# On learning context-aware rules to link RDF datasets

ANDREA CIMMINO[*], *Universidad Politécnica de Madrid, ETSI Informática, Calle de los Ciruelos, s/n, E-28660 Boadilla del Monte, Spain.*

RAFAEL CORCHUELO[**], *Universidad de Sevilla, ETSI Informática, Avda. Reina Mercedes, s/n, E-41012 Sevilla, Spain.*

## Abstract

Integrating RDF datasets has become a relevant problem for both researchers and practitioners. In the literature, there are many genetic proposals that learn rules that allow to link the resources that refer to the same real-world entities, which is paramount to integrating the datasets. Unfortunately, they are context-unaware because they focus on the resources and their attributes but forget about their neighbours. This implies that they fall short in cases in which different resources have similar attributes but refer to different real-world entities or cases in which they have dissimilar attributes but refer to the same real-world entities. In this article, we present a proposal that learns context-aware rules that take into account both the attributes of the resources and their neighbours. We have conducted an extensive experimentation that proves that it outperforms the most advanced genetic proposal. Our conclusions were checked using statistically sound methods.

## 1 Introduction

An RDF dataset is a collection of resources that describe real-world entities. Such datasets are commonly used to feed a variety of automated business processes [5]. Typically, this requires to integrate them by linking the resources that refer to the same real-world entities [1]. The resources are described by means of properties that can be either data properties or object properties; the former model the attributes of the resources and the latter relate them to their neighbours.

There are several state-of-the-art proposals that use genetic approaches to learn link rules [9, 10, 16, 17]. Such rules basically compute the similarity of two resources by comparing their attributes using a series of string transformations and similarity functions. If the resources are similar enough, then they are linked because they are assumed to describe the same real-world entity; otherwise, they are kept apart. It is not difficult to realize that such context-unaware rules are imprecise in cases in which two resources have similar attributes but describe different real-world entities (e.g. different people who have similar names or ages) or have dissimilar attributes but refer to the same real-world entities (e.g. resources that describe different facets of a person).

In this article, we present an approach to learn context-aware rules building on the context-unaware rules learnt by any of the previous proposals. By context-aware, we mean that the rule takes into account the attributes of the resources being linked and the attributes of their neighbours. This is a novel approach since our analysis of the related work reveals that this is the first time that context-aware rules have been explored in this context. We have also performed an extensive experimental

[*]E-mail: cimmino@fi.upm.es
[**]E-mail: corchu@us.es

study in which we sought to prove two hypothesis, namely, (i) exploring the context helps improve the effectiveness of the link rules and (ii) learning context-aware rules helps improve the efficiency of the linking process. Our experimental results and the statistical analysis that we have conducted validate these hypotheses, which prove that our proposal is very promising. Our proposal is related to a previous one in which our goal was to link two datasets on the fly [3]; our experimentation confirms that our new proposal is as effective as the previous one but increases efficiency significantly.

The rest of the article is organized as follows: Section 2 reports on the related work; Section 3 provides the details of our proposal; Section 4 presents our experimental analysis; finally, Section 5 summarizes our conclusions.

## 2 Related work

Learning link rules originated in the field of relational databases, where the problem was known as de-duplication [15], collective matching [18] or entity matching [13]. Unfortunately, it is not straightforward to adapt these results to RDF datasets because of the gap between the underlying data models.

Some authors have developed a number of proposals that are specifically tailored to working with RDF datasets. Unfortunately, some of them work on a single dataset [8, 14] and others require the datasets to be modelled using OWL ontologies [4, 6, 7, 12], which hinders their general applicability. There are a few proposals that work on two RDF datasets without an explicit model [9, 10, 16, 17, 20], which makes them generally applicable. We analyse them below.

Isele and Bizer [9, 10] devised GenLink, which is a genetic approach that uses a tournament selection operator, a generational replacement operator, custom cross-over and mutation operators and a fitness function that relies on the Matthews correlation coefficient. It can use a variety of custom string transformation functions and the Levenshtein, Jaccard, numeric, geographic and date string similarity functions. Ngomo and Lyko [16] devised EAGLE, another genetic approach that uses a tournament selection operator, a $\mu + \beta$ replacement operator, tree cross-over and mutation operators and a fitness function that relies on the $F_1$ score. It does not use any transformation functions, but the Levenshtein, Jaccard, cosine, Q-grams, overlap and trigrams string similarity functions. Nikolov *et al.* [17] contributed with another genetic proposal that uses a roulette-wheel selection operator, an elitist replacement operator, a tree cross-over operator, a custom mutation operator and a pseudo $F_1$ fitness function. Transformations are not considered, but the library of similarity functions includes Jaro, Levenshtein and I-Sub. The proposal by Soru and Ngomo [20] transforms the input RDF datasets into feature vector sets to which any standard machine-learning technique can be applied to learn a classification model that makes the resources to be linked apart from the others. It does not take any transformation functions into account, but it can work with the Q-grams, the cosine and the Levenshtein string similarity functions.

Unfortunately, the previous proposals can learn context-unaware rules only, i.e. rules that compare the resources building on their attributes, but not their neighbours. This means that they have difficulties to deal with resources that are similar according to their attributes but are actually different or vice versa. Analysing the neighbourhood of the resources may help discern the previous cases. We have devised two instance-driven approaches [2, 3] that take the neighbours of the resources to be linked into account, which proved to be a good idea regarding effectiveness. Unfortunately, none of the approaches is efficient enough since they do not learn any rules; i.e. the effort spent at executing them on two particular datasets cannot be reused when dealing with similar datasets. This motivated us to work on the current proposal.

```
1: method $learnContextAwareRule(R, S, D_1, D_2, \gamma, \theta)$
2:     $K := computeCorrespondences(R, S, D_1, D_2, \theta)$
3:     $P := \{C \mid \exists T, P_1, P_2 : C = |links(K, T, P_1, P_2)|\}$
4:     $V := \{(T, P_1, P_2) \mid |links(K, T, P_1, P_2)| \geq \gamma \max P\}$
5:     $W := instantiate(R, V, \theta)$
6:     return $W$
```

FIGURE 1. Method to learn context-aware rules.

## 3 Our proposal

Our proposal[1] consists in a component that learns a set of context-unaware rules using any existing technique and a component that analyses each rule individually and learns its context-aware version. Below, we focus on the second component, which is our original contribution; first, we describe our method to learn link rules; then, we present two ancillary methods; next, we illustrate the proposal with an example; finally, we analyse its computational complexity.

### *3.1 Learning a context-aware rule*

Figure 1 provides the pseudo-code to the method to learn context-aware rules. It works on a base rule $R$, a set of support rules $S$, two datasets $D_1$ and $D_2$ and two parameters $\gamma$ and $\theta$, which we explain below. We assume that the rules have been learnt with the first component, i.e. they are context-unaware; we also assume that the datasets provide resources that are representative enough of the resources that we wish to link in future. Our goal is to learn a context-aware rule that combines base rule $R$ with a subset of support rules $S$ to improve the precision when linking similar datasets. Our proposal works in three steps: it first learns a set of correspondences, next filters some of them out, and then instantiates a template to produce the resulting rule.

The first step consists in computing a set of correspondences $K$ at line 2. We describe the details of this ancillary method in the following subsection. To understand the main method, it suffices to know that the correspondences are tuples of the form $(A, B, T, P_1, P_2)$, where $A$ and $B$ denote two resources that are linked by means of base rule $R$; $P_1$ and $P_2$ denote two paths, i.e. two sequences of object properties that relate resources $A$ and $B$ to two subsets of direct or indirect neighbours; and $T$ denotes a support rule that establishes some links among the previous subsets of neighbours. There can be many correspondences, but the method filters out the ones in which the neighbours cannot be considered similar enough according to the links found by support rule $T$. Intuitively, the correspondences keep the links whose context can be considered similar enough not to be false positive links.

The second step consists in selecting the support rules in $K$ that have generated enough links, which is performed by the statements at lines 3–4. We use a threshold $\gamma$ to set the minimum number of links that a support rule must have generated so that it can be selected. (The most appropriate value of $\gamma$ for a particular learning problem can be easily set using grid search.) First, the statement at line 3 computes a set with the counters of links in $K$ that have been generated by each support rule $T$ given two paths $P_1$ and $P_2$. It relies on an ancillary function called *links*, which is defined as follows:

$$links(K, T, P_1, P_2) \quad = \quad \{(A, B) \mid (A, B, T, P_1, P_2) \in K\}.$$

[1] Our prototype is available at https://github.com/AndreaCimminoArriaga/Sorbas.

The set of support rules selected is then defined as the set of support rules that generate at least $\gamma$ percent the maximum number of links generated by a support rule, which is computed and stored in variable $V$ at line 4. Note that $V$ stores triples in which each support rule is accompanied by two paths; the reason is that a link rule may generate many links, but we are interested in links among resources that are directly or indirectly related to the original ones only.

The final step consists in generating the resulting context-aware rule by means of the statement at line 5. Method *instantiate* takes the base rule $R$, the set of links $V$ and the parameter $\theta$ as inputs and instantiates the following template:

$$\begin{aligned} \mathit{link}(A,B) \quad \text{if} \quad & \mathbf{R}(A,B) \wedge \exists(T,P_1,P_2) \in \mathbf{V}: \\ & X = \mathit{findResources}(A,P_1) \wedge Y = \mathit{findResources}(B,P_2) \wedge \\ & E = \{(U,V) \mid U \in X \wedge V \in Y \Rightarrow T(U,V)\} \wedge \\ & \mathit{computeSimilarity}(X,Y,E) \geq \boldsymbol{\theta} \end{aligned}$$

Intuitively, resources $A$ and $B$ can be linked if they are linked by the base rule and their neighbourhoods are similar enough. The neighbours are similar enough if there is at least one support rule $T$ with paths $P_1$ and $P_2$, such that let $X$ denote the neighbours of resource $A$ by following path $P_1$, let $Y$ denote the neighbours of resource $B$ by following path $P_2$, and let $E$ be the set of links that support rule $T$ finds among $X$ and $Y$; the neighbours are similar enough if $X$ and $Y$ are similar enough according to the links found by the support rule.

Method *findResources* is very simple, so we do not provide any additional details. In the subsections below, we delve into the intricacies of computing correspondences and similarities.

## *3.2 Computing correspondences*

Figure 2 provides the pseudo-code to the method to compute correspondences. It works on a base rule $R$, a set of support rules $S$, two datasets $D_1$ and $D_2$ and a similarity threshold $\theta$. It returns a set of correspondences $K$.

The statements at lines 2–5 perform the initialization steps. The method first initializes $K$ to an empty set and stores the source and the target classes of the base rule in sets $R_S$ and $R_T$, respectively, and the links that result from applying it to the input datasets in set $L_1$. For the sake of efficiency, we also compute the models of the input datasets, i.e. we infer the classes and their connecting properties from their resources in the input datasets.

The main loop at lines 6–15 then iterates over the set of support rules using variable $T$. It first computes the sets of source and target classes involved in rule $T$, which are stored in variables $R_S$ and $R_T$, respectively; next, it finds the set of paths $Q_1$ that connect the source classes in $R_S$ with the source classes in $T_S$ in dataset $D_1$; similarly, it finds the set of paths $Q_2$ that connect the target classes in $R_T$ with the target classes in $T_T$ in dataset $D_2$. The idea is to find the paths that relate the resources linked by the base rule with the resources linked by the support rule, which is done by the middle and the inner loops.

The middle loop at lines 10–15 iterates over the set of pairs of paths $(P_1,P_2)$ from the Cartesian product of $Q_1$ and $Q_2$. If there is at least a pair of paths, it then means that the resources involved in the links returned by base rule $R$ might have some neighbours that might be linked by support rule $T$. The inner loop at lines 11–15 iterates over the collection of links $(A,B)$ in set $L_1$, i.e. the links returned by the base rule. It first finds the set of resources $X$ that are related to resource $A$ using path $P_1$ in dataset $D_1$ and the set of resources $Y$ that are related to resource $B$ using path $P_2$ in dataset $D_2$.

```
1: method $computeCorrespondences(R, S, D_1, D_2, \theta)$
2:     $K := \emptyset$
3:     $(R_S, R_T) := (sourceClasses(R), targetClasses(R))$
4:     $L_1 := apply(R, D_1, D_2)$
5:     $(M_1, M_2) := (computeModel(D_1), computeModel(D_2))$
6:     for each rule $T \in S$ do
7:         $(T_S, T_T) := (sourceClasses(T), targetClasses(T))$
8:         $(Q_1, Q_2) := (findPaths(R_S, T_S, M_1), findPaths(R_T, T_T, M_2))$
9:         $L_2 = apply(T, D_1, D_2)$
10:        for each $(P_1, P_2) \in Q_1 \times Q_2$ do
11:            for each link $(A, B) \in L_1$ do
12:                $(X, Y) := (findResources(A, P_1, D_1), findResources(B, P_2, D_2))$
13:                $E := L_2 \cap (X \times Y)$
14:                if $computeSimilarity(X, Y, E) \geq \theta$ then
15:                    $K := K \cup \{(A, B, T, P_1, P_2)\}$
16:    return $K$
```

FIGURE 2. Method to learn correspondences.

```
1: method $computeSimilarity(X, Y, E)$
2:     $E^\star := Floyd\text{-}Warshall(E)$
3:     $X' := reduce(X, E^\star)$
4:     $Y' := reduce(Y, E^\star)$
5:     $W := intersect(X', Y', E^\star)$
6:     $D := |W| / \min\{|X'|, |Y'|\}$
7:     return $D$
```

FIGURE 3. Method to compute similarity.

Next, the method applies support rule $T$ to datasets $D_1$ and $D_2$ and intersects the resulting links with $X \times Y$ in order to filter out the resources that are not neighbours of $A$ or $B$; the result is stored in set $E$. It then computes the similarity of sets $A$ and $B$. If it is greater than or equal to threshold $\theta$, then correspondence $(A, B, T, P_1, P_2)$ is added to set $K$; otherwise, it is filtered out.

Method *findPaths* returns the collection of paths between two given sets of classes from a given dataset using Dijkstra's algorithm. Method *computeModel* is implemented using the approach by Rivero *et al.* [19]. We do not provide any additional details regarding methods *sourceClasses*, *targetClasses*, *findResources* and *apply* because they are straightforward.

## *3.3 Computing similarity*

Figure 3 shows our method to compute similarity. Its input consists of sets $X$ and $Y$, which are two sets of resources, and $E$, which is a set of links between them. It returns the Szymkiewicz–Simpson overlapping coefficient, namely,

$$overlap(X, Y) \quad = \quad \frac{|X \cap Y|}{\min\{|X|, |Y|\}}$$

The previous formula assumes that there is an implicit equality relation to compute $X \cap Y$, $|X|$, or $|Y|$. In our context, this relation must be inferred from the set of links $E$ by means of

Floyd–Warshall's algorithm to compute the reflexive, commutative, transitive closure of relation $E$, which we denote as $E^\star$.

The method to compute similarities relies on two ancillary methods, namely, *reduce*, which given a set of resources $X$ and a set of links $E$ returns a set whose elements are subsets of $X$ that are equal according to $E^\star$, and *intersect*, which given two reduced sets of resources $X$ and $Y$ and a set of links $E$ returns the intersection of $X$ and $Y$ according to $E^\star$. Their definitions are as follows:

$$\begin{aligned} \mathit{reduce}(X, E) &= \{W \mid W \propto W \subseteq X \wedge W \times W \subseteq E^\star\} \\ \mathit{intersect}(X, Y, E) &= \{W \mid W \propto W \subseteq X \wedge \exists W' : W' \subseteq Y \wedge W \times W' \in E^\star\} \end{aligned}$$

where $X \propto \phi$ denotes the maximal set $X$ that fulfils predicate $\phi$, i.e.

$$X \propto \phi \quad \iff \quad \phi(X) \wedge (\nexists X' : X \subseteq X' \wedge \phi(X'))$$

The method to compute similarities then works as follows: it first computes the closure of $E$ at line 2; it then reduces the input sets of resources $X$ and $Y$ according to the closure of $E$ at lines 3–4; next, it computes the intersection of both reduced sets at line 5; finally, it computes the similarity using Szymkiewicz–Simpson's formula on the reduced sets at line 6.

## 3.4 *An illustration*

Figure 4 presents an excerpt of the DBLP dataset and an excerpt of the NSF dataset. The first component learns the following rules in this scenario, where *levenshtein* and *jaccard* denote the well-known string similarity functions, *lfname* denotes a function that transforms names into format 'last name, first name', and *lowercase* denotes a function that changes a string into lowercase:

$$\begin{aligned} r_1: \quad \mathit{link}(A, R) \quad & \text{if } \mathit{rdf}{:}\mathit{type}(A) = \mathit{dblp}{:}\mathit{Author}, \mathit{rdf}{:}\mathit{type}(R) = \mathit{nsf}{:}\mathit{Researcher}, \\ & N_A = \mathit{dblp}{:}\mathit{name}(A), N_R = \mathit{nsf}{:}\mathit{name}(R), \\ & \mathit{levenshtein}(\mathit{lfname}(N_A), \mathit{lfname}(N_R)) \geq 0.80. \end{aligned}$$

$$\begin{aligned} r_2: \quad \mathit{link}(A, P) \quad & \text{if } \mathit{rdf}{:}\mathit{type}(A) = \mathit{dblp}{:}\mathit{Article}, \mathit{rdf}{:}\mathit{type}(P) = \mathit{nsf}{:}\mathit{Paper}, \\ & T_A = \mathit{dblp}{:}\mathit{title}(A), T_P = \mathit{nsf}{:}\mathit{title}(P), \\ & \mathit{jaccard}(\mathit{lowercase}(T_A), \mathit{lowercase}(T_P)) \geq 0.65. \end{aligned}$$

$$\begin{aligned} r_3: \quad \mathit{link}(A, U) \quad & \text{if } \mathit{rdf}{:}\mathit{type}(A) = \mathit{dblp}{:}\mathit{Affiliation}, \mathit{rdf}{:}\mathit{type}(U) = \mathit{nsf}{:}\mathit{Award}, \\ & T_A = \mathit{dblp}{:}\mathit{name}(A), T_U = \mathit{nsf}{:}\mathit{uid}(U), \\ & \mathit{jaccard}(\mathit{lowercase}(T_A), \mathit{lowercase}(T_U)) \geq 0.95. \end{aligned}$$

Intuitively, rule $r_1$ is applied to a resource $A$ of type *dblp:Author* and a resource $R$ of type *nsf:Researcher*; it computes the Levenshtein similarity between the names of the author and the researcher after transforming them; if it is at least 0.80, then the corresponding resources are linked. Rules $r_2$ and $r_3$ should now be easy to understand.

Let us analyse the case in which the base rule is $r_1$ and the support rules are $r_2$ and $r_3$; i.e. we are interested in linking DBLP authors and NSF researchers. Rule $r_1$ returns the following links: $\{(\mathit{dblp}{:}\mathit{weiwang}, \mathit{nsf}{:}\mathit{weiwang}_1), (\mathit{dblp}{:}\mathit{weiwang}, \mathit{nsf}{:}\mathit{weiwang}_2), (\mathit{dblp}{:}\mathit{binliu}, \mathit{nsf}{:}\mathit{binwliu})\}$; note that the

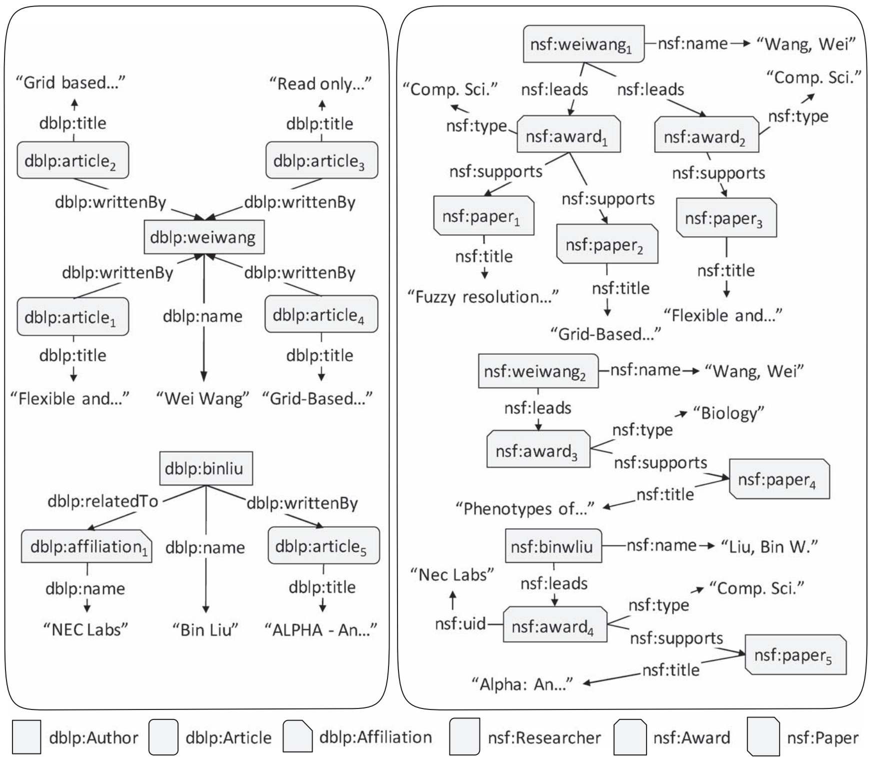


FIGURE 4. Running example.

first and the third links are true positive links, but the second one is a false positive link. Rule $r_2$ returns the following links: $\{(dblp{:}article_1, nsf{:}paper_3), (dblp{:}article_2, nsf{:}paper_2), (dblp{:}article_4, nsf{:}paper_2), (dblp{:}article_5, nsf{:}paper_5)\}$, which are true positive links. Finally, rule $r_3$ returns $\{(dblp{:}affiliation_1, nsf{:}award_4)\}$, which are also true positive links. The sets of paths between the source and target classes of $r_1$ and $r_2$ are $\{\langle dblp{:}writtenBy\rangle\}$ and $\{\langle nsf{:}leads, nsf{:}supports\rangle\}$. Furthermore, the paths between the source and target classes of $r_1$ and $r_3$ are $\{\langle dblp{:}relatedTo\rangle\}$ and $\{\langle nsf{:}leads\rangle\}$, respectively.

Link $l_1 = (dblp{:}weiwang, nsf{:}weiwang_1)$ is analysed first. The method finds $X = \{dblp{:}article_1, dblp{:}article_2, dblp{:}article_3, dblp{:}article_4\}$ by following resource *dblp:weiwang* through path $\langle dblp{:}writtenBy\rangle$; similarly, it finds $Y = \{nsf{:}paper_1, nsf{:}paper_2, nsf{:}paper_3\}$ by following resource *nsf:weiwang*$_1$ through path $\langle nsf{:}leads, nsf{:}supports\rangle$. Now, support rule $r_2$ is applied and the results are intersected with $X \times Y$ so as to keep the links that are related to $l_1$ only; the result is $E = \{(dblp{:}article_1, nsf{:}paper_3), (dblp{:}article_2, nsf{:}paper_2), (dblp{:}article_4, nsf{:}paper_2)\}$. Then, the similarity of $X$ and $Y$ in the context of $E$ is computed, which returns 0.67; intuitively, there are chances that $l_1$ is a true positive link.

Link $l_2 = (dblp{:}weiwang, nsf{:}weiwang_2)$ is analysed next. The method finds $X = \{dblp{:}article_1, dblp{:}article_2, dblp{:}article_3, dblp{:}article_4\}$ by following resource *dblp:weiwang* through path $\langle dblp{:}writtenBy\rangle$; next, it finds $Y = \{nsf{:}paper_4\}$ by following resource *nsf:weiwang*$_2$ through path $\langle nsf{:}leads, nsf{:}supports\rangle$. Now support rule $r_2$ is applied and the result is intersected with $X \times Y$, which results in $E = \emptyset$. In this case, the similarity is zero, which intuitively indicates that it is very likely that $l_2$ is a false positive link.

Link $l_3 = (\mathit{dblp{:}binliu}, \mathit{nsf{:}binwliu})$ is analysed next. The method finds $A_1 = \{\mathit{dblp{:}article}_5\}$ by following resource *dblp:binliu* through path $\langle \mathit{dblp{:}writtenBy} \rangle$; next, it finds $Y_1 = \{\mathit{nsf{:}paper}_5\}$ by following resource *nsf:binwliu* through path $\langle \mathit{nsf{:}leads}, \mathit{nsf{:}supports} \rangle$. Now support rule $r_2$ is applied and the result is intersected with $X_1 \times Y_1$, which results in $E = \{(\mathit{dblp{:}article}_5, \mathit{nsf{:}paper}_5)\}$. The similarity is now 1.00, i.e. it is very likely that link $l_3$ is a true positive link. The method then finds $X_2 = \{\mathit{dblp{:}affiliation}_1\}$ by following resource *dblp:binliu* through path $\langle \mathit{dblp{:}relatedTo} \rangle$; next, it finds $X_2 = \{\mathit{nsf{:}award}_4\}$ by following resource *dblp:binwliu* through path $\langle \mathit{nsf{:}supports} \rangle$. Now support rule $r_3$ is applied and the result is intersected with $X_2 \times Y_2$, which results in $E = \{(\mathit{dblp{:}affiliation}_1, \mathit{nsf{:}award}_4)\}$. The similarity is 1.00, which means that $l_3$ is likely to be a true positive link.

Assume that $\theta = 0.50$ and $\gamma = 0.70$, which are intended for illustration purposes only. Method *computeCorrespondences* returns set $K = \{(\mathit{dblp{:}weiwang}, \mathit{nsf{:}weiwang}_1), (\mathit{dblp{:}binliu}, \mathit{nsf{:}binwliu})\}$. We now have to analyse support rules $r_2$ and $r_3$ and the links that they produced given the paths shown previously. Support rule $r_2$ produces links $l_1$ and $l_3$, and support rule $r_3$ produces link $l_3$ only; therefore, every support rule that returns at least $2\,\gamma = 1.40$ links using the previous paths is selected. In other words, support rule $r_2$ is selected and support rule $r_3$ is discarded. The resulting context-aware rule is as follows:

$$\begin{aligned}
p^*\!: \quad \mathit{link}(A,B) \quad & \text{if } r_1(A,B),\\
& X = \mathit{findResources}(A, \langle \mathit{dblp{:}writtenBy} \rangle),\\
& Y = \mathit{findResources}(B, \langle \mathit{nsf{:}leads}, \mathit{nsf{:}supports} \rangle),\\
& E = \{(U,V) \mid U \in X \wedge V \in Y \Rightarrow r_2(A,B)\},\\
& \mathit{computeSimilarity}(X,Y,E) \geq 0.50.
\end{aligned}$$

The intuitive interpretation is that $p^*$ links resources $A$ and $B$ if rule $r_1$ links them and they have some neighbours (using paths $\langle \mathit{dblp{:}writtenBy} \rangle$ and $\langle \mathit{nsf{:}leads}, \mathit{nsf{:}supports} \rangle$), respectively) that can be linked by means of rule $r_2$, and it results in two sets of resources whose similarity is at least 0.50.

## *3.5 Complexity analysis*

Our goal is to prove that our proposal is computationally tractable, i.e. there is a polynomial upper limit to the time and space required to run it.

LEMMA 1
Method *computeSimilarity* does not require more than $O(n^3)$ time and $O(n^2)$ space to execute, where $n$ denotes the number of resources in the largest input dataset.

PROOF. The proof proceeds in three steps: first, we present some preliminaries, next analyse the time complexity, and, finally, study the space complexity.

Preliminaries: the method works on two sets of resources $X$ and $Y$ and a set of links $E$. We denote $n = \max\{|D_1|, |D_2|\}$. We can safely assume that $O(n)$ is an upper bound to both the number of resources in $X$ and $Y$ and that $O(n\,(n-1)/2) \subseteq O(n^2)$ is an upper limit to the number of links in $E$ since this is the maximum number of edges in a graph with $n$ vertices.

Time complexity: the statement at line 2 applies Floyd–Warshall's algorithm to the input set of links $E$. This algorithm requires cubic time in the number of vertices of the input graph, which implies that $O((n+n)^3) \subseteq O(n^3)$ is an upper limit to the execution time. The statements at lines 3–4 reduce the input sets $X$ and $Y$ using the closure of $E$. A trivial implementation checks the pairs in the self-Cartesian products of $X$ and $Y$; thus, $O(n^2)$ is an upper limit to the time that each reduction requires. The statement at line 5 intersects the reductions of $X$ and $Y$. A trivial implementation checks the pairs in the Cartesian product of $X$ and $Y$, which implies that $O(n^2)$ is an upper limit to the time required. The statement at line 6 performs a simple arithmetic operation that is not expected to require more than $O(1)$ time. As a conclusion, method *computeSimilarity* does not require more than $O(n^3 + 2\,n^2 + n^2 + 1) \subseteq O(n^3)$ time to execute.

Space complexity: the statement at line 2 relies on Floyd–Warshall's algorithm, which is quadratic in the number of vertices of the input graph, which implies that it does not require more than $O(n^2)$ in our proposal. The statements at lines 3–4 reduce the input sets $X$ and $Y$, which may not require more than $O(n)$ space since this is an upper limit to the sizes of the input sets of resources. The statement at line 5 intersects the reductions of $X$ and $Y$, which cannot result in more than $O(n)$ resources. The statement at line 6 performs a simple arithmetic operation that requires $O(1)$ space. As a conclusion, method *computeSimilarity* does not require more than $O(n^2 + 2\,n + n + 1) \subseteq O(n^2)$ space to execute. □

LEMMA 2
Method *computeCorrespondences* does not require more than $O(n^4)$ time and $O(n^2)$ space to execute, where $n$ denotes the number of resources in the largest input dataset.

PROOF. The proof proceeds in three steps: first, we present some preliminaries, next analyse the time complexity and, finally, study the space complexity.

Preliminaries: the method works on a rule $R$, a set of support rules $S$, two datasets $D_1$ and $D_2$ and a threshold $\theta$. We denote the classes from the input datasets that are involved in these rules as $C_1$ and $C_2$, respectively; we also denote $n = \max\{|D_1|, |D_2|\}$, $s = |S|$ and $c = \max\{|C_1|, |C_2|\}$.

Time complexity: we first analyse the initialization statements. Line 2 does not require more than $O(1)$ time to run. Line 3 does not require more than $O(1)$ time because the subsets of source and target classes of rule $R$ can be stored with the rule so that it does not require any additional computations. Line 4 requires to iterate over the Cartesian product of $D_1$ and $D_2$ to apply rule $R$ to each pair; we can safely assume that applying a rule can be implemented in $O(1)$ time since the overall computation is dominated by computing the Cartesian product, which requires $O(n^2)$ time. Blocking techniques can be used to reduce the cardinality of the Cartesian product [11], but we only need to find a computationally tractable upper limit. Thus, we can be safely assume that the actual implementation of line 4 will not require more than $O(n^2)$ time. Line 5 computes the models of the input datasets, which we can perform using Rivero *et al.* [19] approach. It runs in $O(e)$ time, where $e$ denotes the number of properties in the input dataset; since such number cannot exceed $n\,(n-1)/2$, then the statement at line 5 does not require more than $O(n^2)$ time to execute. As a conclusion, the initialization statements do not require more than $O(1 + 1 + n^2 + n^2) \subseteq O(n^2)$ time.

We now analyse the main loop at lines 6–15, which executes $s$ times. The statement at line 7 does not consume more than $O(1)$ time if the classes involved in a rule are stored with the rule. The statement at line 8 finds the paths in the models of the input datasets that connect the source classes to the target classes. We use Dijkstra's algorithm, which means that computing the path from a single class requires $O(c + e\,\log e)$ time, where $e$ denotes the number of properties in the model; note that $e$ cannot exceed $c\,(c-1)/2$, which implies that computing the path from a single class does not require more than $O(c + c\,(c-1)/2\,\log(c\,(c-1)/2) \subseteq O(c^2\,\log c)$; simply put: computing a maximum of $c$ paths cannot require more than $O(c\,c^2\,\log c) \subseteq O(c^3\log c)$ time. The statement at line 9 applies rule $T$ to the input datasets, which we concluded does not require more than $O(n^2)$ time. The middle loop at lines 10–15 cannot execute more than $O(c)$ times because we compute a path per class. The inner loop at lines 11–15 does not execute more than $O(n^2)$ times, which would be the case if every resource could be linked to every other resource. Line 12 finds the resources that are reachable from two given resources along two given paths, which may not require more than $O(2\,n) \subseteq O(n)$ time. Line 13 is dominated by the computation of the Cartesian product, which does not require more than $O(n^2)$ time. Finally, the if statement at lines 14–15 does not require more than $O(n^2)$ time, which we proved before. Thus, the inner loop does not require more than $O(n^2\,(n + n^2 + n^2)) \subseteq O(n^4)$ time to execute; consequently, the middle loop does not require more than $O(c\,n^4)$ time to execute; as a conclusion, the main loop does not require more than $O(s\,(1+c^3\,\log c+c\,n^4)) = O(s+s\,c^3\,\log c+s\,c\,n^4)$ time. Realize that the computation is dominated by the number of resources $n$, which is expected to be much larger than the number of support rules $s$ or the number of classes $c$; i.e. the previous time order is a subset of $O(n^4)$.

Summing up: method *computeCorrespondences* does not require more than $O(n^2 + n^4) \subseteq O(n^4)$ time to execute.

Space complexity: we first analyze the initialization statements. Line 2 requires no more than $O(1)$ space because it simply stores an empty set; line 3 computes the sources and target classes, which may not require more than $O(2\,c) \subseteq O(c)$ space; line 4 requires to store the links in the Cartesian product of $D_1$ and $D_2$, which may not require more than $O(n^2)$ space. (As it was the case before, blocking techniques can be used [11], but this will not contradict the previous upper limit.) That is, the initialization steps do not require more than $O(1 + c + n^2)$ space; since the number of classes is expected to be much smaller than the number of resources, then we can safely assume that the initialization does not require more than $O(n^2)$ space.

We now analyse the main loop. The only statement that accumulates data is the one at line 15, which computes the resulting set of correspondences. The correspondences involve the resources in the input datasets, the support rules and the paths from the source classes to the target classes. That means that $O(n^2\,s\,c)$ is an upper limit to the number of correspondences; taking into account that the number of resources $n$ is expected to be much larger than the number of support rules or classes, we can then safely assume that storing the resulting correspondences does not require more than $O(n^2)$ space. In the main loop, the statement at line 7 does not require more than $O(2\,c) \subseteq O(c)$ space to store the classes, the statement at line 8 does not require more than $O(2\,c) \subseteq O(c)$ space to store the paths and the statement at line 9 does not require more than $O(n^2)$ space to store the links. In the inner loop, the statement at line 12 may not require more than $O(2\,n)$ space to store the resources, the statement at line 13 may not require more than $O(n^2)$ space to store the links and the statement at line 14 was proven not to require more than $O(n^2)$ space. As a conclusion, the computation in the main loop is dominated by accumulating the resulting correspondences, which does not require more than $O(n^2)$ space.

Summing up: method *computeCorrespondences* does not require more than $O(n^2 + n^2) \subseteq O(n^2)$ space. □

THEOREM 1
$O(n^4)$ and $O(n^2)$ are upper limits to the time and space complexity of method *learnContextAwareRule*, where $n$ denotes the number of resources in the largest input dataset.

PROOF. The proof proceeds in three steps: first, we present some preliminaries, next analyse the time complexity and, finally, study the space complexity.

Preliminaries: the method works on a rule $R$, a set of support rules $S$, two datasets $D_1$ and $D_2$ and two thresholds $\gamma$ and $\theta$. We denote $n = \max\{|D_1|, |D_2|\}$ and $s = |S|$.

Time complexity: the statement at line 2 computes the correspondences, which was proved not to require more than $O(n^4)$ time. The statements at lines 3–4 compute the counters of links and filter them out; since there cannot be more than $O(n^2)$ links, it is safe to assume that this is an upper limit to the time required to execute these statements. Finally, the statement at line 5 simply instantiates a rule template, which does not require more than $O(1)$ time. As a conclusion, the method does not require more than $O(n^4 + 2\,n^2 + 1) \subseteq O(n^4)$ time. Please, realize that this is an upper limit which proves that the algorithm is computationally tractable. Our experimental study confirms that the timings on a commodity workstation are very good.

Space complexity: the statement at line 2 returns the correspondences, which are stored locally. Before, we proved that the space requirements for the correspondences do not exceed $O(n^2)$ space. The statements at lines 3–4 do not require more than $O(n^2)$ space to store the selected links. The final statement at line 5 does not require more than $O(1)$ space. As a conclusion, the method does not require more than $O(n^2 + 2\,n^2 + 1) \subseteq O(n^2)$ space. Please, realize that this is an upper limit which proves that the algorithm is computationally tractable. Our experimental study confirms that the memory consumption is within the limits of a commodity workstation. □

## 4 Experimental analysis

In this section, we first describe our experimental environment and then comment on our results regarding effectiveness and efficiency. The research hypothesis behind the two experiments presented in this section are as follows:

- $H_1$: exploiting the context of the resources improves the results in terms of effectiveness. More specifically, it can increase precision without degrading recall significantly.
- $H_2$: learning context-aware rules improves the results in terms of efficiency. More specifically, there is a balance between the time required to learn a rule and the time required to apply it.

### *4.1 Experimental environment*

We run our experiments on a workstation that was equipped with four Intel Xeon E5-2690 cores at 2.60 GHz and 4 GiB of RAM. The operating system was CentOS Linux 7.3. We implemented our proposal with Java 1.8 and the following components: the GenLink implementation provided by Silk 2.6.0 to generate context-unaware link rules; Jena TDB 3.2.0 to work with RDF datasets; ARQ 3.2.0 to work with SPARQL queries; and Simmetrics 1.6.2, SecondString 2013-05-02, and JavaStringSimilarity 1.0.1 to compute string similarities.

We set up the following evaluation scenarios[2] : (i) researchers, which focuses on the top 100 DBLP authors and the top 130 NSF researchers with the same names; (ii) authors, which focuses on the 9 076 DBLP authors with the same names who are known to be different people; (iii) films,

[2]The datasets are available at http://dx.doi.org/10.5281/zenodo.2555034.

which focuses on 691 BBC movies and 445 DBpedia films that have similar titles; (iv) movies, which focuses on 96 DBpedia movies and 101 IMDb films that have similar titles; (v) publications, which focuses on 108 RAE publications and 98 Newcastle papers that are similar; (vi) restaurants, which focuses on 113 and 752 restaurants from Restaurantz; (vii) Persons1, which focuses on 500 and 500 instances of people whose names are very similar but are different people and vice versa; and (viii) Persons2, which is similar to the previous one but focuses on 400 and 600 different people. We also used some Doremus scenarios, which provide data about music works that are catalogued by the French National Library and the Paris Philharmonic, namely, (ix) Doremus16-9ht, which focuses on 40 and 40 music works that are exactly the same but have differences in spelling or classification; (x) Doremus16-fp, which focuses on 85 and 41 music works that are different but have very similar attributes; (xi) Doremus17-ht, which focuses on 238 and 238 music works that are the same, but have different attributes; and (xii) Doremus17-fp, which focuses on 75 and 75 music works that are different but have similar attributes.

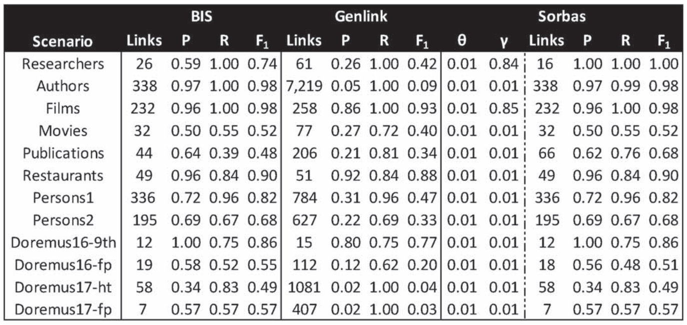
The experimental competitors are BIS, which is our instance-driven proposal [3], and GenLink, which is a state-of-the-art proposal in the literature [9]. Herein after, we refer to the proposal that is described in this article as Sorbas.

### 4.2 *Effectiveness results*

Figure 5(a) reports on the number of links generated by each proposal and their average precision, recall and $F_1$ score using 2-fold cross validation. In the case of Sorbas, we also report on the values learnt for thresholds $\theta$ and $\gamma$.

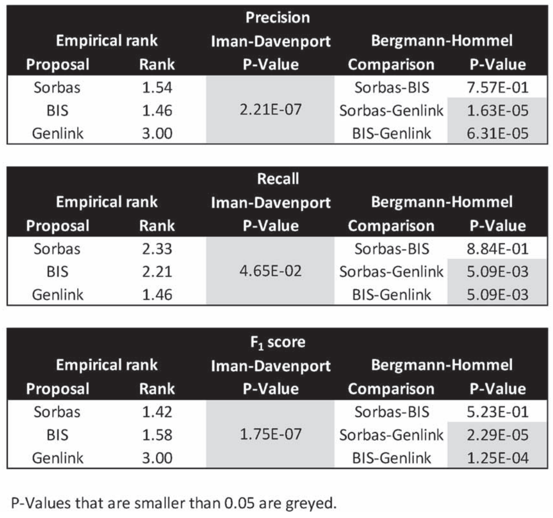
Note that GenLink generates more links than BIS or Sorbas, but many of them are false positives. The average precision of GenLink is $0.34 \pm 0.33$, whereas the average precisions of BIS and Sorbas are $0.71 \pm 0.21$ and $0.74 \pm 0.23$, respectively. Recall that we are dealing with scenarios in which the datasets have many similar resources that are actually different and also many dissimilar resources that are actually the same. Both BIS and Sorbas take the context of the resources to be linked into account, which helps make a difference between the previous cases. Note that the average recall of GenLink is $0.87 \pm 0.14$, which is relatively high, but not that interesting because of the low precision; the recalls of BIS and Sorbas are $0.76 \pm 0.21$ and $0.78 \pm 0.19$, respectively, which are smaller, but we think that the improvement in precision clearly compensates the slight loss regarding recall. Realize that the $F_1$ score of GenLink is $0.41 \pm 0.31$, whereas the $F_1$ scores of BIS and Sorbas are $0.71 \pm 0.19$ and $0.75 \pm 0.2$, respectively. The conclusion is that both Sorbas and BIS are similar to each other and better than GenLink regarding their effectiveness.

We have confirmed the previous conclusion by means of a statistical analysis, cf. Figure 5(b). Let us focus on the $F_1$ score since it combines precision and recall. Note that Sorbas ranks at position 1.42 in average, BIS ranks at position 1.58, and GenLink ranks at position 3.00; in other words, Sorbas and BIS seem to compete for the first position, whereas GenLink always ranks at the third position. Iman–Davenport's test returns $p$-value $1.75E{-}07$, which is a strong indication that the differences in rank are statistically significant. We then compared the proposals to each other using Bergmann–Hommel's test. The comparison of Sorbas to BIS returns $p$-value $5.23E{-}01$, which is larger than the standard confidence level; thus, it is a clear indication that the differences in rank are statistically significant; on the contrary, the comparison of Sorbas to GenLink returns $p$-value $2.29E{-}05$ and the comparison of BIS to GenLink returns $p$-value $1.25E{-}04$, which are strong indications that the differences in rank are statistically significant. These results confirm our conclusion regarding our proposal being more effective than GenLink in the scenarios in which we have experimented. Therefore, hypothesis $H_1$ has been validated.



| Scenario | BIS Links | P | R | $F_1$ | Genlink Links | P | R | $F_1$ | θ | γ | Sorbas Links | P | R | $F_1$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Researchers | 26 | 0.59 | 1.00 | 0.74 | 61 | 0.26 | 1.00 | 0.42 | 0.01 | 0.84 | 16 | 1.00 | 1.00 | 1.00 |
| Authors | 338 | 0.97 | 1.00 | 0.98 | 7,219 | 0.05 | 1.00 | 0.09 | 0.01 | 0.01 | 338 | 0.97 | 0.99 | 0.98 |
| Films | 232 | 0.96 | 1.00 | 0.98 | 258 | 0.86 | 1.00 | 0.93 | 0.01 | 0.85 | 232 | 0.96 | 1.00 | 0.98 |
| Movies | 32 | 0.50 | 0.55 | 0.52 | 77 | 0.27 | 0.72 | 0.40 | 0.01 | 0.01 | 32 | 0.50 | 0.55 | 0.52 |
| Publications | 44 | 0.64 | 0.39 | 0.48 | 206 | 0.21 | 0.81 | 0.34 | 0.01 | 0.01 | 66 | 0.62 | 0.76 | 0.68 |
| Restaurants | 49 | 0.96 | 0.84 | 0.90 | 51 | 0.92 | 0.84 | 0.88 | 0.01 | 0.01 | 49 | 0.96 | 0.84 | 0.90 |
| Persons1 | 336 | 0.72 | 0.96 | 0.82 | 784 | 0.31 | 0.96 | 0.47 | 0.01 | 0.01 | 336 | 0.72 | 0.96 | 0.82 |
| Persons2 | 195 | 0.69 | 0.67 | 0.68 | 627 | 0.22 | 0.69 | 0.33 | 0.01 | 0.01 | 195 | 0.69 | 0.67 | 0.68 |
| Doremus16-9th | 12 | 1.00 | 0.75 | 0.86 | 15 | 0.80 | 0.75 | 0.77 | 0.01 | 0.01 | 12 | 1.00 | 0.75 | 0.86 |
| Doremus16-fp | 19 | 0.58 | 0.52 | 0.55 | 112 | 0.12 | 0.62 | 0.20 | 0.01 | 0.01 | 18 | 0.56 | 0.48 | 0.51 |
| Doremus17-ht | 58 | 0.34 | 0.83 | 0.49 | 1081 | 0.02 | 1.00 | 0.04 | 0.01 | 0.01 | 58 | 0.34 | 0.83 | 0.49 |
| Doremus17-fp | 7 | 0.57 | 0.57 | 0.57 | 407 | 0.02 | 1.00 | 0.03 | 0.01 | 0.01 | 7 | 0.57 | 0.57 | 0.57 |

(a) Effectiveness results.



| Precision | | | | |
|---|---|---|---|---|
| Empirical rank | | Iman-Davenport | Bergmann-Hommel | |
| Proposal | Rank | P-Value | Comparison | P-Value |
| Sorbas | 1.54 | 2.21E-07 | Sorbas-BIS | 7.57E-01 |
| BIS | 1.46 | | Sorbas-Genlink | 1.63E-05 |
| Genlink | 3.00 | | BIS-Genlink | 6.31E-05 |

| Recall | | | | |
|---|---|---|---|---|
| Empirical rank | | Iman-Davenport | Bergmann-Hommel | |
| Proposal | Rank | P-Value | Comparison | P-Value |
| Sorbas | 2.33 | 4.65E-02 | Sorbas-BIS | 8.84E-01 |
| BIS | 2.21 | | Sorbas-Genlink | 5.09E-03 |
| Genlink | 1.46 | | BIS-Genlink | 5.09E-03 |

| $F_1$ score | | | | |
|---|---|---|---|---|
| Empirical rank | | Iman-Davenport | Bergmann-Hommel | |
| Proposal | Rank | P-Value | Comparison | P-Value |
| Sorbas | 1.42 | 1.75E-07 | Sorbas-BIS | 5.23E-01 |
| BIS | 1.58 | | Sorbas-Genlink | 2.29E-05 |
| Genlink | 3.00 | | BIS-Genlink | 1.25E-04 |

P-Values that are smaller than 0.05 are greyed.

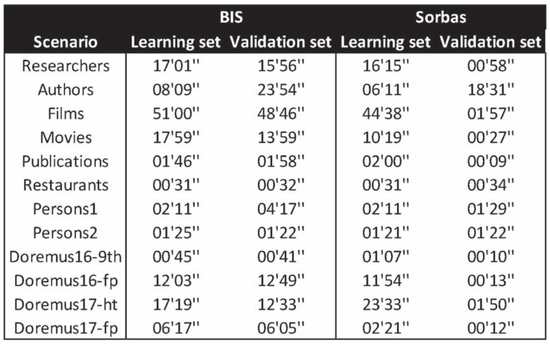
(b) Effectiveness analysis.

FIGURE 5. Experimental effectiveness.

## 4.3 *Efficiency results*

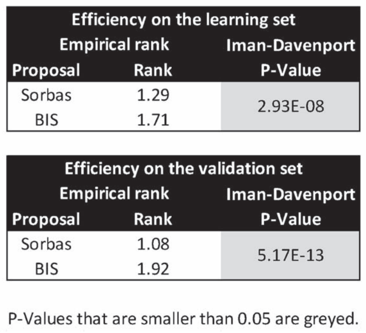
Figure 6 shows our experimental results regarding efficiency, which discards GenLink because it was not effective enough. Note that BIS is an instance-driven proposal that does not attempt to learn any rules, whereas Sorbas is a rule learner. We divided our scenarios into learning and validation sets in order to perform *k*-fold cross validation. In Figure 6(a), the column regarding the learning set must be interpreted as the time taken to link the resources in this set in the case of BIS and the time taken to learn context-aware rules from this set in the case of Sorbas; the column regarding the validation set must be interpreted as the time taken to link the resources in this set in the case of BIS and the time taken to apply the context-aware rules learnt previously in the case of Sorbas.

Regarding the learning set, the average time taken by BIS is $16'22'' \pm 23'58''$ and the average time taken by Sorbas is $16'54'' \pm 22'38''$. That is, it seems that the time taken to link the dataset and to learn context-aware rules from it are very similar. Regarding the validation set, the difference is more clear: note that the average time taken by BIS is $16'54'' \pm 22'38''$ and the average time taken



| | BIS | | Sorbas | |
|---|---|---|---|---|
| Scenario | Learning set | Validation set | Learning set | Validation set |
| Researchers | 17'01'' | 15'56'' | 16'15'' | 00'58'' |
| Authors | 08'09'' | 23'54'' | 06'11'' | 18'31'' |
| Films | 51'00'' | 48'46'' | 44'38'' | 01'57'' |
| Movies | 17'59'' | 13'59'' | 10'19'' | 00'27'' |
| Publications | 01'46'' | 01'58'' | 02'00'' | 00'09'' |
| Restaurants | 00'31'' | 00'32'' | 00'31'' | 00'34'' |
| Persons1 | 02'11'' | 04'17'' | 02'11'' | 01'29'' |
| Persons2 | 01'25'' | 01'22'' | 01'21'' | 01'22'' |
| Doremus16-9th | 00'45'' | 00'41'' | 01'07'' | 00'10'' |
| Doremus16-fp | 12'03'' | 12'49'' | 11'54'' | 00'13'' |
| Doremus17-ht | 17'19'' | 12'33'' | 23'33'' | 01'50'' |
| Doremus17-fp | 06'17'' | 06'05'' | 02'21'' | 00'12'' |

(a) Efficiency results.



| Efficiency on the learning set | | |
|---|---|---|
| Empirical rank | | Iman-Davenport |
| Proposal | Rank | P-Value |
| Sorbas | 1.29 | 2.93E-08 |
| BIS | 1.71 | |

| Efficiency on the validation set | | |
|---|---|---|
| Empirical rank | | Iman-Davenport |
| Proposal | Rank | P-Value |
| Sorbas | 1.08 | 5.17E-13 |
| BIS | 1.92 | |

P-Values that are smaller than 0.05 are greyed.

(b) Efficiency analysis.

FIGURE 6. Experimental efficiency.

by Sorbas is $02'19'' \pm 05'09''$. That is, it seems that applying a rule that was learnt previously helps save much computing time.

To confirm the previous conclusions, we analysed the experimental results using Iman–Davenport's test, cf. Figure 6. Note that the *p*-value is nearly zero in both cases, which is a strong indication that the differences in rank are statistically significant. Simply put, the experimental results support the idea that Sorbas is more efficient than BIS. Therefore, hypothesis $H_2$ has been validated.

## 5 Conclusions

Many RDF datasets provide data that do not have any explicit models. Linking their resources is very important to integrate them. In the literature, there are several genetic proposals that learn link rules. Unfortunately, they are context-unaware, which means that they do not take the neighbours of the resources to be linked into account. This is problematic insofar there are resources that are similar in terms of their attributes but are actually different and vice versa. In such cases, analysing their neighbours may help make a difference. In this article, we have presented a proposal that leverages any existing genetic proposal to learn a set of context-unaware rules and, then combined them in order to learn context-aware rules that have proven to be more effective and efficient.

From our formalization, theoretical analysis, and empirical analysis, we have learnt the following lessons:

1. Exploring the context of the resources has proven to be a good idea, since it helps improve the effectiveness of the rules learnt by other systems.
2. Learning rules has also proven to be a good idea, since the time required to learn them is comparable to the time required to link two datasets, but they can be applied to similar datasets more efficiently.
3. Computing the shortest possible paths among the source and the target classes has proven to provide a good balance between effectiveness and efficiency. Our experiments reveal that most paths range from one to three or four properties in length. In other words, the context that allows to discern if two resources must be linked or not is typically very close to them.
4. The proposals that learn context-unaware rules behave in two different ways: they either produce rules with nearly perfect precision but nearly zero recall, or they produce rules that have a relatively high recall but very low precision. It seems that the behaviour depends completely on the input datasets and the degree of similarity of their resources in terms of their attributes.

Our future research will focus on devising heuristics to explore the neighbours more efficiently. We have profiled our prototype and we have found out that computing the correspondences is the major source of inefficiency; thus, boosting its performance will be our main goal. Note that our proposal is computationally tractable and that the time it takes to work with the experimental datasets is good enough, but we clearly need to improve it to deal with really big datasets.

## Funding

Supported by the Spanish R&D programme (grants TIN2013-40848-R, TIN2016-75394-R, and P18-RT-1060). The computing facilities were provided by the Andalusian Scientific Computing Centre (CICA). We are grateful to Dr. Carlos R. Rivero and Dr. David Ruiz for their early work on the ideas that led to this article.

## References

[1] C. Bizer, T. Heath and T. Berners-Lee. Linked data: principles and state of the art. In *WWW (Invited Talks)*, vol. 1, 2008.

[2] A. Cimmino and R. Corchuelo. A hybrid genetic-bootstrapping approach to link resources in the web of data. In *Hybrid Artificial Intelligent Systems - International Conference*, vol. 10870, pp. 145–157, Lecture Notes in Computer Science, Springer, 2018a.

[3] A. Cimmino and R. Corchuelo. On feeding business systems with linked resources from the web of data. In *Business Information Systems - International Conference*, vol. 320, pp. 307–320, Lecture Notes in Business Information Processing, Springer, 2018b.

[4] I. F. Cruz, F. P. Antonelli and C. Stroe. AgreementMaker: efficient matching for large real-world schemas and ontologies. *Proceedings of the VLDB Endowment*, **2**, 1586–1589, 2009.

[5] K. Forsberg and L. Dannstedt. Extensible use of RDF in a business context. *Computer Networks*, **33**, 347–364, 2000.

[6] M. Holub, O. Proksa and M. Bieliková. Detecting identical entities in the semantic web. In *SOFSEM*, vol. 8939, pp. 519–530, Lecture Notes in Computer Science, Springer, 2015.

[7] W. Hu and Y. Qu. Falcon-AO: a practical ontology matching system. *Journal of Web Semantics*, **6**, 237–239, 2008.

[8] J. Huber, T. Sztyler, J. Nößner and C. Meilicke. CODI: combinatorial optimization for data integration. In *International Workshop on Ontology Matching*, vol. 814, pp. 134–141, CEUR Workshop Proceedings, CEUR-WS.org, 2011.

[9] R. Isele and C. Bizer. Learning expressive linkage rules using genetic programming. *Proceedings of the VLDB Endowment*, **5**, 1638–1649, 2012.

[10] R. Isele and C. Bizer. Active learning of expressive linkage rules using genetic programming. *Journal of Web Semantics*, **23**, 2–15, 2013.

[11] R. Isele, A. Jentzsch and C. Bizer. Efficient multidimensional blocking for link discovery without losing recall. In *International Workshop on the Web and Databases*, 2011.

[12] E. Jiménez-Ruiz and B. C. Grau. LogMap: logic-based and scalable ontology matching. In *International Semantic Web Conference*, vol. 7031, pp. 273–288, International Workshop on the Web and Databases, Springer, 2011.

[13] H. Köpcke and E. Rahm. Frameworks for entity matching: a comparison. *Data & Knowledge Engineering*, **69**, 197–210, 2010.

[14] S. Lacoste-Julien, K. Palla, A. Davies, G. Kasneci, T. Graepel and Z. Ghahramani. SIGMa: a simple greedy matching for aligning large knowledge bases. In *International Conference on Knowledge Discovery and Data Mining*, pp. 572–580, ACM, 2013.

[15] A. E. Monge and C. Elkan. The field matching problem: algorithms and applications. In *International Conference on Knowledge Discovery and Data Mining*, pp. 267–270, AAAI Press, 1996.

[16] A. C. N. Ngomo and K. Lyko. EAGLE: efficient active learning of link specifications using genetic programming. In *Extended Semantic Web Conference*, vol. 7295, pp. 149–163, Springer, 2012.

[17] A. Nikolov, M. d'Aquin and E. Motta. Unsupervised learning of link discovery configuration. In *Extended Semantic Web Conference*, vol. 7295, pp. 119–133, Springer, 2012.

[18] V. Rastogi, N. N. Dalvi and M. N. Garofalakis. Large-scale collective entity matching. *Proceedings of the VLDB Endowment*, **4**, 208–218, 2011.

[19] C. R. Rivero, I. Hernández, D. Ruiz and R. Corchuelo. Discovering and analysing ontological models from big RDF data. *Journal of Database Management*, **26**, 48–61, 2015.

[20] T. Soru and A. C. N. Ngomo. A comparison of supervised learning classifiers for link discovery. In *International Conference on Semantic Systems*, pp. 41–44, ACM, 2014.