Linking and disambiguating entities across heterogeneous RDF graphs

Manel Achichi, Zohra Bellahsene, Mohamed Ben Ellefi, Konstantin Todorov

Linking and Disambiguating Entities Across Heterogeneous RDF Graphs

Manel Achichi[a], Zohra Bellahsene[a], Mohamed Ben Ellefi[b], Konstantin Todorov[a,*]

[a] *LIRMM, University of Montpellier, CNRS*
161 rue Ada, 34000 Montpellier, France
[b] *LIS, Aix-Marseille University,*
163 Avenue de Luminy, 13288 Marseille, France

Abstract

Establishing identity links across RDF datasets is a central and challenging task on the way to realising the Data Web project. It is well-known that data supplied by different sources can be highly heterogeneous—two entities referring to the same real world object are often described, structured and valued differently, or in a complementary fashion. In this paper, we explore the origins and the multiplicity of *data heterogeneity* problems, proposing a novel classification that allows to isolate challenges and to position our and future work. Many state-of-the-art data linking approaches rely on sets of discriminative properties, provided by the user or by specialised tools, which, in the lack of knowledge of the nature of the data, do not allow to account automatically for a large number of structural heterogeneities. In addition, similarity measures and thresholds need to be selected and tuned manually or learned by specialized algorithms. We propose a solution covering an important number of heterogeneities, attempting to reduce the user configuration effort, based on: (i) *Property filtering*, or automatic data cleaning of "problematic" attributes; (ii) *Instance profiling* allowing to represent each resource by a sub-graph considered relevant for the comparison task; and (iii) *Instance vector representation* allowing to compare resources. To reduce the false positives rate, we apply a (iv) *Post-processing* step based on hierarchical clustering and key ranking techniques aiming to disambiguate highly similar, though not identical instances. This pipeline is implemented in *Legato*—a data linking tool, showing to outperform or to perform as well as state-of-the-art tools on highly heterogeneous and diverse benchmark datasets, yet keeping the user configuration effort low.

Keywords: RDF Data Linking, Knowledge Graphs, Linked Open Data, Data Heterogeneities

1. Introduction

Linked data and its underlying technologies have been gaining popularity over the past years, due to the means they offer for data reuse and federation, increased visibility and data sharing on the web and facilitated exchange of metadata. The web of data, and particularly the Linked Open Data (LOD) project,[1] has been growing in size over the past years, with hundreds of datasets published following the semantic web principles. To fully unlock the potential of the open data project, related resources across datasets need to be linked together—a process that cannot be handled manually at the web scale. Data linking is the semantic web research field that has taken the challenge of proposing methods and providing tools for the automatic detection of relations between cross-dataset resources. A plethora of data linking approaches and systems has been proposed in the past years, surveyed in [1, 2, 3]. The majority of these approaches attempt to solve the problem of discovering identity relations (often of arity 1:1), declared as `owl:sameAs` statements, between similarly-typed resources of two RDF datasets.

Preparing the datasets prior to linking and configuring the linking tools are challenging problems that often require in-depth knowledge of the data. Several state-of-the-art tools [4, 5]

[*] Correspodning author
Email address: `todorov@lirmm.fr` (Konstantin Todorov)

[1] `http://linkeddata.org`

require link specification files where one has to indicate property names, select and tune similarity measures. This process is handled either manually, or by specialised tools [6, 7]. Making a linking tool self-dependent in that respect is among the challenging issues that the community faces.

On the instance matching level, a data linking system has to be able to deal with a large variety of data heterogeneities, taking into account mismatches in the descriptions on value, ontological and logical level, as well as differences in the quality of the input datasets. While heterogeneities on literals are rather well-handled by similarity measures and data unification techniques, ontological discrepancies (regarding structure and properties) appear to be more challenging.

Finally, as we show in our experiments, most current approaches fail to handle correctly datasets containing blocks of highly similar in their descriptions, but yet distinct resources. Datasets with such characteristics are prone to the generation of false positives in the linking process (for example, two datasets containing all piano sonatas of Beethoven, where two works differ very little in their descriptions).

In this paper, we attempt to address the challenges given above. We aim at reducing the difficulty of manual configuration when it comes to data-related parameters, such as properties to compare. With respect to similarity measures or threshold settings, we rely on an empirical approach which shows satisfactory results on our tests, although we make no assumptions on its generalisation properties. We propose a system, called *Legato*, which, contrarily to property-based instance matchers, applies indexing techniques that allow to project each instance in a vector space defined by an appropriately chosen set of literals that describe that instance, derived from the Concise Bounded Descriptions (CBD) of the resources.[2] In addition to avoiding the selection of properties, this representation addresses in its mechanism a number of data heterogeneities without requiring user input. An automatic property filtering module allows to decrease noise prior to instance matching. We pay particular attention to discriminating highly similar, yet distinct resources, implementing an unsupervised learning post-processing strategy combined with a key selection and ranking algorithm [8] that reduces the number of false positives and increases precision.

Legato has been conceived in the framework of the DOREMUS project,[3] which develops methods to describe, publish, connect and contextualize music catalogs from major cultural institutions[4] on the web of data [9]. The data collected and handled in this project has served as a main motivation for the development of this system, which aimed to respond to the difficulties of linking these highly heterogeneous datasets, while remaining as generic as possible. We evaluate *Legato* on benchmarks from the Ontology Alignment Evaluation Initiative (OAEI)[5] instance matching tracks form 2015, 2016 and 2017. Note that two of the benchmarks of these campaigns released in 2016 and 2017 are real-world music-metadata datasets issued from the DOREMUS project. *Legato* has participated to the 2017 edition of OAEI. The experimental results show that our system performs as good as the state-of-the-art link discovery tools and it outperforms them particularly on heterogeneous real-world data and in the presence of difficult to disambiguate cross-graph instances. In addition, we show that *Legato* achieves better results in terms of F-measure than established matching tools that implement an automatic link specification learning strategy. Among the drawbacks of the system, we underline certain scaling issues encountered on several datasets. *Legato* is an open source, freely available system.[6] For the sake of reproducibility of our experiments, all datasets and configuration files (where applicable) used in the evaluation are made available (links and references are provided in the evaluation section).

To wrap up, the contributions of this paper are as follows:

- A classification of the different *data heterogeneity* types based on a large number of examples from real-world and synthetic datasets.

[2] `https://www.w3.org/Submission/CBD/`

[3] `http://www.doremus.org/`

[4] The French National Library, Philharmonie de Paris and Radio France

[5] `http://oaei.ontologymatching.org`

[6] `https://github.com/DOREMUS-ANR/legato`

- A new CBD-based instance profiling framework allowing to represent and compare resources at the matching phase.
- A novel preprocessing strategy aiming at the automatic identification and removal of “problematic” properties across two datasets.
- A new post-processing mechanism to select and repair erroneous links generated in the matching step.
- A multi-facet reproducible empirical evaluation on a large variety of openly available benchmarks.
- An open source implementation of our system with a simple user interface.

The rest of the paper is structured as follows. In Section 2, we present our account on RDF web data heterogeneity types, while in Section 3, we focus on challenges related to the problem of reducing the user effort in the data linking tool configuration process. These two sections are intended to be read as an account of different challenging issues that allow to structure the related work proposed to deal with these challenges and to identify open issues and problems. On these bases, Section 4 introduces the approach and workflow of *Legato*, which is discussed and positioned with respect to the related work in Section 5. We report on our experiments in Section 6 before we conclude and draw further directions of work and discuss lessons learned in Section 7.

2. RDF Datasets Heterogeneity Types

Understanding data heterogeneity in its multiple forms allows to identify and analyse the origins of the data linking problem and hence propose better solutions. In the context of web data linking, we will refer to data heterogeneity as any difference in the expression of a given piece of information across two graphs, observed in terms of schema (classes, properties), values, or general data structure. Ferrara et al. [10] consider three major levels of data heterogeneity (or requirements, as given in the paper): *value*, *structural* and *logical* levels. We base ourselves on this classification and extend it in an attempt to provide a comprehensive inventory of data heterogeneity types. Our classification emerges as a result of observations and tests on two types of data: (1) highly heterogeneous real-world data about classical music,[7] (2) a number of synthetic benchmark datasets released between 2015 and 2017 by the Instance Matching track of OAEI (IM@OAEI). Two different methodologies have been followed in the two cases. In case (1), we have worked tightly with librarian experts and archivists from the BnF, Radio France and the Philharmonie de Paris (partners on the DOREMUS project). They have identified collaboratively and listed a set of possible heterogeneities, given their expert knowledge of the data.[8] (As a matter of fact, this work is the basis of the creation of the DOREMUS benchmarks at OAEI 2016 and 2017, as discussed below.) Regarding (2), we have considered the respective benchmark generation strategies (e.g., altering string values or value types) as a basis and completed with these the list of heterogeneities identified in (1). Finally, the resulting list has been expanded by additional cases that we have observed through our work with the data. In order to form the taxonomical skeleton of our classification, we have used and extended the three axes identified by [10].

The resulting classification reflects the authors consensus and does not claim universality. For illustration purposes, we will use a fictional example, given in Figure 1, showing the descriptions of a real-world entity (the composer Ludwig van Beethoven) in two different graphs. For readability reasons, the example outlines only several of the heterogeneity types given below.

2.1. Value Dimension

Datatype properties are an ample source of heterogeneities, both when it comes to string or numerical attributes.

Terminological heterogeneity. We refer to differences between the lexical labels used to denote the same information across graphs. This comprises well-known issues related to

[7] `https://github.com/DOREMUS-ANR/knowledge-base/tree/master/data`

[8] Throughout the working process, these heterogeneities have been outlined together with concrete examples in a table that can be found here (in French): `https://docs.google.com/spreadsheets/d/19dLjabt_fggTVNuM7XW9CUZkuB1JgoH9xKWQgLZv_Y4/edit#gid=1271677916`

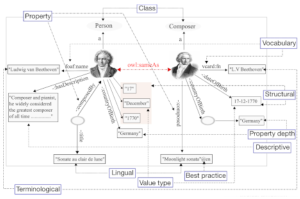

Figure 1: Open web data heterogeneity types: several examples (in blue-lined boxes).

synonymy, polysemy or variations in spelling (typos, acronyms, abbreviations, etc.). The problems of synonymy and polysemy have been largely addressed in the literature by term disambiguation techniques assisted by lexico-semantic resources [11]. A long tradition of research in the field of string similarity measures, at the core of the instance comparison modules of state-of-the-art systems [4, 12, 13], has allowed to handle the case of orthographical variations among labels [14, 15]. Several works propose solutions allowing to find the full form of an acronym or an abbreviation [16, 17].

Lingual heterogeneity. Multilingualism has been outlined as a major challenge to the open data community, already several years ago [18]. Looking at the datasets about music openly published as RDF graphs by the BNF (French National Library) [19] and Freebase [20] in French and English, respectively, one notices that very few string literals are directly comparable across these graphs. Several studies propose solutions based on machine translation [21, 22] or, alternatively, relying on a lexico-semantic resource such as BabelNet [11] as a mediator to bridge the language gap, as proposed in [23] (in a similar spirit as an earlier ontology matching method [24]). The latter approach anchors the resources as vectors of BabelNet identifiers where each of them represents a sense of a term allowing to compute vector distances as a proxy for instance similarity. Combining machine translation with concept embeddings, [25] translates each resource description to English and then a Wikipedia-based representation (a set of concepts) is generated for the resources in order to compare them.

Datatype properties vs. object properties. A piece of information (e.g., the genre of a music work) can be given by a string literal via a datatype property, or by a URI, that is used to identify the same element in a controlled vocabulary (e.g., a SKOS vocabulary of music genres[9]). Since the two objects are not directly comparable, a linking tool needs to access the string label associated to the URI in the respective vocabulary before proceeding to the comparison. In the same line of thought, comparing object property values (different URI's identifying the same object) can potentially lead to a similar type of difficulty.

2.2. *Ontological Dimension*

We discuss schema-related differences across RDF graphs.

Vocabulary heterogeneity. A recurrent discrepancy across data providers is the use of different ontologies. This is a challenging problem in the context of the open web of data, where we have an abundance of models and vocabularies/ontologies with different degrees of explicit rigour of their semantics, leading to different interpretations and usages.[10] Ontology matching techniques have been adopted by certain data linking systems [26], but can also be applied independently.

Structural heterogeneity. The description of an entity can be done at different levels of granularity, as is the case of the birth date of Beethoven in Fig. 1, given in a single information field or distributed over multiple properties—a challenge for property-based instance matchers. To the best of our knowledge, this heterogeneity type is only partially resolved by using inverted indexes and Natural Language Processing (NLP) techniques by several data linking approaches [12, 22, 23, 27, 28]

[9] Examples of such vocabularies: `https://github.com/DOREMUS-ANR/knowledge-base/blob/master/vocabularies/`

[10] The Linked Open Vocabularies catalog `http://lov.okfn.org/dataset/lov/` (LOV) allows to browse over 600 vocabularies from the linked open data cloud, but, although very useful, it is to date not exhaustive (certain established graphs such as Yago, Freebase or MusicBrainz are not indexed).

that propose to index each resource by the literals collected at a given distance $n \geqslant 1$ in the RDF graph.[11] Then, a vector space model is used to represent each resource description and select linking candidates on the basis of vectors proximity. By doing so, the resources are compared with respect to their literals without taking into account the properties describing them. The quality of the alignment depends strongly on the distance parameter.

Property depth heterogeneity. The same piece of information can be found at different distances to the resource in two different graphs. In our example, we can observe this problem with regard to the name of the country of birth of Beethoven. This problem can also be solved by indexing the scope of literals describing each resource. For each entity, the distance at which the literals are collected can be fixed (e.g., [29] choose a distance of 1). Again, the trade-off while setting this parameter is between too small a distance not allowing for a complete description and a too large distance, increasing the likelihood to collect irrelevant information.

Descriptive heterogeneity. A resource can be described with more information (a larger set of properties and types) in one dataset than in another, as we can see in our running example. The lack of information narrows down the intersection of the resources respective descriptions and hence the common ground where to look for commonalities or differences.

Key heterogeneity. Key identification algorithms [6, 30, 31] aim to discover discriminative properties on two datasets independently and thus identify potential candidates for link specifications of property-based state-of-the-art tools [4, 12]. However, in certain cases, comparing the values of such properties will lead to deciding negatively on the equality of two identical instances and to the generation of false negatives. We take two examples: (1) a property that is valued by unstructured textual information (e.g., a free-text description as the one given by the *-:hasDescription* property in our example) and (2) a property used to provide dataset-specific individual identifiers, e.g., the ID's of bibliographical entries of two libraries. In both cases the values of these key properties are not comparable across datasets.

2.3. Logical Dimension

In a number of cases, the equivalence between two pieces of information across two datasets is implicit but can be inferred. We outline two main heterogeneity problems in that group.

Class heterogeneity. This is the case of two resources belonging to different classes for which an explicit or an implicit hierarchical relationship is defined ("*Person*" and "*Composer*" in Fig. 1). Moreover, two instances referring to the same object can belong to two different subclasses of a given class.

Property heterogeneity. At this level, the equivalence between two values is deduced after performing a reasoning task (cf. Fig. 1): $<< i2 >, - : composed, ``Moonlight\ sonata'' @en >$, $<< i4 >, - : composedBy, < i1 >>, << i4 >, - : title, ``Sonate\ au\ clair\ de\ lune'' >$. The comparison process has to go beyond the value and property levels by comparing explicitly and implicitly specified values of the two entities.

2.4. Data Quality Dimension

Quality related issues can be observed on any of the three levels discussed above and can appear as a source of heterogeneity, therefore, we consider these aspects as a separate (transversal) category. The topic of data quality has been of interest for many years to the semantic web community [32, 33]. We will provide several examples of heterogeneities related to data quality that can potentially hinder the instance matching task.

Transgression to best practices. Data representations can differ depending on the degree to which the semantic web best practices are respected in the data publishing process. The list of transgressions is long: a missing language tag, the introduction of inappropriate symbols such as '♯' that are supposed to replace missing information (while a good practice would be to ignore what we do not know), the use of a string literal instead of a URI to identify an object, and so forth.

Value type heterogeneity. This heterogeneity type concerns differences in encoding data, as for example, representing an

[11] A distance in an RDF graph is defined as the minimal number of edges (properties) connecting two resources or a resource and a literal.

age-value as a string or as a number, or not representing the date in a standard date format, but as a string. Multiple data unification techniques can be applied to solve this problem. The benchmark data generators SPIMBENCH [34] and LANCE [35] focus on these issues by applying value transformations.

Dataset currentness. The temporal evolution (or the lack thereof) of data and its dynamicity [33] can lead to conceptual issues across datasets. For example similar or identical classes in terms of semantics can be applicable to a given group of instances only during defined periods of time (e.g., "Orchestra-Conductor").

In Section 5, we discuss the positioning of our approach with regard to the heterogeneities and the methods for their resolution presented here.

3. Doing Linked Open Data with Less User Effort

The data linking process commonly follows a pipeline consisting of three main steps [1]: (1) preprocessing, where data is prepared for linking and a number of system parameters are set, (2) matching, where instances are compared by the help of an aggregation of similarity measures and (3) post-processing, where erroneous links are removed and / or new links are inferred. For extensive surveys of data linking approaches, we refer the reader to [1, 2, 3]. Here, we focus in more detail on the phase that takes place before the actual instance comparison. We argue that the preparation of data and the configuration of the linking tool constitute a major part of the effort with regard to the linking task. Moreover, this effort is often required from the user, leading to a pressing need of automation of this process. Therefore, we pay particular attention to approaches that propose (semi-)automatic solutions to the preprocessing and configuration tasks.

Several of the most commonly used linking tools [4, 5, 36] require prior knowledge provided by either the user or another tool in order to proceed to the linking task. This knowledge is expressed in the form of *linking rules*, describing under which conditions two instances should be compared and linked. There are two main configuration groups of elements to feed to the linking tool: (1) types (classes) of instances to align as well as a set of properties across the two datasets whose values to compare, and (2) a set of similarity measures, together with thresholds and possibly an aggregation function. We discuss these two groups in the following subsections.

3.1. Selecting Classes and Properties

The choice of types of instances that defines the pool of linking candidates is often left to the user (considering a dataset as a set of resources belonging all to the same class), although certain systems attempt to identify the equivalent classes automatically by applying ontology matching techniques [26, 36].

The properties to compare are selected manually or by the help of key discovery tools—this choice is crucial for it predetermines the outcome of the linking task. Intuitively, instances having common values for highly discriminant sets of properties (keys) are likely to be representing the same real-world objects. While many approaches to automatic key discovery from RDF data exist [6, 30, 37, 38, 39, 40, 41], their use for data linking is not always straightforward. Most of these tools produce large numbers of keys valid on a single dataset with no assessment given of their likelihood to discover links. For example, a property containing a record's identifier in a bibliographical database will be identified as a key in two datasets containing the entries of musical works of two libraries independently on one another, but it will be of no use for the linking task, since the two libraries use different identifiers for the same work. An exception is [30], which considers keys valid on two datasets. In addition, key discovery systems do not consider the heterogeneity of the properties used to describe instances across datasets, which compromises the usefulness of the keys for the linking task. In an attempt to overcome this issue, the authors of [37] present measures of the quality of link keys, valid on two datasets, in order to facilitate their selection. Two recent studies [8, 42] propose approaches that attempt to close the gap between key discovery and data linking tools, allowing to produce a list of keys, valid on two datasets simultaneously and ranked with respect to their usefulness for the particular data linking task at hand.

3.2. Learning Link Specifications

Link specification is defined in [43] as (i) the setting of the elements to compare from two knowledge bases, (ii) the setting of a complex similarity metric via the combination of several atomic similarity measures, and (iii) the setting of thresholds for these similarity measures. The (semi-)automatic link specification approaches of which we know have focused predominantly on (ii) and (iii)—configuration parameters of type (2) that can either be set by the user or learned from data in a semi–supervised or unsupervised manner. Two main categories of *semi-supervised* learning methods emerge: *active* [44, 43, 45] and *batch* [7, 46] approaches. *Batch* approaches require a large amount of candidate links as input to learn the classifiers while *active* approaches proceed iteratively and for each iteration the user is asked to label a set of generated links until the maximal number of iterations is reached or the fitness value is greater than a given threshold. *Unsupervised learning* methods attempt to surpass the necessity of human labeled examples [47]. A method based on a refinement operator that only needs positif examples that are more often available than negative ones is proposed in [48], while [49] propose an approach implemented in KnoFuss [50], based on a genetic programming algorithm learning iteratively the optimal similarity parameters. However, it is required from the user to set the fitness function and to specify the fitness measures, thresholds and the maximum number of iterations. Certain releases of the well-known data linking tool LIMES [4] include both EAGLE [44] and WOMBAT [48] as link specification algorithms,[12] while SILK [5] includes ActiveGenLink [7].

We discuss how our approach positions with respect to end-user configuration effort reduction in Section 5, right after presenting it.

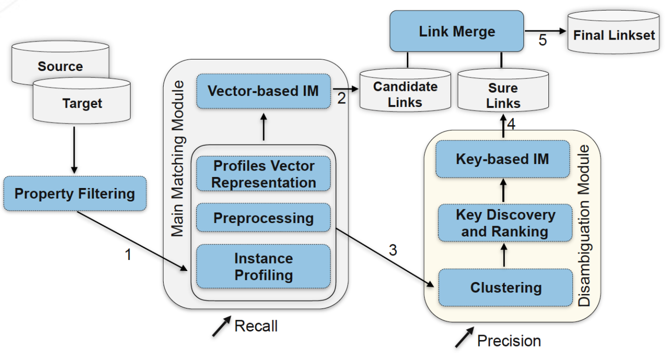

Figure 2: Processing pipeline of *Legato*.

4. Data Linking with *Legato*

We proceed to present the *Legato* framework, illustrated in Figure 2. The system takes as an input two RDF graphs (particularly, instances of the same type). The datasets are automatically preprocessed and prepared for comparison, and then a set of links is generated, as a result of an instance matching, instance disambiguation and link selection (or link merging) procedures. Note that in its default release, the system takes one parameter as input: a pair of types (classes) of instances and optionally a global similarity threshold value. However, for the data-aware user, a customisation of *Legato* is possible with regard to two additional parameters, giving rise to two version of the tool - an automatic and a manual one (see below for details). The approach and the results presented in this paper concern the automatic default release of *Legato*.

Before we proceed, we introduce definitions and notation. In the context of this work, an RDF graph or an RDF dataset[13] is defined in accordance to the dataset definition in the Vocabulary of Interlinked Datasets (VoID):[14] “A dataset is a set of RDF triples that are published, maintained or aggregated by a single provider”. An RDF triple is a set of elements $< s, p, o >$, where s is a subject (a URI or a blank node), o is an object (a URI, a literal or a blank node) and p is a predicate (property, relation) (identified by a URI). We use the terms “instance” or

[12] E.g., *limes-core-1.2.1*.

[13] The two terms are used interchangeably. To improve readability, we use the term “dataset” as a shortcut to “RDF dataset” and the term “graph” as a shortcut to “RDF graph”.

[14] http://vocab.deri.ie/void

"resource" to identify an entity of interest described in a graph (e.g., a musical work or a composer).

A key is a central notion to the functioning of our system. We comply to the open world assumption-compatible definition of a key used in [6].

Definition 4.1 (RDF Dataset Key)**.** Let G be an RDF graph, let $subj(G)$ be the set of resources in G and let $pred(G)$ be the set of properties in G. We define a *key* denoted by K as the set $K = \{P : P \subseteq pred(G)$ and $\nexists s_1, s_2 \in subj(G)$ such as $p(s_1) = p(s_2)$ $\forall p \in P\}$, where $p(s_1)$ and $p(s_2)$ are the values of the property p for the resources s_1 and s_2, respectively. A set of properties P is considered as a *minimal key* if it is a key and there is no subset of P, which is a key.

A *CBD*, for Concise Bounded Description, allows to represent a given resource r by a subgraph such that all triples of this subgraph have as a subject r or a blank node connected to r or are reifications of statements of that subgraph. We cite the definition given by w3c.[15]

Definition 4.2 (Concise Bounded Description)**.** The Concise Bounded Description *CBD* of a resource r in an RDF graph G is a subgraph of G denoted by $CBD(r)$ identified as follows:

- Include in $CBD(r)$ all statements in G where the subject of the statement is r;
- Recursively, for all statements identified in $CBD(r)$ thus far having a blank node object, include in $CBD(r)$ all statements in G where the subject of the statement is the blank node in question and which are not already included in $CBD(r)$.
- Recursively, for all statements included in $CBD(r)$ thus far, for all reifications of each statement in the source graph, include the *CBD*(rdf:Statement) of each reification.

Definition 4.3 (Data Linking)**.** Given two graphs G and G' containing two equivalent classes C and C', respectively, the data linking problem consists in discovering all relations of identity across the instances of these classes. The outcome of this task is a set of links declared by `owl:sameAs` statements on a subset of the cartesian product of the elements of C and C'. We refer to G and G' as a source and target dataset, respectively, while the resources of C and C' are referred to as source and target resources, respectively.

Note that the given definition restricts the linking task to identity relations only, which are in the scope of this study. Relations of arbitrary types can be of interest in the general case. In that, Definition 4.3 serves the purposes of this paper which deals with the special data linking problem of deduplication, and therefore provides a particular case of a larger problem.

With these definitions at hand, we proceed to describe the framework of *Legato*.

Property Filtering. As we have seen in Section 2, *key heterogeneities* hinder the resources comparison, mainly because properties concerned with this heterogeneity type are erroneously likely to be considered as linking rules parameters, as discussed in Section 3.1 (certain OAEI datasets from 2016 and 2017 are rich with such examples[16]). If a linking tool uses these keys to compare instances, it will fail to find a correspondence. A way of going around this problem is to remove properties with such values, that we will call *problematic properties*, before proceeding to data comparison. We propose to identify automatically these properties by discovering all mono-property keys that are valid over *both* datasets to be linked (in that we consider the union of the two input datasets as a single dataset), i.e., each object for such a property has at most one subject in *both* graphs.

Note that the property filtering module can be seen as a preprocessing step. We analyse its impact on the global linking quality in our experiments (Section 6).

Main Matching Module. The main matching module consists of the following components.

CBD-based Instance Profiling. A core feature of our ap-

[15] `https://www.w3.org/Submission/CBD/`

[16] `http://islab.di.unimi.it/content/im_oaei/2016`

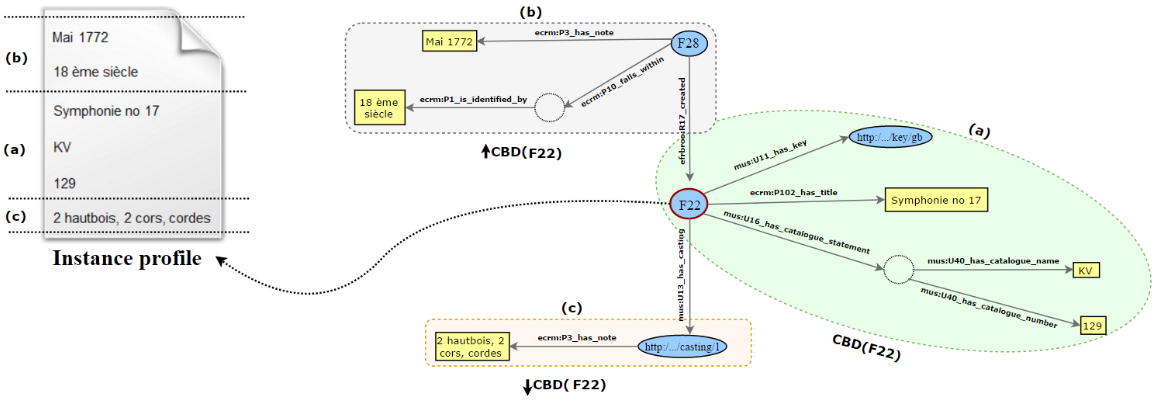

Figure 3: Constructing an instance profile by using the CBD of a resource and the CBDs of its successors and predecessors in the graph.

proach is the representation of instances as text documents and their projection to a vector space. Particularly, each resource is represented by a set of literals considered as relevant to its description, based on a choice of *CBD* subgraphs (Fig. 3). We extend the *CBD* definition by considering the descriptions of neighbouring nodes of a resource r in its graph. A *CBD* subgraph is a directed one, where the orientation is implied by the order of the subject and the object in a triple. This allows us to introduce the notion of successors of a node r (the nodes that are in a triple of which r is a subject) and the predecessors of r (the nodes that are in a triple of which r is an object). We provide the following definitions.

- $\uparrow CBD(r)$ defines the scope of resource description including $CBD(r)$ and the CBDs of its direct *predecessors*.
- $\downarrow CBD(r)$ defines the scope of resource description including $CBD(r)$ and the CBDs of its direct *successors*.
- $\updownarrow CBD(r)$ defines the scope of resource description including $CBD(r)$, $\uparrow CBD(r)$ and $\downarrow CBD(r)$.
- $CBD^*(r)$ defines the scope of resource description including one of the *CBDs* cited above, i.e., $CBD(r)$, $\uparrow CBD(r)$, $\downarrow CBD(r)$ or $\updownarrow CBD(r)$.

We refer to an instance representation obtained on that basis as an *instance profile*, defined as follows.

Definition 4.4 (Instance Profile)**.** Let G be an RDF graph, let r be a resource in G and let $L(G)$ be the set of literals found in G. We define the *instance profile* of r as the set

$$f(r) = \{l_r : l_r \in L(G) \wedge l_r \in CBD^*(r)\}.$$

Fig. 3 provides an illustration for an instance of the class F22 from the DOREMUS ontology[17] (a class of music works, example taken from the DOREMUS OAEI 2017 data). Selecting the most relevant profile depends on the way the resources are modeled in the graph. Without any user intervention, the default setting of *Legato* represents the instances only by literals found in their *CBDs*, which shows to produce good results in terms of F-measure on all benchmarks in our evaluation (Section 6). Note however, that this parameter is modifiable in the open source release of our system. Avoiding property-based comparison addresses the remaining ontology-level heterogeneities introduced in Section 2.

Instance Profiles to Vectors. Once all resources in both datasets are profiled, the resulting documents are processed in order to prepare data for the matching task. This includes tokenization and stop-words removal by applying NLP filters. The set of *instance profiles* in both datasets are indexed in a standard manner by using all remaining terms. We project the instance profiles to a vector space of a dimension limited to the number of these terms and weight them by using their TF-IDF (Term Frequency-Inverse Document Frequency) scores per instance.

Vector-based Instance Matching. The correlation between the vectors of the resources, expressed by the cosine similarity

[17] www.data.doremus.org

measure, is used as a proxy for the similarity of resources. The use of a similarity measure is tangled to the choice of a similarity threshold. In this step, it is empirically fixed to 0.2 (deliberately low) in order to capture a large number of links and ensure high recall. In order to ensure $1:1$ type of matching, for each instance from the source dataset, the one from the target dataset that has the highest similarity score greater than the threshold is selected. In case of ties, the instances are handled by the disambiguation module described below. As an outcome of this process, a first linkset (a short for "set of links") is produced, called *candidate links* (Fig. 2).

The main matching module ensures high recall. To improve the matching quality and precision, we perform a post-processing step, described next, allowing to filter out erroneous links that may have been generated at this step and add new quality links.

Instance Disambiguation Module. Taking as an input the vector space representations of the indexed instance profiles, the algorithm proceeds to cluster *within each data set* highly similar (in terms of their vector space similarity) instances by relying on the generic agglomerative bottom-up *hierarchical clustering algorithm* [51]. This results in the formation of a number of clusters of instances within each dataset. A cluster matching procedure across the two datasets, using a distance metric on the cluster centroids, allows to isolate pairs of corresponding clusters, where the first one belongs to the source dataset and the second one—to the target dataset. Each pair of corresponding clusters is then analyzed separately and their respective instances are compared, this time on a property basis. The effectiveness of the comparison process depends on the quality of the selected properties. The RANKey algorithm [8] is developed to discover keys that are valid on *two* datasets, ranked with respect to the performance achieved by a linking system using these keys in its configuration. This allows to select the set of properties over two graphs that guarantee the best linking result. We apply that algorithm independently on each pair of corresponding clusters, considering them as a source and target datasets to be linked. In that, we identify the discriminant properties among these clusters that would have remained "diluted" in the global graphs. This allows to disambiguate the highly similar instances in each pair of clusters and maximises the rate of correct alignments. This component of *Legato* is illustrated in Fig. 4. As a result of this process, we end up with a second set of links, that we call *sure links* (the choice of this name is motivated by the fact that the instance clustering and property-based link discovery ensures high precision and high quality of these links).

Note that this component of *Legato* shares certain similarities with the well-known blocking techniques used in ontology matching, although it differs in its mechanism, motivation and application. Blocking aims at isolating disjoint sets of potential matching candidates based on certain property values so as to ensure that comparison is performed only among comparable entities and thus reduce the search space and computational effort. As an example, works by the same composer would form a block. Our instance disambiguation module has a different motivation: we aim at creating clusters of instances that, rather than being similar with respect to a small set of property values, are different with respect to only very few property values. An example would be all piano sonatas by Beethoven that would only differ by their music keys (e.g., G minor vs. A major, although same composer, genre, title, instrument, etc.). The motivation comes from observations on real-life data containing many blocks of highly similar entities. The comparison of instances belonging to such clusters allows to determine the discriminating property (the music key in our example), which would have been difficult to determine by taking the entire dataset or a block (of all works by Beethoven).

Link Merge. Finally, a merge operation is performed on the two linksets generated previously. The set of *sure links* will be taken as a catalyser on the links in the *candidate links* set and directly fed to the final linkset, because of the high precision in the process of generation of the links that it contains. For each link between two resources r_s and r_t l=(r_s, r_t) in the set of *candidate links*, the module searches over the set of *sure links* for a link between a source resource r_s and a target resource $r'_t \neq r_t$.

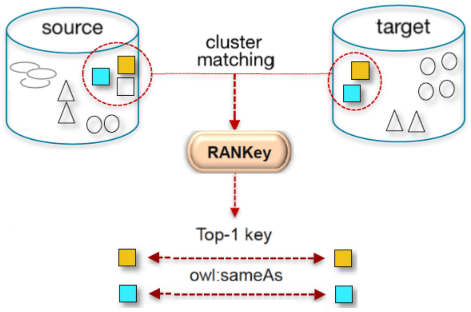

Figure 4: Instance disambiguation by clustering, cluster matching and key ranking techniques.

If found, the link l=(r_s, r_t) is deleted from the *candidate links* set. The remaining links in the *candidate links* set, merged with those from the *sure links* set, are then fed to the *final linkset*.

The disambiguation module followed by the link merge procedure can be seen as post-processing steps. In our experiments, we analyse their impact on the global linking quality.

5. Discussion and Positioning

After having introduced our approach, in this section we discuss how it stands with respect to state of the art methods and issues presented in Sections 2 and 3.

5.1. Positioning with Respect to Dataset Heterogeneities

Most of the heterogeneities on *value level*, as well as certain quality-related heterogeneities (such as the value-type heterogeneities) are relatively well-understood and assisted by various similarity measures, external resources and data unification methods [15]. A large number of instance matching tools of reference rely on string similarity measures and their combinations, coupled with thresholds that condition the matching decision [4, 12, 13]. *Legato* is only partially an exception to this tradition—we do consider instance similarities, but similarity between strings is not explicitly computed. Instead, we represent instances as bags of words, which allows their information retrieval kind of indexing and a projection onto a vector space, in a similar spirit as [22, 28]. We rely more strongly on the data presentation, or instance profiling (taking place at the preprocessing step) and matching selection (post-processing) than on the actual instance similarity computation.

While *logical heterogeneities* are in the realm of reasoning—a field of research in its own right—*Legato* focuses on issues from the *ontological level* that appear to be more challenging for the majority of the linking tools. The representation of instances that we adopt allows to avoid a number of structure- and property-related heterogeneities found on ontological level. Particularly, the use of CBD vs. a fixed distance n to the resource, as proposed for example in [29], tackles heterogeneities on ontological level (excluding key heterogeneity discussed below), allowing to ensure that literal values of relevance will be included in the description of an instance. Note that [49] defines and compares attribute sets across graphs that are equivalent to CBDs. In contrast, we also look into the CBDs of the neighbouring nodes of the resources of interest, in order to ensure that information found at a greater depth is taken into consideration.

The presence of unstructured information, such as properties containing textual descriptions, although common in real-world examples is, to our knowledge, not directly handled by linking systems. In the same line of thought, other types of single-property keys, such as instance identifiers specific to a single data provider, present a hindrance for property-based linking systems. *Legato* adresses these issues by the help of a property filtering method, allowing to automatically identify and remove from the process “problematic” attributes. This issue pertains to the key heterogeneity category outlined above.

Finally, the paradigm adopted by state-of-the-art systems traditionally stands on the plausible premise that heterogeneities are a major hindrance to the instance comparison task. However, the problem of disambiguating between very similar descriptions of yet different resources has received less attention. An example can be two different music works by the same composer, written in the same genre and key for the same instrument. Comparing their property values is likely to lead to the generation of false positives. The identification of the discriminative properties for such groups of instances can be a difficult task for automatic key-discovery methods, because these groups of instances are found in larger datasets where these properties are not keys. Therefore, in addition to data heterogeneity, *Legato* pays attention to the problem of data similarity, in an attempt to disambiguate cross-datasets entities by introducing an unsupervised learning post-processing module. It aims to identify and isolate clusters of highly similar instances across the two datasets in order to enable efficient key identification on these clusters.

5.2. Level of Automaticity

We position our approach with respect to user configuration effort reduction methods, reviewed in Section 3.

With regard to property selection, in contrast to state-of-the-art tools [4, 5], our system takes a holistic stance by representing an instance as the set of literals collected from a CBD-defined subgraph. In that, no property selection is required from the user at the main matching module. At the instance disambiguation and link selection phase, to the best of our knowledge, *Legato* is the first tool of its kind to apply a key discovery algorithm combined with a key ranking tool in its internal mechanics, which allows to automatically select the discriminative set of properties that guarantee the best linking performance. The types of instances to compare has to be manually specified.

As we have seen in the discussion in Section 3, fully automatic link specification remains a challenge, even for specialised approaches. In all cited methods, the user input is required under one form or another (we detail on that in Section 6.6). With regard to the choice of similarity measures and thresholds, we take a different and simpler approach as compared to machine learning-based techniques. In the first place, given the vector space framework that we adopt, the choice of the cosine similarity combined with a TF-IDF weighting scheme that allows to take into account the overall textual content of the instance descriptions, appears to be natural. *Legato* depends on a single threshold for the cosine similarity. It has been fixed as an outcome of an extensive empirical analysis, although the user is given the possibility to easily modify this parameter. In a similar spirit as [52], during the main matching phase, the similarity threshold is deliberately kept very low, so that the system can discover a large number of candidate links, ensuring high recall. Improving precision is handled at the time by the preprocessing module (particularly the filter on problematic properties), and by the instance disambiguation and link merging module (see details in the preceding section).

While we do not claim that the tool provides a fully automatic solution, we have attempted to offer punctual solutions to several issues that are not integrally handled automatically by a couple of the most popular state-of-the-art tools, such as (1) property filtering, (2) property selection and (3) similarity measures combination and tuning. A direct comparison in terms of user configuration to other tools is difficult, because of the varying underlying principles of these tools. However, we attempt to compare *Legato* to several other popular and freely available systems - SILK, LIMES, AML, as well as three automatic configuration versions of LIMES (relying on batch, active or Unsupervised learning) (cf. Table 1). The comparison criteria in the table come from the union of the sets of parameters in the configuration files of SILK and LIMES, as well as the set of "potential" parameters of *Legato* and AML. We use the word "potential" to indicate that a number of parameters are currently hard-coded both in AML and Legato (giving rise to automatic and manual versions of these tools). For fairness of comparison, we have chosen to include both versions of the two systems to the table. Regarding AML, note that there is no officially released user-tuneable version of the system: in the current release all parameter values are hard-coded except for the type restriction and a threshold. We therefore consider as a manual version of the tool a one containing all parameters that could be included in a configuration file of the tool or could be set by a user with programming skills.[18] The table allows us to conclude that *Legato* and AML are least demanding in terms of user intervention, while the experiments reported in the following section give an advantage to *Legato* in terms of performance.

[18] Source: personal exchanges with the authors of AML.

Configuration element	*Legato*	*Legato*	Silk, Limes	Limes +	Limes +	Limes +	AML	AML
	(manual)	(automatic)	(standard)	Batch L.	Active L.	Unsupervised L.	(manual)	(automatic)
Types	y (=yes)	y	y	y	y	y	y	y
Properties	y	n (=no)	y	y	y	y	n	n
Similarity measures	n	n	y	n	n	n	y	n
Local sim. thresholds	n	n	y	n	n	n	y	n
Global sim. threshold	y	y	y	y	y	y	y	y
Instance profile type	y	n	n	n	n	n	n	n
Machine learning alg.	n	n	n	y	y	y	n	n
Training data	n	n	n	y	n	n	n	n
Label link candidates	n	n	n	n	y	n	n	n
Matching strategy	n	n	n	n	n	n	y	n
Total ratio	4/10	2/10	5/10	5/10	5/10	4/10	5/10	2/10

Table 1: End-user configuration effort comparison.

6. Experimental Evaluation

The experiments reported in this section aim (1) to assess the effectiveness of the internal modules of *Legato*, (2) compare the system to other linking tools and (3) compare it to approaches for automatic link specification. *Legato* was implemented in Java 8 and the experiments were conducted on a machine running under Windows 10 over an Intel Core i5-5300U, with 2.30 GHz CPU and 16 GBytes RAM. Note that the automatic (default) version of *Legato* is evaluated here. The system is available as an open source release at `https://github.com/DOREMUS-ANR/legato`.

6.1. Experimental Setting

We begin by describing the evaluation framework that we have established.

Datasets. For the various experiments carried out and reported in this paper, we have relied on data coming from the Instance Matching evaluation campaigns of OAEI (IM@OAEI) from 2015 to 2017.

• **DOREMUS datasets.** One of the main results of the DOREMUS project is the representation of the catalogs of three French cultural institutions as knowledge graphs following a specifically designed for this purpose model [9] and their publication on the web. This has resulted in the creation of (currently) three knowledge graphs—one per partner institution.[19] The DOREMUS benchmarks have been built together with librarian experts form the BnF and the Philharmonie de Paris. The basis for the construction of the benchmark is a set of pairs of identical works (given in a synthetic table) that have been manually selected by the experts such that one work in each pair belongs to the catalog of the BnF and the other - to that of the Philharmonie. In that process, the experts have identified the heterogeneities that each pair of works manifest. As it can be seen in the table,[20] this resulted in a set of heterogeneity types, specific to the DOREMUS data (note that these types have been generalized in our heterogeneity types categorization provided in Section 2): (1) numbers vs. letters in the tiles, arabic vs. roman numbers in the titles, (2) differences in spelling, (3) missing catalog numbers, (4) different catalogues, (5) multilingual titles, (6) specific characters, (7) differences in the lengths of the property chains that lead to the value of interest (graph depths), (8) different property names for the same entity types, (9) missing descriptions (missing property values), (10) missing titles, (11) use of synonyms. Whenever a given pair of works manifests one of these heterogeneity types, this has been indicated

[19] `https://github.com/DOREMUS-ANR/knowledge-base/tree/master/data`

[20] `https://docs.google.com/spreadsheets/d/19dLjabt_fggTVNuM7XW9CUZkuB1JgoH9xKWQgLZv_Y4/edit#gid=1271677916`

in the table. This resulted in two RDF datasets benchmarks released by OAEI in 2016 and 2017 (the datasets are not identical, the underlying model as well as their sizes have evolved from one year to the other):

- DOREMUS 2016 consists of three datasets of different sizes and scopes: **9-HT**, **4-HT** (for heterogeneities) and **FP-trap** (for false positives trap). The data is available and described at `http://islab.di.unimi.it/content/im_oaei/2016/#doremus`.
- DOREMUS 2017 consists of **HT** (for heterogeneities) and **FPT** (for false positives trap). The data is available and described at `http://islab.di.unimi.it/content/im_oaei/2017/#doremus`.

The particularity of these benchmarks is that they contain datasets that were particularly designed to challenge the capacity of linking tools to correctly disambiguate highly similar in their descriptions instances (FP-trap (2016) and FPT (2017)). All data follow the same model and therefore share significant number of vocabulary terms. Nonetheless, these datasets are highly heterogeneous in terms of all other ontology-dimension heterogeneities and many value-dimension heterogeneities (Section 2) .

- **Synthetic datasets.** We additionally evaluated *Legato* on synthetic benchmark datasets from three consecutive years of the OAEI campaign.

- SPIMBENCH 2015: This includes three benchmark datasets generated through the Semantic Publishing Instance Matching Benchmark (SPIMBENCH) [34] by transforming the source instances based on their values and semantics (the **Val-Sem** dataset), on their values and structures (the **Val-Struct** dataset) and on their values, structures and semantics (the **Val-Struct-Sem** dataset). The data is available and described at `http://oaei.ontologymatching.org/2015/im/index.html`.
- SPIMBENCH 2016: This comprises **SPIMBENCH small**, that we denote **SB-s** and **SPIMBENCH large**, that we denote **SB-l**, two datasets of different sizes, produced by following the same strategy as described above. The data is available and described at `http://islab.di.unimi.it/content/im_oaei/2016/#synthetic`.
- SPIMBENCH 2017: This includes **SPIMBENCH sandbox**, that we denote **SB-s** (the year of edition helps disambiguate the two SB-s notations) and **SPIMBENCH mainbox**, that we denote **SB-m**, datasets of different sizes, produced by following the SPIMBENCH transformation patterns.

Scenarii. We consider five evaluation scenarii. First, we evaluate *Legato* with respect to three of its core components, by assessing (1) the efficiency of the automatic property filtering module by measuring the impact of automatically identified problematic properties on the quality of the generated links (Section 6.2), (2) the impact of the choice of instance profile (Section 6.3) and (3) the use of keys to efficiently disambiguate instances and assess and select links by improving recall (Section 6.4). Then, (4) we assess the overall performance of *Legato* by comparing it to state-of-the-art systems and participant-systems to the IM@OAEI campaigns on a large variety of datasets (Section 6.5). Finally, (5) we confront *Legato* with EAGLE and WOMBAT (in its two versions)—two automatic link specification methods implemented with LIMES (Section 6.6).

We use three well-known performance measures: Precision (P), Recall (R) and F-Measure (F-m). P and R evaluate the *correctness* and the *completeness* of the generated links, respectively, while F-m is their harmonic mean.

Tuning. We have conducted a series of experiments by varying the cosine similarity threshold value (cf. Section 4) observing its impact on F-m, P and R. We observed that the best results of *Legato* on all data were achieved with a threshold of 0.2. We report as an example the results obtained on the DOREMUS 2017 HT dataset in the form of couples (F-m, threshold): (0.92, 0.1), (0.93, 0.2), (0.81, 0.3), (0.59, 0.4), (0.29, 0.5), (0.14, 0.6), (0.1, 0.7), (0.02, 0.8), (0.0, 0.9). Analogical behaviour has been

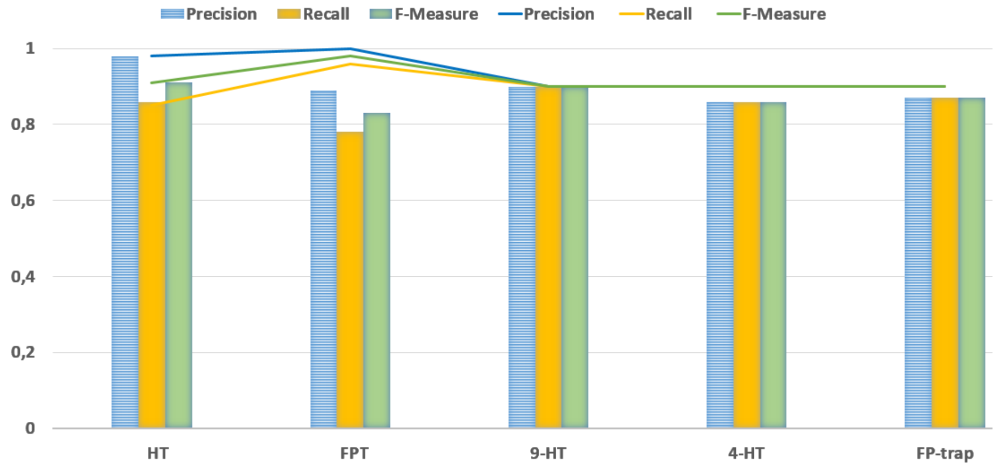

Figure 5: Property filtering evaluation on DOREMUS datasets: without (histograms) and with (curves) automatic removal of problematic properties.

observed with other datasets. This threshold is low, which guarantees a fair trade-off between the matching module (ensuring high recall) and the disambiguation and merge module (improving precision).

6.2. *Effectiveness of Property Filtering*

In this experiment (scenario 1), instances are compared first by considering all the properties and then after removing the automatically identified problematic ones. An improved performance is expected after property filtering. We experiment on the DOREMUS datasets from 2016 and 2017, as reported in Fig. 5, articulating the assessment in two phases: without (histograms) and with (curves) the automatic removal of problematic properties. The experiments show that applying the automatic property filtering module allows to improve the linking quality for all datasets except for HT and 9-HT, where this module has no impact. Further experiments and analysis on these two datasets reveal that removing all the other properties separately does not improve the results either, which indicates that no problematic property has been "missed" by the module.

6.3. *Effectiveness of Instance Profiling*

In these experiments (scenario 2), we analyse the behaviour of *Legato* with respect to different choices of instance profiles, expressed as four CBD-based instance representations (Section 4): CBD, $\uparrow CBD$, $\downarrow CBD$ or $\updownarrow CBD$. We have selected two real-world datasets from OAEI 2017, as well as four synthetic ones from OAEI 2015 and 2016. As expected, the effectiveness of instance profiling depends on how the data is modelled (Figure 6). Particularly, these tests show that the choice of a $\updownarrow CBD$ profile is relevant for the real-world datasets HT and FPT data, as the highest F-measure scores are achieved with that representation (91% and 98% for HT and FPT, respectively), while for the synthetic datasets the relevant information is located in their direct CBDs. For those datasets, we can also deduce that taking into account the description of predecessors does not impact the matching decision. The results do not allow to conclude on the choice of an instance profile in the general case. An understanding of how data is modelled (where to look for important information) is needed in order to guarantee a choice of a profile that maximises the outcome. Based on our results, we set $\updownarrow CBD$ and CBD as profile parameters for the real-world and the synthetic datasets, respectively.

6.4. *Effectiveness of Post-processing*

We evaluate the efficiency of the post-processing step (scenario 3) of *Legato* consisting of an instance disambiguation and a link merge module. In that, we execute *Legato* with and without performing this step. By taking as a reference the set of *candidate links*, generated at the main matching step, we measure the proportion of links that fall on its intersection with the *sure links* set, dubbed **#safe_links**, as well as the propor-

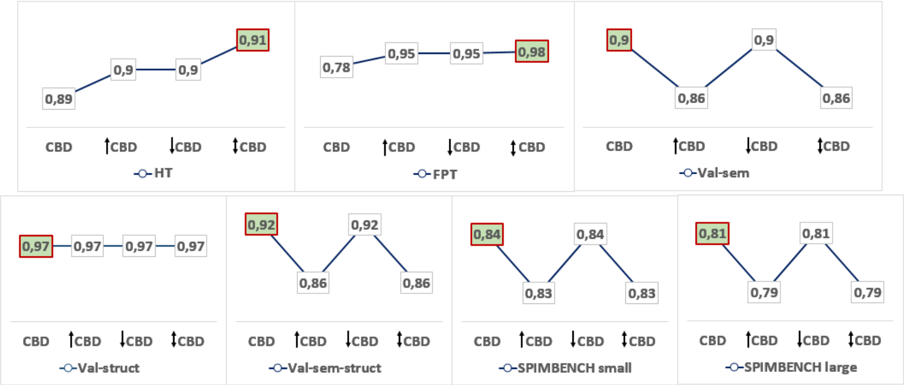

Figure 6: CBD-based instance profiling evaluation on reference datasets (F-measures).

dataset / profile		#safe_links	#deleted_links	#added_links
HT	*CBD*	≃ 10%	≃ 2%	0%
	↓ *CBD*	≃ 14%	≃ 2%	0%
	↑ *CBD*	10.5%	≃ 2%	0%
	↕ *CBD*	≃ 5%	≃ 1%	0%
FPT	*CBD*	≃ **55**%	≃ **18**%	≃ 3%
	↓ *CBD*	≃ 21%	≃ 3%	0%
	↑ *CBD*	≃ 15%	0%	0%
	↕ *CBD*	≃ 15%	0%	0%

Table 2: Post-processing evaluation on OAEI 2017 datasets.

tion of links deleted or added by the merge module, dubbed **#deleted_links** and **#added_links**, respectively, in order to form the *final linkset*. Table 2 shows the results on the DOREMUS 2017 data. We observe, as hypothesised, that performing the post-processing step is significant in the presence of highly similar instances (dataset FPT). We also notice that the precision boost is pronounced in the case of a simple *CBD* profile, which (or variants of which) is most commonly used in the existing instance representation approaches (cf. Section 3). This shows the potential of the post-processing module of *Legato* to make up for possible flaws in the instance representation.

6.5. General Evaluation

In this experiment (scenario (4)), we assess the performance of *Legato* in its complete automatic version, performing both *property filtering* and *post-processing*. We compare *Legato* to the participant tools to IM@OAEI on all benchmarks from the years 2015, 2016 and 2017 and additionally with SILK [5] for the DOREMUS 2017 data. In 2015, two systems participated on the three tasks: STRIM [53] and LogMap[21] [36]. In 2016, the systems LogMapIm [54], AML[22] [55, 52] and RiMOM [28] participated on the two proposed tasks. In 2017, in addition to *Legato*, which participated to OAEI for the first time, AML, I-Match [56] and LogMap participated to the SPIMBENCH task, while AML, I-Match, NjuLink[23] [57] and LogMap participated to the DOREMUS task. In addition to the participant systems, we have included to the comparison SILK[24] in its 2.6.1 version on the 2017 data. Note that the comparison to SILK was made by using the best keys in its link specification as identified by the RANKey algorithm [8], namely the

[21] `https://github.com/ernestojimenezruiz/logmap-matcher/`

[22] `https://github.com/AgreementMakerLight/AML-Project`

[23] `https://github.com/nju-websoft/njuLink`

[24] `https://github.com/silk-framework/silk`

Benchmark (year)	System	P	R	F-m	size
HT (2017)	*Legato*	0.930	0.920	0.930	476
	AML	0.851	0.479	0.613	
	I-Match	0.680	0.071	0.129	
	LogMap	0.406	0.882	0.556	
	NjuLink	**0.966**	**0.945**	**0.955**	
	SILK	0.34	0.12	0.18	
FPT (2017)	*Legato*	**1.000**	**0.980**	**0.990**	150
	AML	0.914	0.427	0.582	
	I-Match	**1.000**	0.053	0.101	
	LogMap	0.119	0.880	0.210	
	NjuLink	0.959	0.933	0.946	
	SILK	0.45	0.2	0.27	
SB-s (2017)	*Legato*	**0.980**	0.730	0.840	≃ 1800
	LogMap	0.938	0.763	0.841	
	AML	0.849	**1.000**	0.918	
	I-Match	0.854	0.997	**0.920**	
SB-m (2017)	*Legato*	**0.970**	0.700	0.810	≃ 1800
	LogMap	0.893	0.709	0.790	
	AML	0.855	**1.000**	**0.922**	
	I-Match	0.856	0.997	0.921	
9-HT (2016)	*Legato*	0.9	**0.9**	0.9	60
	AML	**0.96**	0.87	**0.91**	
	RIMOM	0.81	0.81	0.81	
4-HT (2016)	*Legato*	0.9	**0.9**	**0.9**	400
	AML	**0.93**	0.77	0.84	
	RIMOM	0.74	0.74	0.74	
FP-trap (2016)	*Legato*	0.9	**0.9**	**0.9**	80
	AML	**0.92**	0.85	0.88	
	RIMOM	0.7	0.7	0.7	
SB-s (2016)	*Legato*	**0.98**	0.74	0.84	≃ 380
	LogMapIm	0.95	0.76	0.85	
	AML	0.9	0.74	0.82	
	RiMOM	**0.98**	**1.0**	**0.99**	
SB-l (2016)	*Legato*	0.96	0.71	0.81	≃ 1800
	LogMapIm	0.98	0.69	0.81	
	AML	0.9	0.74	0.81	
	RiMOM	**0.99**	**1.0**	**0.99**	

Table 3: Results on different benchmark datasets for *Legato*, compared to other linking tools.

`U16_has_catalogue_statement` property from the DOREMUS ontology.[25] For the sake of reproducibility, we make available the SILK configuration files used for the HT and the FPT data at the following link: `https://github.com/manoach/SILK-Evaluation`.

Note that, among the cited systems RIMOM and I-Match do not have openly available source code or executable versions. The results reported for these tools are taken from the OAEI web site for the year 2015 or via the SEALS platform to which we had access as co-organisers of the OAEI tracks of 2016 and 2017. The runtimes of the systems participating in the different OAEI campaign editions are not reported, for which reason comparison with respect to that criterion to *Legato* is not feasible (runtimes of our system are reported in the following experiment).

The results of the evaluation are presented in Table 3. For the sake of readability, we provide only the results of the evaluation on data from OAEI 2016 and 2017. The complete results can be found at `https://github.com/DOREMUS-ANR/legato/blob/master/Legato-Results.png`.

We can observe that *Legato* has comparable (and in certain cases better) performance as the state-of-the-art systems on all benchmarks. These results show in particular that *Legato* is well-suited for dealing with real-world data containing difficult to disambiguate instances (FPT and FP-trap), which confirms the effectiveness of the post-processing module. Overall, *Legato* performs well when data heterogeneity is related to *descriptive* differences and all remaining heterogeneity types of the *ontological dimension* (Section 2) thanks to its *property filtering* and *indexing* techniques. As expected, our system is less effective on the synthetic data transformed with SPIMBENCH. This is explained by the fact that the heterogeneities of *value dimension* are not considered by *Legato* (for example, we did not implement string unification methods in the indexing process). Nevertheless, we consider the results of *Legato* satisfactory on these data, given that it provides comparable results as the systems, to which it was confronted, but requires significantly less user-tuning effort than many of these tools.

Table 4: Automatic LS approaches vs. *Legato* on SPIMBENCH data.

Method	System	F-m	P	R	Execution (ms)
SB-s (2016)					
UNSUPERVISED	EAGLE	0.62	0.63	0.61	74531
	WOMBAT simple	0.64	0.61	0.67	14905
	WOMBAT complete	0.64	0.61	0.67	64275
ACTIVE	EAGLE	0.36	0.28	0.50	100373
	WOMBAT simple	0.64	0.61	0.67	15958
	WOMBAT complete	0.64	0.61	0.67	61612
BATCH	EAGLE	0.64	0.77	0.55	93857
	WOMBAT simple	0.64	0.61	0.67	14079
	WOMBAT complete	0.64	0.61	0.67	61730
Legato		**0.84**	**0.98**	**0.74**	**7000**
SB-l (2016)					
UNSUPERVISED	EAGLE	0.43	0.32	0.63	1597173
	WOMBAT simple	0.43	0.32	0.63	413450
	WOMBAT complete	0.43	0.32	0.63	364643
ACTIVE	EAGLE	0.43	0.32	0.63	1668132
	WOMBAT simple	0.43	0.32	0.63	282928
	WOMBAT complete	0.43	0.32	0.63	254423
BATCH	EAGLE	0.45	0.38	0.55	1032956
	WOMBAT simple	0.43	0.32	0.63	249196
	WOMBAT complete	0.43	0.32	0.63	226922
Legato		**0.81**	**0.96**	**0.71**	**31000**

6.6. *Comparison to Automatic Link Specification Methods*

These experiments (scenario (5)) confront *Legato* with two approaches to automatic links specification (LS), EAGLE [44] and WOMBAT [48], that are included in the linking system LIMES. We have used the *limes-core-1.2.1* release of the system,[26] where two versions of WOMBAT are implemented: simple and complete. The simple version of WOMBAT learns links specifications and combines them to improve F-measure, while the complete version implements a refinement operator, which guarantees the best specification.

Regarding the level of automation of the linking process, in the case of the three methods a manual configuration is required. At the beginning, the user is asked to select the preferred link specification method (EAGLE or WOMBAT) and afterwards a number of parameters need to be manually fed to the system. In the first place, the user has to select the type of learning approach to apply (supervised, active or batch, see Section 3). Then, the names of the properties to compare across the source and the target datasets (types) have to be specified.

[25] `http://data.doremus.org/ontology#U16_has_catalogue_statement`

[26] `https://github.com/dice-group/LIMES/releases`

Table 5: Automatic LS approaches vs. *Legato* on DOREMUS data.

Method	System	F-m	P	R	Execution (ms)
HT (2017)					
UNSUPERVISED	EAGLE	0.45	0.97	0.29	1882
	WOMBAT simple	0.45	1.0	0.29	390
	WOMBAT complete	0.45	1.0	0.29	472
ACTIVE	EAGLE	0.44	1.0	0.28	1227
	WOMBAT simple	0.45	1.0	0.29	395
	WOMBAT complete	0.45	1.0	0.29	565
BATCH	EAGLE	0.44	1.0	0.28	7733
	WOMBAT simple	0.45	1.0	0.29	**387**
	WOMBAT complete	0.45	1.0	0.29	459
Legato		**0.93**	0.93	**0.92**	69000
FPT (2017)					
UNSUPERVISED	EAGLE	0.57	1.0	0.4	911
	WOMBAT simple	0.57	1.0	0.4	**232**
	WOMBAT complete	0.57	1.0	0.4	294
ACTIVE	EAGLE	0.57	1.0	0.4	1399
	WOMBAT simple	0.57	1.0	0.4	238
	WOMBAT complete	0.57	1.0	0.4	294
BATCH	EAGLE	0.57	1.0	0.4	813
	WOMBAT simple	0.57	1.0	0.4	235
	WOMBAT complete	0.57	1.0	0.4	294
Legato		**0.99**	1.0	**0.98**	16000

For each pair of properties to compare, the user does not have to select the similarity measure and its associated threshold, but a global similarity threshold needs to be manually set. For the batch algorithm, a small sample of `owl:sameAs` links over the data is also required. In our experiments, the choice of properties to compare has been done again by using the RANKey key selection system (just as described in the previous subsection). This resulted in the choice of the following keys on each of the datasets:

SB-s: {bbc[27]:bbc/primaryContentOf, bbc:creativework/description}

SB-l: {bbc:creativework/title, bbc:creativework/about}

HT & FPT: {mus[28]:#U16_has_catalogue_statement}

The results are given in Tables 4 and 5. For the sake of reproducibility of the experiments, all configuration files used for the different LS tools for LIMES, together with the data and the results are made available.[29] In that experiment, we also report on the respective execution times of the evaluated systems. We see that *Legato*, in its fully automatic version where only the classes to compare need to be specified, achieves better results than the three tested methods in terms of performance. In terms of execution times, our system outperforms the other methods on the synthetic data but scales much worse on the DOREMUS real-world datasets.

We explain this difference by the characteristics of the datasets. Particularly, the sizes of the respective *CBD*-based profiles and hence documents generated in each of the two cases are different. The number of literals collected by the *CBD*-profile of the DOREMUS resources (using both successors and predecessors) is larger than those of the SPIMBENCH data (using only the *CBD* of the resource), leading to larger in size documents and hence vectors of larger dimension, which has a direct impact on the computational efficiency.

7. Conclusion and Future Work

In this work, we propose, implement and evaluate *Legato*, an open source framework for discovery of identity links across RDF graphs in the context of the web of open data. We provide an extensive inventory of dataset heterogeneities, which lie at the origin of the data linking problem. We show that *Legato* addresses efficiently many of these heterogeneities, particularly those at ontological level, by implementing efficient property filtering (data cleaning) and link selection modules, acting, respectively, at the pre- and post-processing steps of the linking pipeline. A core feature of *Legato* is its capacity to avoid the generation of false positives by disambiguating effectively highly similar instances across datasets by the help of a clustering method and a key selection and ranking algorithm. In addition, the system has the advantage of proposing a fully automatic version (only the types of instances to compare need to be indicated) - in that, nor property, neither similarity selection are performed manually.

Fully automating the data linking process is a highly challenging task. As we have seen in our related work discussions, several approaches have been proposed to handle automatic system configurations at different levels (e.g., selecting properties, or tuning similarity measures and thresholds). These approaches, even when included in the internals of a tool, do not

[27] bbc=http://www.bbc.co.uk/ontologies/

[28] mus=http://data.doremus.org/ontology

[29] `https://github.com/manoach/LIMES-Evaluation`

guarantee a process with zero user implication—to our knowledge, there does not exist to date an entirely automatic data linking system, that is also agnostic to the underlying data. In reality, there is a multitude of factors that make it difficult (or even futile) to think of a generic fully automatic linking system, and these factors strongly relate to the specificities of data with respect to domains, structure, coverage, their degree of heterogeneity, and varieties of presentation. For that reason, we have left the possibility to the alerted user to set two parameters in a customised version of *Legato*: the threshold for the similarity measure and the shape of the CBD-based instance profile.

We evaluate *Legato* on real-world music-related data and on synthetic datasets coming from OAEI 2015, 2016 and 2017. The results showed that our system is in competition with state-of-the-art tools, outperforming them on datasets containing highly heterogeneous or difficult to disambiguate instances. Additionally, we evaluate the performance of *Legato* with respect to a version of LIMES that uses automatic link specification modules and obtain better results.

Scalability is an important factor given the sizes of datasets that one has to deal with. *Legato* can be improved in that respect. One way that we deal with this issue in real-wolrd matching tasks is to partition the data with respect to the values of a property that we know that will generate groups of potential linking candidates across the two datasets and treat these groups separately (in the case of the DOREMUS data, this property is the name of the composer of a music work).[30] It is a subject of future work to automate this process and thus reduce the execution time of the system.

As observed before, our approach is sensitive to the underlying data model, particularly with regard to the CBD-based profiling. Our experiments do not allow to conclude on which CBD presentation is most appropriate in the general case. More experiments need to be performed on larger sets of benchmarks (for example, those coming form the HOBBIT project[31]) in order to confirm this observation. The choice of centroid vectors to represent clusters of instances in the disambiguation method requires further evaluation as well, under the hypothesis that such representation would provide skewed results in the presence of outliers.

In future work, we also aim to address the problem of information complementarity across datasets—how to handle entities that are described by complementary sets of properties in two different graphs and therefore have little information in common in order to compare them. We will explore the application of knowledge graph augmentation techniques in order to reconstitute the missing descriptions intersection.

Acknowledgements

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

References

[1] A. Ferrara, A. Nikolov, F. Scharffe, Data linking for the semantic web, Semantic Web: Ontology and Knowledge Base Enabled Tools, Services, and Applications 169.

[2] M. Achichi, Z. Bellahsene, K. Todorov, A survey on web data linking, Revue des Sciences et Technologies de l'Information - ISI.

[3] M. Nentwig, M. Hartung, A.-C. Ngonga Ngomo, E. Rahm, A survey of current link discovery frameworks, Semantic Web 8 (3) (2017) 419–436.

[4] A. N. Ngomo, S. Auer, LIMES - A time-efficient approach for large-scale link discovery on the web of data, in: IJCAI, 2011, pp. 2312–2317.

[5] A. Jentzsch, R. Isele, C. Bizer, Silk-generating rdf links while publishing or consuming linked data, in: ISWC, 2010.

[6] D. Symeonidou, V. Armant, N. Pernelle, F. Sais, Sakey: Scalable almost key discovery in RDF data, in: ISWC, 2014, pp. 33–49.

[7] R. Isele, C. Bizer, Active learning of expressive linkage rules using genetic programming, J. Web Sem. 23 (2013) 2–15.

[8] M. Achichi, M. Ben Ellefi, D. Symeonidou, K. Todorov, Automatic key selection for data linking, in: EKAW, Springer, 2016, pp. 3–18.

[9] M. Achichi, P. Lisena, K. Todorov, R. Troncy, J. Delahousse, Doremus: A graph of linked musical works, in: International Semantic Web Conference, Springer, 2018, pp. 3–19.

[30] Note that this manipulation has not been applied to the experiments reported in this paper and therefore does not impact the execution times reported here.

[31] `https://project-hobbit.eu/`

[10] A. Ferrara, D. Lorusso, S. Montanelli, G. Varese, Towards a benchmark for instance matching, in: Ontology Matching-Volume 431, CEUR-WS. org, 2008, pp. 37–48.

[11] R. Navigli, S. P. Ponzetto, Babelnet: Building a very large multilingual semantic network, in: 48th annual meeting of the association for computational linguistics, ACL, 2010, pp. 216–225.

[12] J. Volz, C. Bizer, M. Gaedke, G. Kobilarov, Silk-a link discovery framework for the web of data., LDOW 538.

[13] D. Faria, C. Pesquita, B. S. Balasubramani, C. Martins, J. Cardoso, H. Curado, F. M. Couto, I. F. Cruz, OAEI 2016 results of AML, in: Ontology Matching ISWC, CEUR, Vol. 1766, 2016.

[14] D. Ngo, Z. Bellahsene, K. Todorov, Extended tversky similarity for resolving terminological heterogeneities across ontologies, in: OTM On the Move to Meaningful Internet Systems", Springer, 2013, pp. 711–718.

[15] M. Cheatham, P. Hitzler, String similarity metrics for ontology alignment, in: ISWC 2013, Springer, 2013, pp. 294–309.

[16] Y. Yamamoto, A. Yamaguchi, H. Bono, T. Takagi, Allie: a database and a search service of abbreviations and long forms, Database 2011.

[17] C. Li, L. Ji, J. Yan, Acronym disambiguation using word embedding, in: AAAI, 2015, pp. 4178–4179.

[18] J. Gracia, E. Montiel-Ponsoda, P. Cimiano, A. Gomez-Perez, P. Buitelaar, J. McCrae, Challenges for the multilingual web of data, Web Semantics: Science, Services and Agents on the World Wide Web 11 (2012) 63–71.

[19] A. Simon, R. Wenz, V. Michel, A. D. Mascio, Publishing bibliographic records on the web of data: Opportunities for the bnf (french national library), in: ESWC 2013, 2013, pp. 563–577.

[20] K. D. Bollacker, R. P. Cook, P. Tufts, Freebase: A shared database of structured general human knowledge, in: AAAI, 2007, pp. 1962–1963.

[21] F. Scharffe, Y. Liu, C. Zhou, Rdf-ai: an architecture for rdf datasets matching, fusion and interlink, in: IJCAI 2009 workshop IR-KR, 2009.

[22] T. Lesnikova, J. David, J. Euzenat, Interlinking english and chinese rdf data sets using machine translation, in: ESWC workshop Know@ LOD, Vol. 2013, 2014.

[23] T. Lesnikova, J. David, J. Euzenat, Interlinking english and chinese RDF data using babelnet, in: ACM DocEng, 2015, pp. 39–42.

[24] A. N. Tigrine, Z. Bellahsene, K. Todorov, Light-weight cross-lingual ontology matching with lyam++, in: OTM: On the Move to Meaningful Internet Systems, Springer, 2015, pp. 527–544.

[25] F. Narducci, M. Palmonari, G. Semeraro, Cross-lingual link discovery with TR-ESA, Inf. Sci. 394 (2017) 68–87.

[26] A. Nikolov, V. Uren, E. Motta, Knofuss: A comprehensive architecture for knowledge fusion, in: K-Cap, ACM, 2007, pp. 185–186.

[27] S. Rong, X. Niu, E. W. Xiang, H. Wang, Q. Yang, Y. Yu, A machine learning approach for instance matching based on similarity metrics, in: ISWC, Springer, 2012, pp. 460–475.

[28] C. Shao, L. Hu, J. Li, Z. Wang, T. L. Chung, J. Xia, Rimom-im: A novel iterative framework for instance matching, J. Comput. Sci. Technol. 31 (1) (2016) 185–197.

[29] M. Kejriwal, D. P. Miranker, Semi-supervised instance matching using boosted classifiers, in: ESWC, 2015, pp. 388–402.

[30] D. Symeonidou, N. Pernelle, F. Saïs, KD2R: A key discovery method for semantic reference reconciliation, in: On the Move to Meaningful Internet Systems: OTM 2011 Workshops, 2011, pp. 392–401.

[31] T. Soru, E. Marx, A. N. Ngomo, ROCKER: A refinement operator for key discovery, in: WWW, 2015, pp. 1025–1033.

[32] C. Bizer, R. Cyganiak, Quality-driven information filtering using the wiqa policy framework, Web Semantics: Science, Services and Agents on the World Wide Web 7 (1) (2009) 1–10.

[33] M. B. Ellefi, Z. Bellahsene, J. Breslin, E. Demidova, S. Dietze, J. Szymanski, K. Todorov, Rdf dataset profiling-a survey of features, methods, vocabularies and applications, Semantic Web.

[34] T. Saveta, E. Daskalaki, G. Flouris, I. Fundulaki, M. Herschel, A.-C. Ngonga Ngomo, Pushing the limits of instance matching systems: A semantics-aware benchmark for linked data, in: WWW, ACM, 2015, pp. 105–106.

[35] T. Saveta, E. Daskalaki, G. Flouris, I. Fundulaki, M. Herschel, A. N. Ngomo, LANCE: piercing to the heart of instance matching tools, in: ISWC, 2015, pp. 375–391.

[36] E. Jimenez-Ruiz, B. C. Grau, Logmap: Logic-based and scalable ontology matching, in: ISWC, Springer, 2011, pp. 273–288.

[37] M. Atencia, J. David, J. Euzenat, Data interlinking through robust linkkey extraction., in: ECAI, 2014, pp. 15–20.

[38] T. Soru, E. Marx, A.-C. Ngonga Ngomo, Rocker: a refinement operator for key discovery, in: WWW, ACM, 2015, pp. 1025–1033.

[39] M. Atencia, J. David, F. Scharffe, Keys and pseudo-keys detection for web datasets cleansing and interlinking, in: EKAW, 2012, pp. 144–153.

[40] D. Symeonidou, I. Sanchez, M. Croitoru, P. Neveu, N. Pernelle, F. Sais, A. Roland-Vialaret, P. Buche, A. Muljarto, R. Schneider, Key discovery for numerical data: Application to oenological practices, in: ICCS, 2016, pp. 222–236.

[41] D. Symeonidou, L. Galarraga, N. Pernelle, F. Saïs, F. Suchanek, VICKEY: Mining Conditional Keys on Knowledge Bases, in: ISWC, 2017.

[42] H. Farah, D. Symeonidou, K. Todorov, Keyranker: Automatic rdf key ranking for data linking, in: K-Cap, ACM, 2017, p. 7.

[43] A.-C. N. Ngomo, J. Lehmann, S. Auer, K. Höffner, Raven-active learning of link specifications, in: Ontology Matching, CEUR-WS. org, 2011, pp. 25–36.

[44] A.-C. N. Ngomo, K. Lyko, Eagle: Efficient active learning of link specifications using genetic programming, in: ESWC, Springer, 2012, pp. 149–163.

[45] A. N. Ngomo, K. Lyko, V. Christen, COALA - correlation-aware active learning of link specifications, in: ESWC, 2013, pp. 442–456.

[46] R. Isele, C. Bizer, Learning linkage rules using genetic programming, in: Ontology Matching-Volume 814, CEUR-WS. org, 2011, pp. 13–24.

[47] M. Kejriwal, D. P. Miranker, An unsupervised instance matcher for

schema-free RDF data, J. Web Sem. 35 (2015) 102–123.

[48] M. A. Sherif, A.-C. N. Ngomo, J. Lehmann, Wombat–a generalization approach for automatic link discovery, in: ESWC 2017, Springer, 2017, pp. 103–119.

[49] A. Nikolov, M. d'Aquin, E. Motta, Unsupervised learning of link discovery configuration, in: ESWC, 2012, pp. 119–133.

[50] A. Nikolov, V. S. Uren, E. Motta, A. N. D. Roeck, Integration of semantically annotated data by the knofuss architecture, in: EKAW, 2008, pp. 265–274.

[51] L. Rokach, O. Maimon, Clustering methods, in: The Data Mining and Knowledge Discovery Handbook, 2005, pp. 321–352.

[52] D. Faria, B. S. Balasubramani, V. R. Shivaprabhu, I. Mott, C. Pesquita, F. M. Couto, I. F. Cruz, Results of AML in OAEI 2017, in: OM@ISWC, 2017, pp. 122–128.

[53] A. Khiat, M. Benaissa, M. A. Belfedhal, Strim results for oaei 2015 instance matching evaluation., in: OM, 2015, pp. 208–215.

[54] E. Jiménez-Ruiz, B. C. Grau, V. Cross, Logmap family participation in the OAEI 2017, in: OM@ISWC, 2017, pp. 153–157.

[55] D. Faria, C. Pesquita, E. Santos, M. Palmonari, I. F. Cruz, F. M. Couto, The agreementmakerlight ontology matching system, in: OTM: On the Move to Meaningful Internet Systems, Springer, 2013, pp. 527–541.

[56] A. Khiat, M. Mackeprang, I-match and ontoidea results for OAEI 2017, in: OM@ISWC, 2017, pp. 135–137.

[57] X. Lyu, Q. Zhang, W. Hu, Z. Sun, Y. Qu, njulink: results for instance matching at OAEI 2017, in: OM@ISWC, 2017, pp. 158–165.