Automatic Key Selection for Data Linking

Manel Achichi[1], Mohamed Ben Ellefi[1], Danai Symeonidou[2], Konstantin Todorov[1]

{achichi,benellefi,todorov}@lirmm.fr, danai.symeonidou@supagro.inra.fr
[1]LIRMM / University of Montpellier, France, [2]INRA, MISTEA Joint Research Unit, UMR729, F-34060 Montpellier, France

Abstract. The paper proposes an RDF key ranking approach that attempts to close the gap between automatic key discovery and data linking approaches and thus reduce the user effort in linking configuration. Indeed, data linking tool configuration is a laborious process, where the user is often required to select manually the properties to compare, which supposes an in-depth expert knowledge of the data. Key discovery techniques attempt to facilitate this task, but in a number of cases do not fully succeed, due to the large number of keys produced, lacking a confidence indicator. Since keys are extracted from each dataset independently, their effectiveness for the matching task, involving two datasets, is undermined. The approach proposed in this work suggests to unlock the potential of both key discovery techniques and data linking tools by providing to the user a limited number of merged and ranked keys, well-suited to a particular matching task. In addition, the complementarity properties of a small number of top-ranked keys is explored, showing that their combined use improves significantly the recall. We report our experiments on data from the Ontology Alignment Evaluation Initiative, as well as on real-world benchmark data about music.

1 Introduction

In recent years, the Web of Data has been constantly growing both in terms of quantity of the RDF datasets published publicly on the web and in terms of diversity of the domains that they cover. One of the most important challenges in this setting is creating semantic links among these data [1]. Among all possible semantic links that could be declared between resources found in different datasets, identity links, defined by the `owl:sameAs` statement, are of great importance and the ones that most of the attention is given to. Indeed, `owl:sameAs` links allow to see currently isolated datasets as one global dataset of connected resources. Considering the small number of existing `owl:sameAs` links on the Web today, this task remains a major challenge [1].

Due to the large amount of data already available on the Web, defining manually `owl:sameAs` links would not be feasible. Therefore, many approaches try to answer to this challenge by providing different strategies to automate this process. Datasets conforming to different ontologies, data described using

different vocabularies, datasets described in different languages are only several of the examples that make this problem hard to solve.

Many of the existing link discovery approaches are semi-automatic and require manual configuration. Some of these approaches use *keys*, declared by a domain expert, to link. A key represents a set of properties that uniquely identifies every instance of a given class. Keys can be used as logical rules to link data ensuring high precision results in the linking process. Additionally, they can be exploited to construct more complex rules. Nevertheless, keys are rarely known and are very hard to declare even for experts. Indeed, experts may not know all the specificities of a dataset leading to overlook certain keys or even introduce erroneous ones. For this reason, several automatic key discovery approaches have been already proposed in the context of the Semantic Web [2,3,4,5,6].

In spite of that fact, applying the output of these approaches directly is, in most of the cases, impossible due to the characteristics of the data. Ontology and data heterogeneity are not the only issues that can arise while trying to apply keys directly for data linking. Even if the datasets conform to the same ontology and the vocabulary of the properties is uniform, this does not ensure the success of the linking process. Very often, key discovery approaches discover a very large number of keys. The question that arises is whether all the keys are equally important among them, or there are some that are more significant than others. So far, no approach provides a strategy to rank the discovered keys, by taking in consideration their effectiveness for the matching task at hand.

Bridging the gap between key discovery and data linking approaches is critical in order to obtain successful data linking results. Therefore, in this paper we propose a new approach that, given two datasets to be linked, provides a set of ranked keys, valid for both of them. We introduce the notion of "effectiveness" of a discovered key. Intuitively, a key is considered as effective if it is able to provide many correct `owl:sameAs` links. In order to measure the effectiveness of keys, a support-based key quality criterion is provided. Unlike classic approaches using support for the discovered keys, in this work we introduce a new global support for keys valid for a set of (usually two) datasets.

The proposed approach can be summarized in the following main steps. *(1) Preprocessing:* in this step, given two datasets to be linked, only properties that are shared by both datasets are kept. This ensures that a key can be applied on both the source and the target datasets, and not only on each of them independently. At this point it is important to state that we consider that the datasets use either common vocabularies or that the explicit mapping between the respective vocabularies is known. *(2) Merge:* the key candidates discovered in each dataset are then merged by computing their cartesian product (recall that a key is a set of properties). *(3) Ranking:* we introduce a ranking criterion on the set of merged keys that is a function of the respective supports of each merged key in each dataset, normalized by the dataset sizes. *(4) Keys combination:* finally, the combined use of several top-ranked merged keys is evaluated, showing an improvement of the recall of a given link discovery tool.

The rest of the paper is structured as follows. Section 2 overviews data linking and automatic keys discovery and link specification approaches. Then, Section

3 presents our key ranking technique, evaluated in Section 4. Conclusions and future work are provided in Section 5.

2 Related Work

Let us look onto the process of data linking from a global perspective. The majority of the existing linking tools implement a process that consists of three steps: (1) configuration and pre-processing, (2) instance matching and (3) post-processing. Step (1) aims on the one hand to reduce the search space by identifying sets of linking candidates and key properties to compare, and on the other hand – to model instances by using a suitable representation that renders them comparable (one can think of indexing techniques, automatic translation, *etc.*). Step (2) aims at deciding on a pair of instances whether they are equivalent or not, mostly relying on similarity of property values, evaluated by similarity measures defined in step (1). The output of step (2) is a set of matched instances, also known as a *link set*. Finally, step (3) allows to filter out erroneous matches or infer new ones, based on the link set provided in step (2).

The configuration step of the linking workflow described above contains two important sub-steps: (a) the choice of properties (or keys) across the two datasets whose values need to be compared, and (b) the choice of similarity measures to apply and their tuning. Our approach is tightly related to these sub-steps, although it does not fit into either of these categories. Indeed, we are not aware of the existence of other approaches that address the problem of key quality evaluation with respect to data linking, therefore, the current section looks into approaches relevant to both (a) and (b), as well as to the data linking process as a whole.

2.1 Automatic Linking Tools Configuration

Key Discovery. In order to link, many data linking approaches require a set of linking rules. Some data linking approaches use keys to build such rules. A key is a set of properties that uniquely identifies every resource of a given class. Nevertheless, keys are rarely known and also very hard to define even for expert.

In the context of Semantic Web, different key discovery approaches have been already proposed. Both [5] and [2] propose a key discovery approach that follows the semantics of a key as defined by OWL. This definition states that two instances are referring to the same real world entity if at least one value per property appearing in a key is equal. Unlike [5], [2] proposes a method that scales on large datasets, taking also into account errors or duplicates in the data. In [6] and [4], the authors propose an alternative definition for the keys that is valid when the data are locally complete. In this case, to consider that two instances are equal, all the set of values per property appearing in a key should be the same. Finally, in [7], a key discovery approach for numerical data is proposed.

Atencia *et al.* [3] observe that key extraction is conducted by state-of-the-art tools in an independent manner for two input datasets without taking into

consideration the linking task ahead. The authors introduce the concept of a *linkkey* – a set of properties that are a key for two classes *simultaneously*, implying equivalence between resources that have identical values for the set of these properties.

Automatic Link Specification Algorithms. We consider the work on automatic link specification as related in terms of motivation to our approach and complementary in terms of application. Link specification is understood as the process of automatically building a set of linking rules (restrictions on the instances of the two datasets), choosing similarity measures to apply on corresponding property values across datasets together with their respective thresholds [8]. Several approaches have been introduced so far, mostly based on machine learning techniques, such as FEBRL [9], an extension of SILK [10], RAVEN [11] or, more recently, EAGLE [8]. Contrarily to key discovery methods, these approaches mainly focus on the automatic selection, combination and tuning of similarity measures to apply on the values of comparable properties. The identification of properties to compare is done by matching algorithms and no key computation is implied in this process. The efficiency of these algorithms can be improved if the system knows on which properties and on what types of values the similarity measures will be applied.

2.2 Data Linking

Data linking has evolved as a major research topic in the semantic web community over the past years, resulting in a number of approaches and tools addressing this problem. Here, instead of making an inventory of these techniques, surveyed in [12] and [13], we scrutinize the main characteristics that unite or differentiate the most common approaches.

The majority of the off-the-shelf linking tools [14,15,16,17,18,19] produce an RDF *linkset* of `owl:sameAs` statements relating equivalent resources and the linking process is commonly semi-automatic. As discussed above, the user has to configure manually a number of input parameters, such as the types of the instances to compare (with certain exceptions like [18] where ontology matching techniques are applied to identify the equivalent classes automatically), the properties (or property chains) to follow, since most linking tools adopt a property-based link discovery philosophy, the similarity measure(s) and thresholds to apply on the literals and possibly an aggregation function for several measures. The bigger part of the existing approaches are conceived as general purpose linking methods and are designed to handle monolingual RDF data.

What differentiates these tools in the first place is the techniques of automatic preprocessing that are embedded in their architecture. Scalability and computational efficiency are major issues when dealing with data linking problems on the web scale. To reduce the search space, [19] cluster data items, based on their similarity with respect to their properties. Indexing techniques are used to reduce the number of instance comparisons by Rong *et al.* [20] using similarity of vectors as a proxy for instance relatedness. Similarly, Shao *et al.* [16] and

Kejriwal *et al.* [21] apply a blocking technique, which consists in using inverted indexing to generate candidate linking sets. SILK [14] relies on indexing all target resources by the values of one or more properties used as a search term. LIMES [15] relies on the triangle inequality property of metric spaces to reduce the number of comparisons and thus the time complexity of the task.

The linking tools vary with respect to their abilities to handle different degrees and types of data heterogeneity. Indeed, most of the tools are able to cope with minor differences in spelling in the string literals by applying string matching techniques, but only a few are able to deal with more complex heterogeneities and just a couple of them try to resolve the problem of multilingualism (using different natural languages in data description), as Lesnikova *et al.* do, although in a very restricted scenario of only two languages [22].

2.3 Positioning

The approach that is proposed in this paper attempts to close the gap between automatic key discovery algorithms and the data linking process. As observed above, the majority of key discovery techniques do not effectively facilitate the task of selection of properties whose values to compare in the linking process, due the large number of keys produced and the lack of confidence indicator coupled with the keys. Our method suggests to unlock the potential of key-based techniques by providing to the user of a data linking tool a limited number of quality keys, well-suited to the particular matching task. The only key-based approach that looks into the usefulness of keys for two datasets simultaneously, and not independently from one another, is [3]. In contrast to our approach, the set of *linkkeys* produced in [3] is unordered which does not allow to effectively select a key or decide on the use of one key as opposed to another.

As compared to automatic link specification algorithms cited in Subsection 2.1, our approach can be seen as complementary: we focus on the identification of a limited set of properties that can be used to effectively link datasets, while leaving the choice of the similarity measures, their combination and tuning to the user, or to the auto-configuring link specification methods given above. The automatic selection of keys can potentially improve the quality of link specification methods by restricting considerably the similarity space.

3 Automatic Key Ranking Approach

Given two RDF datasets, candidates to be linked, our approach aims at ranking the keys that are valid for *both* datasets. These keys can be used successfully as link specifications by link discovery frameworks. Before introducing the approach, recall the OWL definition of a key. A key is a set of properties, such that if two resources share at least one value for every property participating in this key, these resources are considered as equal, or formally:

$$\forall X, \forall Y, \forall Z_1, \ldots, Z_n, \wedge c(X) \wedge c(Y) \bigwedge_{i=1}^{n} (p_i(X, Z_i) \wedge p_i(Y, Z_i)) \Rightarrow X = Y, \quad (1)$$

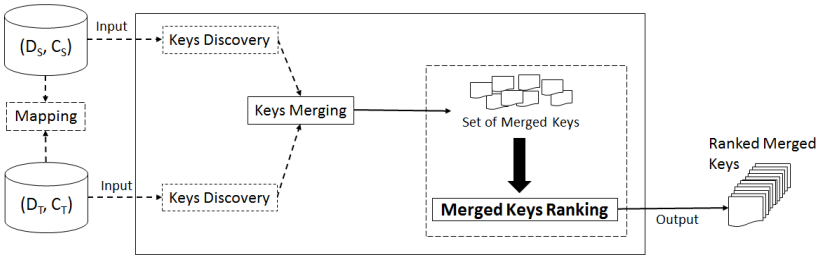


Fig. 1: The processing pipeline of Alg. 1.

where X and Y are instances of the class c and $p_i(X, Z_i) \wedge p_i(Y, Z_i)$ expresses that both X and Y share the same value Z_i for every property p_i in the key.

In next section, we describe how do we select keys that are valid for the two datasets. Afterwards, we describe our ranking approach on the set of these keys.

3.1 Selecting Mutual Keys for Two Datasets and Merging

We start by giving one of our initial hypothesis. The number of available vocabularies has been growing with the growth of the LOD cloud, resulting in datasets described by a mixture of reused vocabulary terms. It is therefore often the case that two different datasets to be linked are described by different vocabularies. To answer to that, ontology alignment methods [23] are used in order to create mappings between vocabulary terms. In this paper, we assume that equivalence mappings between classes and properties across the two input datasets are declared (either manually, or by the help of an ontology matching tool). These mappings will be used to obtain keys that are valid for *both* datasets.

Algorithm 1 gives an overview of the main steps of our approach, also depicted in Figure 1. Overall, given two datasets to be linked, this algorithm returns a set of ranked keys valid for both datasets. In addition to that, every proposed key is given a score, allowing to rank keys according to their impact on the data linking process. This process is described step by step below.

First, given the datasets D_S and D_T containing instances of a class C, a set of property mappings M between the two datasets is computed. As described in [5], property mappings allow the identification of properties that belong to both datasets simultaneously.

A key discovery step is applied to both datasets independently allowing the discovery of valid keys in each dataset. Only mapped properties, appearing in M, will be contained in the discovered keys. For this step, existing key discovery tools such as SAKey [2] or ROCKER [4] can be used to obtain keys for a given class C.

However, even if keys consist of properties that belong to both datasets, nothing ensures that the discovered keys found in each dataset independently will be the same. Indeed, there can be cases where something found as a key in one dataset it is not true in the other. Since key discovery approaches learn

keys from the data, the generality of each dataset affects the generality of the discovered keys. For example, if a dataset contains people working in a specific university, it is possible to discover that the last name is a key. Thus, to deal with this challenge a merging step is performed. Indeed, merging keys coming from different datasets allows to verify the validity of discovered keys and to obtain more meaningful keys since they are applicable to more than one datasets. Different strategies for key merging could be applied. In this work, we apply a merging strategy proposed in [5] providing minimal keys valid in both datasets.

The result is a set of merged keys considered as valid for *both* datasets. However, the number of merged keys produced by the algorithm can be significantly high, which makes manual selection difficult, particularly in the lack information of the keys suitability for the data linking task. Therefore, we introduce a novel ranking method for merged keys to identify the most suitable keys to be used in the link specification, introduced in next section.

Algorithm 1: The merged keys ranking algorithm.

Input: D_S and D_T, a pair of datasets candidates to be linked.
Output: A set of merged and ranked keys: *rankedMergedKeys*

```
1 M ← Mapping(D_S, D_T);
2 KeysD_S ← keysDiscovery(D_S, M);
3 KeysD_T ← keysDiscovery(D_T, M);
4 MergedKeys ← keysMerging(KeysD_S, KeysD_T);
5 rankedMergedKeys ← mergedKeysRanking(D_S, D_T, MergedKeys);
6 return rankedMergedKeys;
```

3.2 Merged Keys Ranking

As described before, the merged keys are valid for both datasets. However, these keys may vary in terms of "effectiveness" in the linking process. Therefore, we propose to first to assign a score reflecting the "effectiveness" of a discovered key and second use this score to rank the discovered keys among them.

In general, it is very common that not all the properties are used to describe every instance of a given class. This happens often due to the nature of the property or the incompleteness of the data and may have significant impact on the quality of the discovered keys with respect to the linking task. While many properties apply to every instance of a class, there exist cases of properties that have values only for certain instances (the property "spouse" for a person applies only to people that are married). In addition, in the case when data are incomplete, an instance may not have a value for a specific property even if a value exists in reality. This can lead to the discovery of wrong keys since not all the possible scenarios are visible in the data. Since it is very hard to differentiate these two cases automatically and a manual identification would not be feasible due to the size of the existing datasets, we use the notion of support to measure the completeness of a key. The support measures the presence of a

set of properties in a dataset. Intuitively, we tend to trust more keys that are valid for many instances in the data, i.e., keys with high support.

Basing ourselves on the support definition initially given by Atencia *et al.* in [6], we redefine this measure in order to provide a ranking score for properties with respect to a given dataset.

Let D be an RDF dataset described by an ontology O. For a given class $C \in O$, let I_C be the set of instances of type C and P the set of properties having an element of I_C as a subject and let G_C be the subgraph defined by the set of triples of I_C and P, $G_C = \{< i, p, . >: i \in I_C, p \in P\}$.

Definition 1 (Property Support Score).
The support of a property $p \in P$ with respect to the pair (D, C) is defined by:

$$supportProp(p, D, C) = \left| \bigcup_{i \in I_C} < i, p, . > \right| \frac{1}{|I_C|}.$$

In other words, $supportProp(p, D, C) = N\frac{1}{|I_C|}$ means that N instances of type C in the dataset D have a value for the property p ($supportProp(p, D, C) \in [0, 1]$).

As keys for a given class can be composed of one or several properties, we introduce a ranking score for keys based on the supports of their properties, again with respect to their dataset.

Definition 2 (Key Support Score).
Let $K = \{p_1, ..., p_n\}$ be a key corresponding to the pair (D, C), where $p_j \in P, j \in [1, n]$. We define the support of K with respect to (D, C) as

$$supportKey(K, D, C) = \left| \bigcup_{i \in I_C} < i, K, . > \right| \frac{1}{|I_C|},$$

where $< i, K, . >$ means that $\forall p_j \in K, \exists < i, p_j, . > \in G_C$.

In other words, $supportKey(K, D, C)$ can be seen as a measure of the **co-occurrence** of $\{p_1, ..., p_n\}$ in G_C.

To illustrate, let us consider a source dataset D_S having 300 instances of type C_S. Respectively, let D_T be a target dataset having 100 instances of type C_T, where C_S and C_T are two mapped (equivalent) classes, potentially sharing instances. Let K_i and K_j be two merged keys, obtained as described in Algorithm 1, with the following supports for (D_S, C_S) and (D_T, C_T), respectively:

$$supportKey(K_i, D_S, C_S) = \frac{160}{300}; \quad supportKey(K_i, D_T, C_T) = \frac{40}{100};$$
$$supportKey(K_j, D_S, C_S) = \frac{110}{300}; \quad supportKey(K_j, D_T, C_T) = \frac{90}{100}.$$

Obviously, the challenge that arises here is how to rank the merged keys in order to ensure a maximum instance representativeness.

We note that key support score expresses the importance of a merged key with respect to each dataset, however, it is still necessary to provide a ranking function allowing to measure the importance of the merged keys for *both* datasets *simultaneously.*

An intuitive strategy to compute the final support of a merged key, given the supports computed locally in each dataset, would be to compute the average score of these supports. Nevertheless, this strategy would fail to capture all the different scenarios that could lead to a support value. For example, a key having supports 1 and 0.4 in datasets 1 and 2, would have the same merged support than a key having supports of 0.7 and 0.7 in datasets 1 and 2 respectively. Thus, we propose a multiplication function between already computed key supports which ensures better results in the context of data linking evaluation. Consequently, we adopt this ranking function as defined below.

Definition 3 (Merged Keys Rank Function). *We define the rank of a merged key K with respect to two datasets D_S and D_T and two classes C_S and C_T as:*

$$mergedKeysRank(K) = supportKey(K, D_S, C_S) \times supportKey(K, D_T, C_T).$$

Applying the ranking to our example, we obtain the following scores:
$globalRank(K_j) = 0.33; \quad globalRank(K_k) = 0.22; \quad globalRank(K_i) = 0.21;$

Therefore, in this example, the key K_j is more important than K_i which means that intuitively should lead to better data linking results.

4 Evaluation

In order to confirm the effectiveness of the proposed approach, we have conducted an experimental evaluation applying two state-of-the-art key discovery tools: SAKey and ROCKER. We have used two different datasets, a real-world dataset coming from the DOREMUS project[1] and a synthetic benchmark provided by the Instance Matching Track of the Ontology Alignment Evaluation Initiative (OAEI) 2010[2]. The current experiments were applied on links generated semi-automatically using the linking tool SILK. In this evaluation, we highlight a set of issues raised during these experiments. But first, let us define the criteria and the measures used for this evaluation. Two aspects are taken into account through the keys ranking performed using our approach, first the *correctness* that determines whether the discovered links are correct and second, the *completeness* that determines whether all the correct links are discovered. These criteria are evaluated by the help of three commonly used evaluation metrics:

- ***Precision***: expresses the ratio between the cardinalities of the set of valid matchings and all matching pairs identified by the system.
- ***Recall***: expresses the ratio between the cardinalities of the set of valid matchings and the all matching pairs that belong in the reference alignment.
- ***F-Measure***: is computed by the following formula :

$$\text{F-Measure} = 2 * \frac{Precision * Recall}{Precision + Recall}$$

[1] `http://www.doremus.org`
[2] `http://oaei.ontologymatching.org/2010/`

We note that all considered pairs of datasets are using the same ontology model, hence, the ontology mapping process is not considered in our experiments. We first execute SAKey or ROCKER on each dataset in order to identify the set of keys. However, we emphasize the fact that advanced key exceptions like pseudo-keys or almost keys are not the focus of this paper, therefore, only traditional keys are discovered. These keys are then merged and ranked according to their support score. We launch SILK iteratively as many times as the number of the retrieved keys and produce an F-measure at each run by the help of the reference alignment of our benchmark data. We expect to find a monotonic relation between the ranks of keys and the F-measure values produced by SILK by using these keys. Note that the purpose of these experiments is not to evaluate the performance of the linking tools, but to evaluate the quality of the automatically computed ranks of keys. In other words, we assess whether the generated links are increasingly correct in an ascending order of the ranked keys.

4.1 Experiments on the DOREMUS Benchmark

The data in our first experiment come from the DOREMUS project and consists of bibliographical records found in the music catalogs of two major French institutions – *La Bibliothque Nationale de France (BnF)* and *La Philharmonie de Paris (PP)*. These data describe music works and contain properties such as work titles ("Moonlight Sonata"), composer (Beethoven), genre (sonata), opus number, *etc.*. The benchmark datasets were built based on these data with the help of music librarian experts of both institutions, providing at each time sets of works that exist in both of their catalogs, together with a reference alignment. The data were converted from their original MARC format to RDF using the marc2rdf prototype[3] [24]. We consider two benchmark datasets[4], each manifesting a number data heterogeneities:

1) DS1 is a small benchmark dataset, consisting of a source and a target dataset form the BnF and the PP, respectively, each containing 17 music works. These data show recurrent heterogeneity problems such as letters and numbers in the property values, orthographic differences, missing catalog numbers and/or opus numbers, multilingualism in titles, presence of diacritical characters, different value distances, different properties describing the same information, missing properties (lack of description) and missing titles. SAKey produced eight keys in this scenario. The three top-ranked merged keys using our approach are:

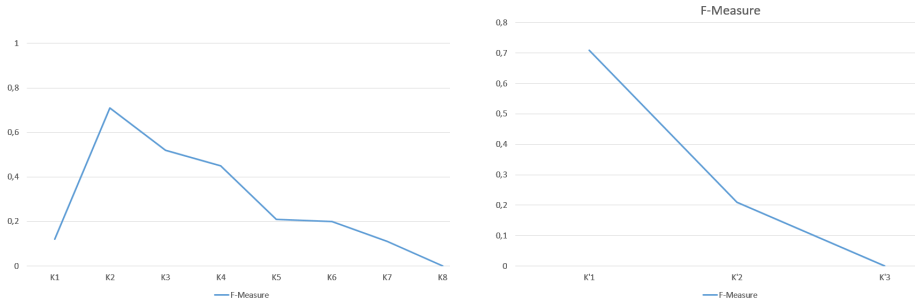
1. K1: $\{P3\_has\_note\}$
2. K2: $\{P102\_has\_title\}$
3. K3: $\{P131\_is\_identified\_by, P3\_has\_note\}$,

where $P3\_has\_note$, $P102\_has\_title$, $P131\_is\_identified\_by$ and $P3\_has\_note$ correspond to a *comment*, *title*, *composer* and *creation date* of a musical work, respectively.

[3] `https://github.com/DOREMUS-ANR/marc2rdf`

[4] Doremus datasets, together with their reference alignments, are available at `http://lirmm.fr/benellefi/doremus-bench`

As we can see in Figure 2(a), our ranking function ensures a decrease of the F-measure with the decrease of the key-rank, in the prominent exception of the top-ranked key, which obtains a very low value of F-Measure. This is explained by the nature of the property *P3_has_note*. This property describes a comment in a free format text written by a cataloguer providing information on the works, creations or authors of such works. The values for this property for the same work are highly heterogeneous (most commonly they are completely different) across the two institutions, which introduces noise and considerably increases the alignment complexity between these resources. Thus, we decided



(a) (b)

Fig. 2: Results by using SAKey on DS1: (a) by considering all properties, (b) without the property *has_note*.

to conduct a second experiment on the same data by removing the property *has_note* in order to confirm our observation. Figure 2 (b) reports the results of this experiment and shows a net decrease of the curve. Overall, the experiment showed that our ranking approach is efficient and the misplaced key is due to the heterogeneous nature of data.

The same experiment has been conducted using this time the key discovery approach ROCKER. The results are reported in Figure 3 showing that the keys were well ranked. Note that, due to the different keys identification definition used by ROCKER, the problematic property *has_note* did not appear in the keys produced by the system.

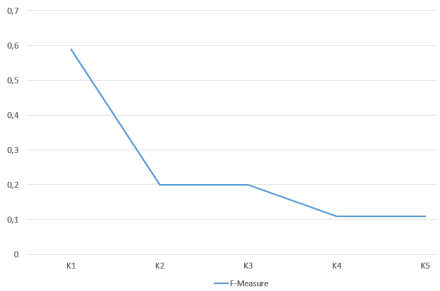


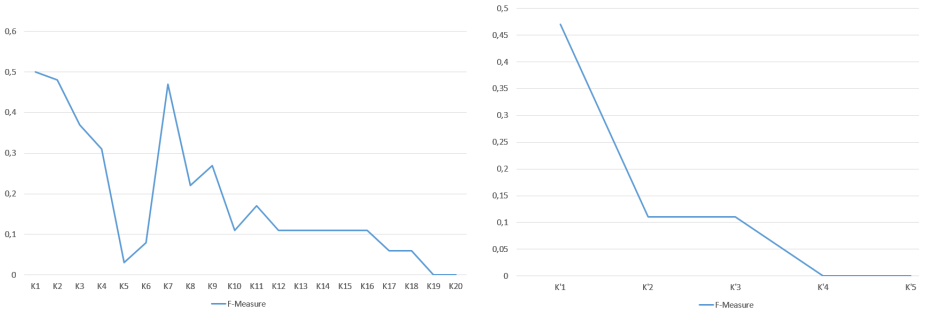
Fig. 3: Results on DS1 by using ROCKER.

2) DS2 is a benchmark dataset consisting of a source and a target dataset from the BnF and the PP, respectively, each composed of 32 music works. Contrarily to DS2, these datasets consist of blocks that are highly similar in their description works (i.e., works of the same composer and with same titles).

The results on this dataset by using SAKey are reported in Figure 4(a). The three top-ranked merged keys are:

1. K1: $\{P3\_has\_note, P102\_has\_title, P131\_is\_identified\_by\}$
2. K2: $\{P3\_has\_note, P102\_has\_title, U35\_had\_function\_of\_type\}$
3. K3: $\{P3\_has\_note, P131\_is\_identified\_by, P3\_has\_note\}$

As their names suggest the properties $P3\_has\_note$ (in *K1* and the first property in *K2*), $P102\_has\_title$,
$P131\_is\_identified\_by$, $U35\_had\_function\_of\_type$ and $P3\_has\_note$ (the third property in *K3*) correspond to a *creation date*, *title*, *composer*, *function of the composer* and *comment* on a musical work, respectively.



(a) (b)

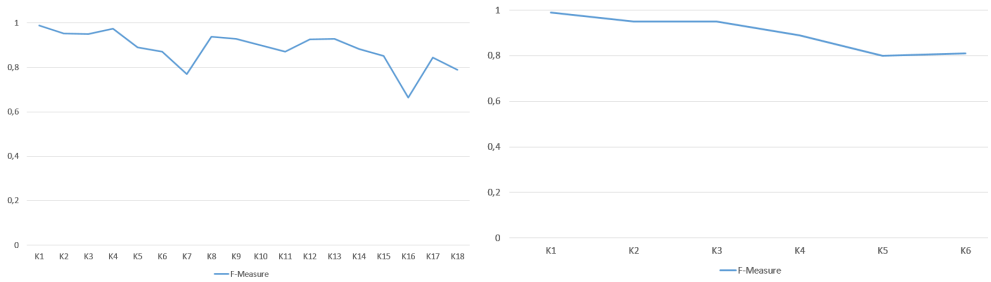
Fig. 4: Results by using SAKey on DS2: (a) by considering all properties, (b) without the property *has_note*.

The results of this experiment are similar to the first one. Not considering the property $P3\_has\_note$ improves considerably (see Figure 4 (a) and (b)) the keys ranking. Indeed, as shown in Figure 4 (a), the key $K5$ which is composed by the properties $P102\_has\_title$, $U35\_had\_function\_of\_type$ and $P3\_has\_note$ has significantly lowered the *f-measure* value; which is not the case of the keys in Figure 4 (b).

4.2 Experiments on the OAEI Benchmark Data

In the second series of experiments, we apply our ranking approach on keys identified in datasets proposed in the instance matching track of OAEI 2010. In this work, we report the obtained results on the dataset $Person1$. The results by using SAKey and ROCKER are shown in Figure 5(a) and (b), respectivey, where one can notice that there is an overall decrease in the F-Measure values in the two cases. Note that in Figure 5(a), there are some problematic key-ranks, showing increase in F-measure while the ranks descend. We observed that SILK achieves

better results comparing string characters than numeric characters. Indeed, this explains why we have had an increasing curve between the keys $K7$ and $K8$, knowing that they are composed of *street* and *house_number* properties (*street* and *surname* properties), respectively.



(a) (b)

Fig. 5: Results on the dataset Person1: (a) by using SAKey, (b) by using ROCKER.

The three top ranked merged keys (in Figure 5(a)) on the dataset Person1 using SAKey are:

1. K1: $\{soc\_sec\_id\}$,
2. K2: $\{given\_name, postcode\}$
3. K3: $\{surname, postcode\}$,

where the properties *soc_sec_id*, *given_name*, *surname* and *postcode* correspond to the *social security number*, *given name*, *surname* and *postal code address* of a person, respectively. In the same manner, we reiterated the experiment using ROCKER which gives better results as shown in Figure 5(b).

4.3 Top Ranked Keys Complementarity

In this evaluation, we want to examine whether using the k (we have taken $k = 3$) top-ranked keys in combination can improve the linking scores as compared to using only one of the top-ranked keys (e.g., the first one) for linking. As discussed above, even if a key is discovered as a first-rank key, nothing ensures that the vocabulary used in both datasets to describe that key is homogeneous. To answer to that, combining a set of top ranked keys would lead to better linking results.

	SAKey						ROCKER			
	Dataset 1			Dataset 2			Dataset 1			Dataset 2
	F	P	R	F	P	R	F	P	R	
K1	*0.12*	*0.12*	*0.11*	*0.5*	**0.75**	*0.37*	*0.59*	**0.8**	*0.47*	No merged key has been identified.
K2	0.71	0.9	0.58	0.48	0.7	0.37	0.2	0.66	0.11	
K3	0.52	1	0.35	0.37	0.56	0.28	0.2	0.66	0.11	
K1+K2+K3	**0.54**	**0.44**	**0.7**	**0.51**	*0.63*	**0.43**	**0.62**	*0.75*	**0.52**	

Table 1: Results of the combination of the three top-ranked keys on the DOREMUS datasets.

Notice that by doing so, the *recall* value remains the same or increases as compared to the single key approach, while the *precision* may increase (if the proportion of the positive matching pairs becomes larger than the negative matching pairs) as it may as well decrease.
As shown in Table 1, the experiments on DOREMUS datasets using the three top ranked keys increased relatively (in bold in the table) the *F-Measure* with respect to the first-rank key (where the improved values are in italics) and significantly the *recall* scores (more positive matching pairs were recovered). Thus, it seems reasonable to conclude that merging the matching results retrieved from the top ranked keys allows to improve significantly the results in terms of *recall*, while this cannot guarantee an improvement in *precision*.

5 Conclusion and Future Work

This paper presents an approach that allows to select automatically a number of merged keys, relevant for a given pair of input datasets, and rank them with respect to their "effectiveness" for the task of discovering `owl:sameAs` links between them. The effectiveness of a merged key is defined as a function of the combination of its respective supports on each of the two input datasets. The proposed method allows to reduce significantly the user effort in the selection of keys used as a parameter of a data linking tool, such as SILK or LIMES. In this way, we attempt to bridge the gap between configuration-oriented approaches, such as automatic key discovery and automatic link specification, and the actual process of data linking. We also look into the complementarity properties of a small set of top-ranked keys and show that their combined use improves significantly the recall. To demonstrate our concepts, we have conducted a series of experiments on data coming from the OAEI campaign, as well as on real-world data from the field of classical music cataloguing.

In near future, we plan to improve our ranking criterion by defining it as a function of the estimated intersection of the sets of instances covered by a given key across two datasets.

Acknowledgements

This work has been partially supported by the French National Research Agency(ANR) within the DOREMUS Project, under grant number ANR-14-CE24-0020.

References

1. C. Bizer, T. Heath, and T. Berners-Lee, "Linked data-the story so far," *Semantic Services, Interoperability and Web Applications*, pp. 205–227, 2009.
2. D. Symeonidou, V. Armant, N. Pernelle, and F. Saïs, "SAKey: Scalable Almost Key discovery in RDF data," in *ISWC 2014*, Springer Verlag, 2014.

3. M. Atencia, J. David, and J. Euzenat, "Data interlinking through robust linkkey extraction.," in *ECAI*, pp. 15–20, 2014.
4. T. Soru, E. Marx, and A. N. Ngomo, "ROCKER: A refinement operator for key discovery," in *WWW 2015*, pp. 1025–1033, 2015.
5. N. Pernelle, F. Saïs, and D. Symeonidou, "An automatic key discovery approach for data linking," *Journal of Web Semantics*, vol. 23, pp. 16–30, 2013.
6. M. Atencia, J. David, and F. Scharffe, "Keys and pseudo-keys detection for web datasets cleansing and interlinking," in *EKAW*, pp. 144–153, 2012.
7. D. Symeonidou, I. Sanchez, M. Croitoru, P. Neveu, N. Pernelle, F. Saïs, A. Roland-Vialaret, P. Buche, A. Muljarto, and R. Schneider in *ICCS*, pp. 222–236, 2016.
8. A.-C. N. Ngomo and K. Lyko, "Eagle: Efficient active learning of link specifications using genetic programming," in *ESWC*, pp. 149–163, Springer, 2012.
9. P. Christen, "Febrl-: an open source data cleaning, deduplication and record linkage system with a graphical user interface," in *SIGKDD*, pp. 1065–1068, ACM, 2008.
10. R. Isele, A. Jentzsch, and C. Bizer, "Efficient multidimensional blocking for link discovery without losing recall.," in *WebDB*, 2011.
11. A.-C. N. Ngomo, J. Lehmann, S. Auer, and K. Höffner, "Raven-active learning of link specifications," in *International Conference on Ontology Matching*, pp. 25–36, CEUR-WS. org, 2011.
12. A. Ferrara, A. Nikolov, and F. Scharffe, "Data linking for the semantic web," *Semantic Web: Ontology and Knowledge Base Enabled Tools, Services, and Applications*, vol. 169, 2013.
13. M. Nentwig, M. Hartung, A.-C. N. Ngomo, and E. Rahm, "A survey of current link discovery frameworks," *Semantic Web*, no. Preprint, pp. 1–18, 2015.
14. A. Jentzsch, R. Isele, and C. Bizer, "Silk–generating rdf links while publishing or consuming linked data," in *ISWC*, Citeseer, 2010.
15. A. N. Ngomo and S. Auer, "LIMES - A time-efficient approach for large-scale link discovery on the web of data," in *IJCAI*, pp. 2312–2317, 2011.
16. C. Shao, L. Hu, J. Li, Z. Wang, T. L. Chung, and J. Xia, "Rimom-im: A novel iterative framework for instance matching," *J. Comput. Sci. Technol.*, vol. 31, no. 1, pp. 185–197, 2016.
17. E. Jiménez-Ruiz and B. C. Grau, "Logmap: Logic-based and scalable ontology matching," in *ISWC*, pp. 273–288, Springer, 2011.
18. A. Nikolov, V. S. Uren, E. Motta, and A. N. D. Roeck, "Integration of semantically annotated data by the knofuss architecture," in *EKAW*, pp. 265–274, 2008.
19. S. Araujo, J. Hidders, D. Schwabe, and A. P. De Vries, "Serimi-resource description similarity, rdf instance matching and interlinking," *arXiv preprint arXiv:1107.1104*, 2011.
20. S. Rong, X. Niu, E. W. Xiang, H. Wang, Q. Yang, and Y. Yu, "A machine learning approach for instance matching based on similarity metrics," in *ISWC*, pp. 460–475, Springer, 2012.
21. M. Kejriwal and D. P. Miranker, "Semi-supervised instance matching using boosted classifiers," in *ESWC*, pp. 388–402, 2015.
22. T. Lesnikova, J. David, and J. Euzenat, "Interlinking english and chinese rdf data using babelnet," in *Proceedings of the 2015 ACM Symposium on Document Engineering*, pp. 39–42, ACM, 2015.
23. P. Shvaiko and J. Euzenat, "Ontology matching: state of the art and future challenges," *IEEE Transactions on knowledge and data engineering*, vol. 25, no. 1, pp. 158–176, 2013.
24. M. Achichi, R. Bailly, C. Cecconi, M. Destandau, K. Todorov, and R. Troncy, "Doremus: Doing reusable musical data," in *ISWC PD*, 2015.