we could not mine and use REs of depth 2 for linking. We compare our results to the 3 systems with the best performances in the competition[12]: Lily, AML, and FTRLIM, as well as the baseline solution. Looking at Table 4, we observe that the random baseline approach is the least effective and that REs alone have comparable performance to the other 3

[12] http://ceur-ws.org/Vol-2536/oaei19_paper0.pdf.

systems. However, when combined with the instantiation of keys, resulting in the full set of REs, they outperform all other systems achieving an F-measure of 99%. The average confidence of the discovered links is 85.5%, whereas this number increases to 97.9% among the links that are picked through the voting strategy described in Sect. 5.

Relevancy of Diversity. We also performed another set of experiments to observe the effects discovering diversified REs brings to the data linking task. By modifying the RE-miner algorithm, we discovered all minimal REs on the same 3 classes of DBpedia-YAGO dataset presented in Table 2. We observed a considerable increase in the number of discovered REs (e.g., it almost doubled for the class book); whereas the recall and F-measure of the linking task either remained the same or slightly decreased (e.g., for the class book, it dropped by 2%). These results support that the use of diversity as a quality criterion for referring expressions proves beneficial in limiting the number of REs while preserving the quality of data linking.

To sum, we showed that using REs improves data linking results compared to previous works and the random baseline. The results were verified on different datasets containing classes with 5 to 23 properties and 300 to 137k instances.

7 Conclusion

In this paper, we proposed an approach that efficiently discovers referring expressions by reducing the search space thanks to the use of maximal non-keys. The generated REs are adapted to a data linking task through the notions of minimality and diversification and the post-processing step of expansion.

We showed that *RE-Miner* can scale to classes consisting of millions of triples and that the defined REs can significantly improve the performance of instance matching and increase the recall of rule-based data linking methods.

As future work, we aim to refine REs by virtue of data linking, whereby if a RE finds more than one match in the target dataset, we can deduce that some information had been missing in the source dataset and hence can add the new relevant facts to the source knowledge graph.

Acknowledgements. This work has been supported by the project PSPC AIDA: 2019-PSPC-09 funded by BPI-France.

References

1. Al-Bakri, M., Atencia, M., Lalande, S., Rousset, M.: Inferring same-as facts from linked data: an iterative import-by-query approach. In: Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, Texas, USA, pp. 9–15. AAAI Press (2015)
2. Areces, C., Koller, A., Striegnitz, K.: Referring expressions as formulas of description logic. In: Fifth International Natural Language Generation Conference, pp. 42–49. ACL (2008)

3. Atencia, M., et al.: Defining key semantics for the RDF datasets: experiments and evaluations. In: Hernandez, N., Jäschke, R., Croitoru, M. (eds.) ICCS 2014. LNCS (LNAI), vol. 8577, pp. 65–78. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-08389-6_7
4. Atencia, M., David, J., Euzenat, J., Napoli, A., Vizzini, J.: Link key candidate extraction with relational concept analysis. Discrete Appl. Math. **273**, 2–20 (2019)
5. Aumueller, D., Do, H.H., Massmann, S., Rahm, E.: Schema and ontology matching with coma++. In: Proceedings of the 2005 ACM SIGMOD International Conference on Management of Data, pp. 906–908. Association for Computing Machinery (2005)
6. Croitoru, M., Van Deemter, K.: A conceptual graph approach for the generation of referring expressions. In: IJCAI, pp. 2456–2461 (2007)
7. Dale, R.: Cooking up referring expressions. In: 27th Annual Meeting of the association for Computational Linguistics, pp. 68–75 (1989)
8. Dale, R.: Generating Referring Expressions: Constructing Descriptions in a Domain of Objects and Processes. The MIT Press, Cambridge (1992)
9. Dale, R., Reiter, E.: Computational interpretations of the Gricean maxims in the generation of referring expressions. Cogn. Sci. **19**(2), 233–263 (1995)
10. Doan, A., Domingos, P., Levy, A.: Learning source description for data integration. In: Proceedings of the Third International Workshop on the Web and Databases, in Conjunction with ACM PODS/SIGMOD 2000, pp. 81–86, January 2000
11. Euzenat, J., Shvaiko, P.: Ontology Matching, 2nd edn. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-38721-0
12. Fan, W., Fan, Z., Tian, C., Dong, X.L.: Keys for graphs. PVLDB **8**(12), 1590–1601 (2015)
13. Farah, H., Symeonidou, D., Todorov, K.: KeyRanker: automatic RDF key ranking for data linking. In: Proceedings of the Knowledge Capture Conference, K-CAP 2017, TX, USA, 4–6 December 2017, pp. 7:1–7:8. ACM (2017)
14. Faria, D., Pesquita, C., Santos, E., Palmonari, M., Cruz, I.F., Couto, F.M.: The AgreementMakerLight ontology matching system. In: Meersman, R., et al. (eds.) OTM 2013. LNCS, vol. 8185, pp. 527–541. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-41030-7_38
15. Ferrara, A., Nikolov, A., Scharffe, F.: Data linking for the semantic web. Int. J. Semant. Web Inf. Syst. (IJSWIS) **7**(3), 46–76 (2011)
16. Galárraga, L., Delaunay, J., Dessalles, J.: REMI: mining intuitive referring expressions on knowledge bases. In: Proceedings of the 23nd International Conference on Extending Database Technology, EDBT 2020, Denmark, pp. 387–390. OpenProceedings.org (2020)
17. Gardent, C.: Generating minimal definite descriptions. In: Proceedings of the 40th Annual Meeting on Association for Computational Linguistics, pp. 96–103. ACL (2002)
18. Hassanzadeh, O., et al.: Discovering linkage points over web data. Proc. VLDB Endow. **6**(6), 445–456 (2013)
19. Horacek, H.: On referring to sets of objects naturally. In: Belz, A., Evans, R., Piwek, P. (eds.) INLG 2004. LNCS (LNAI), vol. 3123, pp. 70–79. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-27823-8_8
20. Huber, J., Sztyler, T., Noessner, J., Meilicke, C.: CODI: combinatorial optimization for data integration - results for OAEI 2011. In: Proceedings of the 6th International Conference on Ontology Matching, vol. 814. pp. 134–141. CEUR-WS.org (2011)
21. Krahmer, E., Erk, S.V., Verleg, A.: Graph-based generation of referring expressions. Comput. Linguist. **29**(1), 53–72 (2003)

22. Krahmer, E., Van Deemter, K.: Computational generation of referring expressions: a survey. Comput. Linguist. **38**(1), 173–218 (2012)
23. Ma, H., Alipourlangouri, M., Wu, Y., Chiang, F., Pi, J.: Ontology-based entity matching in attributed graphs. Proc. VLDB Endow. **12**(10), 1195–1207 (2019)
24. Ngonga Ngomo, A.C., Auer, S.: Limes - a time-efficient approach for large-scale link discovery on the web of data. In: IJCAI 2011, Proceedings of the 22nd International Joint Conference on Artificial Intelligence, Barcelona, Catalonia, Spain, pp. 2312–2317, January 2011
25. Rahm, E., Bernstein, P.: A survey of approaches to automatic schema matching. VLDB J. **10**, 334–350 (2001)
26. Ren, Y., Van Deemter, K., Pan, J.Z.: Charting the potential of description logic for the generation of referring expressions. In: Proceedings of the 6th International Natural Language Generation Conference, pp. 115–123. ACL (2010)
27. Ren, Y., Van Deemter, K., Pan, J.Z.: Generating referring expressions with OWL2. In: 23rd International Workshop on Description Logics DL2010, p. 420 (2010)
28. Soru, T., Marx, E., Ngonga Ngomo, A.C.: Rocker: a refinement operator for key discovery. In: Proceedings of the 24th International Conference on WWW, pp. 1025–1033 (2015)
29. Suchanek, F.M., Abiteboul, S., Senellart, P.: Paris: probabilistic alignment of relations, instances, and schema. Proc. VLDB Endow. **5**(3), 157–168 (2011)
30. Symeonidou, D., Armant, V., Pernelle, N., Saïs, F.: SAKey: scalable almost key discovery in RDF data. In: Mika, P., et al. (eds.) ISWC 2014. LNCS, vol. 8796, pp. 33–49. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-11964-9_3
31. Symeonidou, D., Galárraga, L., Pernelle, N., Saïs, F., Suchanek, F.: VICKEY: mining conditional keys on knowledge bases. In: d'Amato, C., et al. (eds.) ISWC 2017. LNCS, vol. 10587, pp. 661–677. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-68288-4_39
32. Symeonidou, D., Pernelle, N., Saïs, F.: KD2R: a key discovery method for semantic reference reconciliation. In: Meersman, R., Dillon, T., Herrero, P. (eds.) OTM 2011. LNCS, vol. 7046, pp. 392–401. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-25126-9_51
33. Udrea, O., Getoor, L., Miller, R.J.: Leveraging data and structure in ontology integration. In: Proceedings of the 2007 ACM SIGMOD International Conference on Management of Data, SIGMOD 2007, pp. 449–460. Association for Computing Machinery (2007)
34. Volz, J., Bizer, C., Gaedke, M., Kobilarov, G.: Silk - a link discovery framework for the web of data. In: LDOW (2009)