On Feeding Business Systems with Linked Resources from the Web of Data

Andrea Cimmino(✉) and Rafael Corchuelo(✉)

ETSI Informática, University of Seville,
Avda. Reina Mercedes, s/n., Sevilla, Spain
{cimmino,corchu}@us.es

Abstract. Business systems that are fed with data from the Web of Data require transparent interoperability. The Linked Data principles establish that different resources that represent the same real-world entities must be linked for such purpose. Link rules are paramount to transparent interoperability since they produce the links between resources. State-of-the-art link rules are learnt by genetic programming and build on comparing the values of the attributes of the resources. Unfortunately, this approach falls short in cases in which resources have similar values for their attributes, but represent different real-world entities. In this paper, we present a proposal that leverages a genetic programming that learns link rules and an ad-hoc filtering technique that boosts them to decide whether the links that they produce must be selected or not. Our analysis of the literature reveals that our approach is novel and our experimental analysis confirms that it helps improve the F_1 score by increasing precision without a significant penalty on recall.

1 Introduction

The feasibility of many emerging business relies on the availability and interoperability of suitable on-line datasets [1]. The Web of Data provides business with islands of data (availability) that can be transparently used as required by the business models (interoperability). The Web of Data is ruled by the Linked-Data principles that support the idea that resources within different datasets that represent the same real-world entities must be linked so as to facilitate data interoperability [4]. Link rules are intended to help link resources automatically however they may be linked by different kind of relations, e.g., spatial or temporal coverage, we focus only on owl:sameAs that relates resources representing the same real-world entity.

Supported by the Spanish R&D programme (grants TIN2013-40848-R and TIN2016-75394-R). 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 earlier ideas that led to the results in this paper. We also thank Dr. Francisco Herrera for his hints on statistical analyses and sharing his software with us.

The literature provides several proposals to machine learn link rules by means of genetic programming [11,12,19,20]. Such rules build on transformation and similarity functions that are applied to the values of the attributes of two resources to check if they can be considered similar enough (by attributes we mean their data-type properties); if they are, then the input resources are linked; otherwise, they are kept apart. Such link rules fall short where there are many resources that represent different real-world entities but have attributes with similar values.

In this paper, we present a novel approach to the problem: first, we leverage a state-of-the-art genetic programming to learn a set of link rules; we then select a link rule and apply it in order to obtain a collection of candidate links; the remaining rules are then boosted to analyse the neighbours of the resources involved in each candidate link (the neighbours are the resources that can be reached by means of their object properties); finally, we analyse how similar they are in order to decide which of the candidate links must be selected as true positives and which must be discarded as false positives. Our analysis of the related work unveils that this is a novel approach since current state-of-the-art link rules do no take the neighbours into account. Our experimental analysis confirms that precision can be improved by 77% in average, with an average −9% impact on recall; overall, the average improvement regarding the F_1 score is 58%. We also conducted the Iman-Davenport test to check that these differences are statistically significant regarding precision and the F_1 score, but not regarding recall. Our conclusion is that ours is a good approach to help software agents integrate the data that they fetch from the Web of Data, which will definitely help many business systems have access to many datasets in the Web of Data and integrate them with others.

A hypothetical busyness that may benefit from our approach is a collector that aims at selling food products from several supermarkets choosing always the one with the lower fair. In this scenario is paramour to identify the products that are the same in the different supermarkets, so the cheapest can be suggested. Furthermore the collector may also consider other datasets to add busyness value to their data. For instance a food-health dataset and a datasets containing recipes, ingredients, menus and diets which along the food-health data may allow to sell packages of food that conform healthy menus and recipes. As a result, relying on a highly precise approach like ours to identify the same products sparse along the different datasets is paramour.

The rest of the article is organised as follows: Sect. 2 reports on the related work; Sect. 3 provides the details of our proposal; Sect. 4 presents our evaluation results; finally, Sect. 5 summarises our conclusions.

2 Related Work

The earliest techniques to learn link rules were devised in the field of traditional databases, namely: de-duplication [7,17], collective matching [2,3,6,14,21], and entity matching [15]. They set a foundation for the researchers who addressed

the problem in the context of the Web of Data, where data models are much richer and looser than in traditional databases.

Some of the proposals that are specifically-tailored to web data work on a single dataset [10,16], which hinders their general applicability; there are a few that attempt to find links between different datasets [5,8,9,13], but they do not take the neighbours of the resources being linked into account, only the values of the attributes; that is, they cannot make resources with similar values for their attributes apart in cases in which they represent different real-world entities. An additional problem is that they all assume that data are modelled by means of OWL ontologies. Unfortunately, many common datasets in the Web of Data do not rely on OWL ontologies, but on simple RDF vocabularies that consists of classes and properties whose relationships and constraints are not made explicit.

The previous problems motivated several authors to work on techniques that are specifically tailored to work with RDF datasets. Most such proposals rely on genetic programming [11,12,19,20] in which chromosomes encode the link rules as trees, which facilitates performing cross-overs and mutations. They differ regarding the expressivity of the language used to encode the link rules and the heuristics used to implement the selection, replacement, cross-over, and mutation operators, as well as the performance measure on which the fitness function relies. Isele and Bizer [11,12] contributed with a supervised proposal called Genlink. It is available with the Silk framework [24], which is gaining impetus thanks to many real-world projects [23]. It uses a tournament selection operator, a generational replacement operator, custom cross-over and mutation operators, and its fitness function 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 measures. An interesting feature is that the size of the link rules must not be pre-established at design time, but it is dynamically adjusted during the learning process. Ngomo and Lyko [19] contributed with a supervised proposal called Eagle, which is available with the LIMES framework [18]. It uses a tournament selection operator, a $\mu + \beta$ replacement operator, tree cross-over and mutation operators, and its fitness function relies on the F_1 score. It does not use transformation functions, but the Levenshtein, Jaccard, Cosine, Q-Grams, Overlap, and Trigrams string similarity functions. The maximum size of the link rules must be pre-established at design time. Nikolov et al. [20] contributed with an unsupervised proposal. It 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 taken into account, but the library of similarity functions includes Jaro, Levenshtein, and I-Sub. The maximum size of the link rules rules is also set at design time. There is a diverging proposal by Soru and Ngomo [22]. It supports the idea of using common machine-learning techniques on a training set that consists in a vectorisation of the Cartesian product of the resources in terms of the similarity of their attributes. Transformation functions cannot be used and the string similarity functions available are Q-Grams, Cosine, and Levenshtein. Whether the size of the rules must be pre-set or not depends on the underlying machine learning technique.

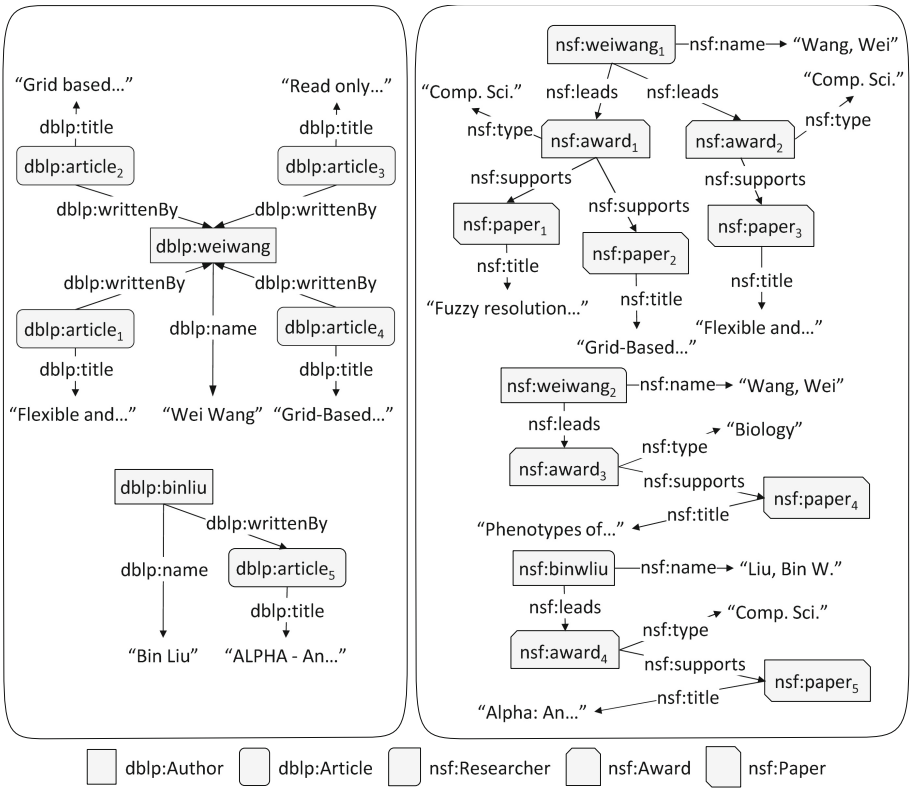

Fig. 1. Running example.

Unfortunately, none of the proposals for RDF datasets take the neighbourhoods into account (Fig. 1).

Clearly, the state of the art does not account for a proposal to link resources in RDF datasets that takes their neighbours into account. Ours is tailored to work with such datasets and it is novel in that it is not intended to generate link rules, but leverages the rules that are learnt with other proposals and boosts them in order to analyse the neighbours, which our experimental analysis confirms that has a positive impact on precision without degrading recall.

3 Our Proposal

Our proposal consists in three components, namely: the first one learns link rules, the second one filters out the links that they produce, and the third relies on two voting strategies to select most reliable filtered links.

The link rule learner is based on Genlink [12], which is a state-of-the-art genetic programming that has proven to be able to learn good link rules for many common datasets. Genlink is feed with a set of examples representing

resources that should be linked, then it learns a set of link rules from them. As a result Genlink returns a set of link rules (all of them aim at linking the same kind of resources). A quality score ranks the output rules, usually the best is selected and the rest discarded. Doing this task several times for different kind of resources a set of link rules is generated, one of these rules is selected to be improved using our approach and the rest are the supporting rules.

The filter is an ad-hoc component that works as follows: it takes a link rule and executes it to produce a set of candidate links; then, it analyses the neighbours of the resources involved in each candidate link by boosting the remaining rules; links in which the corresponding neighbours are similar enough are preserved as true positive links while the others are discarded as false positive links. The selector is an ad-hoc component that works as follows: it takes the filtered links and the supporting rules used to filter them; then, it performs a voting strategy regarding how many rules filtered the same link and another voting strategy regarding how many links were filtered by each rule; finally, a subset of the filtered links is selected and preserved as the rest are discarded.

Below, we present the details of the filter and selector, plus an ancillary method that helps measure how similar the neighbours of two resources are.

Example 1. Listing 1 presents two sample datasets that are based on the DBLP and the NSF datasets. The resources are depicted in greyed boxes whose shapes encode their classes (i.e., the value of property *rdf*:*type*), the properties are represented as labelled arrows, and the literals are encoded as strings. The genetic component learns the following link rules in this scenario, which we represent using a Prolog-like notation for the sake of readability:

$$r_1\text{: } \mathit{link}(A, R) \text{ if } \mathit{rdf}{:}\mathit{type}(A) = \mathit{dblp}{:}\mathit{Author}, \mathit{rdf}{:}\mathit{type}(R) = \mathit{nsf}{:}\mathit{Researcher},\\ N_A = \mathit{dblp}{:}\mathit{name}(A), N_R = \mathit{nsf}{:}\mathit{name}(R),\\ \mathit{levenstein}(\mathit{lfname}(N_A), \mathit{lfname}(N_R)) > 0.80.$$

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

where *levenstein* and *jaccard* denote the well-known string similarity functions (normalised to interval $[0.00, 1.00]$), *lfname* is a function that normalises people's names as "last name, first name", and *lowercase* is a function that changes a string into lowercase.

Intuitively, link rule r_1 is applied to a resource A of type *dblp*:*Author* and a resource R of type *nsf*:*Researcher*; it computes the normalised Levenshtein similarity between the normalised names of the author and the researcher; if it is greater than 0.80, then the corresponding resources are linked. Link rule r_2 should now be easy to interpret: it is applied to a resource A of type *dblp*:*Article* and a resource P of type *nsf*:*Paper* and links them if the normalised Jaccard similarity amongst the lowercase version of the title of article A and the title of paper P is greater than 0.65.

```
 1: method filterLinks(r, S, D1, D2, θ, μ, ρ) returns K
 2:     K := ∅
 3:     (C1, C2) := (sourceClasses(r), targetClasses(r))
 4:     L1 := apply(r, D1, D2)
 5:     for each link rule r' ∈ S do
 6:         (C1', C2') := (sourceClasses(r'), targetClasses(r'))
 7:         (P1, P2) := (findPath(C1, C1', D1), findPath(C2, C2', D2))
 8:         L2 = apply(r', D1, D2)
 9:         for each (p1, p2) ∈ P2 × P2 do
10:             for each link (a, b) ∈ L1 do
11:                 (A, B) := (findResources(a, p1, D1), findResources(b, p2, D2))
12:                 E := L2 ∩ (A × B)
13:                 w := computeSimilarity(A, B, E)
14:                 if w ≥ θ then
15:                     K := K ∪ {(a, b)}
16:                 end
17:             end
18:         end
19:     end
20:     K := selectLinks(K, μ, ρ)
21: end
```

Listing 1. Method to filter links.

It is not difficult to realise that link rule r_1 links resources *dblp*:*weiwang* and *nsf*:$weiwang_1$ or *dblp*:*binliu* and *nsf*:*binwliu*, which are true positive links, but also *dblp*:*weiwang* and *nsf*:$weiwang_2$, which is a false positive link. In cases like this, the only way to make a difference between such resources is to analyse their neighbours, be them direct (e.g., *dblp*:*weiwang* and *dblp*:$article_2$) or transitive (e.g., *nsf*:$weiwang_1$ and *nsf*:$paper_2$). ■

3.1 Filtering Links

Listing 1 presents the method to filter links. It works on a link rule r, a set of supporting link rules S, a source dataset D_1, a companion dataset D_2, and a threshold θ that we explain later. It returns K, which is the subset of links produced by base link rule r that seem to be true positive links.

The method first initialises K to an empty set, stores the source and the target classes of the base link rule in sets C_1 and C_2, respectively, and the links that result from applying it to the source and the companion datasets in set L_1.

The main loop then iterates over the set of supporting link rules using variable r'. In each iteration, it first computes the sets of source and target classes involved in link rule r', which are stored in variables C_1' and C_2', respectively; next, it finds the set of paths P_1 that connect the source classes in C_1 with the source classes in C_1' in dataset D_1; similarly, it finds the set of paths P_2 that connect the target classes in C_2 with the target classes in C_2' in dataset D_2. By

path between two sets of classes, we mean a sequence of object properties that connect resources with the first set of classes to resources with the second set of classes, irrespective of their direction. Simply put: the idea is to find the way to connect the resources linked by the base link rule with the resources linked by the supporting link rule, which is done by the intermediate and the inner loops.

The intermediate loop iterates over the set of pairs of paths (p_1, p_2) from the Cartesian product of P_1 and P_2. If there is at least a pair of such paths, it then means that the resources involved in the links returned by base link rule r might have some neighbours that might be linked by supporting link rule r'.

The inner loop iterates over the collection of links (a, b) in set L_1. It first finds the set of resources A that are reachable from resource a using path p_1 in source dataset D_1 and the set of resources B that are reachable from resource b using path p_2 in the companion dataset D_2. Next, the method applies supporting link rule r' to the source and the companion dataset and intersects the resulting links with $A \times B$ so as to keep resources that are not reachable from a or b apart; the result is stored in set E. It then computes the similarity of sets A and B; intuitively, the higher the similarity, the more likely that resources a and b refer to the same real-world entity. If the similarity is equal or greater than threshold θ, then link (a, b) is added to set K; otherwise, it is filtered out. When the main loop finishes, set K contains the collection of links that involve neighbours that are similar enough according to the supporting rules.

We do not provide any additional details regarding the algorithms to find paths or resources since they can be implemented using Dijkstra's algorithm to find the shortest paths in a graph. Computing the similarity coefficient is a bit more involved, so we devote a subsection to this ancillary method below.

Example 2. In our running example, link rule r_1 is the base link rule, i.e., we are interested in linking authors and researchers, and we use link rule r_2 as the support link rule, i.e., we take their articles and papers into account. Their source classes are $C_1 = \{dblp{:}Author\}$ and $C_1' = \{dblp{:}Article\}$, respectively, and their target classes are $C_2 = \{nsf{:}Researcher\}$ and $C_2' = \{nsf{:}Paper\}$, respectively. Link rule r_1 returns the following links: $L_1 = \{(dblp{:}weiwang, nsf{:}weiwang_1), (dblp{:}weiwang, nsf{:}weiwang_2), (dblp{:}binliu, nsf{:}binwliu)\}$; note that the first and the third links are true positive links, but the second one is a false positive link. Link rule r_2 returns the following links: $L_2 = \{(dblp{:}article_1, nsf{:}paper_3), (dblp{:}article_2, nsf{:}paper_2), (dblp{:}article_4, nsf{:}paper_2), (dblp{:}article_5, nsf{:}paper_5)\}$, which are true positive links.

The sets of paths between the source and target classes of r_1 and r_2 are $P_1 = \{\langle dblp{:}writtenBy\rangle\}$ and $P_2 = \{\langle nsf{:}leads, nsf{:}supports\rangle\}$. Now, the links in L_1 are scanned and the resources that can be reached from the resources involved in each link using the previous paths are fetched.

Link $l_1 = (dblp{:}weiwang, nsf{:}weiwang_1)$ is analysed first. The method finds $A = \{dblp{:}article_1, dblp{:}article_2, dblp{:}article_3, dblp{:}article_4\}$ by following resource $dblp{:}weiwang$ through path $\langle dblp{:}writtenBy\rangle$; similarly, it finds $B = \{nsf{:}paper_1, nsf{:}paper_2, nsf{:}paper_3\}$ by following resource $nsf{:}weiwang_1$ through path $\langle nsf{:}leads, nsf{:}supports\rangle$. Now supporting link rule r_2 is applied and the results

1: method $computeSimilarity(A, B, E)$ returns d
2: $A' := reduce(A, E)$
3: $B' := reduce(B, E)$
4: $W := intersect(A', B', E)$
5: $d := |W| / \min\{|A'|, |B'|\}$
6: end

Listing 2. Method to compute similarity.

are intersected with $A \times B$ so as to keep 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 A and B 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 $A = \{dblp{:}article_1, dblp{:}article_2, dblp{:}article_3, dblp{:}article_4\}$ by following resource $dblp{:}weiwang$ through path $\langle dblp{:}writtenBy\rangle$; next, it finds $B = \{nsf{:}paper_4\}$ by following resource $nsf{:}weiwang_2$ through path $\langle nsf{:}leads, nsf{:}supports\rangle$. Now supporting link rule r_2 is applied and the result is intersected with $A \times B$, which results in $E = \emptyset$. In such a case the similarity is zero, which intuitively indicates that it is very likely that l_2 is a false positive link.

Link $l_3 = (dblp{:}binliu, nsf{:}binwliu)$ is analysed next. The method finds $A = \{dblp{:}article_5\}$ by following resource $dblp{:}binliu$ through path $\langle dblp{:}writtenBy\rangle$; next, it finds $B = \{nsf{:}paper_5\}$ by following resource $nsf{:}binwliu$ through path $\langle nsf{:}leads, nsf{:}supports\rangle$. Now supporting link rule r_2 is applied and the result is intersected with $A \times B$, which results in $E = \{(dblp{:}article_5, nsf{:}paper_5)\}$. The similarity is now 1.00, i.e., it is very likely that link l_3 is a true positive link.

Assuming that θ is set to, e.g., 0.50, the *filterLinks* method would return $K = \{(dblp{:}weiwang, nsf{:}weiwang_1), (dblp{:}binliu, nsf{:}binwliu)\}$. Note that the previous value of θ is intended for illustration purposes only because the running example must necessarily have very little data. ■

3.2 Computing Similarity

Listing 2 shows our method to compute similarities. Its input consists of sets A and B, which are two sets of resources, and E, which is a set of links between them. It returns the Szymkiewicz-Simpson overlapping coefficient, namely:

$$overlap(A, B) = \frac{|A \cap B|}{\min\{|A|, |B|\}}$$

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

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

where $X \propto \phi$ denotes the maximal set X that fulfils predicate ϕ, that is:

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

The method to compute similarities then works as follows: it first reduces the input sets of resources A and B according to the set of links E; it then computes the intersection of both reduced sets; finally, it computes the similarity using Szymkiewicz-Simpson's formula on the reduced sets.

Example 3. Analysing link $l_1 = (dblp{:}weiwang, nsf{:}weiwang_1)$ results in sets $A = \{dblp{:}article_1, dblp{:}article_2, dblp{:}article_3, dblp{:}article_4\}$, $B = \{nsf{:}paper_1, nsf{:}paper_2, nsf{:}paper_3\}$, and $E = \{(dblp{:}article_1, nsf{:}paper_3), (dblp{:}article_2, nsf{:}paper_2), (dblp{:}article_4, nsf{:}paper_2)\}$. If E is interpreted as an equality relation by computing its reflexive, symmetric, transitive closure, then it is not difficult to realise that $dblp{:}article_2$ and $dblp{:}article_4$ can be considered equal, because $dblp{:}article_2$ is equal to $nsf{:}paper_2$ and $nsf{:}paper_2$ is equal to $dblp{:}article_4$. Thus, set A is reduced to $A' = \{\{dblp{:}article_1\}, \{dblp{:}article_2, dblp{:}article_4\}, \{dblp{:}article_3\}\}$ and set B is reduced to $B' = \{\{nsf{:}paper_1\}, \{nsf{:}paper_2\}, \{nsf{:}paper_3\}\}$. As a conclusion, $|A' \cap B'| = |\{\{dblp{:}article_1, nsf{:}paper_3\}, \{dblp{:}article_2, dblp{:}article_4, nsf{:}paper_2\}\}| = 2$, $|A'| = 3$, and $|B'| = 3$; so the similarity is 0.67.

When link $l_2 = (dblp{:}weiwang, nsf{:}weiwang_2)$ is analysed, $A = \{dblp{:}article_1, dblp{:}article_2, dblp{:}article_3, dblp{:}article_4\}$, $B = \{nsf{:}paper_4\}$, and $E = \emptyset$. Since the equality relation $E^\star$ is then empty, the similarity is zero because the intersection between the reductions of sets A and B is empty.

In the case of link $l_3 = (dblp{:}binliu, nsf{:}binwliu)$, $A = \{dblp{:}article_5\}$, $B = \{nsf{:}paper_5\}$, and $E = \{(dblp{:}article_5, nsf{:}paper_5)\}$. As a conclusion, $|A' \cap B'| = |\{\{dblp{:}article_5, nsf{:}paper_5\}\}| = 1$, $|A'| = 1$, and $|B'| = 1$, where A' and B' denote, respectively, the reductions of sets A and B; so the similarity is 1.00. ■

3.3 Selecting Links

Listing 3 shows the method to select the best links out of a set of correspondences. Its input consists of a set of correspondences K, a threshold μ to the minimum number of times that a link must have been selected by a supporting link rule so that it can be selected by this method (top links), and an additional threshold ρ to the minimum number of times that a supporting link rule must have selected a link so that links that have been selected by that link rule can be selected by this method (top link rules) even if they are not top links (Fig. 2).

This method relies on two ancillary functions, namely: *links*, which given a supporting link rule r' returns the set of links that it has selected, and *rules*,

1: method $selectLinks(K, \mu, \rho)$ returns L
2: $M := \{(s,t) \mid \exists r' : (s,t,r') \in K\}$
3: $S := \{r' \mid \exists s,t : (s,t,r') \in K\}$
4: $g := \mu \max\{n \mid \exists a,b : (a,b) \in M \wedge n = |rules(a,b,K)|\}$
5: $d := \rho \max\{m \mid \exists r' : r' \in S \wedge m = |links(r',K)|\}$
6: $U := \{(a,b) \mid (a,b) \in M \wedge |rules(a,b,K)| \geq g\}$
7: $V := \{r' \mid r' \in S \wedge links(r',K) \geq d\}$
8: $L := \{(a,b) \mid (a,b) \in M \wedge ((a,b) \in U \wedge rules(a,b,K) \cap V \neq \emptyset)\}$
9: end

Listing 3. Method to select filtered links.

	Genlink							Our proposal							
Scenario	links	P	R	F_1	θ	μ	ρ	links	P	R	F_1	ΔLinks	ΔP	ΔR	ΔF_1
DBLP - NSF	127	0.25	0.97	0.40	0.01	0.02	0.74	33	0.97	0.97	0.97	-94	0.96	0.00	0.95
DBLP - DBLP	78,348	0.12	1.00	0.21	1.00	0.07	0.03	9,069	1.00	1.00	1.00	-69,279	1.00	0.00	1.00
BBC - DBpedia	525	0.85	1.00	0.92	0.10	0.00	0.03	461	0.96	1.00	0.98	-64	0.74	0.00	0.72
DBpedia - IMDb	118	0.27	0.55	0.36	0.60	0.00	0.03	41	0.68	0.48	0.57	-77	0.57	-0.12	0.32
RAE - Newcastle	404	0.22	0.82	0.35	0.30	0.13	0.03	68	0.72	0.45	0.56	-336	0.64	-0.45	0.32
Rest - Rest	103	0.90	0.83	0.87	0.10	0.08	0.58	96	0.97	0.83	0.89	-7	0.69	0.00	0.19
												Average Δ	0.77	-0.09	0.58
												Iman-Davenport's test	0.00	0.25	0.00

Fig. 2. Experimental results.

which given a link (a,b) returns the set of link rules that have selected it. The previous functions are formally defined as follows:

$$links(r',K) = \{(a,b) \mid \exists r' : (a,b,r') \in K\}$$
$$rules(a,b,K) = \{r' \mid \exists a,b : (a,b,r') \in K\}$$

The method to select links first projects the set of correspondences K onto the set of links M and the set of supporting link rules S. It then computes g as a percentage, according to μ, of the maximum number of link rules that have selected each candidate link; it also computes d as a percentage, according to ρ, of the maximum number of links that a support link rule has selected as candidates. Next, it computes the set of top links U as the set of links in M that have been selected by at least g link rules; similarly, it computes the set of top link rules V as the set of link rules in S that have selected at least d links. The resulting set of links L is computed as the subset of links in M that are either top links or have been selected by top link rules.

4 Experimental Analysis

In this section, we first describe our experimental environment and then comment on our results.

Computing facility: We run our experiments on a virtual computer 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.

Prototype:[1] We implemented our proposal with Java 1.8 and the following components: the Genlink implementation from the Silk Framework 2.6.0 to generate link rules, Jena TDB 3.2.0 to work with RDF data, 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.

Evaluation datasets:[2] We used the following datasets: DBLP, NSF, BBC, DBpedia, IMDb, RAE, Newcastle, and Rest. We set up the following scenarios: (1) DBLP–NSF, which focuses on the top 100 DBLP authors and 130 principal NSF researchers with the same names; (2) DBLP–DBLP, which focuses on the 9 076 DBLP authors with the same names who are known to be different people; (3) BBC–DBpedia, which focuses on 691 BBC movies and 445 DBpedia films that have similar titles; (4) DBpedia–IMDb, which focuses on 96 DBpedia movies and 101 IMDb films that have similar titles; (5) RAE–Newcastle, which focuses on 108 RAE publications and 98 Newcastle papers that are similar; and (6) Rest–Rest, which focuses on 113 and 752 restaurants published by OAEI. The scenarios DBLP-DBLP, Rest-Rest and RAE-Newcastle rely on datasets from the literature which have not being curated nor modified, the gold standard of the former two is released with the data and the rest have being extracted from LinkLion[3]. BBC-DBpedia, DBLP-NSF and DBPEDIA-IMDb uses a subset of the original datasets due to their size, but again data has not being modified and gold standard have being generated by hand thanks to the size reduction.

Baseline: Our baseline was the Genlink implementation from the Silk Framework 2.6.0, which is a state-of-the-art genetic programming to learn link rules.

Measures: On the one hand we explored a large portion of the parameter space to establish optimal values for θ, μ, ρ in each scenario. On the other hand we measured the number of links returned by each proposal (*Links*), precision (P), recall (R), and the F_1 score (F_1) using 2-fold cross validation. We also computed the normalised differences in precision (ΔP), recall (ΔR), and F_1 score (ΔF_1), which measure the ratio from the difference found between the baseline and our proposal and the maximum possible difference for each performance measure.

Results: The results are presented in Listing 2. We analyse them regarding precision, recall, and the F_1 score below.

The results regarding precision clearly show that our technique improves the precision of the rules learnt by GenLink in every scenario. In average, the difference in precision is 77%. The worst improvement is 57% in the DBpedia–IMDb scenario since these datasets are clearly unbalanced: movies in DBpedia have related actors, writers, or directors that IMDb may not contain at all; this obviously makes it impossible for our proposal to find enough context to make a decision. The best improvement is 100% in the DBLP–DBLP scenario since there

[1] The prototype is available at https://github.com/AndreaCimminoArriaga/TeidePlus.

[2] The datasets are available at https://goo.gl/Pu76SU.

[3] http://www.linklion.org/.

are 9 069 authors with very similar names, which makes it almost impossible for GenLink to generate rules with good precision building solely on the attributes of the resources. The p-value computed by Iman-Davenport's test is 0.00; since it is clearly smaller than the standard confidence level, we can interpret it as a strong indication that there is enough evidence in our experimental data to confirm the hypothesis that our proposal works better than the baseline regarding precision.

The normalised difference of recall ΔR shows that our proposal generally retains the recall of the link rules learnt by GenLink, except in the DBpedia–IMDb and the Rae–Newcastle scenarios. The problem with the previous scenarios was that there are many incomplete resources, that is, many resources without neighbours. For instance, there are 43 papers in the Newcastle dataset that are not related to any authors. The incompleteness of data has also a negative impact on the recall of the base link rules. In our prototype, we are planning on implementing a simple check to identify incomplete resources so that the links in which they are involved are not discarded as false positives, but identified as cases on which our proposal cannot make a sensible decision. Iman-Davenport's test returns 0.25 as the corresponding p-value; since it is larger than the standard confidence level, it may be interpreted as a strong indication that the differences in recall found in our experiments are not statistically significant. In other words, the cases in which data are that incomplete do not seem to be common-enough for them to have an overall impact on our proposal.

We also studied ΔF_1, which denotes the normalised difference in F_1 score. Note that it is 58% in average and that the corresponding Iman-davenport's p-value is 0.00, which can be interpreted as a strong indication that the difference is significant from a statistical point of view. Overall, this result confirms that our proposal helps improve precision without degrading recall and that the improvement in precision is enough for the F_1 score to improve significantly.

5 Conclusions

Data interoperability of business systems based on Web of Data requires to link the resources that are available in different datasets and represent the same real-world entities. Such links are generated by link rules that take the values of the attributes of the resources into account, but not their neighbours, which sometimes results in false positives that have a negative impact on their precision. We have presented a novel proposal that leverages a genetic programming to learn a set of link rules and then boosts them, which has proven to improve the overall F_1 score.

References

1. Alili, H., Belhajjame, K., Grigori, D., Drira, R., Ghezala, H.H.B.: On enriching user-centered data integration schemas in service lakes. In: Abramowicz, W. (ed.) BIS 2017. LNBIP, vol. 288, pp. 3–15. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-59336-4_1
2. Ananthakrishna, R., Chaudhuri, S., Ganti, V.: Eliminating fuzzy duplicates in data warehouses. In: VLDB, pp. 586–597 (2002)
3. Bhattacharya, I., Getoor, L.: Collective entity resolution in relational data. TKDD **1**(1), 1–36 (2007)
4. Bizer, C., Heath, T., Berners-Lee, T.: Linked data: principles and state of the art. In: WWW (Invited talks) (2008). https://www.w3.org/2008/Talks/WWW2008-W3CTrack-LOD.pdf
5. Cruz, I.F., Antonelli, F.P., Stroe, C.: AgreementMaker: efficient matching for large real-world schemas and ontologies. PVLDB **2**(2), 1586–1589 (2009)
6. Dong, X., Halevy, A.Y., Madhavan, J.: Reference reconciliation in complex information spaces. In: SIGMOD, pp. 85–96 (2005)
7. Hernández, M.A., Stolfo, S.J.: The merge/purge problem for large databases. In: SIGMOD Conference, pp. 127–138 (1995)
8. Holub, M., Proksa, O., Bieliková, M.: Detecting identical entities in the semantic web data. In: Italiano, G.F., Margaria-Steffen, T., Pokorný, J., Quisquater, J.-J., Wattenhofer, R. (eds.) SOFSEM 2015. LNCS, vol. 8939, pp. 519–530. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-46078-8_43
9. Hu, W., Qu, Y.: Falcon-AO: a practical ontology matching system. J. Web Sem. **6**(3), 237–239 (2008)
10. Huber, J., Sztyler, T., Nößner, J., Meilicke, C.: CODI: combinatorial optimization for data integration. In: OM, pp. 134–141 (2011)
11. Isele, R., Bizer, C.: Learning expressive linkage rules using genetic programming. PVLDB **5**(11), 1638–1649 (2012)
12. Isele, R., Bizer, C.: Active learning of expressive linkage rules using genetic programming. J. Web Sem. **23**, 2–15 (2013)
13. Jiménez-Ruiz, E., Cuenca Grau, B.: LogMap: logic-based and scalable ontology matching. In: Aroyo, L., et al. (eds.) ISWC 2011. LNCS, vol. 7031, pp. 273–288. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-25073-6_18
14. Kalashnikov, D.V., Mehrotra, S., Chen, Z.: Exploiting relationships for domain-independent data cleaning. In: SDM, pp. 262–273 (2005)
15. Köpcke, H., Rahm, E.: Frameworks for entity matching: a comparison. Data Knowl. Eng. **69**(2), 197–210 (2010)
16. Lacoste-Julien, S., Palla, K., Davies, A., Kasneci, G., Graepel, T., Ghahramani, Z.: SIGMa: Simple greedy matching for aligning large knowledge bases. In: KDD, pp. 572–580 (2013)
17. Monge, A.E., Elkan, C.: The field matching problem: algorithms and applications. In: KDD, pp. 267–270 (1996)
18. Ngomo, A.C.N., Auer, S.: LIMES: a time-efficient approach for large-scale link discovery on the Web of data. In: IJCAI, pp. 2312–2317 (2011)
19. Ngomo, A.-C.N., Lyko, K.: EAGLE: efficient active learning of link specifications using genetic programming. In: Simperl, E., Cimiano, P., Polleres, A., Corcho, O., Presutti, V. (eds.) ESWC 2012. LNCS, vol. 7295, pp. 149–163. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-30284-8_17

20. Nikolov, A., d'Aquin, M., Motta, E.: Unsupervised learning of link discovery configuration. In: Simperl, E., Cimiano, P., Polleres, A., Corcho, O., Presutti, V. (eds.) ESWC 2012. LNCS, vol. 7295, pp. 119–133. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-30284-8_15
21. Rastogi, V., Dalvi, N.N., Garofalakis, M.N.: Large-scale collective entity matching. PVLDB **4**(4), 208–218 (2011)
22. Soru, T., Ngomo, A.C.N.: A comparison of supervised learning classifiers for link discovery. In: SEMANTICS, pp. 41–44 (2014)
23. Szekely, P., et al.: Building and using a knowledge graph to combat human trafficking. In: Arenas, M., et al. (eds.) ISWC 2015. LNCS, vol. 9367, pp. 205–221. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-25010-6_12
24. Volz, J., Bizer, C., Gaedke, M., Kobilarov, G.: Silk: a link discovery framework for the web of data. In: LDOW (2009)