A Framework to Restore Semantically Affected Links in LOD Datasets

André Gomes Regino [a,*], Julio Cesar dos Reis [a] and Rodrigo Bonacin [b]
[a] *Institute of Computing, Unicamp, Campinas, SP, Brazil*
E-mail: andre.regino@students.ic.unicamp.br
[b] *Center for Information Technology Renato Archer, CTI, Campinas, SP, Brazil*
E-mails: jreis@ic.unicamp.br, rodrigo.bonacin@cti.gov.br

Abstract. RDF triples are the basis for semantically interlinking data in the Web. As a major component of Semantic Web, RDF triples express data elements linking machine-readable resources throughout the Web. The literature presents various alternatives for exploring (semi-)automatic algorithms for connecting distributed RDF datasets. However, RDF datasets are constantly changing, which frequently affects links between web resources. This makes the maintenance of RDF links consistency a too arduous task to be manually performed. This article proposes a framework for supporting the (semi-)automatic maintenance of RDF links between resources from different RDF repositories. We present the LODMF framework, which enables detecting change operations on RDF triples deals with affected links to other repositories. Our solution defines a set of link maintenance actions applicable to affected links based on users assistance. We present scenarios illustrating the application of our framework in real-world situations.

Keywords: Web of Data evolution, Link evolution, Change Operations, Link changes, Link Repairment

1. Introduction

Linked data technology has become essential for semantic interoperability and knowledge discovery. Nowadays, the interconnection of Resource Description Framework (RDF) statements via explicit links plays a central role in assuring data linkage on the Web. Links between datasets have been stimulated to produce interconnected datasets instead of deploying isolated data repositories as information islands. RDF triples and links to external data elements must be updated to keep existing repositories consistent over time, maintaining the evolutionary characteristic of the datasets.

Linked Open Data (LOD) datasets may change drastically over time [1]. For instance, in the biomedical area, new species are discovered, new medication is created, and existing diseases suffer mutation. Such dynamics must result in change operations in existing datasets, which in turn might influence well-formed links between RDF datasets, turning the maintenance of connections over time a challenging task to ensure consistency. Manual maintenance is unfeasible due to several aspects, including:

- Size: it would take lots of time to maintain and keep up-to-date huge datasets; the DBpedia[1], for instance, has 3 billion triples, 50 million links, and 4.58 million resources;
- Language: several datasets are multilingual, which requires a costly team of maintainers;

*Corresponding author. E-mail: andre.regino@students.ic.unicamp.br.
[1] dbpedia.org

– Error: the manual maintenance might derive errors to the process due to the complexity of existing RDF triples.

Changes affecting RDF triples may lead to broken links caused by structural or semantic changes in resources. Structurally broken links occur when associated resources are removed, then the subject or the link object is no longer dereferenceable [2]. Semantically broken links appear when the semantics of associated resources does not further express the meaning intended by the triple's author. Existing literature usually addresses cases of structurally broken links by detecting the removed resource associated with the link [3]. On the other hand, semantically broken links are harder to detect, fix, and prevent than structurally broken links [3] due to the difficulties in detecting changes in the meaning of resources.

Existing literature on link maintenance has studied the process of supporting the identification of broken links. This includes the identification of changes between RDF datasets [4]; monitoring and preservation of data quality [5]; storage of triples metadata [6]; and notification of dataset changes [3]. The fixing of broken links was investigated with the use of SPARQL templates [2] and recalculation of existing links [3]. The problem of broken links still requires further investigations regarding (semi)-automatic link maintenance techniques. This is necessary to reduce human workload and keep the RDF datasets' connections in a consistent state based on how datasets change.

In our previous study, we empirically analyzed several cases of link modifications by studying the evolution of datasets in the domain of life sciences [7]. Our study investigated if there was a correlation between changes in triples and changes in links by considering twelve scenarios involving addition, removal, and modification of triples and links in the Agrovoc[2] dataset. We discovered that complex changes, like modifications of resources and triples, affect links more than simple changes.

In addition, we developed a methodology to identify semantically broken links caused by modifying a given LOD dataset [8]. Our study combined syntactic and semantic similarity computation methods to detect semantically broken links. Our methodology was evaluated with the use of two different datasets, Geonames [3] and Wikidata [4].

This article proposes the Linked Open Data Maintenance Framework (LODMF), aiming to support the maintenance of links between RDF datasets. Our framework receives an RDF dataset in two different periods. The solution is composed of three steps, which includes: detection of RDF changes by identifying RDF changes performed at triple-level between two RDF dataset versions; identification of integrity inconsistencies by locating links that are genuinely affected by changes, which make links broken; the decision of which action to be taken and which of the substitute candidates will be used to replace the broken part of the link. The LODMF generates a list of suggested actions to support the users in choosing the best maintenance alternative. In the end, the framework updates the datasets links based on the user's maintenance choice, therefore improving the consistency and integrity of the datasets.

This article refers to an extended and enhanced version of our preliminary work presented in Regino et. al. [9]. In the current study, we present improvements in the framework and deeply describe the last step of the LODMF, which is conceived and developed to fix incorrect links. We also provide a real-world example of a link maintenance action using our framework. We present a detailed explanation of a novel algorithm developed to select candidates to replace the broken part of the affected links and a set of actions suggested to the dataset manager (to fix the links).

The remaining of this article is organized as follows: Section 2 discusses related work; Section 3 presents the description and formalization of the framework; Section 4 presents an application scenario with a practical example, which illustrates the LODMF functioning. Lastly, Section 5 discuss relevant aspects about our software tool and the presented example; Section 6 draws conclusion remarks.

2. Related Work

To the best of our knowledge, complete solutions to fully and automatically address the link maintenance problem in the LOD context are scarce in the literature. Most current literature deals with broken links and their consequences

[2]http://aims.fao.org/standards/agrovoc/linked-data
[3]https://www.geonames.org/
[4]https://www.wikidata.org/

by investigating alternatives to avoid them from occurring, *i.e.* they do not focus on how to recover existing broken links. Previous studies also address structurally broken links, not semantically broken ones [10].

Our proposed LODMF framework is suited to detect and maintain broken links. The framework can be used mainly by ontology managers to help them find cases of broken links that are either hard to manually detect (if there are plenty of links in the broken state) or hard to preserve. Our main finding is a tool that not only detects semantically broken links but also selects the right resource to correct the state of the link from a list of predefined actions, maintaining its semantics. In the following, we present relevant strategies used by state-of-the-art investigations to handle broken links inside and outside the scope of LOD.

A recent paper [11] focuses on one of the most recurrent problems in the ontology field, which is the presence of semantic drifts caused by changes in the real world domain and concepts, reflecting on changes in the ontology's representation, relations, and meanings. The OntoDrift approach tries to address this issue by evaluating drifts. For each resource, OntoDrift identifies changes patterns in tags, classes, and URIs (*e.g. Sport* was compared based on its label, URI, equivalent classes, and subclasses through many versions of the DBpedia dataset) [11]. The OntoDrift deals with the evolution of resources through many versions of the same dataset, whereas our framework handles links between different datasets in two distinct versions.

The precursor on that kind of study in Semantic Web was the framework DSNotify [12], a tool that recognizes structurally broken links and fixes them, supervised by a specialist. DSNotify notifies the maintainer of the datasets if something changes in the dataset.

DSNotify served as an inspiration to subsequent alternatives, like Delta-LD [2], which fixes structurally broken links removing the inconsistent resource and reconnecting the link using SPARQL templates. DSNotify and Delta-LD cannot repair semantically broken links semi-automatically, differing from our approach.

In terms of concept level, Dos Reis et al. [13] conducted a literature survey on mapping maintenance techniques for Knowledge Organization Systems (KOS), which are a set of structures aiming to represent knowledge, including ontologies, thesauri, vocabularies, among others. Mapping maintenance is a key issue of the Ontology Engineering field. It is responsible for fixing possible errors caused by the evolution of the ontology. The semantic update of a given concept can cause this type of semantic error, invalidating an existing mapping. The authors categorized existing studies into four categories: mapping revision, calculation, adaptation, and representation.

They concluded that most of the existing techniques of mapping maintenance use logical inference, thus depending on KOS with many logical formalizations [13]. The authors also pointed out the importance of searching for a small part of the KOS instead of the entire model to perform new reconnections. This can avoid time-consuming techniques. Thus, we adopted the strategy of not searching the entire dataset. We focus on specifically selected candidates to reconnect a broken link instead.

Pourzaferani and Nematbakhsh [5] proposed a tool for detecting and fixing the structurally broken link. This tool implements a method to identify broken links based on changes in the source code of the link. Thus, it uses an approach that differs from that of the majority of the methods reported in the literature, which focuses on the destination node. The method uses two sub ontologies containing the subjects and objects of the dataset that changed over time. These ontologies support finding possible resource similarities and new resources to be connected after discovering broken links. Using datasets from the "people" domain, the tool fixed almost 90 percent of the broken links. However, this tool cannot repair semantically broken links, as our framework does.

COLIBRI [14] is a software tool that creates links by discovering candidate resources in different datasets using unsupervised machine learning algorithms. Unlike other approaches, COLIBRI does not focus only on *owl:sameAs* links. Unlike COLIBRI, our framework emphasizes discovering broken links between distinct RDF datasets in two different releases.

Another approach related to this work is the idMesh [15]. It explores identity links (*e.g. owl:sameAs*) in datasets of the Web of Data. The idMesh uses a reasoner and the symmetric and transitive property of *owl:sameAs* predicate to discover inconsistent ones. It relied on the trustworthiness of some datasets to correct these links. In our approach, the transitivity property of the *owl:sameAs* predicate is also used to judge if a given resource can replace the broken resource of the link.

Our literature analysis indicated that there is no evidence of a complete solution that can identify RDF changes, track broken links, and automatically fix them as a unique system. Existing approaches mostly explore solutions based on the notification of syntactic broken links. Studies mainly emphasized auxiliary ontologies for documenting

the versioning of RDF changes. Our proposal advances state of the art in addressing how to handle semantic broken links as automatically as possible. We propose a complete framework containing maintenance actions that support users in addressing link maintenance.

3. LODMF framework

We propose LODMF to serve as a link maintenance tool for RDF datasets. This section formalizes the problem (subsection 3.1), describes the framework general overview (subsection 3.2) and its specific modules (subsections 3.3, 3.4 and 3.5).

3.1. Problem Formalization

This section presents the preliminary definitions followed by the link maintenance problem formalization.

A RDF triple (t) refers to a data entity composed of subject (s), predicate (p) and object (o) defined as $t = (s, p, o)$. A dataset ($\mathcal{R}$) in the context of Linked Data is a conglomeration of a finite number (n) of RDF triples in a domain. Formally, $\mathcal{R} = \{t_1, t_2, t_3, ..., t_n\}$.

In addition to the triples in a dataset, the linkage among several distinct datasets by interconnecting RDF repositories plays a key role in the LOD. To this end, it is necessary that a predicate establishes a relation between a subject of a first dataset (source) and an object of a second dataset (target). Formally, we define a link as $l =< r_a, p, r_b >$ connecting a pair of resources r_a and r_b, in which $r_a \in \mathcal{R}^S$ and $r_b \in \mathcal{R}^T$, such that $\mathcal{R}^S$ (source dataset) differs from $\mathcal{R}^T$ (target dataset). The studied resources (r_a and r_b) reside in a datasets of instances ($\mathcal{R}^S$ and $\mathcal{R}^T$) and are defined by their related classes, present in ontologies ($\mathcal{O}^S$ and $\mathcal{O}^T$, respectively, *cf.* Figure 1).

For the definition of p, we consider well-established properties to express the predicates of links including: *owl:SameAs*, *rdfs:seeAlso*, *owl:differentFrom* and SKOS mapping properties vocabulary[5]. We define a finite (k) set of links between $\mathcal{R}^S$ and $\mathcal{R}^T$ as follows: $\mathcal{L}_{\mathcal{ST}} = \{l_0, l_1, l_2, ..., l_k\}$.

The evolution of RDF datasets in terms of changes affecting their triples may invalidate previously well-established links. This hampers data linkage consistency over time. This scenario might derive a set of broken links, which are not able to represent their intended meaning as before the RDF evolution (*i.e.*, the meaning intended by the triple's author). In order to maintain the consistency of RDF datasets, their links should remain in an integrity state, even with the evolution of the associated RDF data resources.

There are two types of broken links: (1) *structurally broken links* occur when the source (where the subject lies) or the target (where the object lies) of the link cannot be dereferenceable [2]; and, (2) *semantically broken links* occur when the semantics do not represent the intention aimed by the triple's author anymore (*i.e.*, there are changes in meaning), even when the resource (subject or object) structure remains consistent.

As RDF datasets evolve, it is necessary to define the notion of time j. For an RDF dataset, we denote $\mathcal{R}^{S^j}$ as the initial version of the dataset $\mathcal{R}^S$ at time j and $\mathcal{R}^{S^{j+1}}$ as its evolved version (release) at time $j+1$. Consider a link l^j at time j and a link l^{j+1} based on distinct releases of the associated datasets (cf. Figure 1). Modifications occurring in a resource of the link (r_a) from a release of the dataset in a specific time j to a time $j+1$ can invalidate such link $l^j(r_a \rightarrow r_b)$. For example, r_a can exist at time j ($r_a \in \mathcal{R}^{S^j}$), but be modified ($r_a$ to r_z) at time $j+1$ ($r_a \notin \mathcal{R}^{S^{j+1}}$). This change can turn the link invalid if the link between r_z is inadequate to be semantically associated with r_b. Therefore, l^j should be updated to a new state $l^{j+1}(r_z \rightarrow r_b)$, turning link valid again.

The changed link $l^{j+1}(r_z \rightarrow r_b)$ should be evaluated in terms of semantic consistency to check if the link after the change remains healthy (not broken). For the consistency check, we define similarity comparisons (*e.g* between r_z and r_b). Based on Euzenat and Shvaiko [16], the definition of similarity is: given a set of strings K (in our case, labels of resources), the similarity is a given function that associates a pair of K values to a real number $\Re$, expressing how similar the compared K values (*cf.* Equation 1).

$$similarity : K \times K \Rightarrow \Re \tag{1}$$

[5]https://www.w3.org/TR/skos-reference/#mapping

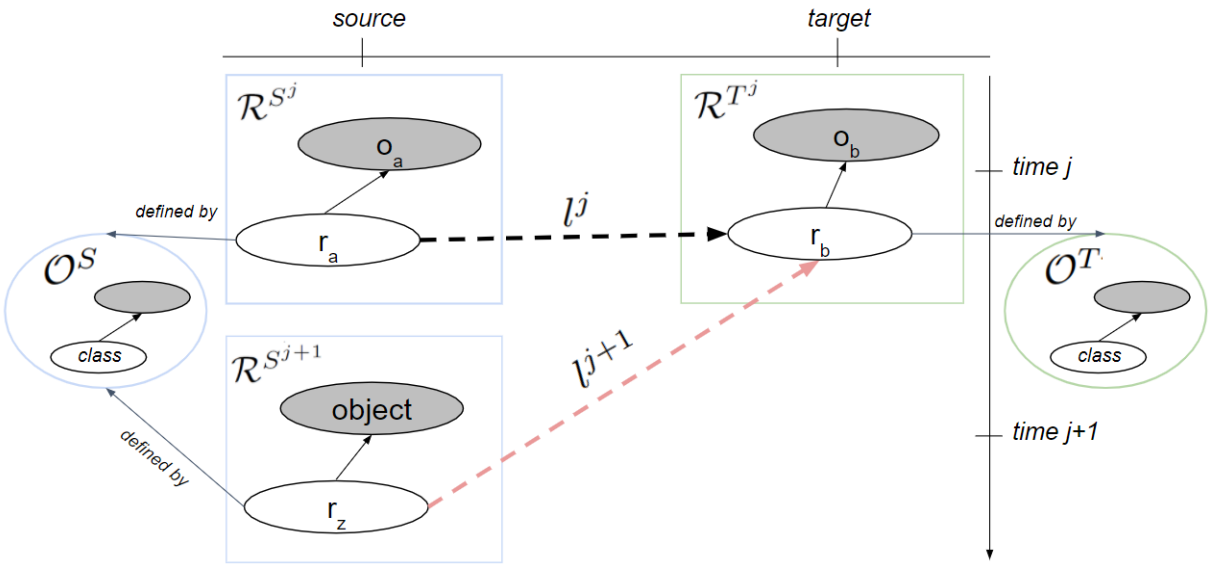

Fig. 1. Problem Formalization

The numeric value should be positive (*c.f* Equation 2) and the max similarity of K values should not be higher between K and itself (*cf.* Equation 3).

$$\forall x, y \in K, \quad sim(x, y) \geqslant 0 \tag{2}$$

$$\forall x, y, z \in K, \quad similarity(x, x) \geqslant similarity(y, z) \tag{3}$$

We assume basic premises for the complete functioning of our solution. First, some types of resources should be presented in the RDF dataset (*e.g.,* label, type) because our procedure uses them to obtain candidate resources. Without them, the designed algorithm cannot guarantee appropriate similarity computation among links. In addition, our proposed framework requires two different RDF dataset versions. The difference between $\mathcal{R}^{S^j}$ and $\mathcal{R}^{S^{j+1}}$ should not be null including the difference between their links.

3.2. General description of the framework

The LODMF framework defines maintenance actions to be performed and fix structurally and semantically broken links when a dataset evolves. Figure 2 presents our proposed framework responsible for keeping links up-to-date in LOD datasets.

The required input refers to interconnected datasets, consisting of the $\mathcal{R}^{S^j}$, $\mathcal{R}^{T^j}$ and the links $\mathcal{L}^j_{ST}$. Given a new version of one of the datasets (assuming that only one of the datasets evolves per time) $\mathcal{R}^{S^{j+1}}$, the goal is to obtain the new up-to-date version of the links $\mathcal{L}^{j+1}_{ST}$. In order to apply maintenance actions, the LODMF framework performs three steps (cf. Figure 2). Step A is responsible for searching and retrieving all changes between two releases of the involved datasets. Step B identifies, among these changes, which ones positively and negatively affect links in place. Step C fixes the negatively affected links.

Algorithm 1 defines a link maintenance procedure developed according to the LODMF framework. The algorithm requires the input datasets and a normalized β value. This value refers to a similarity threshold responsible for

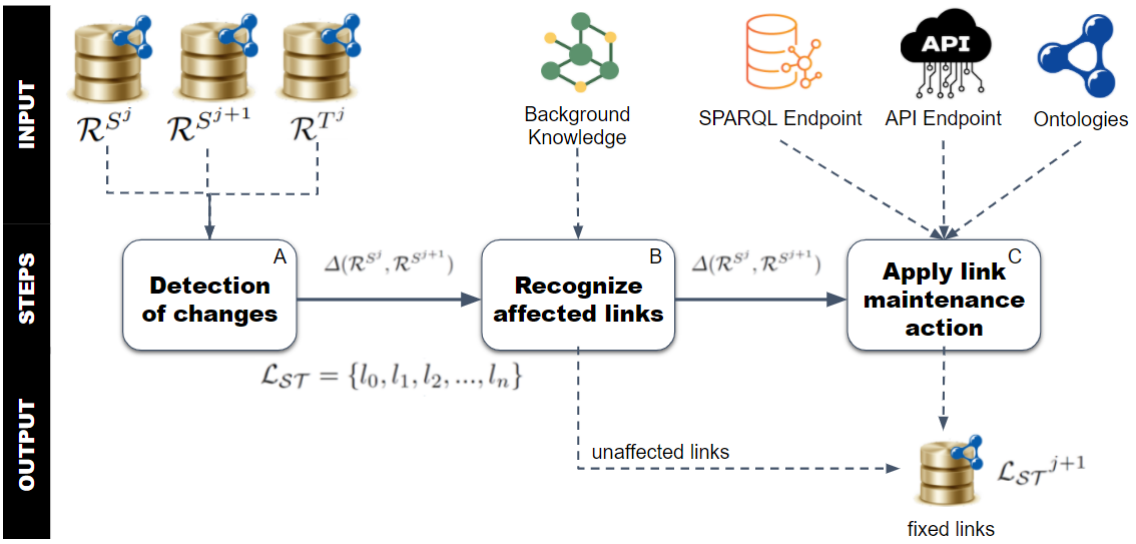

Fig. 2. LODMF Framework

defining resources candidates that can replace the broken part of the link. Resources with values lower than β are not suitable candidates for the replacement. The resulting output refers to updated links $\mathcal{L}_{ST}^{j+1}$. Line 3 of Algorithm 1 refers to Step A (cf. Figure 2), lines 4, 5 and 6 refer to Step B and lines 7 to 10 refer to Step C.

Algorithm 1 Link maintenance algorithm.

Require: $\mathcal{R}^{S^j}, \mathcal{R}^{S^{j+1}}, \mathcal{L}_{ST}^{j}, \beta, CAND, l_a$

1: $actions \leftarrow \emptyset$
2: $\mathcal{L}_{ST}^{j+1} \leftarrow \emptyset$
3: $\Delta \leftarrow detectChanges(\mathcal{R}^{S^j}, \mathcal{R}^{S^{j+1}})$;
4: $\mathcal{L}_{ST}^{aff} \leftarrow recAffecLinks(\Delta, \mathcal{L}_{ST}^{j})$;
5: $\mathcal{L}_{ST}^{unaff} \leftarrow recUnAffecLinks(\Delta, \mathcal{L}_{ST}^{j})$;
6: $\mathcal{L}_{ST}^{j+1} \leftarrow \mathcal{L}_{ST}^{j+1} \cup \mathcal{L}_{ST}^{unaff}$;
7: **for all** $l_i \in \mathcal{L}_{ST}^{aff}$ **do**
8: $CAND \leftarrow Algorithm2(l_i)$;
9: actions$[l_i] \leftarrow Algorithm3(l_i, CAND, \beta)$;
10: **end for**
11: **return** actions;

3.3. Step A: Detection of Changes

The initial step (Step A in Figure 2) is responsible for detecting changes that occurred in a given period of time based on two releases of the involved LOD datasets. These changes can be simple changes (like simple addition or removal of triples) or complex changes (update action) of the knowledge stored in datasets (Line 2 of Algorithm 1). We consider the following RDF triple changes:

- $RemovedTriple(t_k) \longrightarrow (t_k \in \mathcal{R}^{S^j} \wedge t_k \notin \mathcal{R}^{S^{j+1}})$;
- $ModifiedSubTriple(t_k) \longrightarrow (t_k \in \mathcal{R}^{S^j} \wedge t_k \in \mathcal{R}^{S^{j+1}}, r_a^j \neq r_a^{j+1})$
- $ModifiedPredTriple(t_k) \longrightarrow (t_k \in \mathcal{R}^{S^j} \wedge t_k \in \mathcal{R}^{S^{j+1}}, p^j \neq p^{j+1})$

– $ModifiedObjTriple(t_k) \longrightarrow (t_k \in \mathcal{R}^{S^j} \wedge t_k \in \mathcal{R}^{S^{j+1}}, r_b^j \neq r_b^{j+1})$

Algorithm 1 requires the dataset $\mathcal{R}^S$ in two different release versions for computing the changes. We consider one release at time j and its evolution at time $j+1$. The output of this step is a delta of changes defined by $\Delta(\mathcal{R}^{S^j}, \mathcal{R}^{S^{j+1}})$.

3.4. Step B: Recognize Affected Links

The second step aims to recognize changes identified during Step A which affected existing links in place. Step B (cf. B in Figure 2) requires as entry data the set of all changes detected (*i.e.*, $\Delta(\mathcal{R}^{S^j}, \mathcal{R}^{S^{j+1}})$) and the set of all links ($\mathcal{L}_{\mathcal{ST}}$). Lines 4, 5 and 6 of Algorithm 1 define this step.

Line 5 of Algorithm 1 detects the unaffected links. If an input link is not associated with any RDF triple change, it is classified as unaffected. Our approach measures the degree of impact of a triple change in the link semantics. The degree of impact is calculated by comparing a similarity value between the resources before evolution and the similarity value after evolution. Similarity refers to a function that computes a value that defines how close one resource is to another. In particular, our investigation explored semantic similarity functions, which rely on background knowledge, and explored the degree of proximity of the input resources in the background knowledge based on different techniques, *e.g.* path-based ones. The similarity returned is a normalized value $[0-1]$.

In our approach, if the "before evolution similarity value" outperforms the "after evolution similarity value", then we have a possible case of a semantic broken link [8]. We understand that in a semantically broken link, the similarity among involved resources decreased compared to the original link at time j [8].

If the occurred change affects the meaning of the link between two datasets, this link is added to $\mathcal{L}_{ST}^{aff}$ to be further processed (Line 4 of Algorithm 1). Otherwise, the link is inserted in a list of unaffected links (Line 5 of Algorithm 1). Line 6 of Algorithm 1 computes a union of the unaffected links to those that will be the final result. In this sense, the unaffected links are reused and added to the final up-to-date set of links $\mathcal{L}_{ST}^{j+1}$.

3.5. Step C: Apply Link Maintenance Action

This step (C in Figure 2) focuses on applying corrective actions in a set of affected links discovered during step B. To perform the corrective actions, our framework searches for candidates that can replace the broken part of the link. The candidates are resources in the source dataset. Then, Algorithm 3 decides which maintenance action is the one that effectively repairs the affected link.

Lines 7 to 10 of Algorithm 1 define this step. In line 7, the algorithm runs over all affected links. First, Algorithm 1 (at line 8) generates the list of candidates. Then, based on that list, the algorithm chooses which maintenance action should be taken - after the user review and determines how to reestablish the unbroken state of a link (line 9). The LODMF considers the suggestion of more than one action to users. In the end, only one maintenance action is applied for each triple change operation.

Subsection 3.5.1 presents the procedure for the generation of the list of candidates. Subsection 3.5.2 explores the possible actions to be taken over the links.

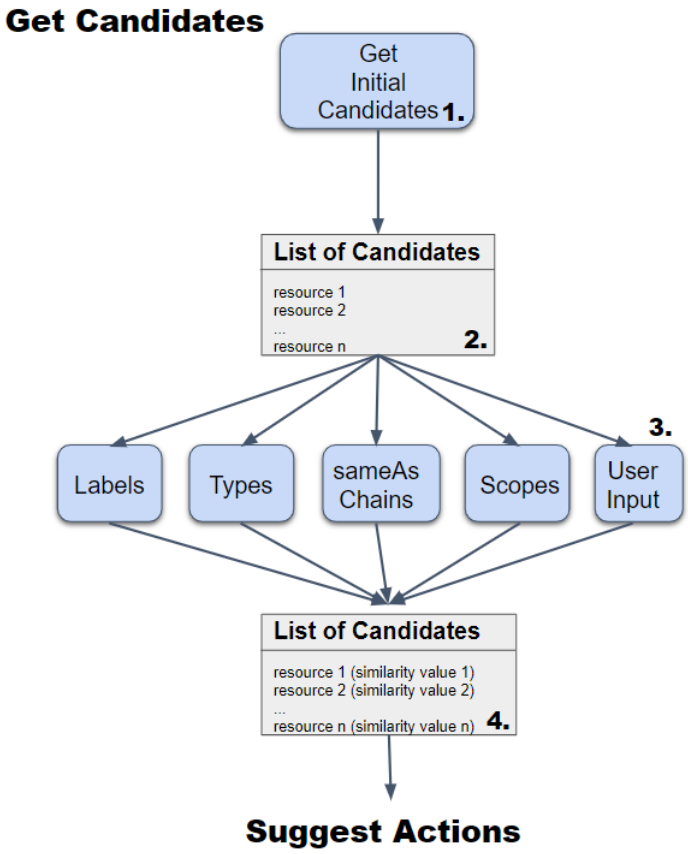

Fig. 3. Methodology to generate a list of resource candidates *CAND*

3.5.1. Generation of resource candidates

In this step the LODMF generates a list of resource candidates $CAND = \{r_{c_0}, r_{c_1}, r_{c_2}, ..., r_{c_k}\}$ to substitute the broken part of an affected link. The broken part of a link is a resource (cf. Figure 1 – the subject). Algorithm 2 specifies the steps of the generation of a list of resource candidates. Figure 3 illustrates the step-by-step methodology used to generate the list of candidates, which starts getting the initial list of candidates (number 1 at Figure 3) and ends generating a list of candidates with their attached values of similarities (number 4 at Figure 3).

The list of pairs (before and after evolution) of affected links is one of the inputs, which is denoted by $\mathcal{L}_{ST}^{aff}$ and represented as triples. Each link before the change l^j is composed by a subject r_a, predicate p and object r_b. Each broken link after the change l^{j+1} is composed by a modified subject r_z, the same predicate p and object r_b. A link after the change includes a resource that changed and affected the semantics of the link.

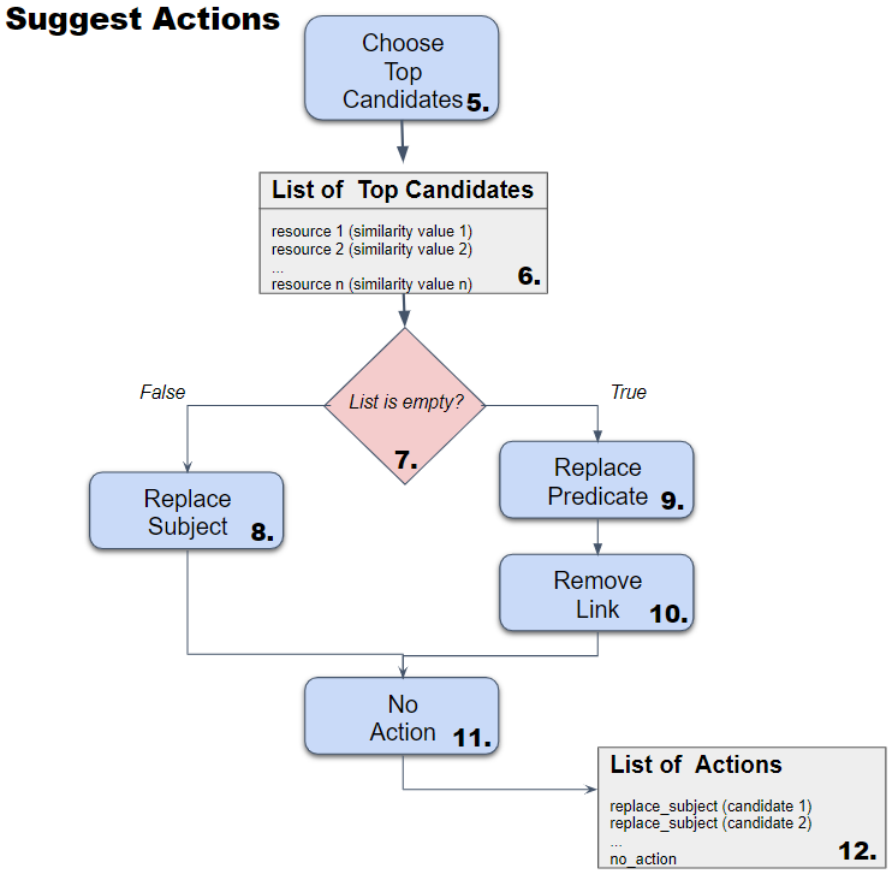

Fig. 4. Methodology to suggest actions to the user based on the list of candidates *CAND*

Our approach requires as input two ontologies containing the classes of both R^S and R^T datasets for the selection of candidates ($\mathcal{O}^{S^{j+1}}$ and $\mathcal{O}^{T^j}$, respectively), as long as a background knowledge B performs the comparisons to decide which one is the appropriate to substitute the resource. Optionally, users can access classes and instances of both R^S and R^T using SPARQL or an API in case that $\mathcal{O}^{S^{j+1}}$ and $\mathcal{O}^{T^j}$ are not provided (*cf.* Figure 2). Any operation in step three involving the source dataset R^S is performed in the evolved version $R^{S^{j+1}}$. The version R^{S^j} is only used in the first and second steps of the LODMF framework.

The output is a list of top chosen candidates that could replace the resource that changed from r_a to r_z and turned the link broken.

Algorithm 2 Algorithm to retrieve candidates and their respective values of similarity.

Require: $\mathcal{O}^{S^{j+1}}, \mathcal{O}^{T^j}, B$

1: **function** $getCandidates(l_i)$
2: $CAND \leftarrow getInitialCandidates(l_i, B)$;
3: $CAND \leftarrow compareLabels(l_i, CAND)$;
4: $CAND \leftarrow compareTypes(l_i, CAND, \mathcal{O}^{S^{j+1}}, \mathcal{O}^{T^j})$;
5: $CAND \leftarrow compareSameAsChain(l_i, CAND)$;
6: $CAND \leftarrow compareScopes(l_i, CAND)$;
7: $CAND \leftarrow compareUserInput(l_i, CAND)$;
8: **return** $CAND$

Algorithm 2 starts with the search in $R^{S^{j+1}}$ for all resources that have the same label of the object (line 2 of Algorithm 2 and step 1 of Figure 3). For example, the resource with label "Niagara Falls" in Wikidata (R^{T^j}) returns 15 resources at DBpedia ($R^{S^{j+1}}$) search. The rationale behind this task is that resources sharing the same label in $R^{S^{j+1}}$ and R^{T^j} might refer to the same real-world object.

The results from the initial search are expanded using the background knowledge. Our solution looks for synonyms of the label (line 2 of Algorithm 2 and step 1 of Figure 3). For instance, in the "Niagara Falls" running example, the query expansion returns the synonym "Niagara". Our solution searched for resources that have the label "Niagara" at $R^{S^{j+1}}$ and returned 13 results in addition to the 15 found with the label "Niagara Falls" (running example). The set containing these 28 results (15 from "Niagara Falls" and 13 with "Niagara") composes the list of resources. From now on, we refer to this list of resources from the $R^{S^{j+1}}$ as the list of candidates (*CAND*).

Based on the list of candidates (step 2 of Figure 3), algorithm 2 computes comparisons to determine which are the best candidates to fit the place of the broken resource r_z. Comparisons between each candidate of the list *CAND* and the object resource r_b are executed. The reliable part of the link is the resource that did not change between versions. If from version R^{S^j} to $R^{S^{j+1}}$, the subject of the link changed and created a broken link, then the object is the reliable resource of the link.

The comparison outputs values of similarity using semantic similarity algorithms. Semantic similarity computes if two input terms share the same meaning even though they differ in a lexical way. Usually, algorithms to calculate the semantic similarity depend on external sources of knowledge, like vocabularies, thesauri, dictionaries and ontologies. These external sources are used to calculate the similarity among items – named here background knowledge. In particular, LODMF explores two semantic similarities algorithms: path-based using WordNet and Nasari using BabelNet.

The comparisons (Step 3 of Figure 3) are:

Comparing Labels: Algorithm 2 (line 3) compares labels of each resource from the list of candidates with the label of object resource of the link (r_b) (*c.f* Comparison 1 of Figure 5). Algorithm 2 uses every resource from *CAND* that contains the property value *rdfs:label*, *skos:altLabel*, *skos:preferedLabel* or *foaf:name* to obtain the labels. These properties are input parameters of the framework. Each pair of the compared elements receives a normalized value indicating the degree of similarity.

Comparing Instance Of/Type: While Comparison 1 focuses on instance level, at this stage, the algorithm compares ontology classes (*rdf:type* predicate) that are instancing the candidates and the reliable resource (*cf.* Comparison 2 of Figure 5). In this sense, the second comparison performed is related to the labels of ontology classes which are instantiating each candidate and the link object (line 4 of Algorithm 2).

The search for classes at the ontology level ($\mathcal{O}^{S^{j+1}}$ and $\mathcal{O}^{T^j}$) is necessary. To this end, the algorithm selects every resource of the list of candidates *CAND* and from r_b that has the predicate *rdf:type* (*c.f* Comparison 1 of Figure 5). Then, the labels are selected, and each pair of the compared elements receives a normalized value indicating the degree of similarity.

Code 1 shows a SPARQL query used in the framework to select all labels of classes that are instantiating a given resource. The SPARQL query is used in the case that the user in the framework did not provide the ontology.

Comparing SameAs Chains: For this comparison, the LODMF uses the transitive property of *owl:sameAs* predicate. This states that if a given resource r_a is *owl:sameAs* with another resource r_b, and r_b is *owl:sameAs* with another resource r_c, then r_a is *owl:sameAs* r_c. This chain is used by our framework to identify what are the resources that r_b is externally linked to (*c.f* Comparison 3 of Figure 5). The label of these resources is compared to each label of the list *CAND*. As with other comparisons, the "sameAs Chain comparison" produces values of similarity among the resource candidates and r_b (line 5 of Algorithm 2).

Comparing Scopes: The rationale behind the scope's comparison is that a resource from *CAND* sharing the same broader label with r_b may represent the same real-world object. The same rationale is applied to narrower labels (*cf.* Comparison 4 of Figure 5).

At this step, Algorithm 2 selects labels of resources that have broader scopes than the individuals of the list of candidates (*e.g.* "animal" has a broader scope than "mammal"). Afterwards, the same procedure is applied in r_b. These labels are then semantically compared (line 6 of Algorithm 2). The same procedure is performed with

narrower scopes. Resources with wider scope are retrieved by our algorithm looking for the property *skos:broader*, while getting the resources with smaller scope requires the property *skos:narrower*.

Comparing User Input: At this step, Algorithm 2 compares properties explicitly provided by the user to identify similarities among the list of candidates and the reliable resource r_b (line 7 of Algorithm 2). Some datasets, such as Geonames, do not use properties listed at the previous comparisons: *rdfs:label*, *rdf:type*, *owl:sameAs*, and others. This situation can hamper the comparisons and hinder the well-functioning of our framework. To overcome this issue, our solution benefits from the user's expertise. The user, which in most cases is a dataset maintainer, can setup which properties must be compared (*c.f* Comparison 5 of Figure 5).

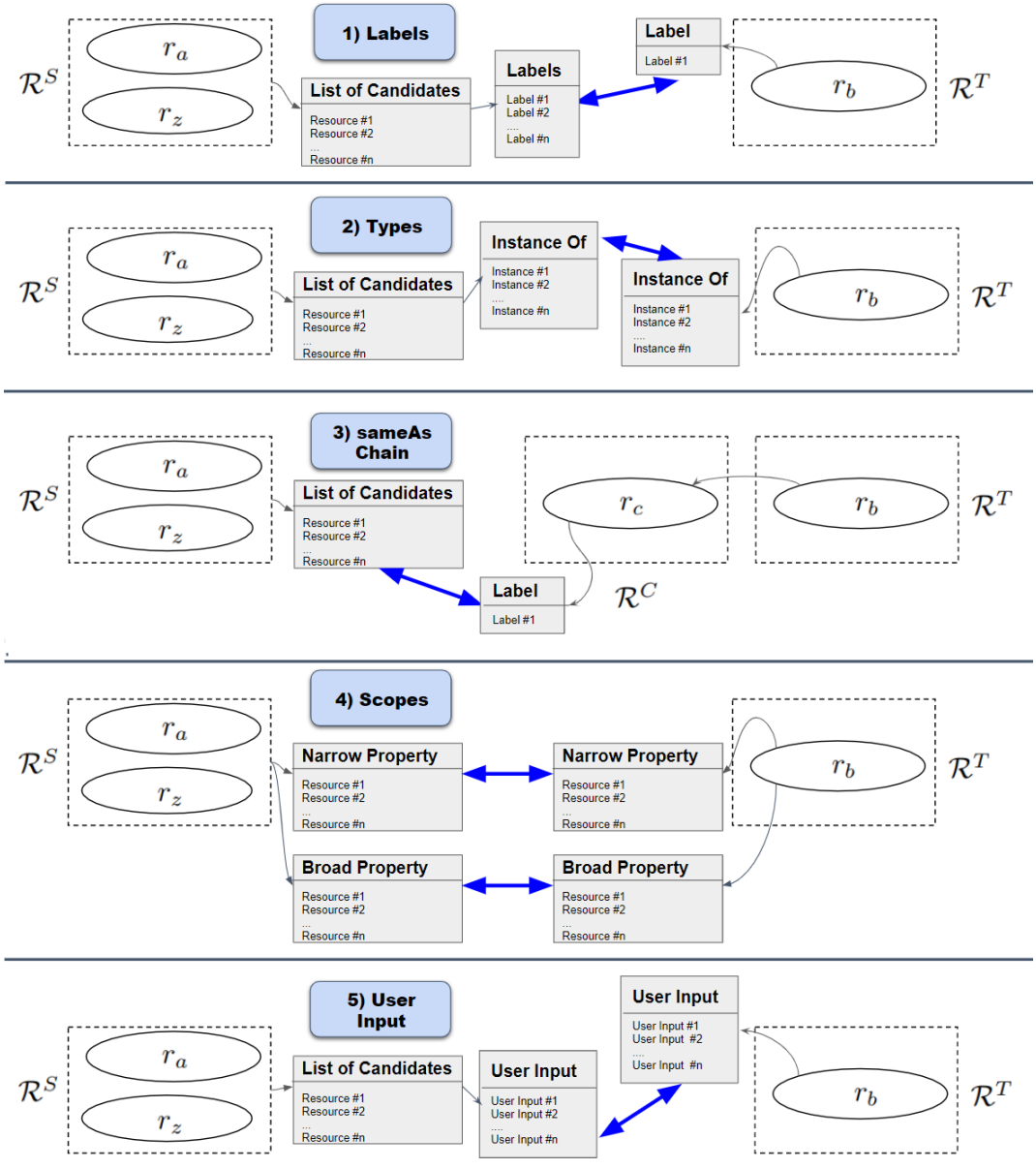

Fig. 5. List of actions performed in each candidate of *CAND*. Each type of comparison is presented inside the blue squares. The extremities of the blues arrows indicate which labels are compared in terms of similarity.

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX skos: <http://www.w3.org/2004/02/skos/core#>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT DISTINCT ?label where {
    ?subject rdf:type ?object .
    VALUES ?subject { <resource> } .
    ?object rdfs:label ?label .
    FILTER (lang(?label) = 'en')
}
LIMIT 50
```

Listing 1: SPARQL query responsible for selecting classes that are instantiating a resource

Top k Candidates: At this step, each resource from *CAND* receives a similarity value by each of the mentioned comparisons, generating a matrix of similarity values (step 4 of Figure 3). Table 1 presents an example of results of similarity between resources based on each comparison. This example simulates a situation where the list of candidates has four resources. Algorithm 2 calculates the mean value for each resource, represented by the lines in Table 1. The mean is calculated over the values ranging from 0 to 1 and excluding the columns with "X". The value X is attributed in two situations:

- When the property of a given comparison is not presented in the datasets R^S or R^T. At column C4 (Table 1), the property used to evaluate the fourth comparison is not presented in the dataset (*skos:narrower* or *skos:broader*) since none of the resources returned values of similarity;
- When the property of a given comparison is not presented in a specific resource. At column C3 (Table 1), we observe that the resource r_2 does not have the property used in the third comparison (the object may not have an *owl:sameAs* link to create a *sameAs* chain).

In the Table 1 example, by excluding "X" in the mean value, the generated ranking of resources is r_3, r_2, r_4 and r_1. Based on this list, Algorithm 3 proceeds to the next step: choose the maintenance action to be taken.

3.5.2. Maintenance Actions

The list of candidates (*cf.* subsection 3.5.1) is used in the task of selecting actions to be performed on the resources. Algorithm 3 and Figure 4 details the steps to select the maintenance actions to be suggested to the users.

Three parameters are defined at line 2 of Algorithm 3: the link that is being fixed (l_i); the list of resources candidates (*CAND*); a similarity threshold (β), which is defined by the user. When it is not defined, the default threshold value of 0.5 is assumed by Algorithm (lines 3 - 5 of Algorithm 3). The value of each candidate is compared to the threshold value β, generating a list containing the top candidates *topCAND* (lines 6 - 10 of Algorithm 3 and steps 5 and 6 in Figure 4). The list generated from Table 1 contains only resource r_3, the only resource to reach a value higher than β.

	C1	**C2**	**C3**	**C4**	**C5**	**Mean**
r_1	X	0.10	0.21	X	0.00	0.1
r_2	0.31	0.11	X	X	0.72	0.38
r_3	1.00	0.99	0.99	X	0.7	0.92
r_4	0.17	0.4	0.12	X	0.00	0.17

Table 1

Example of a generated matrix of comparisons by candidate resources. The lines refer to each resource in the list of candidates *CAND*. The columns refer to the following comparisons: (C1) Label; (C2) Instance of; (C3) sameAs Chain; (C4) Scopes; (C5) User Input. The values indicate that such candidate (in the row) presents similarity value in the range [0,1] with respect to the link object r_b. The "X" sign indicates that the similarity could no be computed.

Based on the parameters of line 2 and in the list of candidates generated at line 8, Algorithm 3 suggests the best action to the user. Assuming this is a semiautomatic process, the user will have the final decision. The possible decisions are:

- **Reconnection of subject**: $replaceSub(l_i, r_x^{j+1}) \longrightarrow (l_i \in \mathcal{LST}^{j+1} \wedge r_x^{j+1} \in topCAND, r_z^{j+1} \neq r_x^{j+1})$.
  This suggestion of action is shown to the user together with each top-ranked candidate (line 12 of Algorithm 3 and Step 8 in Figure 4). This option is presented in case the value of similarity from the presented candidates is equal or higher than the threshold β (lines 7 and 11 of Algorithm 3). The lack of values higher than the threshold means that there is no candidate in *CAND* (the list is empty *cf.* step 7 of Figure 4), which can replace the subject part of the link (r_z) and be connected to the reliable object r_b using the predicate *owl:sameAs*;
- **Replacement of predicate**: $replacePred(l_i, p^{j+1}) \longrightarrow (l_i \in \mathcal{LST}^{j} \wedge l_k \in \mathcal{LST}^{j+1}, p^j \neq p^{j+1})$.
  This suggestion is based on comparison 4 (scopes) (line 14 of Algorithm 3). If a broader scope is found between R^T and R^S, then the substitute of the broken *owl:sameAs* predicate is a predicate with narrower scope (*skos:narrower*). The opposite situation (narrower scope found and broader replacement) can also occur (*cf.* Step 9 of Figure 4).
  Another possible replacement is to change the *sameAs* predicate to a "lighter" (less restrictive) semantic predicate, such as *skos:closeMatch* (line 14 of Algorithm 3). This replacement option can be applied when none of the resources of the list of candidates reached the value of the threshold β (line 13 of Algorithm 3);
- **Removal of the link**: $removeLink(l_i) \longrightarrow (l_i \in \mathcal{LST}^{j} \wedge l_i \notin \mathcal{LST}^{j+1})$.
  Another possible decision is to remove the link (line 15 of Algorithm 3 and Step 10 of Figure 4). This suggestion is shown to the user when there is no candidate to replace the broken link l_i (*topCAND* is also empty), based on the threshold β (line 13 of Algorithm 3).
- **No action:** $noAction() \longrightarrow (l_i \in \mathcal{LST}^{j} \wedge l_i \in \mathcal{LST}^{j+1})$.
  At least one of the three possible decisions presented above is suggested to the user. However, the framework can produce two types of wrong outputs. The first case is the wrong categorization, stating that a healthy link is broken. The other case is the wrong choice of action to be taken. In both cases, the user can decide to take no action, *i.e.*, to keep the link as it is (line 17 of Algorithm 3 and step 11 of Figure 4).

Line 18 of Algorithm 3 and Step 12 of Figure 4 return a list of possible actions to the user, who chooses the best one. Although more than one action can be suggested to the user, they can select only one action to be performed by the LODMF.

After the user chooses the action, Algorithm 1 proceeds with the commit of the changes as follows:

- If the user chooses one of the modification actions (reconnection of subject or replacement of predicate), then Algorithm 1 deletes the incorrect link and creates a new one, with the updated resource (subject or predicate). The number of triples in the dataset remains the same.
- If the user choose to remove the link, then the link is deleted from the dataset, reducing the total number of the triples in the dataset.
- If the user chooses to perform no action, then no change is performed in the dataset, resulting in no changes in the total number of the triples.

4. Illustrating the LODMF application

This section presents an example of a broken link and how the LODMF execution deals with it. We briefly describe the first and second steps of how our framework works in Figure 2 and focus on the third step. Figure 6 shows two datasets used as an example: $\mathcal{R}^S$ is DBpedia and $\mathcal{R}^T$ represents Geonames. DBpedia is the RDF dataset version of Wikipedia. Geonames is an RDF dataset that presents structured information in the geography domain, such as countries, states, cities. For DBpedia, we provided as input to the algorithm the SPARQL endpoint[6]. For Geonames, the API endpoint[7].

[6] http://dbpedia.org/sparql
[7] http://api.geonames.org/getJSON

Algorithm 3 Algorithm to obtain suggestions of link maintenance actions.

```
topCAND ← ∅; actions ← ∅;
function suggestAction(l_i, CAND, β)
    if β = NULL then
        β ← 0.5
    end if
    for all candidate ∈ CAND do
        if getSimilarityValue(candidate) ⩾ β then
            topCAND ← topCAND ∪ candidate;
        end if
    end for
    if topCAND ≠ ∅ then
        actions ← replaceSubject(l_i, topCAND)
    else
        actions ← replacePredicate(l_i)
        actions ← actions ∪ removeLink(l_i)
    end if
    actions ← actions ∪ noAction()
    return actions
```

We selected one of the DBpedia triples, which states that "Niagara Falls" is at latitude 43.07 at time j. The concept "Niagara Falls" of the DBpedia is linked to the resource "Niagara Falls" of Geonames using the predicate "sameAs". A new version $j+1$ of DBpedia was released, and some maintainer changed "Niagara Falls" to "Niagara River". The previously established link "sameAs" became obsolete because we cannot further assure that "Niagara River" concept of DBpedia is "sameAs" of "Niagara Falls" concept of Geonames. Niagara Falls and Niagara river are quite different in various aspects, and the falls are only a part of the river.

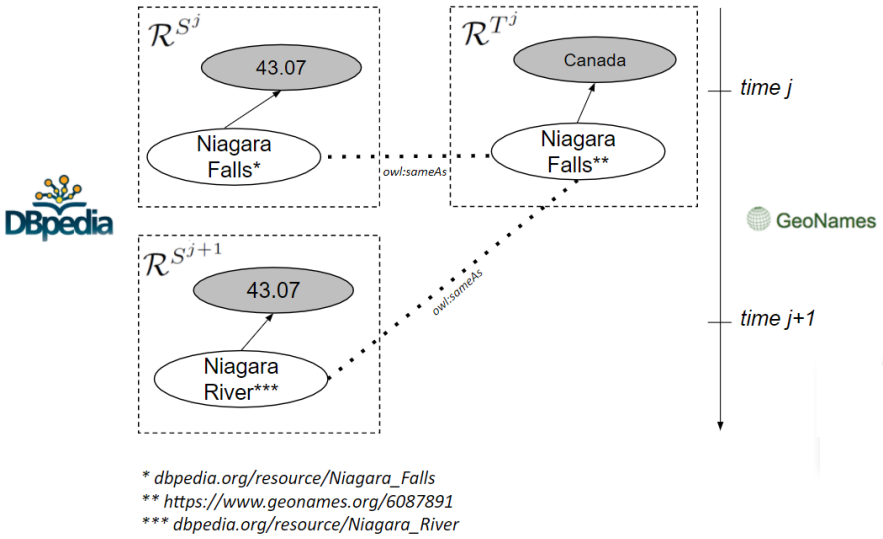

Fig. 6. Example of Broken Link Between DBpedia and Geonames

4.1. Step A: Detection of Changes

Given the scenario presented in Figure 6, we illustrate how LODMF deals with this broken link occurrence. Step A detects all changes between the releases of the DBpedia dataset at times j and $j+1$. Among all detected changes, the framework finds a link change ("Niagara Falls" from DBpedia and "Niagara Falls" from Geonames representing the same thing in the real world). The framework processes the changes and produces a list of changes to be used in the following steps.

4.2. Step B: Recognize Affected Links

The second step categorizes all discovered changes as possible sources of broken links (list of affected links) or changes with no effects on existing links (list of non-affected links). In the example, the change from "Niagara Falls" to "Niagara River" created a semantic inconsistency in the RDF dataset. The inconsistency was found by Algorithm 1 due to semantic difference between the resources: one is related to a cataract, the other is related to a river. On this basis, this link was included in the list of affected links. If the change operation did not modify the meaning of the resource, the link is added to the list of non-affected links. For instance, if the label "Niagara Falls" changed its initial letter to lowercase at time $j+1$ ("niagara falls"), such lexical change would not affect the semantics of the link between DBpedia and Geonames.

To identify semantic changes, we applied a series of comparison/analysis algorithms between the resources, which involves algorithms to compute the semantic degree relation between resources [8]. Lines 4 and 5 of Algorithm 1 shows the creation of both lists of unaffected and affected links, based on the changes found in Step A.

4.3. Step C: Apply Link Maintenance Action

The last LODMF step receives the list of affected links as input, and the framework suggests one or more actions to be performed. The user evaluates these actions and chooses the most appropriate one. For instance, if DBpedia has a semantically close resource to Niagara Falls, then the re-connection should be performed.

The first step in the maintenance procedure is to get all semantically related resources to "Niagara Falls". To this end, Algorithm 2 searches in a background knowledge such as WordNet for all synonyms of the term "Niagara Falls", excluding the label of the subject that is broken (Niagara River). The returned terms are "Niagara" and "Niagara Falls".

In the next step, Algorithm 2 searched DBpedia for resources that have these terms in their labels. This generates a list of candidates *CAND* that can potentially replace the broken resource "Niagara River". Table 2 presents a complete list of resources and their associated labels.

The list of candidates avoids searching the entire dataset for an appropriate substitute. Algorithm 2 searches for a substitute only in the list, reducing the execution time and computational resources needed. We note that some resources are not related to the term "falls" even though they include "Niagara Falls" in their labels. For example, there are selected resources related to movies, bands, and albums (cf. Table 2).

The performed comparisons (number 3 of Figure 3) generated the following outputs:

- **Values of similarity in label comparison.** Each label from Table 2 is semantically compared to the label "Niagara Falls";
- **Values of similarity in the instance comparison.** Each label of the type of the resource from *CAND* is semantically compared to the label "waterfall", which is the label of the type of "Niagara Falls". Figure 7 illustrates some examples of comparisons between some labels and "waterfall";
- **Values of similarity in the "sameAs Chain comparison".** In our running example, no values were retrieved in this comparison because there are no *owl:sameAs* outgoing link at Geonames;
- **Values of similarity in the scope comparison.** No values were retrieved in this comparison since Geonames does not represent the properties *skos:narrower* or *skos:broader*.
- **Values of similarity concerning the user input properties.** We used the property *dboCountry* in DBpedia and *countryName* in Geonames. They both returned the name of the country of the resources. Each retrieved

result from the list of candidates were semantically compared to the country name "Canada", retrieved with the property *countryName* in Geonames.

ID	Label	Resource
1	**Niagara**	**/Niagara
2	**Niagara, British Columbia**	**/Niagara,_British_Columbia
3	**Niagara, Kentucky**	**/Niagara,_Kentucky
4	**Niagara, Toronto**	**/Niagara,_Toronto
5	**Niagara, Western Australia**	**/Niagara,_Western_Australia
6	**Niagara (The Office)**	**/Niagara_(The_Office)
7	**Niagara (band)**	**/Niagara_(band)
8	**Niagara (board game)**	**/Niagara_(board_game)
9	**Niagara (film)**	**/Niagara_(film)
10	**Niagara (palace steamer)**	**/Niagara_(palace_steamer)
11	**USS Niagara (1813)**	**/USS_Niagara_(1813)
12	**Niagara, Wisconsin**	**/Niagara,_Wisconsin
13	**Diocese of Niagara**	**/Diocese_of_Niagara
14	**Niagara (painter and singer)**	**/Niagara_(painter_and_singer)
15	**Niagara Falls**	**/Category:Niagara_Falls
16	**Niagara Falls**	**/Niagara_Falls
17	**Niagara falls**	**/Niagara_falls
18	**Niagara Falls (1932 film)**	**/Niagara_Falls_(1932_film)
19	**Niagara Falls (1941 film)**	**/Niagara_Falls_(1941_film)
20	**Niagara Falls (Chicago song)**	**/Niagara_Falls_(Chicago_song)
21	**Niagara Falls (EP)**	**/Niagara_Falls_(EP)
22	**Niagara Falls (Greg Hawkes album)**	**/Niagara_Falls_(Greg_Hawkes_album)
23	**Niagara Falls (Phish album)**	**/Niagara_Falls_(Phish_album)
24	**Niagara Falls (band)**	**/Niagara_Falls_(band)
25	**Niagara Falls (electoral district)**	**/Niagara_Falls_(electoral_district)
26	**Niagara Falls (provincial electoral district)**	**/Niagara_Falls_(provincial_electoral_district)
27	**Niagara Falls railway station (Ontario)**	**/Niagara_Falls_railway_station_(Ontario)
28	**Niagara Falls station (New York)**	**/Niagara_Falls_station_(New_York)
29	**Niagara Falls, Ontario**	**/Niagara_Falls,_Ontario

Table 2

List of resources and its labels that can replace the broken subject Niagara River. The symbol ** is a shortcut for http://dbpedia.org.br/resource

Table 3 presents all the calculated similarity values for each comparison analysis by each resource from the list of candidates. The last column represents the mean value regarding all similarity values computed.

As mentioned before, the top k candidates aim to define which resources are more similar to the object element of the link, the resource with the label "Niagara Falls" in Geonames. The Geonames resource is related to the waterfalls. This means that Algorithm 2 should return as top candidates in terms of similarity computation those resources related to Niagara waterfalls, *i.e.*, not similar to the movie (1932 or 1941 film) or the steamboat (palace steamer).

The top five ranked candidates based on the simple mean values according to results in Table 3 are:

- Category Niagara Falls with similarity value of 1;
- Niagara Falls and Niagara falls with similarity value of 0.68;

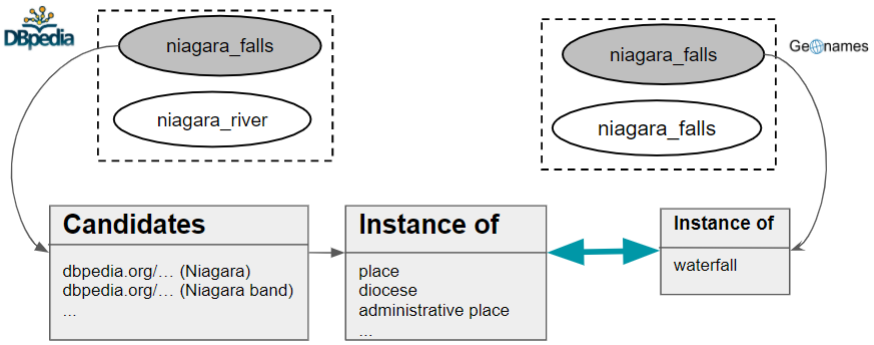

Fig. 7. Type/Instance of Comparison

Label	C1	C2	C3	C4	C5	Mean
Niagara	0.5	X	X	X	X	0.5
Niagara, British Columbia	0	0.03	X	X	1.0	0.34
Niagara, Kentucky	0	0.03	X	X	0	0.01
Niagara, Toronto	0	0.03	X	X	1.0	0.34
Niagara, Western Australia	0	0.19	X	X	0	0.06
Niagara (The Office)	0	0.03	X	X	X	0.01
Niagara (band)	0	0.17	X	X	X	0.08
Niagara (board game)	0	0.03	X	X	X	0.01
Niagara (film)	0	0.03	X	X	X	0.01
Niagara (palace steamer)	0	0.03	X	X	X	0.01
USS Niagara (1813)	0	0.03	X	X	X	0.01
Niagara, Wisconsin	0	0.03	X	X	0	0.01
Diocese of Niagara	0	0.19	X	X	X	0.09
Niagara (painter and singer)	0	0.04	X	X	X	0.02
Niagara Falls	1.0	0.36	X	X	X	0.68
Niagara Falls*	1.0	X	X	X	X	1.0
Niagara falls	1.0	0.36	X	X	X	0.68
Niagara Falls (1932 film)	0	0.03	X	X	X	0.01
Niagara Falls (1941 film)	0	0.03	X	X	X	0.01
Niagara Falls (Chicago song)	0	0.03	X	X	X	0.01
Niagara Falls (EP)	0	0.03	X	X	X	0.01
Niagara Falls (Greg Hawkes album)	0	0.03	X	X	X	0.01
Niagara Falls (Phish album)	0	0.03	X	X	X	0.01
Niagara Falls (band)	0.0	0.17	X	X	X	0.08
Niagara Falls (electoral district)	0	0.19	X	X	1.0	0.39
Niagara Falls (provincial electoral district)	0	0.19	X	X	1.0	0.39
Niagara Falls railway station (Ontario)	0	0.03	X	X	X	0.01
Niagara Falls station (New York)	0	0.03	X	X	X	0.01
Niagara Falls, Ontario	0.5	0.03	X	X	1.0	0.51

Table 3

Generated matrix built over the comparisons among the list of candidates and “Niagara Falls" resource, object of the link under analysis. The lines refer to each resource that is a candidate in DBpedia. The columns refer to the following comparisons: (C1) Label; (C2) Instance of; (C3) sameAs Chain; (C4) Scopes; (C5) User Input. The values indicate that such candidate has similarity value ranging [0,1] in comparison to the label “Niagara Falls". The "X" sign indicates that the similarity could not be computed.
*The label is also “Niagara Falls", but it is related to the DBpedia category name “Niagara Falls", not the individual resource.

– Niagara Falls, Ontario with similarity value of 0.51;
– Niagara, with similarity value of 0.5.

Based on these candidates, Algorithm 3 formulates actions to be taken to repair the link under analysis. In the "Niagara Falls" example, the first recommended action is to replace the subject. This action is presented because at least one candidate reached the similarity value above the threshold β. In our example, five resources are shown to the user. Case A in Figure 8 shows an example of replacement of the subject as a link maintenance action.

In our example, the action of predicate replacement is not suggested. This action would be indicated only if a broader or narrower resource was found between Geonames and DBpedia; or if none of the candidates in the list reached the threshold value (cf. Case B of Figure 8).

The third action is the complete removal of the link. Algorithm 2 does not indicate this suggestion of action to the user since some candidates have values higher than the threshold β in our running example (cf. Case C of Figure 8).

The fourth suggestion is keeping the link (take no action). The user may decide that the link is not broken or that the presented candidates are not good enough to substitute the resource "Niagara River" (cf. Case D of Figure 8).

4.4. Analysis of Results

During the generation of the candidate's list, Algorithm 2 discovered 29 resources that could potentially replace the broken resource "Niagara River". There were cases of Niagara's movies, cities, and songs among those resources. These types of resources would not be suitable to replace the resource in the broken link since the link creator intended to reference waterfalls.

The first comparison (identified by C1 column at Table 3) concerns with labels. Resources that were semantically equivalent to "Niagara Falls" received the max similarity value of 1, indicating that their label was semantically equivalent. The resource with the label "Niagara Falls, Ontario" received a similarity value of 0.5. Ontario is the Canadian province where the waterfalls are situated. This fact can explain why the value was different from 0.

This situation changed in the second comparison (cf. column C2 at Table 3) between the types. The value closer to 0 in "Niagara Falls, Ontario" indicates that "province" is not related to "waterfall". The resources with type "movie", "album", "musical work" and many others also received a low similarity value. The candidate with a higher C2 value was "Niagara Falls" (0.36) with type "body of water". The second highest value (0.19) was calculated with resources with type "region", such as "Niagara Falls, Western Australia" and "Niagara Falls electoral district".

C3 and C4 column values could not be calculated. The DBpedia, our source dataset, has the required properties to perform the comparison, but Geonames does not, making it impossible to calculate C3 and C4.

To overcome situations such as the C3 and C4 ones, or in any situation where an element could not be compared, we allowed users to inform the properties to be compared (case C5). In our example, most of the candidate resources do not possess a country name property (C5), mainly because they are not part of a territory or do not have a geography localization, such as "Niagara (board game)" and "Niagara Falls (Chicago song)". The C5 property was used to separate the resources not located in Canada, like "Niagara, Western Australia", an abandoned town in Australia; and "Niagara, Wisconsin", a small town in the USA. Every resource which has the property *countryName* with value equal to "Canada" has a C5 value of 1.

With all values computed for each comparison, the mean was calculated excluding the X values. The value with the highest similarity is the category "Niagara Falls". Wikipedia and DBpedia have pages used to disambiguate and group terms in common, as seen in one of the resources from $CAND$, the category "Niagara Falls". This resource groups many terms and pages specific to the waterfalls "Niagara Falls".

The second and third highest values of similarity mean were "Niagara Falls" and "Niagara falls". These resources represent the same real-world thing, the waterfalls called "Niagara Falls", which is the resource Algorithm 1 is looking for and the ones that the user would probably choose.

5. Discussion

Handling the impacts of RDF datasets evolution in established links is key to maintaining the consistency and benefits of LOD over time. This is especially valuable because the LOD presupposes a heavy interconnection be-

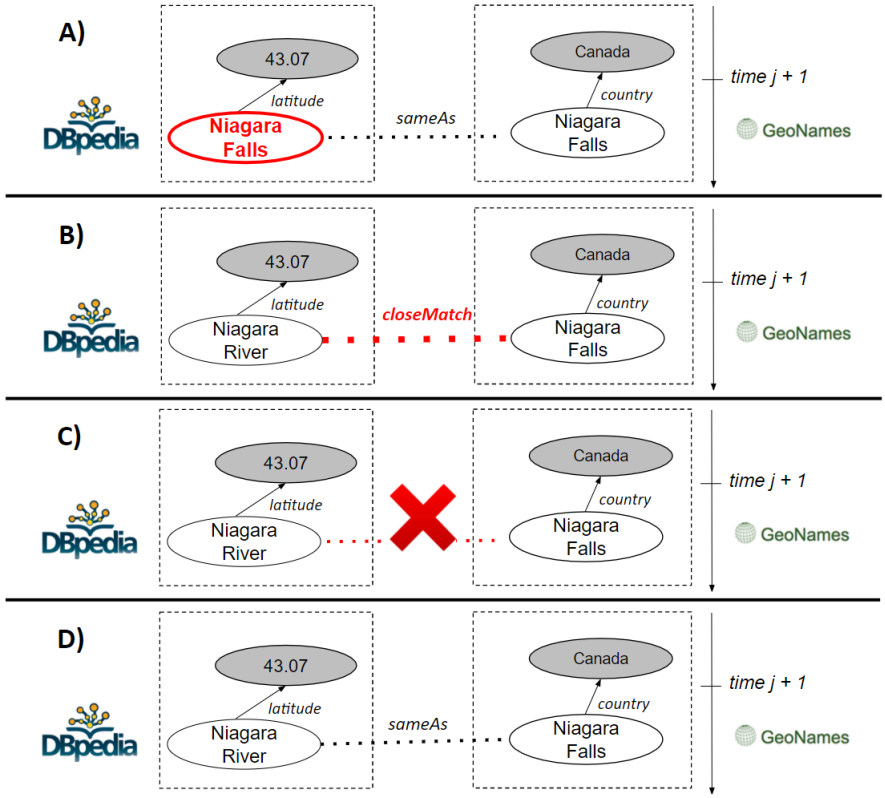

Fig. 8. Example of Maintenance Actions. A) replacement of subject; B) replacement of the predicate; C) removal of the link; D) no action.

tween databases and KOS. We proposed the LODMF framework, which can track and fix cases of semantically broken links. We contributed with a way to improve the quality of dataset linkage in LOD.

We described a real scenario with a broken link identified by our framework. We consider the running example of "Niagara Falls" a complete source to describe the processing of the framework as a whole. First, the framework detected that the link was changed between the RDF dataset versions. Then, using syntactic and semantic similarity measures, the LODMF discovered that the change negatively affected the semantics of the link. Afterward, the solution searched for resources that could replace the broken part of the link and considered other alternatives in addition to the replacement of the subject. The framework recommended actions are presented to the user, who chooses the best action and commits the changes.

It is worth mentioning that the performed validation did not emphasize quantitative metrics and comparisons with other approaches. It should also be noted that there are no ready-to-use gold standard datasets for comparative analyses in the specific context of this study. In addition, existing proposals found in the literature do not deal with the same issue investigated in our framework, which made side-by-side quantitative and statistical comparisons based on objective metrics with other studies impossible at the time. We consider it valuable to calculate precision, recall, f-measure, and time performance metrics. However, this is out of scope for our current study since this demands the creation of standard databases based on long-term studies.

We assume that, in our framework, the quality of the results is correlated with the right choice of background knowledge, which is used in the task of finding suitable candidates to replace the broken part of the link. The choice of domain-specific background knowledge to help maintain links using our framework could improve precision and recall results in a dataset related to this specific domain (e.g., using BioPortal for disease-related datasets). Our approach benefits from the ontology maintainer expertise because the final actions performed by the tool (replacement or exclusion of links) depend on expert approval. We assume that both the background knowledge choice and the expert supervision benefit the correctness of the output results from the tool.

We implemented an assisted solution that requires a user action in the last step (corrective actions). The framework does perform all actions automatically to avoid specific data errors. We understand that the expert's final decision and curation are important in deciding the consistency of the dataset analyzed.

The LODMF framework does not apply consistency checks or a reasoner over the processed links. The framework does not deal with inconsistencies at the concept level (TBox) but at the instance level (Abox) of LOD datasets.

The LODMF optimizes maintaining link consistency because it simplifies the required human actions. If the maintainer had to find these links, they would face a very time-consuming task. The same applies to finding the right candidate for replacement. By restricting the number of possible candidates using our proposed comparisons, the LODMF reduces and optimizes the choice only for minimally consistent resources and not for any resource in the entire dataset. Therefore, the effectiveness of the LODMF is shown in large datasets or datasets that change constantly.

For future work, the exploration of textual data (containing more than one label and string) can be a source of data to be used as a new comparison method. Properties such as *rdfs:comment, dbo:abstract, dct:description*, containing paragraphs about the resource would be handful to help find similarities among resources, complementing the comparisons already performed by the LODMF.

6. Conclusion

RDF statements defining real-world resources are subject to change for coupling with domain updates. Handling the effects of link changes in evolving datasets remains an open research challenge in managing RDF datasets in LOD. This is especially valuable because of the heavy interconnection present in LOD. This investigation proposed the LODMF framework for updating RDF links in a semi-automatic way. We described the detailed steps by including the identification of dataset changes and achieving choice of link maintenance actions. Our study provided an illustrative example fully describing the framework's execution to highlight the potentialities and challenges from our solution for repairing well-established semantically broken links. Future studies involve further experimental evaluations to assess the framework's execution better. We plan to implement complete link maintenance software tools by including the end-user interface for interacting with the candidate resources and maintenance actions. In addition, we intend to conduct a user study to evaluate the long-term use of the tool as well as to create standard databases so that we can execute quantitative and statistical comparisons between maintenance approaches.

References

[1] J. Umbrich, M. Hausenblas, A. Hogan, A. Polleres and S. Decker, Towards dataset dynamics: Change frequency of linked open data sources, in: *LDOW*, 2010.

[2] A. Singh, R. Brennan and D. O'Sullivan, DELTA-LD: A Change Detection Approach for Linked Datasets, in: *4th Workshop on Managing the Evolution and Preservation of the Data Web (MEPDaW)*, 2018.

[3] N. Popitsch and B. Haslhofer, DSNotify: A solution for event detection and link maintenance in dynamic datasets, *Web Semantics: Science, Services and Agents on the WWW* **9**(3) (2011), 266–283.

[4] Y. Roussakis, I. Chrysakis, K. Stefanidis, G. Flouris and Y. Stavrakas, *14th International Semantic Web Conference*, Springer, 2015, pp. 495–512, Chapter A Flexible Framework for Understanding the Dynamics of Evolving RDF Datasets.

[5] M. Pourzaferani and M.A. Nematbakhsh, Repairing Broken RDF Links in the Web of Data, *J. Web Eng. Tech.* **8**(4) (2013), 395–411.

[6] F. Liu and X. Li, Using metadata to maintain link integrity for linked data, in: *Internet of Things (iThings/CPSCom), 2011 International Conference on and 4th International Conference on Cyber, Physical and Social Computing*, IEEE, 2011, pp. 432–437.

[7] A.G. Regino, J.C. dos Reis, J. Matsoui, R. Bonacin, A. Morshed and T. Sellis, Understanding Link Changes in LOD via the Evolution of Life Science Datasets, in: *Semantic Web solutions for large-scale biomedical data analytics (SeWeBMeDA-2019) at the 18th International Semantic Web Conference (ISWC'19), New Zeland*, 2019.

[8] A.G. Regino and J.C. dos Reis, Discovering Semantically Broken Links in LOD Datasets, in: *Workshop Managing the Evolution and Preservation of the Data Web (MEPDaW'20) co-located at the 19th International Semantic Web Conference (ISWC'20), virtual conference. (accepted for publication)*, 2020.

[9] A.G. Regino, J.C. dos Reis and R. Bonacin, LODMF: A Linked Open Data Maintenance Framework, in: *Semantic Technologies for Smart Information Sharing and Web Collaboration (Web2Touch) at the 30th IEEE International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprise (WETICE'20) (accepted for publication).*, 2020.

[10] A.G. Regino, J.C. dos Reis, R. Bonacin, A. Morshed and T. Sellis, Link maintenance for integrity in linked open data evolution: Literature survey and open challenges, *Semantic Web Journal* (2020).

[11] G. Capobianco, D. Cavaliere and S. Senatore, OntoDrift: a Semantic Drift Gauge for Ontology Evolution Monitoring, in: *Workshop Managing the Evolution and Preservation of the Data Web (MEPDaW'20) co-located at the 19th International Semantic Web Conference (ISWC'20), virtual conference. (accepted for publication)*, 2020.

[12] B. Haslhofer and N. Popitsch, DSNotify-detecting and fixing broken links in linked data sets, in: *20th International Workshop on Database and Expert Systems Application*, IEEE, 2009, pp. 89–93.

[13] J.C.D. Reis, C. Pruski and C. Reynaud-Delaître, State-of-the-art on mapping maintenance and challenges towards a fully automatic approach, *Expert Systems with Applications* **42**(3) (2015), 1465–1478. doi:https://doi.org/10.1016/j.eswa.2014.08.047.

[14] A.N. Ngomo, M.A. Sherif and K. Lyko, Unsupervised Link Discovery through Knowledge Base Repair, in: *The Semantic Web: Trends and Challenges - 11th International Conference, ESWC 2014, Anissaras, Crete, Greece, May 25-29, 2014. Proceedings*, V. Presutti, C. d'Amato, F. Gandon, M. d'Aquin, S. Staab and A. Tordai, eds, Lecture Notes in Computer Science, Vol. 8465, Springer, 2014, pp. 380–394. doi:10.1007/978-3-319-07443-6_26.

[15] P. Cudré-Mauroux, P. Haghani, M. Jost, K. Aberer and H. De Meer, idMesh: graph-based disambiguation of linked data, in: *Proceedings of the 18th international conference on World wide web*, 2009, pp. 591–600.

[16] J. Euzenat, P. Shvaiko et al., *Ontology matching*, Vol. 18, Springer, 2007.

[17] A. Groß, J.C.D. Reis, M. Hartung, C. Pruski and E. Rahm, Semi-Automatic Adaptation of Mappings between Life Science Ontologies, in: *Proceedings The 9th International Conference on Data Integration in the Life Sciences*, 2013, pp. 90–104.

[18] M. Atencia, J. David and J. Euzenat, Data interlinking through robust link key extraction, in: *In the proceedingsof the 21st European Conference on Artificial Intelligence - Including Prestigious Applications of Intelligent Systems (PAIS 2014)*, 2014, pp. 15–20.

[19] J.C. Dos Reis, C. Pruski, M. Da Silveira and C. Reynaud-Delaître, Understanding Semantic Mapping Evolution by Observing Changes in Biomedical Ontologies, *Journal of Biomedical Informatics* **47** (2014), 71–82.

[20] J.C. Dos Reis, C. Pruski, M. Da Silveira and C. Reynaud-Delaître, DyKOSMap: A framework for mapping adaptation between biomedical knowledge organization systems, *Journal of Biomedical Informatics* **55** (2015), 153–173.

[21] G. Papastefanatos and Y. Stavrakas, Diachronic linked data: Capturing the Evolution of Structured Interrelated Information on the Web, *Ercim news* **52** (2014), 35–37.

[22] F. Liu and X. Li, Using Metadata to Maintain Link Integrity for Linked Data, in: *4th Int. Conference on Cyber, Physical and Social Computing*, 2011, pp. 432–437.

[23] T. Käfer, A. Abdelrahman, J. Umbrich, P. O'Byrne and A. Hogan, *10th Extended Semantic Web Conference*, Springer, 2013, pp. 213–227, Chapter Observing Linked Data Dynamics.

[24] T.R. Gruber, Toward principles for the design of ontologies used for knowledge sharing, *International Journal of Human-Computer Studies* **43** (1995), 907–928.

[25] J. Euzenat and P. Shvaiko, *Ontology matching*, 2nd edn, Springer-Verlag, Heidelberg, 2013.

[26] D. Spohr, L. Hollink and P. Cimiano, A machine learning approach to multilingual and cross-lingual ontology matching, in: *The Semantic Web–ISWC 2011*, Springer, 2011, pp. 665–680.

[27] B. Fu, R. Brennan and D. O 'sullivan, Cross-lingual Ontology Mapping - An Investigation of the Impact of Machine Translation, in: *The Semantic Web, ASWC 2009*, Springer, 2009, pp. 1–15.

[28] T. Charen, The Etymology of Medicine, *Bulletin of the Medical Library Association* **39** (1951), 216–221.

[29] M.T.P. José Camacho-Collados and R. Navigli, a Novel Approach to a Semantically-Aware Representation of Items, in: *North American Chapter of the Association of Computational Linguistics (NAACL 2015)*, Denver,USA, 2015, pp. 567–577.

[30] M.T.P. José Camacho-Collados and R. Navigli, A Unified Multilingual Semantic Representation of Concepts, in: *53rd Annual Meeting of the Association for Computational Linguistics (ACL 2015)*, Beijing, China, 2015, pp. 741–751.

[31] R. Navigli and S.P. Ponzetto, BabelNet: The Automatic Construction, Evaluation and Application of a Wide-Coverage Multilingual Semantic Network, *Artificial Intelligence* **193** (2012), 217–250.

[32] V.I. Levenshtein, Binary codes capable of correcting deletions, insertions, and reversals, *Soviet Physics Doklady* **10** (1966), 707–710–.

[33] N.F. Noy et al., BioPortal: ontologies and integrated data resources at the click of a mouse, *Nucleic acids research* **37**(suppl 2) (2009), 170–173–.

[34] P. Shvaiko and J. Euzenat, Ontology Matching: State of the Art and Future Challenges, *Knowledge and Data Engineering, IEEE Transactions on* **25**(1) (2013), 158–176. doi:10.1109/TKDE.2011.253.

[35] M. Dragoni, Multilingual Ontology Mapping in Practice: A Support System for Domain Experts, in: *The Semantic Web - ISWC 2015*, Beijing, China, 2015, pp. 169–185.

[36] C. Meilicke, C. Trojahn, O. Šváb-Zamazal and D. Ritze, Multilingual Ontology Matching Evaluation–A First Report on Using MultiFarm, in: *The Semantic Web: ESWC 2012 Satellite Events*, Springer Berlin Heidelberg, 2012, pp. 132–147.

[37] M. Da Silveira, J.C. Dos Reis and C. Pruski, Management of Dynamic Biomedical Terminologies: Current Status and Future Challenges., *Yearbook of medical informatics* **10**(1) (2015), 125–133. doi:10.15265/IY-2015-002.

[38] J. Jung, A. Håkansson and R. Hartung, Indirect Alignment between Multilingual Ontologies: A Case Study of Korean and Swedish Ontologies, in: *Agent and Multi-Agent Systems: Technologies and Applications*, Vol. 5559, Springer, 2009, pp. 233–241.

[39] S. Faisal, K.M. Endris, S. Shekarpour, S. Auer and M.-E. Vidal, Co-evolution of RDF Datasets, in: *International Conference on Web Engineering*, Springer, 2016, pp. 225–243.

[40] T. Galani, G. Papastefanatos and Y. Stavrakas, A Language for Defining and Detecting Interrelated Complex Changes on RDF (S) Knowledge Bases., in: *ICEIS (1)*, 2016, pp. 472–481.

[41] N. Pernelle, F. Saïs, D. Mercier and S. Thuraisamy, RDF data evolution: efficient detection and semantic representation of changes, *Semantic Systems-SEMANTiCS2016* (2016), 4–pages.
[42] R. Vesse, W. Hall and L. Carr, All about that-a uri profiling tool for monitoring and preserving linked data (2009).
[43] J. Volz, C. Bizer, M. Gaedke and G. Kobilarov, Discovering and Maintaining Links on the Web of Data, in: *International Semantic Web Conference (ISWC)*, Springer, 2009, pp. 650–665.
[44] R. Vesse, W. Hall and L. Carr, Preserving linked data on the semantic web by the application of link integrity techniques from hypermedia, 2010.
[45] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak and Z. Ives, DBpedia: A Nucleous for a Web of Open Data, 2009. https://cis.upenn.edu/~zives/research/dbpedia.pdf.
[46] F. Scharffe and J. Euzenat, MeLinDa: an interlinking framework for the web of data, *CoRR* **abs/1107.4502** (2011).
[47] Y. Roussakis, I. Chrysakis, K. Stefanidis, G. Flouris and Y. Stavrakas, A flexible framework for understanding the dynamics of evolving RDF datasets, in: *International Semantic Web Conference*, Springer, 2015, pp. 495–512.