Improving Link Specifications using Context-Aware Information

Andrea Cimmino
University of Seville, Spain
cimmino@us.es

Carlos R. Rivero
Rochester Institute of Technology, USA
crr@cs.rit.edu

David Ruiz
University of Seville, Spain
druiz@us.es

ABSTRACT

There is an increasing interest in publishing data using the Linked Open Data philosophy. To link the RDF datasets, a link discovery task is performed to generate owl:sameAs links. There are two ways to perform this task: by means of a classifier or a link specification; we focus in the latter approach. Current link specification techniques only use the data properties of the instances that they are linking, and they do not take the context information into account. In this paper, we present a proposal that aims to generate context-aware link specifications to improve the regular link specifications, increasing the effectiveness of the results in several real-world scenarios where the context is crucial. Our context-aware link specifications are independent from similarity functions, transformations or aggregations. We have evaluated our proposal using two real-world scenarios in which we improve precision and recall with respect to regular link specifications in 23% and 58%, respectively.

Keywords

Linked Data, Link Discovery, Link Specification, Context-Aware Link Specification

1. INTRODUCTION

In recent years, we have witnessed an increasing interest in the Linked Open Data [9]. As a matter of fact, the number of datasets in 2011 were 452 and in 2014 that number raised to 2,289 [6]. Publicly-available datasets have to fulfill the Linked Data principles, which manly consist in using IRIs as names of things, using HTTP IRIs so that people can look up those names, when someone looks up a IRI provide useful information using the standards (RDF or SPARQL), and finally, include links to other IRIs so that they can discover more things [2]. Since the number of datasets has increased in the recent years and these principles establishes that the datasets must be linked with others and published in RDF formats [1], a huge effort has been done to link these RDF datasets automatically [14]. There are different types of links between datasets, but the most common one is owl:sameAs [7]. To generate the links, a link discovery task must be performed, which aims to find all pair of instances that are describing the same concept [13, 15].

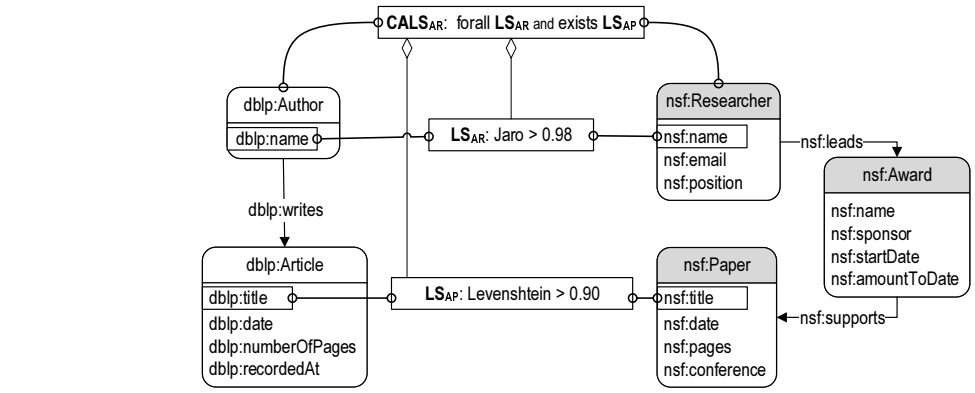
Link discovery can be performed in two different ways: by means of a classifier [22], which links instances with owl:sameAs if it considers them as the same; or generating a link specification, which is a set of restrictions over the data properties of two instances [17, 11]. Each pair of data properties is associated to a similarity function and a threshold, if the similarity function returns a value higher than the threshold, then the restriction is satisfied and the two instances are linked using owl:sameAs. For example, a link specification defines that two instances of Person are the same if both have a data property describing their names, and their literals are exactly the same.

Unfortunately, in some scenarios, this definition is not suitable to generate owl:sameAs links. Under some circumstances, taking only the literals of the data properties of two instances into account may lead to mix up instances that are very similar but actually different, e.g. if two people are different but have the same name. In these cases, a link specification should include conditions over the context information, data properties of other instances that are related with the main two to have more information and improve the effectiveness.

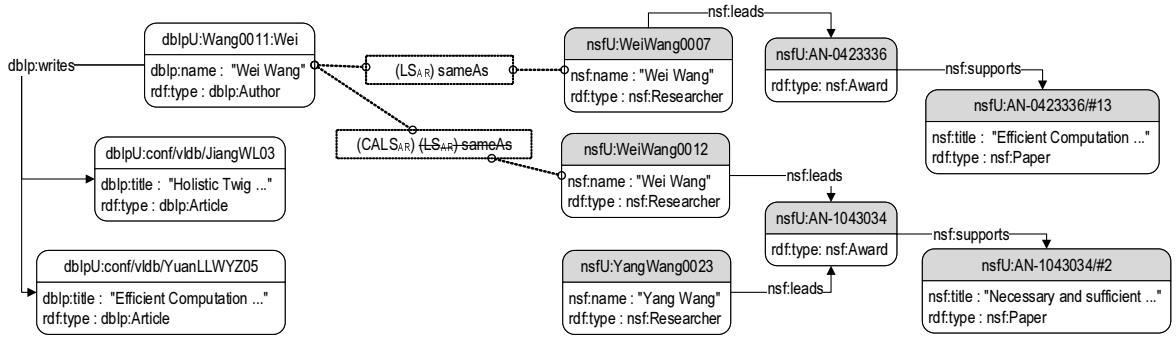
In this paper, we focus on improving current link specifications making them context aware. We aim to extend the definition used by actual techniques [17, 11], and add restrictions over the instances in the context of the pair that we are linking. To achieve this, we introduce the concept of overlap factor. If we define different link specifications over the instances in both contexts, we can handle them as sets potentially overlapped by means of an equity criteria defined by the link specifications. The overlap factor is a function that measures the overlapping between contexts. Thanks to this, we are able to define restrictions over this value. In this paper, we restrict ourselves to two different types of overlap factors, namely: exists or for all. The former means that there is a pair of instances in the context considered the same by means of a link specification. The latter, means that all the instances in both contexts are the same.

Figure 1 depicts a sample scenario where the context is crucial to obtain a good precision. Figure 1(a) shows a part of the data model of DBLP and the National Science Foundation (NSF), both were built using authors that have published in the International Conference of Very Large Data

WWW2016 Workshop: Linked Data on the Web (LDOW2016).



(a) Data model and context-aware link specification



(b) Sample instances and links generated to relate them

Figure 1: Scenario DBLP-NSF: Improving precision using context-aware link specification

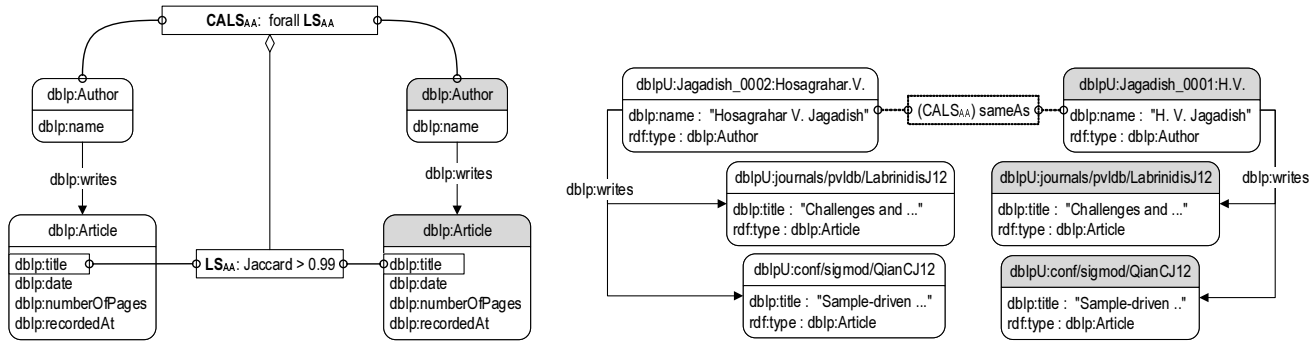
Bases (PVLDB) of 2013. DBLP contains these authors and their articles, the NSF researchers from its portal, with the same name of these authors, that leads awards which supports papers. We wish to identify authors in DBLP that have been awarded with NSF grants using their names and publications. We include two link specifications: LS_{AR}, which links the dblp:Author and nsf:Researcher instances if their literals of dblp:name and nsf:name obtain a score over 0.98 using Jaro; LS_{AP}, which links dblp:Article and nsf:Paper instances if their literals dblp:title and nsf:title obtain a score over 0.90 using Levenshtein. We rely on these link specifications and add overlap factors to them, each of which states how many instances should be linked. The overlap factors for LS_{AR} and LS_{AP} are for all and exists, respectively. The resulting context-aware link specification is interpreted as follows: a pair of dblp:Author and nsf:Researcher instances are the same by means of the context-aware link specification if all the instances covered by LS_{AR}, pairs of dblp:Author and nsf:Researcher instances, are the same, and if at least one pair covered by LS_{AP}, dblp:Article and nsf:Paper instances, are the same.

Actual link specification techniques can generate LS_{AR} or LS_{AP} (notice that LS_{AP} would not link instances of dblp:Author and nsf:Researcher), but they are not able to use LS_{AP} to link instances of dblp:Author and nsf:Researcher. Additionally, they are not able to generate and apply overlap factors over them.

Figure 1(b) depicts a set of sample instances, the link specifications LS_{AR} and LS_{AP} and the context-aware link specification CALS_{AR}. We focus on *"Wei Wang"*, who is an author in DBLP, dblpU:Wang0011:Wei. We see that there are two researchers whose name is *"Wei Wang"* that lead two different NSF grants (nsf:AN-1043034, nsf:AN-0423336). Using LS_{AR} alone, we link the three instances (dblpU:Wang0011:Wei and nsfU:WeiWang0012, and dblpU:Wang0011:Wei and nsfU:WeiWang0007). However, using LS_{AP} we are able to identify one paper in DBLP written by *"Wei Wang"* that also appears in NSF. If we use both links at the same time as part of the context information of the authors, we discard one of the previous links (dblpU:Wang0011:Wei and nsfU:WeiWang0012), which is not correct since it is actually linking another researcher in NSF whose name is *"Wei Wang"*. As a result, we improve precision using context information.

Figure 2 depicts another sample scenario, in which DBLP acts both as source and target, where the context is crucial to obtain a good recall. Figure 2(a) shows a part of the data model, this scenario has several authors and their aliases (different names that refer to the same person); both datasets contains the same authors and their articles but they have different aliases in each dataset. We include a link specification, LS_{AA}, that links dblp:Article instances if the literals of their dblp:title obtain a score over 0.99 using Jaccard. We rely on LS_{AA} and we add to it a for all as overlap factor. The resulting context-aware link specification is interpreted as follows: two instances of dblp:Author are the same by means of the contex-aware link specification if all their dblp:Article instances are the same by means of LS_{AA}.

Figure 2(b) depicts a set of sample instances, the link specification LS_{AA} and the context-aware link specification CALS_{AA}. We focus on link *"Hosagrahar V. Jagadish"* and



(a) Data model and context-aware link specification

(b) Sample instances and links generated to relate them

Figure 2: Scenario DBLP-DBLP: Improving recall using context-aware link specification

"H. V. Jagadish" (dblpU:Jagadish_0002:Hosagrahar.V. and dblpU:Jagadish_0001:H.V.), which are different names of the same person. A regular link specification would compare these literals, instead we rely on their publications. Using LS_{AR} as part of the context information of the authors, we are able to link all their dblp:Article instances and, hence, link the instances dblpU:Jagadish_0002:Hosagrahar.V. and dblpU:Jagadish_0001:H.V. through their publications. We improve the recall because we do not rely on the names of authors, which differs from both datasets, instead we use their publications, which titles do not differ from both datasets. As result, we improve the recall using the context information.

We performed several experiments using the datasets introduced in the Figure 1 and 2, in which we improved 23% in precision and 58% in recall, respectively.

The rest of the article is organized as follows: in Section 2, we report on several related proposals and their features; Section 3 introduces our conceptual framework; Section 4 presents our proposal to generate context-aware link specifications; Section 5 reports the results obtained in our experiments; and, finally, Section 6 recaps on our main conclusions and future work.

2. RELATED WORK

Over the last years, several approaches have been developed to address the link discovery task. There are two ways to face link discovery. The first approach is building different kind of classifiers to establish if two instances are the same [22]. The second approach is through the discovery of accurate link specifications, which specify conditions that must hold true for two entities to be interlinked [11, 12, 17, 18, 19, 21]. The main difference between these two approaches is that the former does not specify why two instances are the same, i.e., it works like a black box that receives as input two instances and outputs whether or not they are the same; the latter generates a specification of why two instances should be the same, describing which data properties have to fulfil the conditions.

We focus only in the link specification approach, although, we have also analyzed some classifiers since they exploit the context information [8, 10, 23]. Holub et al. [10] proposed a technique that works with a fixed formula that takes the instances related directly to the pair that is been linked into account. PARIS [23] is an unsupervised technique developed to exploit the information contained in instances related to the pair that is been analyzed by it. It takes two data models as input and generates a probabilistic model. In the first place, the technique computes the probabilities of equivalences of instances, then, the probabilities for relationships with other instances and, finally, it creates the equivalences between the classes. Hassanzadeh et al. [8] proposed a semi-supervised technique that receives two datasets as input. This technique works as follows: having all the data properties of all the classes in each dataset, the technique iterates over one set and searches in the other set, according to a string distance, the most similar data properties. The technique returns the ranked set of pairs according to the string distance.

Regarding the link specification approaches, Isele and Bizer [11] proposed GenLink, which is a supervised genetic programming technique that generates link specifications as trees. It starts with a population made of random link specifications and some recurrent link specification predefined by the authors. Then, using genetic operations (reproduction, crossover and mutation), the population is evolved and its quality evaluated by means of a fitness function, which uses training data provided by the user. The technique stops when a configured maximum number of iterations is reached, or a link specification obtains a value in the fitness function over a threshold given by the user. Based on GenLink, the same authors proposed ActiveGenLink [12], which aims to reduce the number of labelled examples using active learning. ActiveGenLink selects link candidates to be labelled by the user from a pool of unlabelled instances through a query strategy. Then, once the user labels a given example, it adds the example to the training data and evolves the population using GenLink. Another semi-supervised technique is EAGLE [17], which is based in a genetic programming technique with active learning, and it aims to generate link specifications as trees. It starts detecting similar classes and data properties using RAVEN [16]. Then, EAGLE evolves an initial random population of link specifications according to genetic operators. After that, the technique computes the most informative links and asks the user to label them. This process is repeated until the stop condition is fulfilled, i.e., a maximal number of iterations is reached, or the fitness value of a link specification is over a given threshold. An unsupervised learning technique was proposed by Nikolov et al. [19], which starts with a random population and keeps iterating

over it, applying genetic operators, until a maximal number of iterations is reached, or the fitness of the population does not improve for several iterations. Since this technique does not work with labelled data, the fitness function uses two criteria defined by the authors to evaluate link specifications, namely: pseudo-F-measure and neighborhood growth. When a stop condition is reached, it returns the link specification with the highest fitness value from the population. EUCLID [18] is an unsupervised technique that, using different similarity functions, evaluates the data properties of the instances and generates a space of similarity values. Then, depending on different heuristics, it iterates over that space updating the scores and pruning them until a solution is found or a stop condition is reached. The unsupervised technique proposed by Song and Heflin [21] focuses on metrics to improve the candidate selection to be as more scalable as possible. Candidate selection is a process to pick pairs of instances, each of which has a high probability to be the same. The process is performed by selecting and comparing only part of the data properties of each instance in the pair. It then extracts a set of data properties very useful in disambiguation, which identify why the pair of instances are the same.

As far as we know, none of the previous link specification techniques is able to exploit context information. Holub et al. [10] proposed a technique that takes into account only the instances of the context one-hop related to the pair that is been linked. PARIS [23] takes as input all the datasets and generates a probabilistic model to classify input instances. The technique by Hassanzadeh et al. [8] returns a ranking of most similar data properties using several string distances. None of the previous techniques is able to apply transformations on data properties and only [8] is able to use different string similarity measures, however it only returns a ranked list of data property and not why two instances should be linked. Additionally, [10] only takes one-hop connected instances into account, although, many real-world scenarios require to take more than one-hop related instances into account [20].

Table 1 summarizes all the techniques and their different features. Those that generate link specifications are classified as LS; if they take into account the context, been LS or not, then we classify them as context-aware (C-A). Finally, if the technique is independent of any specific function (aggregations, transformations, string distance measures or string similarities), we classify it as function independent (FI).

Technique	LS	C-A	FI
[10] Holub et al. (2015)	No	Yes	No
[23] Suchanek et al. (2011)	No	Yes	No
[8] Hassanzadeh et al. (2013)	No	Yes	No
[5, 12] Isele and Bizer (2011, 2012)	Yes	No	Yes
[17, 18] Ngonga and Lyko (2012,2013)	Yes	No	Yes
[19] Nikolov et al. (2012)	Yes	No	Yes
[21] Song and Heflin (2011)	Yes	No	No

Table 1: Comparison of current techniques

3. PRELIMINARIES

In the following, we present the formalization of several concepts that we use to describe our proposal. We define its foundations, what a link specification is and what we mean by context-aware link specification.

3.1 Foundations

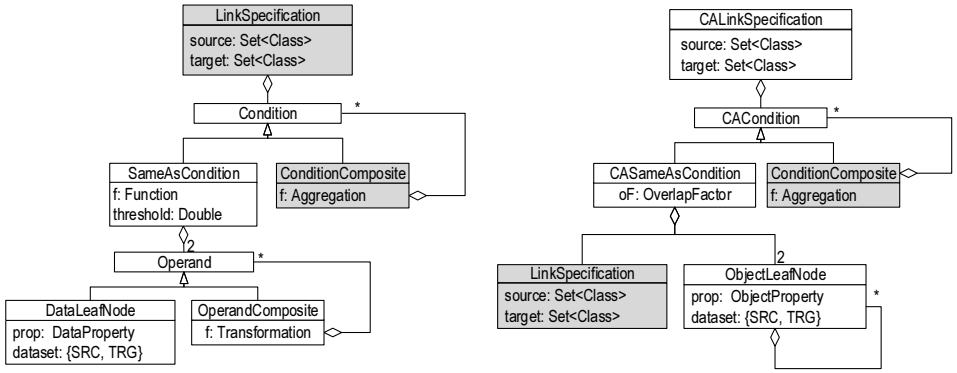
We are focusing on RDF datasets, which are triple stores that contain literals and IRIs. Our proposal focuses on the analysis of different instances, each of which entails several concepts as follows:

- IRI: it uniquely identifies a web location. Note that we use expressions like dblp:name to refer to IRIs, in which dblp: is a prefix. Table 2 summarizes all of our prefixes. For example, some sample IRIs in Figure 1 are dblpU:Wang0011:Wei and nsfU:YangWang0023.
- BlankNode: are placeholders for IRIs whose actual value is unknown. They have only local scope and are purely an artifact of the serialization. Blank nodes are disjoint from IRIs and Literals.
- Instance: an instance is an IRI or a BlankNode that we are interested in linking.

Pref.	IRI
rdf:	http://www.w3.org/1999/02/22-rdf-syntax-ns#type
owl:	http://www.w3.org/2002/07/owl#
dblp:	http://example.org/voc/dblp#
nsf:	http://example.org/voc//nsf#
db-lpU:	http://example.org/urls/dblp#
nsfU:	http://example.org/urls/nsf#

Table 2: IRI prefixes used in the paper

- Class: we can assign classes to Instances, each of which is an IRI that represents a real-world concept. When we assign a Class to an Instance, we are explicitly saying that the Instance belongs to the type of the Class. We use rdf:type to represent this assignment. For example, in Figure 1, Instances related to Classdblp:Author represent authors in DBLP.
- DataProperty: Instances may comprise attributes that describe features of the Instances, which are plain literals. To represent these features, we use data properties, which are IRIs that identify these literals. In Figure 1, the names of the dblp:Author Instances are identified using dblp:name.
- Literal: it denotes a value that a data property takes. For example, in Figure 1 "Yang Wang" for nsfU:YangWang0023 or "Wei Wang" for nsfU:WeiWang0012 are sample literals for the same data property. Depending on the Instance, data properties have different literals.
- ObjectProperty: Instances can be related to other Instances by means of object properties, which are IRIs; a set of Instances related conform a graph. Note that, in RDF, object properties are first-class citizens and they are not subordinated to Instances. Figure 1 shows a sample object property that relates dblp:Author and dblp:Article using dblp:writes. Notice that we can add multiple relations connecting the same Instance to multiple Instances; for example, in Figure 1, the object property dblp:writes may relate one dblp:Author with several dblp:Article Instances.



(a) Link Specification (b) Context-Aware Link Specification

Figure 3: Models for link discovery

3.2 Link Specification

When performing link discovery, we have a source and a target datasets that we wish to relate using owl:sameAs links. To link the Instances of each dataset we generate a link specification. A link specification has been defined in multiple manners in the literature [3, 11, 12, 17]. We have based our work in the definition given by Isele and Bizer [11]. A link specification is a set of restrictions that define the equality between a source and a target sets of classes based on their data properties. For instance, a dblp:Author and a nsf:Researcher are the same if they have very similar literals for dblp:name and nsf:name.

Figure 3(a) depicts how we model link specifications using an UML-like notation. Each DataLeafNode represents a specific data property and the dataset it belongs. OperandComposite specifies one Transformation function to be applied over the literals; examples of these transformations are *lowercase*, *tokenize*, *concatenate* or *remove prefix*. SameAsCondition represents a threshold and a string distance measure, or a string similarity, that defines when two Operands are the same; some of the well-known string distance measures are Levenshtein, Jaccard, Jaro and Jaro-Winkler. ConditionComposite combines different SameAsConditions or other ConditionComposites. Thanks to this, it is possible to define restrictions over data properties and combine the results, for example, using AND or OR Boolean conditions. LinkSpecification contains the sets of source and target classes of the Instances that we are relating with owl:sameAs links.

Figure 1(a) shows two sample link specifications. One of them between the Instances of dblp:Author and nsf:Researcher (LS_{AR}). The Instances of both classes are the same if literals of data properties dblp:name and nsf:name are the same by means of a Jaro comparison and a threshold of 0.98. The second link specification (LS_{AP}) relates dblp:Article and nsf:Paper, which are the same if literals of data properties dblp:title and nsf:title are the same by means of a Levenshtein comparison and a threshold of 0.90.

3.3 Context-Aware Link Specification

A context-aware link specification extends the given definition of link specification by defining when two Instances are the same, like before, but the restrictions are not defined only over their data properties, but also taking data properties of other Instances that belongs to a different set of classes into account, which are connected by object properties.

Figure 3(b) specifies the structure of a context-aware link specification using an UML-like notation. Each ObjectLeafNode represents the object properties that connect the sets of classes in CALinkSpecification with the other sets of classes in the LinkSpecifications. A CASameAsCondition specifies an OverlapFactor over a LinkSpecification. OverlapFactor takes as values for all, if all the Instances are required to be considered the same by means of the LinkSpecification, or exists, if just one pair of Instances is required. ConditionComposite combines different CASameAsConditions or other ConditionComposites, the Aggregation functions are: AND or OR Boolean conditions. Finally, CALinkSpecification represents the two main sets of classes, source and target, that the Instances we wish to link with owl:sameAs belongs.

We present a sample context-aware link specification in Figure 1(a) between dblp:Author and nsf:Researcher, that we refer to as $CALS_{AR}$. The Instances of both classes are considered the same using some of their data properties, dblp:name and nsf:name, but also taking the data properties of Instances belonging to the context into account. It uses two link specifications to link the different kind of Instances, LS_{AR} and LS_{AP}, and over them it defines two overlap factors, for all and exists, respectively.

4. APPROACH

Our proposal aims to generate context-aware link specifications by means of Algorithm 1. The input is an example composed by two Instances from different datasets representing the same concept. The output of the algorithm is a context-aware link specification.

The algorithm takes each input Instance and explores the Instances related with them by means of their object properties, retrieving all the new Instances from both contexts (lines 10-11 in Algorithm 1). Then, the algorithm generates a set of link specifications for the Instances in each context, line 12 in Algorithm 1. For example, receiving as input the Instances dblpU:Wang0011:Wei and nsfU:WeiWang0007 from Figure 1(b), the algorithm firstly retrieves all the Instances that are related with them, dblpU:conf/vldb/JiangWL03 and dblpU:conf/vldb/YuanLLWYZ05 for the first one, nsfU:AN-0423336 and nsfU:AN-0423336/#13 for the second one. It is important to notice that not only the Instances one-hop away are retrieved but also those that are more distant. Then, the algorithm generates two link link specifications, LS_{AR}

Algorithm 1 generateCALinkSpecification

```
1: input
2:    i1, i2: Instance
3: output
4:    cals: CALinkSpecification
5: variables
6:    C1, C2 : ℙ Instance
7:    LS: ℙ LinkSpecification
8:    SO: ℙ CASameAsCondition
9:
10: C1 ← expand (i1)
11: C2 ← expand (i2)
12: LS ← generateLinkSpecifications (C1, C2)
13: SO ← assignOverlapFactor (LS, C1, C2, i1, i2)
14: cals ← createCALinkSpecification (SO, i1, i2)
```

and LS_{AP}, relating the Instances in both contexts. Actual link specification techniques are able to generate LS_{AR} and LS_{AP}, so, in this paper, we do not focus on generating them. We assume that generateLinkSpecifications returns the best link specifications for the Instances in the context.

The algorithm assigns to each link specification an overlap factor. In addition, we store the set of object properties that connect the class of the input Instance with the class of the Instances covered by the link specification (line 13 in Algorithm 1 and Algorithm 2). Finally, it combines with different aggregation functions the results obtained in the previous step, creating the context-aware link specification (line 14 in Algorithm 1).

Algorithm 2 assignOverlapFactor

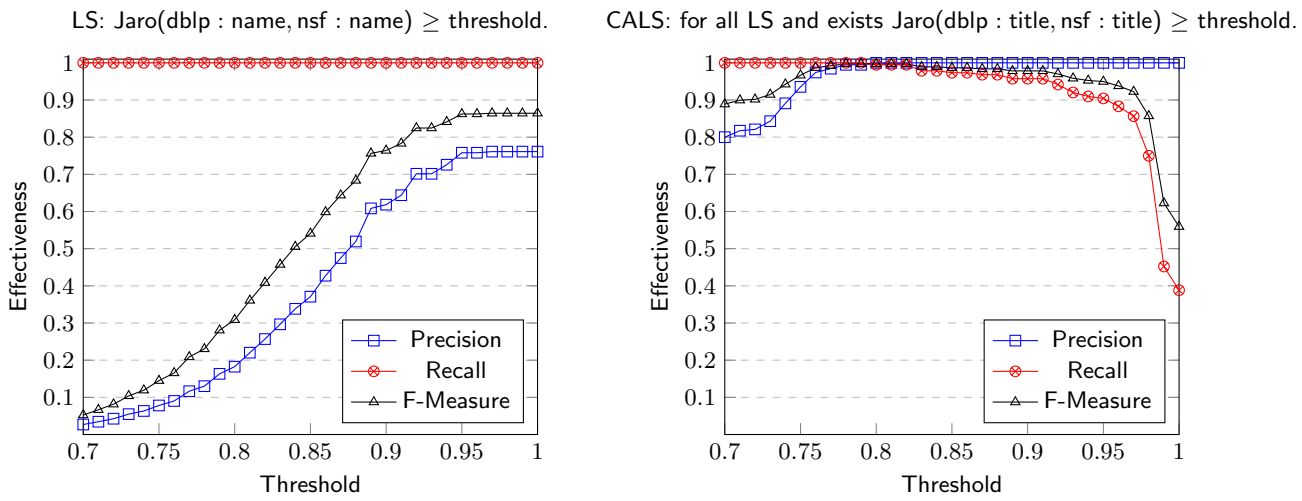
```
1: input
2:    LS: ℙ LinkSpecification
3:    i1, i2 : Instance
4:    C1, C2 : ℙ Instance
5: output
6:    SO: ℙ   CASameAsCondition
7: variables
8:    ls: LinkSpecification
9:    o1, o2 : Double
10:   oF: OverlapFactor
11:   op_src, op_trg: ℙ ObjectLeafNode
12:
13: SO ← ∅
14: oF ← {}
15: for each ls in LS
16:    (o1,o2) ← measureOverlap (ls, C1, C2)
17:    op_src ← objectPropertiesPath (ls, C1, i1)
18:    op_trg ← objectPropertiesPath (ls, C2, i2)
19:    if o1 = 1.0 and o2 = 1.0 then
20:       oF ← {for all }
21:    if o1 > 0.0 and o2 > 0.0 then
22:       oF ← {exists }
23:    if oF = {for all }   or oF = {exists }   then
24:       SO ∪ createCASameAsCond (oF, ls, op_src, op_trg)
```

Algorithm 2 receives as input a set of link specifications, two Instances to be linked, and two sets of Instance that are the contexts; the output of the algorithm is a set of CASameAsCondition. The algorithm starts iterating over the link specifications and, for each of them (line 15 in Algorithm 2), it firstly applies the current link specification to the Instances, and then, it measures the ratio of Instances linked by a owl:sameAs generated with the current link specification (line 16 in Algorithm2). In our technique, if all the Instances in both contexts covered by the link specification are linked by owl:sameAs, then a for all overlap factor is assigned to the current link specification (lines 19-20 in Algorithm 2). If only one owl:sameAs is generated between the Instances in the context, covered by the current link specification, then an exists overlap factor is assigned to it (lines 21-22 in Algorithm 2); otherwise, the link specification is discarded. In Figure 1(b), the algorithm assigns for all to LS_{AR}, since there is only a pair of Instances covered by it and both fulfill the restrictions of LS_{AR}. The algorithm assigns exists to LS_{AP} since only one pair of dblp:Article and nsf:Paper Instances fulfill the conditions of LS_{AP}. Additionally, we also store the sets of object properties that connect the class of the input Instance with the class of the Instances covered by the link specification (lines 17-18 in Algorithm 2). In Figure 1(b), the algorithm relates LS_{AR} with an empty set of object properties because the Instances covered by it are the same that the input. Then the algorithm assigns to LS_{AP} two sets of object properties, {dblp:writes } and {nsf:leads, nsf:supports }, that connect the main Instancesdblp:Author and nsf:Researcher with the class of the Instances covered by LS_{AP}, dblp:Article and nsf:Paper. Finally, for each link specification, its related overlap factor and the sets of object properties, the algorithm creates a CASameAsCondition (line 24 in Algorithm 2). Every CASameAsCondition is added to a set, which is the output of the algorithm when there are no more link specifications to compute.

Algorithm 3 createCALinkSpecification

```
1: input
2:    i1, i2 : Instance
3:    SO: ℙ CASameAsCondition
4: output
5:    cals: CALinkSpecification
6: variables
7:    class_src, class_trg: ℙ Class
8:    aggrAND: ConditionComposite
9:
10: aggrAND ← combineWithAndAggregations (SO)
11: class_src ← extractRDFClass (i1)
12: class_trg ← extractRDFClass (i2)
13: cals ← createCALS (aggrAND, class_src, class_trg)
```

Algorithm 3 receives as input a set of CASameAsCondition and the input Instances, the output of the algorithm is a CALinkSpecification. The algorithm starts combining all the different CASameAsConditions of the input set with and aggregations (line 10 in Algorithm 3). Finally the algorithm extracts the class of each input Instance (lines 11-12 in Algorithm 3) and creates a CALinkSpecification (line 13 in Algorithm 3). In Figure 1(b), the classes of the input Instances are dblp:Author for Wang0011:Wei and nsf:Researcher for WeiWang0007, the final context-aware link specification with the aggregation functions is depicted in this figure. It links dblp:Author and nsf:Researcher Instances if they have similar names (for all LS_{AR}) and some of their publications have similar titles (exists LS_{AP}).



(a) Link specification

(b) Context-Aware link specfication

Figure 4: Effectiveness results when specifications are given by an expert in DBLP-NSF

5. EVALUATION

We use two scenarios in which we study the effectiveness using link specifications and context-aware link specifications. Both scenarios were built with real data using researchers that have published in PVLDB of 2013 extracted from DBLP. Furthermore, there are real-world situations in which taking the context into account is crucial to perform the optimal link discovery task. For each scenario, we did two evaluations: in the first one, an expert defined a link specification, to the best of his/her knowledge, and then, the same expert defined a context-aware link specification. Since link specifications are very sensitive to their acceptance threshold, for each defined specification, we tuned the acceptance threshold value of their string similarity from 0.7 to 1.00 and analyzed for which values the best effectiveness was achieved. In the second evaluation, we used GenLink [11] to generate link specifications between the same classes of the previous experiments, whose goal is to analyze the impact of adding context to a regular link specification generated by a technique.

We make our data, algorithms, and scripts, publicly available [4]. Therefore, our results can be reproduced and tested by third parties and researchers can extend our results to cope with future requirements.

We have implemented our technique in Java 1.8, and Jena 3.0.0. Our experiments were run on a computer that was equipped with a Intel Core i7 2.8 GHz CPU and 16 GB RAM, running on Mac OS 10.9.5 (64-bits).

In section 5.1, we present our first scenario, DBLP-NSF, we describe its characteristics, the relationships between the datasets and how we built them. Section 5.2 follows the same structure, in which we present our second scenario DBLP-DBLP.

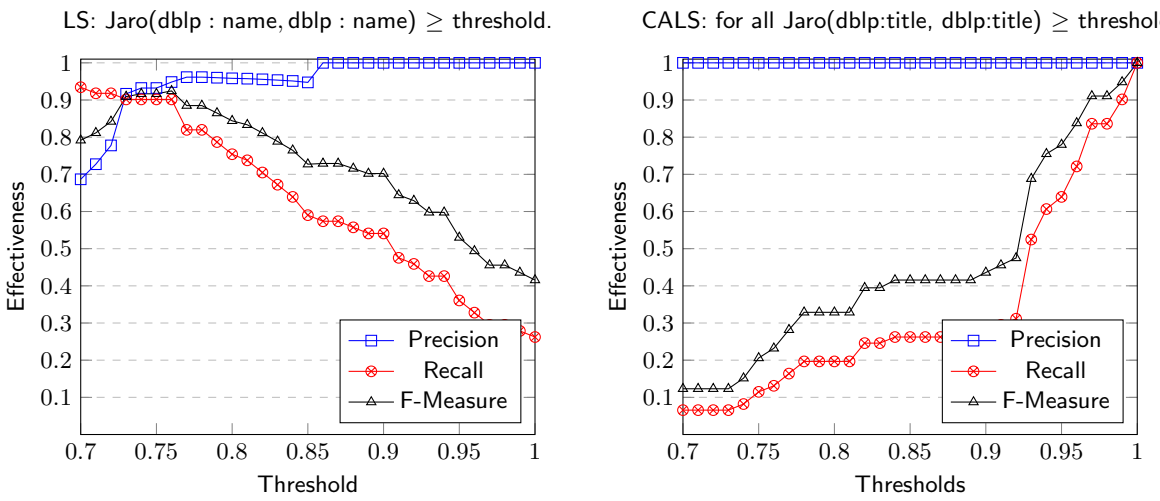
5.1 NSF-DBLP scenario

In this scenario, we have 188 owl:sameAs links between dblp:Author and nsf:Researcher instances, which we consider our gold standard. All of them relate authors and researchers with the same name and publications in common. Between the datasets, we have 57 pair of dblp:Author and nsf:Researcher instances that have the same name but are different authors, therefore, taking some context information into account, like their publications, is crucial to perform a suitable link discovery task.

To build this scenario, firstly, we extracted from DBLP all the articles and authors that have been published in PVLDB of 2013. Then, we looked up their names in the NSF portal and we extracted all their related information. Finally, we created two RDF datasets, whose data models are depicted in Figures 1(b) and 2(b), respectively. The resulting DBLP dataset comprises 764 instances of dblp:Author and 47,225 instances of dblp:Article. The resulting NSF dataset comprises 235 instances of nsf:Researcher, 235 instances of nsf:Award, and 6,877 instances of nsf:Paper. Since NSF has information about different disciplines, in this dataset we have several researchers that have the same name but are different people. For example, in Figure 1, instances WeiWang0012 and WeiWang0007 have the same literal for nsf:name, although they are describing different researchers. In the whole dataset, only 74 instances of nsf:Researcher have different literals for nsf:name.

Figure 4(a) depicts the effectiveness obtained with the link specification provided by the expert, a Jaro comparison over dblp:name and nsf:name, and using all possible acceptance thresholds values from 0.7 to 1.0. Figure 4(b) depicts the effectiveness obtained using the context-aware link specification that extends the previous link specification, adding a link specification composed by Jaro over dblp:title and nsf:title, and using the best threshold for the dblp:Author and nsf:Researcher link specification. The overlap factor for the link specifications between the publications is exists and for the link specification between persons is for all.

The results in Figure 4(a) shows how the effectiveness of the link specification is better if the threshold acceptance is higher, although it never reaches the best precision or F-Measure of 1.0, it always obtains a recall of 1.00. Recall never changes because every dblp:Author that should be linked with a nsf:Researcher has exactly the same name, hence, if the threshold is low, the string metric generates false positives but always recognizes pairs of instances with



(a) Link specification

(b) Context-Aware link specfication

Figure 5: Effectiveness results when specifications are given by an expert in DBLP-DBLP

the same name (covering all the correct links). If the threshold is high, the precision improves by pruning these false positives.

In Figure 4(b), the context-aware link specification obtains a precision that improves when the acceptance threshold is higher, however, the recall decreases for values higher than 0.83 of acceptance threshold. The context-aware link specification reaches 1.00 in precision and recall for thresholds in the range of 0.80-0.83. Recall drops when the threshold is higher because this time we are comparing the names of the authors, and also the titles of their publications, which are written slightly different, e.g., *SmartSaver turning flash drive into a disk energy saver for mobile computers* and *"SmartSaver: turning flash drive into a disk energy saver for mobilecomputers"*. As result, an exact string matching would not recognize them as the same. Due to this issue, recall drops for higher thresholds. On the contrary, precision improves when the threshold is higher, it mainly generates false negatives but the instances linked are always correct.

LS for DBLP-NSF			
LS	P	R	F
LSN_1	0.76	1.00	0.86
LSN_5	0.76	1.00	0.86
LSN_{10}	0.76	1.00	0.86
CALS for DBLP-NSF			
LS and their overlap factors	P	R	F
for all LSN_1 and exists LST_1	0.94	1.00	0.97
for all LSN_5 and exists LST_5	0.97	1.00	0.99
for all LSN_{10} and exists LST_{10}	1.00	0.95	0.98
CALS Best improvement	0.24	-	0.13

Table 3: GenLink results for dblp:Author and nsf:Researcher link specifications and context-aware link specification

Table 3 shows the results obtained using GenLink to generate several link specifications between the classes of previous experiments. We used different number of examples to generate the links specifications. On one hand, for the instances of dblp:Author and nsf:Researcher with 1 and 5 examples, Genlink generated LSN_1 and LSN_5; both have a Jaccard distance ≤ 0.37 over the literals of the data properties dblp:name and nsf:name. Using 10 examples, GenLink generated LSN_{10}, which has a Jaccard distance ≤ 0.21 for the same data properties. On the other hand, the link specifications between dblp:Article and nsf:Paper using 1 example was LST_1, it has a Levenshtein distance ≤ 29.48 over dblp:title and nsf:title, using 5 examples GenLink generated LST_5, which has a Jaccard distance ≤ 0.59 over the same data properties and, finally, using 10 examples GenLink generated LST_{10}, it has a Levenshtein distance ≤ 7.05 over the same data properties.

We analyzed the effectiveness of dblp:Author and nsf:Researcher link specification, and the context-aware link specification for the same classes. The results in Table 3 shows how, when we took context into account, precision improved by 0.18 (1 example), 0.21 (examples) and 0.24 (10 examples). However, recall dropped by 0.05 in the context-aware link specification made by 10 examples because the acceptance threshold was restrictive enough to not recognize titles written slightly different, as we explained before.

5.2 DBLP-DBLP scenario

This scenario has 62 owl:sameAs links between the source and target datasets. Both contains dblp:Author instances with similar names and aliases, which are different enough to produce false positives using comparators with low acceptance thresholds, and false negatives with high acceptance thresholds. This scenario was built using the same authors and publications in the previous scenario. We took the whole list filtered by authors with aliases (like "H. V. Jagadish" and "Hosagrahar Visvesvaraya Jagadish"), then, we split the instances in two datasets, each of which were obtained by using a different alias for the same person. The data model of the source and target datasets is depicted in Figure 2(a). Both datasets contain 58 dblp:Author instances and their publications, which are 5284 dblp:Article in total. We conducted similar experiments in this scenario as previously.

Figure 5(a) shows the results obtained for the link spec-

ification given by the expert, which relates by means of a Jaro distance the dblp:name of dblp:Author instances from the source and target dataset, then, we obtain the precision, recall and F-Measure for each possible threshold acceptance value. Figure 5(b) shows the results for a context-aware link specification that uses a link specification which, by means of a Jaro distance, relates the dblp:title from the source and target dblp:Article instances. The overlap factor for the link specification is for all.

The results of Figure 5(a) shows that the best F-Measure results are obtained for thresholds values between 0.72 and 0.77; however, it never obtains a F-Measure of 1.00. For higher thresholds, recall decreases while precision increases, this tendency is inverted for lower thresholds. Due to authors' aliases, recall behaves in the same way of Figure 4(b) with publication titles; if the threshold is higher, the link specification does not recognize as the same some aliases, e.g., *"H.V. Jagadish"* and *"Hosagrahar V. Jagadish"*. On the contrary, when the threshold is higher, precision improves because the linked instances have similar names.

Figure 5(b) shows that the context-aware link specifications always obtain a precision of 1.00, recall and F-Measure increases when the threshold is higher, achieving 1.00. This situation is the same as Figure 4(a), the titles of the publications in each dataset have exactly the same literal, therefore, when the threshold is higher, the recall improves. Precision is always 1.00 because the CALS of this example only links two instances of dblp:Author if all their publications are exactly the same, due to the for all restriction. If just one publication is not linked, then their authors are also not linked; therefore, if a link is actually generated, it is always correct.

LS for DBLP-DBLP			
LS	P	R	F
LSN_1	1.00	0.26	0.45
LSN_5	1.00	0.30	0.46
LSN_{10}	1.00	0.26	0.45
CALS for DBLP-DBLP			
LS and their overlap factors	P	R	F
for all LST_1	1.00	0.84	0.91
for all LST_5	1.00	0.84	0.91
for all LST_{10}	1.00	0.84	0.91
CALS Best improvement	-	0.58	0.46

Table 4: GenLink results for source and target dblp:Author link specification and context-aware link specification

Table 4 shows the link specifications generated by GenLink for the same classes of the previous experiments. On one hand, for the source and target dblp:Author instances, with 1 example, Genlink generated LSN_1 that relates source and target dblp:name data properties by means of a Jaccard distance ≤ 0.15, with 5 examples generated LSN_5 that relates the same data properties by means of a Levenshtein distance ≤ 1.48 and, finally, with 10 examples it generated LSN_{10}, which relates the same data properties by means of a Levenshtein distance ≤ 1.15. On the other hand, for the source and target dblp:Article instances, with 1 example GenLink generated LST_1 that relates source and target dblp:title by means of a Levenshtein distance ≤ 1.76, with 5 examples it generated LST_5 that relates the same data properties by means of a Levenshtein distance ≤ 1.46, finally, with 10 examples it generated LST_{10} that relates the different dblp:title by means of a Levenshtein distance ≤ 1.76.

We analyzed the effectiveness for the source and target dblp:Author instances using the link specification, and then, the context-aware link specification. The results in Table 4 show how, when we take context into account, precision does not change but recall improves by 0.58 (1 example), 0.54 (5 examples) and 0.58 (10 examples); which entails an improvement in the F-Measure of 0.46 (1 example), 0.45 (5 examples) and 0.46 (10 examples).

6. CONCLUSION AND FUTURE WORK

In the literature, there are several techniques that generate link specifications to perform a link discovery task; however, none of them is able to exploit context information. In this paper, we present a proposal to extend the definition of link specification by means of the concept of overlap factor, which let us exploit context information and define context-aware link specifications. Additionally, we have identified two real-world scenarios where the context is crucial and where, the current techniques, are not able to obtain the best effectiveness without taking the context into account.

Our experimental results prove how context-aware link specifications obtain a better effectiveness in comparison with regular link specifications in our scenarios. We obtained an improvement of 23% in precision and 58% in recall, respectively.

In future work, we plan to develop a technique to navigate through context information of instances by not using all of their object properties, and selecting only those more suitable to build effective context-aware link specifications. Additionally, we plan to add more metrics to calculate the overlap factor extending our current for all and exists restrictions. Finally, this paper is focused on generating owl:sameAs links, but an interesting extension of our work is the generation of other kind of links in an automatic way, depending on the results of the overlap factor.

Acknowledgements

Supported by the Spanish R&D&I program under grant TIN2013-40848-R.

7. REFERENCES

[1] C. Bizer, T. Heath, and T. Berners-Lee. Linked Data: Principles and state of the art. In *WWW*, pages 1–40, 2008.

[2] C. Bizer, T. Heath, and T. Berners-Lee. Linked Data-the story so far. *Int. J. Semantic Web Inf. Syst. 5(3)*, pages 205–227, 2009.

[3] M. G. Carvalho, A. H. Laender, M. A. Gonçalves, and A. S. da Silva. Replica identification using genetic programming. In *SAC*, pages 1801–1806, 2008.

[4] A. Cimmino, C. R. Rivero, and D.Ruiz. Research prototype, repositories and experimental results. *URL http://www.tdg-seville.info/acimmino/Cals*, 2016.

[5] M. G. de Carvalho, A. H. F. Laender, M. A. Gonçalves, and A. S. da Silva. A genetic programming approach to record deduplication. *IEEE Trans. Knowl. Data Eng.*, 24(3):399–412, 2012.

[6] I. Ermilov, M. Martin, J. Lehmann, and S. Auer. Linked Open Data Statistics: Collection and Exploitation. In *KESW*, pages 242–249. 2013.

[7] H. Halpin, P. J. Hayes, J. P. McCusker, D. L. McGuinness, and H. S. Thompson. When owl:sameAs isn't the same: An analysis of identity in Linked Data. In *ISWC*, pages 305–320. 2010.

[8] O. Hassanzadeh, K. Q. Pu, S. H. Yeganeh, R. J. Miller, L. Popa, M. A. Hernández, and H. Ho. Discovering linkage points over web data. *PVLDB*, 6(6):444–456, 2013.

[9] T. Heath and C. Bizer. *Linked Data: Evolving the Web into a Global Data Space*. Morgan & Claypool Publishers, 2011.

[10] M. Holub, O. Proksa, and M. Bieliková. Detecting identical entities in the Semantic Web Data. In *SOFSEM*, pages 519–530. 2015.

[11] R. Isele and C. Bizer. Learning expressive linkage rules using genetic programming. *PVLDB*, 5(11):1638–1649, 2012.

[12] R. Isele and C. Bizer. Active learning of expressive linkage rules using genetic programming. *J. Web Sem.*, 23:2–15, 2013.

[13] R. Isele, A. Jentzsch, and C. Bizer. Efficient multidimensional blocking for link discovery without losing recall. In *ACM SIGMOD workshops*, 2011.

[14] M. Nentwig, M. Hartung, A.-C. N. Ngomo, and E. Rahm. A survey of current Link Discovery frameworks. *Web Sem. J.*, pages 1–18, 2015.

[15] A.-C. N. Ngomo and S. Auer. LIMES: A time-efficient approach for large-scale Link Discovery on the Web of Data. In *IJCAI*, pages 2312–2317, 2011.

[16] A.-C. N. Ngomo, J. Lehmann, S. Auer, and K. Höffner. RAVEN - Active learning of link specifications. In *ISWC workshops*, pages 25–37, 2011.

[17] A.-C. N. Ngomo and K. Lyko. EAGLE: Efficient active learning of link specifications using genetic programming. In *ESWC*, pages 149–163. 2012.

[18] A.-C. N. Ngomo and K. Lyko. Unsupervised learning of link specifications: deterministic vs. non-deterministic. In *ISWC workshops*, pages 25–36, 2013.

[19] A. Nikolov, M. d'Aquin, and E. Motta. Unsupervised learning of Link Discovery configuration. In *ESWC*, pages 119–133. 2012.

[20] C. R. Rivero, I. Hernández, D. Ruiz, and R. Corchuelo. Exchanging data amongst linked data applications. *Knowl. Inf. Syst.*, 37(3):693–729, 2013.

[21] D. Song and J. Heflin. Automatically generating data linkages using a domain-independent candidate selection approach. In *ISWC*, pages 649–664. 2011.

[22] T. Soru and A.-C. N. Ngomo. A comparison of supervised learning classifiers for Link Discovery. In *SEM*, pages 41–44, 2014.

[23] F. M. Suchanek, S. Abiteboul, and P. Senellart. PARIS: Probabilistic alignment of relations, instances, and schema. *PVLDB*, 5(3):157–168, 2011.