Defining Key Semantics for the RDF Datasets: Experiments and evaluations

Manuel Atencia[2,4], Michel Chein[1,4], Madalina Croitoru[1,4], Jérôme David[2,4], Michel Leclère[1,4], Nathalie Pernelle[3], Fatiha Saïs[3], Francois Scharffe[1], Danai Symeonidou[3]

[1]Univ. Montpellier 2, LIRMM, France
{michel.chein, madalina.croitoru,michel.leclere,francois.scharffe}@lirmm.fr
[2]Univ. Grenoble Alpes, LIG, France
{manuel.atencia, jerome.david}@inria.fr
[3]Univ. Paris Sud, LRI, France
{nathalie.pernelle, fatiha.sais}@lri.fr
[4]Inria, France

Abstract. Many techniques were recently proposed to automate the linkage of RDF datasets. Predicate selection is the step of the linkage process that consists in selecting the smallest set of relevant predicates needed to enable instance comparison. We call keys this set of predicates that is analogous to the notion of keys in relational databases. We explain formally the different assumptions behind two existing key semantics. We then evaluate experimentally the keys by studying how discovered keys could help dataset interlinking or cleaning. We discuss the experimental results and show that the two different semantics lead to comparable results on the studied datasets.

1 Introduction

Linked data facilitates the implementation of applications that reuse data distributed on the web. To ensure the interoperability between applications, data issued by different providers has to be interlinked, i.e., the same entity in different datasets must be identified. Many approaches were recently proposed to detect identity links between entities described in RDF datasets (see [5],[4] for a survey). In most of these approaches, linking rules specify conditions that must hold true for two entities to be linked. Some of the approaches are fully automatic and exploit datasets to learn expressive rules: data transformations, similarity measures, thresholds and the aggregation function [12, 8, 11]. These rules are specific to the vocabulary used in the data sets. Other approaches exploit the axioms that are declared in an ontology such as keys or (inverse) functional properties [19, 6]. Indeed, keys figure largely in the Semantic Web (SW), especially since the inclusion in OWL 2 of HasKey construct, which allows to include in an ontology an axiom stating that a set of data or object properties is a key of a particular class. Key axioms can be exploited for different purposes. They can be used as logical rules to deduce identity links with a high precision rate [18]. They can also guide the construction of more complex linking rules for which elementary similarity measures or aggregation functions are chosen by user experts [22, 19]. Finally, these keys can also be exploited to detect entity pairs that do not need to be compared, as it is done with blocking methods for relational data [9, 3].

Additionally, different approaches have been recently proposed which attempt to automatically induce keys from RDF datasets, and then exploit discovered keys for datasets cleansing and interlinking [15, 2]. Nevertheless, these keys may have different semantics. In [15], each pair of class instances that share at least one value for each property of the key should be considered as referring to the same entity. In [2], the authors consider the fact that each pair of instances should coincide for all the property values to be considered as referring to the same entity. This last semantics can be interesting when one can guarantee that local completeness assumption is fulfilled. This paper contributes to formalize different notions of a key in the context of RDF dataset cleansing and interlinking, and to show if these different notions can be useful for dataset cleansing and interlinking.

In section 2, we present the different notions of keys currently proposed in the literature and show how these rules can be applied to an example. Then in section 3 we exploit an unifying logic framework to compare the various key notions and how these keys can be discovered from RDF datasets. In section 4, we give the experimental results of these different key notions for both problems of cleansing and interlinking and discuss the differences between the notions of keys theoretically introduced in previous sections. Finally, we present related work in section 5 and give our conclusion and some future plans in section 6.

2 Problem Statement

Keys can be exploited for duplicate detection and data interlinking (i.e., *owl:sameAs* link discovery) of RDF datasets. In general, if a set of properties is declared to be a key for a class, then not satisfying the key within a dataset may be due to errors or unknown duplicates, whereas individuals from different datasets which do not satisfy the key can be seen as candidates for interlinking. Indeed, each pair of instances that share the same values for all the properties of a key should be considered as referring to the same entity.

Different definitions of a key can be considered in the semantic web. It is the case of the two key extraction methods [15] and [2]. For the first approach, two individuals have to share at least one value for each property of a key to be equal, while, with the second one, they have to share all the values.

In the following, we define two notions of keys: *S*-key and *F*-key. *S*-key roughly corresponds to the `hasKey` axiom of OWL2 and the notion of a key used by [15]. In [2], a tradeoff between *S*-key and *F*-key is considered.

Definition 1 (*S*-key). *The* S*-key* $\{p_1, \ldots, p_n\}$ *for a class expression* C *is the rule defined as follows:*

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

where $C[.]$ *is a unary lambda formula having exactly one free variable.* $C[.]$ *is still named the target expression class of the key.*

Declaring that the set $\{p_1, \ldots, p_n\}$ is a S*-key for an atomic class C is denoted by* S*-key$(C, (p_1, \ldots, p_n))$.*

The definition of the `hasKey` axiom given in OWL 2 (c.f. Section 9.5 of [17] and Section 2.3.5 of [16]) enforces the considered instances to be named (*i.e.* they have to be URIs or literals, but not blank nodes). Hence an OWL 2 key can be defined with the following rule:

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

where $Const$ is a built-in unary predicate which is $true$ for a constant and $false$ otherwise.

Since KD2R [15] approach discovers *S*-keys without considering blank nodes, it follows the above OWL 2 key definition.

S-key and OWL 2 `hasKey` do not require that the two instances x and y of C coincide on all values of the key properties to be equal: it suffices to have at least one pair of values that coincide for all p_i to decide that x and y refer to the same entity. However, in case of not functional properties that represent a full list of items (e.g., a list of authors of a given paper, a list of actors of a given movie) it can be more meaningful to consider the fact that the instances should coincide for all the property values. We call this second semantics *Forall-key* and we give its formal definition in the following.

Definition 2 (*F*-key). *The* F*-key for a class C is the rule defined as follows:*

$$\forall x \forall y (C[x] \wedge C[y] \wedge \bigwedge_{i=1}^{n} (\forall z_i (p_i(y, z_i) \rightarrow p_i(x, z_i)) \wedge$$
$$(\forall w_i (p_i(x, w_i) \rightarrow p_i(y, w_i))) \;\; \rightarrow \;\; (x = y))$$

Declaring that the set $\{p_1, \ldots, p_n\}$ as a F*-key for an atomic class C is denoted by* F*-key$(C, (p_1, \ldots, p_n))$.*

Atencia et al. [2] propose an approach that automatically discovers *F*-keys by considering only class instances for which all the key properties are instantiated. This variant of *F*-key is called *SF*-key.

Illustrative Example. Figure 1 and Figure 2 help to compare the two notions of a key described above in a scenario of datasets cleansing. First, consider the RDF graph $G1$ shown in Figure 1. If the datatype property myLab:hasEMail is declared to be a *S*-key of the class myLab:Researcher then the two researchers myLab:ThomasDupond and myLab:TomDupond must be the same, since they share "thomas.dupond@mylab.org"[1]. If the datatype property myLab:hasEMail is declared to be a *F*-key of the class myLab:Researcher then we cannot infer that these two researchers are the same. In this

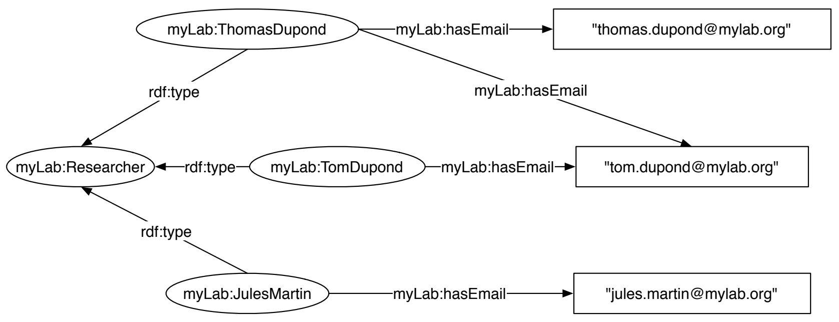

Fig. 1. A RDF graph $G1$

case declaring myLab:hasEMail as a *S*-key is more appropriate. Now, consider the graph $G2$ depicted in Figure 2. It seems not to be appropriate to declare that the object property myLab:isAuthor is a *S*-key for myLab:Researcher. Indeed, this would lead us to infer that myLab:TomDupond and myLab:JulesMartin are the same just because they have coauthored http://papersdb.org/conf/145. On the other hand, it is unlikely that different researchers are authors of exactly the same publications. If we declare that the object property myLab:isAuthor as a *F*-key for myLab:Researcher then we can infer only that the two researchers myLab:ThomasDupond and myLab:TomDupond are the same person. Indeed, using this *F*-key would not lead us to equate myLab:TomDupond and myLab:JulesMartin because the latter is not an author of http://papersdb.org/conf/26.

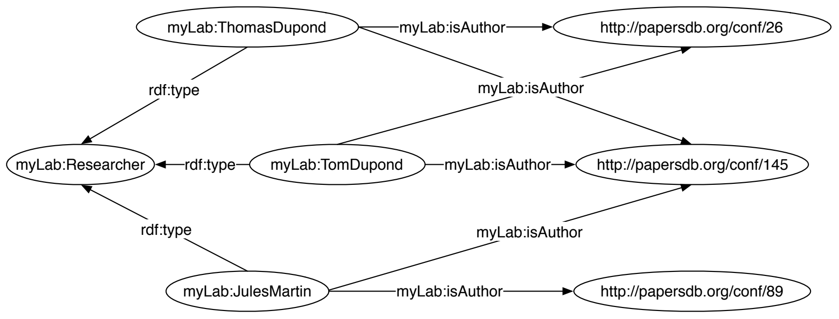

Fig. 2. A RDF graph $G2$

In this paper, we raise the problem of a comparative study of these two different semantics in a practical view (see Section 4).

[1] This is solved by adding myLab:ThomasDupond owl:sameAs myLab:TomDupond to the graph.

3 Semantic comparison of *S*-keys and *F*-keys

The two key discovery algorithms assessed in Section 4 are described using set-based notions and properties corresponding to the interpretations satisfying (i.e. to models of) *S*-keys and *F*-keys. In this section, we define these notions and we briefly present the relationships between *S*-keys and *F*-keys. For simplicity reasons, we consider C as a unary predicate.

3.1 *S*-key and *F*-key Models

Let $\mathcal{C}$ be a set, p be a binary relation over $\mathcal{C}$ and $c, c' \in \mathcal{C}$, $P(c)$ denotes the set of elements in $\mathcal{C}$ related to c by P, i.e. $P(c) = \{u \mid (c, u) \in P\}$.

Definition 3. *A FOL interpretation I with domain Δ^I of the predicates occurring in a* S*-key, or a* F*-key, is given by:*

– $C^I \subseteq \Delta^I$ *the interpretation of the class C;*
– $p_i^I \subseteq \Delta^I \times \Delta^I$ *the interpretation of the predicates p_i;*

Proposition 1. *An interpretation I is a model of the* S*-key $(C, (p_1, \ldots, p_n))$ iff for any c and c' in C^I such that for any $i = 1, \ldots, k$, $p_i^I(c) \cap p_i^I(c') \neq \emptyset$, one has $(c = c')$.*

Proposition 2. *An interpretation I is a model of the* F*-key $(C, (p_1, \ldots, p_n))$ iff for any c and c' in C^I such that for any $i = 1, \ldots, k$, $p_i^I(c) = p_i^I(c')$, one has $(c = c')$.*

Relationships between *S*-keys and *F*-keys are given by the following proposition. The first property states that the *S*-key notion is more restrictive than the *F*-key notion when one considers interpretations in which for any p_i^I there is at most one element of C^I with no value. In relational databases it corresponds to the case where in each column there is at most one null value. The second property states the opposite when one considers interpretations in which any p_i^I is functional. In relational databases it corresponds to the case where there is no multiple values. Finally, the last property states that these notions are equivalent when the interpretation of any p_i, i.e. p_i^I, is a total function. In the relational databases it corresponds to the case where there is neither column with two null values nor multiple values.

Proposition 3. *Let $S\text{-}K$ and $F\text{-}K$ be respectively the* S*-key and the* F*-key associated with $(C, (p_1, \ldots, p_n))$.*

1. *For any interpretation I such as for any $i = 1, \ldots, n$ there is at most one element $c \in C^I$ with $p_i^I(c) = \emptyset$, one has: if I is a model of $S\text{-}K$ then I is a model of $F\text{-}K$ (i.e. if $I \models S\text{-}K$ then $I \models F\text{-}K$).*
2. *For any interpretation I such as for any element $c \in C^I$ and any property p_i one has $card(p_i^I(c)) \leq 1$, then if I is a model of $F\text{-}K$ then I is a model of $S\text{-}K$ (i.e. if $I \models F\text{-}K$ then $I \models S\text{-}K$).*
3. *For any interpretation I such as for any element $c \in C^I$ and any property p_i one has $card(p_i^I(c)) = 1$, then I is a model of $S\text{-}K$ iff I is a model of $F\text{-}K$ (i.e. $I \models F\text{-}K$ iff $I \models S\text{-}K$).*

3.2 Key Discovery

In certain cases the keys are not asserted by domain expert and need to be obtained from the knowledge base. In this section we give the principle of discovering *S*-keys and *F*-keys on RDF datasets. Theoretically, a key K that is discovered on a dataset ensure the logical satisfiabilty of this considered RDF dataset enriched by K. To exploit this theoretical notion, it is needed that all the (not) sameAs of the dataset are expressed. We consider here the problem of key discovery on RDF datasets without considering blank nodes (i.e, with $Const$ atoms). Furthermore, we assume that the RDF datasets have been already saturated using the OWL entailment rules [14].

Definition 4 (Fact). *A* fact relative to a class C *over a vocabulary V is a conjunction of positive atoms built with V.*

Definition 5 (Extended Fact). *An* extended fact *is composed of a fact plus possibly negated equality atoms.*

To check the satisfiability of the fact enriched by a key we consider that all the equality and non-equality atoms are declared.

Definition 6 (Completion of a fact). *An extended fact $\mathcal{F}$ relative to a class C is* complete *if for any pair of terms (t, t'), not necessarily distinct, instances of C, it contains either $t = t'$ or $t \neq t'$. Let $\mathcal{F}$ be a fact, we denote $\mathcal{F}^C$ the* complete fact *obtained by adding $t \neq t'$ atoms for each pair of terms (t, t') such that $t = t'$ is not occurring in $\mathcal{F}$.*

It is straightforward to see that: *A fact $\mathcal{F}$ is consistent iff there is no terms t, t' such that atoms $t = t'$ and $t \neq t'$ both occur in $\mathcal{F}$.* Note that a complete fact is consistent.

Proposition 4. *Let $\mathcal{F}$ be a fact relative to a class C and let S be a* S-*key $(C, (p_1, ..., p_n))$. S is a* S-*key for C in $\mathcal{F}$ iff $(\mathcal{F}^C, S)$ is satisfiable.*

Proposition 5. *Let $\mathcal{F}$ be a fact relative to a class C and let F be a* F-*key $(C, (p_1, ..., p_n))$. F is a* F-*key for C in $\mathcal{F}$ iff $(\mathcal{F}^C, F)$ is satisfiable.*

In practice, knowledge bases often contain erroneous data and/or unknown duplicates. Thus discovering keys strictly on the basis of key definitions leads to miss some useful keys. That is why, [2] introduced the notion of pseudo-keys by allowing discovered keys to have some exceptions. Of course, adding these pseudo-keys to the knowledge base leads to its unsatisfiability.

A pseudo-key can be characterized by a discriminability measure which allows to assess the quality of such discovered keys. We propose here a definition of discriminability that can be used to discover *S*-keys and *F*-keys with exceptions.

Definition 7 (Discriminability). *The discriminability of a key is the ratio between the size of the maximal subset of instances on which the discovered key is a key and the number of instances of the class in $\mathcal{F}$.* [2]

[2] In the case of *S*- and *SF*-keys, discriminability is defined only the subset of instances having at least a value for each property of the key

4 Experimental Results

This section experimentally studies how *S*-key, *F*-key and its *SF*-key variant behave on data cleansing and data interlinking scenarii. In the first experiment, we discovered keys on a benchmark about movies to evaluate the potential of each semantic of key for discovering links between datasets. In a second experiment we discovered keys with exceptions on a subset of DBPedia dataset in order to evaluate the potential of each type of key for discovering duplicates in a dataset. We have used KD2R [15] and [2] to discover the keys.

4.1 Data-interlinking experiments

The first experiment presents the performance of the different semantic of key for a data interlinking task. The second one studies the growth resistance of these different notions of a key.

IIMB dataset IIMB (ISLab Instance Matching Benchmark) is a set of interlinking tasks used by the instance matching track of the Ontology Alignment Evaluation Initiative (OAEI 2011 & 2012)[3]. An initial dataset that describes movies (films, actors, directors, etc.) has been extracted from the web (file 0). Various kinds of transformations, including value transformations, structural and logical transformations, were applied to this initial dataset to generate a set of 80 different test cases. For each test case, the reference alignments are given (i.e. sameAs links between individuals of the generated test case and the ones of the initial dataset).

Qualitative evaluation We evaluate the keys here using the first test case (file 1) in which the modifications only concern data property values (typographical errors, lexical variations). We have focused this experiment on the class $Film$. Each of the two files contains 1228 descriptions of pairwise distinct film instances. These instances are described using six object properties and two datatype properties. The object properties are: $featuring$ that describes all the people involved in, $starring\_in$ that describes the main actors, $directed\_by$, $estimated\_budget$, $filmed\_in$ (i.e. the languages) and $shot\_in$ (i.e. the places where the movie was filmed). The two datatype properties are $name$ and $article$ (a textual detailed description of the film). From all these properties, $featuring$, $starring\_in$, $directed\_by$, $filmed\_in$ and $shot\_in$ are properties that can be multi-valued.

We have discovered *S*-keys, *SF*-keys and *F*-keys on the set of film instances described in file 0. Since we know that the unique name assumption is fulfilled, we have built the extended fact by completing the fact corresponding to the file 0 with all the inequality atoms between all the pairs of class instances. The applied similarity measures are string equality after stop words elimination for data properties and equality for object properties. For the sake of simplicity, we omit the similarity measures in the following.

[3] `http://oaei.ontologymatching.org/2011/`

The three discovered *S*-keys are the following:
$\{(name, directed\_by, filmed\_in\}$, $\{article\}$, $\{estimated\_budget\}$.
The seven *F*-keys are the following:
$\{name, directed\_by, filmed\_in\}$,$\{shot\_in, article, starring\_in\}$, $\{name, article, directed\_by\}$, $\{article, featuring\}$, $\{name, featuring\}$, $\{name, starring\_in\}$, $\{article, directed\_by, starring\_in\}$.
Finally, we have discovered five *SF*-keys:
$\{name, directed\_by, filmed\_in\}$,$\{article\}$, $\{estimated\_budget\}$,
$\{name, featuring\}$, $\{name, starring\_in\}$.

For this dataset, every *S*-key is also an *SF*-key. Because of the absence of some property values and because of the multi-valuation (see Proposition 3), the set of *F*-keys is not comparable to the sets of *S*-keys. Furthermore, some *SF*-keys are not *F*-keys. For example, the property $estimated\_budget$ is not a *F*-key , because there are at least two film instances for which the value of this property is not given.

To evaluate the quality of the discovered keys, they have been applied to infer mappings between film instances of the file 0 and of the file 1. More precisely, we have measured the quality of each set of keys independently from the quality of the possible links that can be found for other class instances. With this goal, we have exploited the correct links appearing in the reference alignments to compare object property values (i.e. complete set of sim_C for all C different from $Film$). In table 1, we present recall, precision and F-measure for each type of keys.

keys	Recall	Precision	F-Measure
S-keys	27.86%	100%	43.57%
F-keys	22.15%	100%	36.27%
SF-keys	27.86%	100%	43.57%

Table 1. Recall, Precision and F-measure for the class film

We notice that the results of *S*-keys and *SF*-keys are the same. Indeed, all the mappings that are found by the two additional *SF*-keys are included in the set of mappings obtained using the three shared keys. Furthermore, the shared keys generate the same links either because the involved properties are monovalued ($estimated\_budget$, $article$), or because the involved multi-valued object properties have the same values in both files for the same film. The recall of *F*-keys is slightly lower in particular because of the absence of the key $\{estimated\_budget\}$.

Some *SF*-keys cannot be *S*-keys and may have a high recall. For example, it is not sufficient to know that two films share one main actor to link them. On the other hand, when the whole sets of actors are the same, they can be linked with a good precision. Moreover, if we consider only instances having at least two values for the property $starring\_in$ (98% of the instances), this property is discovered as an *SF*-key and allows to find links with a precision of 96.3% and a recall of 97.7%.

Growth resistance evaluation We have evaluated how the quality of each type of keys evolve when they are discovered in smaller parts of the dataset. Thus, we have randomly split the file 0 in four parts. Each part contains the complete description of a subset of the Film instances. We have then discovered the keys in a file that contains only 25% of the data, 50% of the data,and finally 75% of the data. Then we have computed the recall, precision and f-measure that are obtained for each type of keys. The larger the dataset, the more specific are the keys. Also, for all types of keys, precision increases and recall decreases with dataset' size. To obtain a good precision, *S*-keys need to be discovered in at least 50% of this dataset, while *F*-keys and *SF*-keys obtain a rather good precision when the keys are learnt using only 25% of the dataset. Furthermore, some *F*-keys and some *SF*-keys have also a very high recall when they are learnt on a subpart of the dataset even if the precision is not 100%. Indeed, new RDF descriptions are introduced that prevent the system from discovering keys that can be very relevant. So, it seems particularly suitable to allow to have exceptions when these two kinds of keys are discovered.

S-keys	Recall	Precision	F-Measure
25%	27.85%	77.55%	40.98%
50%	27.85%	99.42%	43.51%
75%	27.85%	99.42%	43.51%
100%	27.85%	100%	43.56%

SF-keys	Recall	Precision	F-Measure
25%	100%	94.1%	96.96%
50%	100%	99.03%	99.51%
75%	27.85%	99.42%	43.51%
100%	27.85%	100%	43.56%

F-keys	Recall	Precision	F-Measure
25%	100%	94.1%	96.96%
50%	100%	99.03%	99.51%
75%	22.14%	99.27%	36.21%
100%	22.14%	100%	36.27%

Table 2. Recall, Precision and F-measure for the *S*-keys, *SF*-keys & *F*-keys

4.2 Dataset cleansing experiment

This experiment aims at studying the differences between *S*-keys, *SF*-keys and *F*-keys on the application consisting in identifying duplicate pairs inside a dataset. The protocol is to learn pseudo keys using a given discriminability threshold less than 1. All pseudo keys will be viewed as key and their exception pairs as deduced similarities. Deduced similarities will be evaluated allowing us to compute precision and recall.

DBPedia Persons dataset DBPedia Persons[4] is a subset of DBPedia describing persons (date and place of birth etc.) extracted from the English and German Wikipedia and described using the FOAF vocabulary. This dataset contains 966,460 instances and makes use of 8 properties: foaf:name, foaf:surname, foaf:givenName, dc:description, dbpedia:birthPlace, dbpedia:deathPlace, dbpedia:birthDate, dbpedia:deathDate.

[4] for these experiments, we use the version 3.8 of DBPedia

Experimental protocol The goal of this experiment is to evaluate and compare the similarities that can be deduced for each kind of keys: *S*-keys, *F*-keys and *SF*-keys. To that extent, we propose two evaluations. The first one aims at assessing the quality of the similarity deduced by discovered keys. The second evaluation consists in comparing the resistance of discovered keys to dataset alterations.

The first evaluation consists in computing, for each kind of key, the set of pseudo keys on a dataset D for a given discriminability threshold σ_d, then deducing similarities using the discovered keys and finally evaluating the set of deduced similarities. Usually such an evaluation is performed by computing precision and recall. Recall requires a reference set of similarity to be provided. The size of DBPedia prohibits us to manually build such a reference. Then we can only provide a relative recall evaluation. For each kind of key $x \in \{S-key, F-key, SF-key\}$, a set of similarities E_x is computed. All discovered similarities will be evaluated manually allowing to identify true positives TP_x and false positives FP_x. The precision of keys x is $P(x) = |TP_x|/|E_x|$ and its relative recall is $R(x) = |TP_x|/|TP_{S-key} \cup TP_{F-key} \cup TP_{SF-key}|$.

For the second evaluation, we randomly remove $x\%$ triples from the dataset and we compute keys on it. We repeated this procedure for x varying from 0 to 90 with a step of 10%. The ideal evaluation would be to measure precision and recall foreach case. However, it is not straightforward to manually check all exceptions on each case. Thus, we restrict ourselves to compare extracted keys: What is the proportion of keys extracted from the original dataset which are equals to and more general than those extracted from an altered dataset.

Qualitative evaluation Quality evaluation has been performed on DBPedia Persons dataset. For the key discovery, we choose a discriminability threshold equal to $99,90\%$ as it generates an reasonable amount of similarities to be checked. For threshold of 99%, there were several thousand generated similarities.

There is no result for F-keys because on the DBPedia dataset properties are not instantiated for each instance. In average properties are defined for 60% of 1 million of instances then it implies a lot of exceptions to the key. For that kind of dataset, F-keys are clearly not suitable.

	Precision	Relative Recall	F-measure
E_{S-key}	63%	86%	73%
E_{SF-key}	65%	96%	78%
$E_{S-key} \cup E_{SF-key}$	59%	100%	74%
$E_{S-key} \cap E_{SF-key}$	73%	82%	77%
$E_{SF-key} - E_{S-key}$	40%	14%	21%
$E_{S-key} - E_{SF-key}$	15%	4%	6%

Table 3. Precision, Relative recall and F-measure on DBPedia Person

Table 3 shows results obtained for *S*-keys and *SF*-keys. For this dataset, *SF*-keys get slightly better results since they get a better relative recall. Comparatively, the difference between similarities deduced by *S*-keys and *SF*-keys is marginal since they share respectively 83% and 76% of deduced similarities. The precision of similarities shared by the two approaches is much higher than those discovered by only one approach.

In conclusion, these first results do not show any major difference between the similarities discovered by *S*-keys and *SF*-keys. Marginally, both approaches generate their own good similarities but the precision values of these exclusive parts are very low.

Since the precision of both approaches is not perfect, we propose two analysis at the key level. The goal of this analysis is to see if one approach is more suitable than the other for filtering similarities in function of the keys generating them. The first one aims at studying if false-positives are specific to some keys. To that extent, Figure 3 provides the distributions of discovered keys in function of precision and recall for both *SF*-keys and *S*-keys. This chart shows that the keys having the highest recall tend to have good precision, but it does not validate the hypothesis that false-positive are mainly deduced from particular keys. This trend is observed both for *SF*-keys and *S*-keys. The second analysis aims at showing similarities reaching a certain level of consensus among both types of keys positively influence precision and recall. Figure 5 shows that, for both approaches, precision increases significantly and quickly when the minimal consensus increase. Then it becomes stable from 5 keys on. Recall and F-Measure follow almost the inverse trend. It clearly demonstrates that consensus is not sufficient for selecting both only and all good similarities.

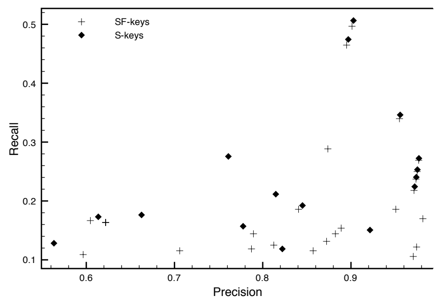

Fig. 3. Distribution of keys in function of the precision and recall

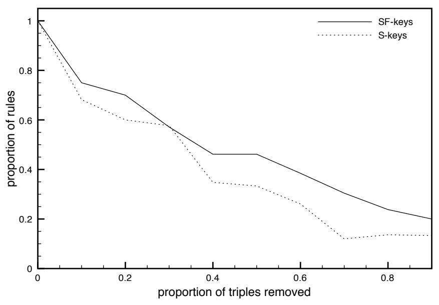

Fig. 4. Evolution of the proportion of keys equals to or generalized by initial keys in function of the quantity of triples removed

To conclude, this qualitative comparison of *SF*-key and *S*-key shows that both approaches are able to discover good similarities among a real dataset. When we try to filter only and the most of true positives the two approaches have the same behavior. The most suitable criteria is to only consider similarities which are common to both approaches.

Growth-resistance evaluation Figure 4 shows the loss of generality of the initial set of keys in regard to the keys discovered when triples are randomly removed. *S*-keys

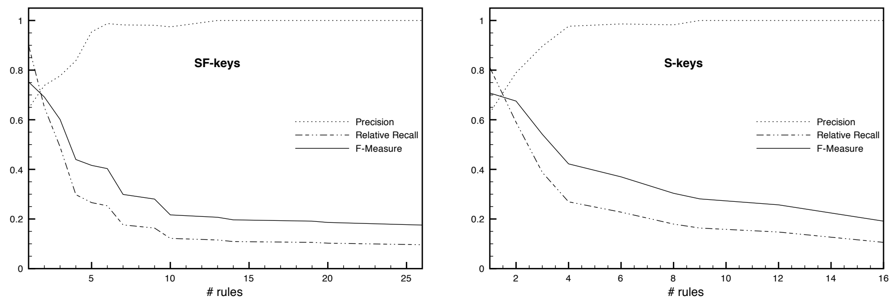

Fig. 5. Evolution of precision, relative recall and F-measure for the deduced similarities in function of the minimal number of keys from which they are deduced

curve decreases faster than those of ADS, but the difference between the two approches remains low.

5 Existing Work

The problem of key discovery from RDF datasets is similar to the key discovery problem in relational databases. In both cases, the key discovery problem is a sub-problem of Functional Dependencies (FDs) discovery. A FD states that the value of one attribute is uniquely determined by the values of some other attributes. In relational data keys or FDs can be used in tasks related to data integration, anomaly detection, query formulation, query optimization, or indexing. Discovering FDs in RDF data differs from the relational case as null values and multi-valued properties are to be considered.

In the semantic web, data linking approaches exploit S-key semantics that can be declared using OWL2 when they are restricted to named entities. Indeed, these keys can be used for different purposes. Blocking methods aim at using approximate S-keys to reduce the number of instance pairs that have to be compared by a data linking tool ([10],[20]). In [20], discriminating data type properties (i.e approximate keys) are discovered from a data set. Then, only the instance pairs that have similar literal values for such discriminating properties are selected. These properties are chosen using unsupervised learning techniques.

Other approaches use approximate *S*-keys to infer possible identity links. In [21], the authors discover (inverse) functional properties from data sources where the unique name assumption (UNA) hypothesis is fulfilled (i.e. non composite *S*-keys). The functionality degree of a property is computed to generate probable identity links. More precisely, for one instance, the local functionality degree of a property is the number of distinct values (or instances) that are the object of the property when the considered instance is the subject. The functionality degree of one property is the harmonic mean of the local functionality degrees across all the instances; the inverse functionality degree is defined analogously. In the context of Open Linked Data, [13] have proposed a supervised approach to learn (inverse) functional properties on a set of linked data (i.e.

non composite S-keys). Other approaches aim to enrich the ontology and/or use these *S- keys* to generate identity links between pairs of instances that can be propagated to other pairs of instances ([19, 1]). Such approaches, are called collective or global approaches of data linking. For example, if the approach can find that two paintings are the same, then their museums can be linked and this link will lead to generate identity links between the cities where the museums are located in. KD2R [15] aims to discover S-keys that are correct with regard to a set of data sources. The approach does not need training data and exploits data sources where the unique name assumption hypothesis is fulfilled. One important feature of KD2R is that it can discover composite keys.

In [2], the authors developed an approach based on TANE [7] algorithm to discover pseudo-keys for which a few instances may have the same values for the properties of a key. In this work, the discovered keys are particular F-keys for which only class instances for which all the key properties are instantiated are considered (i.e *SF*-keys).

The notion of keys employed by the above papers varies depending on the approach taken. Nevertheless, even if some of them exploit approximated or non composite keys, they are all based on the semantics of F-keys, SF-keys or *S-keys*.

6 Conclusion

Keys are useful in the web of data for interlinking and cleansing tasks. We have shown that different notions of a key coexist in the semantic web. This paper provides experimental evaluations of the different semantics of key for both interlinking and cleansing scenarios. Results show that learning F-keys from data is not suitable when properties are not almost fully instanciated. The SF-key variant allows to fix this problem by relaxing the equality constraint when instances have no value for a property. In term of data interlinking or data cleansing , S-keys and SF-keys have almost the same relevance in term of precision and recall. SF-keys and F keys seems to be more robust than S-keys to instance removal.

Each semantics of key has its own advantages and we think that it could be interesting to define and discover hybrid keys (i.e. keys composed of F properties and S properties) when data knowledge and/or ontology axioms can be used to decide how properties can be handled.

Acknowledgement. This work has been partially supported by the ANR projects Qualinca (12-CORD-0012) and Lindicle (12-IS02-0002).

References

1. A. Arasu, C. Ré, and D. Suciu. Large-scale deduplication with constraints using dedupalog. In *ICDE*, pages 952–963, 2009.
2. M. Atencia, J. David, and F. Scharffe. Keys and pseudo-keys detection for web datasets cleansing and interlinking. In *EKAW*, pages 144–153, 2012.
3. R. Baxter, P. Christen, and T. Churches. A comparison of fast blocking methods for record linkage. In *KDD 2003 WORKSHOPS*, pages 25–27, 2003.

4. A. K. Elmagarmid, P. G. Ipeirotis, and V. S. Verykios. Duplicate record detection: A survey. *IEEE Transactions on Knowledge and Data Engineering*, 19:1–16, 2007.
5. A. Ferrara, A. Nikolov, and F. Scharffe. Data linking for the semantic web. *Int. J. Semantic Web Inf. Syst.*, 7(3):46–76, 2011.
6. W. Hu, J. Chen, and Y. Qu. A self-training approach for resolving object coreference on the semantic web. In *WWW*, pages 87–96, 2011.
7. Y. Huhtala, J. Kärkkäinen, P. Porkka, and H. Toivonen. Tane: An efficient algorithm for discovering functional and approximate dependencies. *The Computer Journal*, 42(2):100–111, 1999.
8. R. Isele and C. Bizer. Learning expressive linkage rules using genetic programming. *PVLDB*, 5(11):1638–1649, 2012.
9. R. Isele, A. Jentzsch, and C. Bizer. Efficient multidimensional blocking for link discovery without losing recall. In *Proceedings of the 14th International Workshop on the Web and Databases (WebDB)*, Greece, 2011.
10. M. Michelson and C. A. Knoblock. Learning blocking schemes for record linkage. In *AAAI*, pages 440–445, 2006.
11. A.-C. N. Ngomo and K. Lyko. Eagle: Efficient active learning of link specifications using genetic programming. In *9th Extended Semantic Web Conference (ESWC)*, pages 149–163, 2012.
12. A. Nikolov, M. d'Aquin, and E. Motta. Unsupervised learning of link discovery configuration. In *9th Extended Semantic Web Conference (ESWC)*, pages 119–133, Berlin, Heidelberg, 2012. Springer-Verlag.
13. A. Nikolov and E. Motta. Data linking: Capturing and utilising implicit schema-level relations. In *Proceedings of Linked Data on the Web workshop at 19th International World Wide Web Conference(WWW)'2010*, 2010.
14. P. F. Patel-Schneider, P. Hayes, and I. Horrocks. OWL Web Ontology Language Semantics and Abstract Syntax Section 5. RDF-Compatible Model-Theoretic Semantics. Technical report, W3C, Dec. 2004.
15. N. Pernelle, F. Sais, and D. Symeonidou. An automatic key discovery approach for data linking. *Web Semantics: Science, Services and Agents on the World Wide Web*, (0):–, 2013.
16. W. Recommendation. Owl 2 web ontology language: Direct semantics. In C. G. B. Motik B., Patel-Schneider P. F., editor, *http://www.w3.org/TR/owl2-direct-semantics/*. W3C, 27 October 2009.
17. W. Recommendation. Owl 2 web ontology language: Structural specification and functional-style syntax. In P. B. Motik B., Patel-Schneider P. F., editor, *http://www.w3.org/TR/owl2-syntax/*. W3C, 27 October 2009.
18. F. Saïs, N. Pernelle, and M.-C. Rousset. L2r: A logical method for reference reconciliation. In *Proceedings of the Twenty-Second AAAI Conference on Artificial Intelligence, Vancouver, British Columbia, Canada*, pages 329–334, 2007.
19. F. Saïs, N. Pernelle, and M.-C. Rousset. Combining a logical and a numerical method for data reconciliation. *Journal on Data Semantics*, 12:66–94, 2009.
20. D. Song and J. Heflin. Automatically generating data linkages using a domain-independent candidate selection approach. In *International Semantic Web Conference (1)*, pages 649–664, 2011.
21. F. M. Suchanek, S. Abiteboul, and P. Senellart. Paris: Probabilistic alignment of relations, instances, and schema. *The Proceedings of the VLDB Endowment(PVLDB)*, 5(3):157–168, 2011.
22. J. Volz, C. Bizer, M. Gaedke, and G. Kobilarov. Discovering and maintaining links on the web of data. In *Proceedings of the 8th International Semantic Web Conference*, ISWC '09, pages 650–665, Berlin, Heidelberg, 2009. Springer-Verlag.