Undefined 0 (2015) 1–0
IOS Press

A Survey of Current Link Discovery Frameworks

Markus Nentwig [a,*], Michael Hartung [a] Axel-Cyrille Ngonga Ngomo [b] and Erhard Rahm [a]
[a] *Database Group, University of Leipzig, Augustusplatz 10, 04109 Leipzig, Germany*
E-mail: {nentwig, hartung, rahm}@informatik.uni-leipzig.de
[b] *AKSW, University of Leipzig, Augustusplatz 10, 04109 Leipzig, Germany*
E-mail: ngonga@informatik.uni-leipzig.de

Abstract: Links build the backbone of the Linked Data Cloud. With the steady growth in size of datasets comes an increased need for end users to know which frameworks to use for deriving links between datasets. In this survey, we comparatively evaluate current Link Discovery tools and frameworks. For this purpose, we outline general requirements and derive a generic architecture of Link Discovery frameworks. Based on this generic architecture, we study and compare the features of state-of-the-art linking frameworks. We also analyze reported performance evaluations for the different frameworks. Finally, we derive insights pertaining to possible future developments in the domain of Link Discovery.

1. Introduction

Over the last years, the Linked Open Data (LOD) Cloud has been the most well-known incarnation of the Linked Data Principles. The intention behind this set of interlinked datasets is to create the initial seed for the machine-readable extension of the current Web dubbed the Data Web. While partly very large datasets are being added to the LOD Cloud on a regular basis (e.g., Linked TCGA [47]), they are only sparsely linked with other datasets. Recent studies show that 44% of the LOD datasets are not connected to other datasets at all [48]. This problem is of major importance as links are central for manifold applications including federated queries [46] and answering complex questions [49,53]. The main reason for this blatant lack of links in the LOD Cloud lies in the creation of links being a very tedious process when carried out manually. This is especially true when dealing with large knowledge bases which contain a very large number of resources. For example, creating links between DBpedia[1] (4.5 million resources) and LinkedGeoData[2] (1+ million resources) would last several decades if checking whether two resources should be linked lasted 1ms.

Several software tools and frameworks have already been developed to address the link discovery problem especially to identify semantically equivalent objects in different data sources. The basic intuition behind most of these approaches is to reduce the link discovery problem to a similarity computation problem: Given two sets of resources S and T, the goal is to automatically find pairs of resources in $S \times T$ that should be linked with each other, e.g., according to a `owl:sameAs` relationship. Two main problems arise when dealing with link discovery in this manner: achieving both a high effectiveness and a high efficiency of the linking process. A high effectiveness requires finding (almost) all links between two given sources without deriving incorrect links. Achieving this goal requires finding a suitable link configuration or specification [20,33] specifying the similarity condition(s) two resources $s \in S$ and $t \in T$ have

*Corresponding author. E-mail: nentwig@informatik.uni-leipzig.de

[1] `http://dbpedia.org`
[2] `http://linkedgeodata.org`

to comply with in order to count as a being in the input relation. Even when given a suitable link specification, we have to address the efficiency problem since a naïve implementation which compares all elements of S with all elements of T would have a complexity of $O(|S| \cdot |T|)$.

Link discovery and the related problems of entity resolution or object matching are being studied extensively. A large number of techniques have already been described in several surveys and books, e.g., [13,55,7], In contrast to these works, we focus on surveying and comparing the currently available link discovery tools and frameworks. The goal is thus to survey the state-of-the-art in existing solutions which could be applied to solve specific linking tasks. Our comparison is based on numerous criteria derived from major requirements as well as from the steps of a generic link discovery workflow that we will present in the following sections. The workflow takes into account the newest developments in this research area including support for learning-based configurations and human interaction. We will first present a functional comparison of ten current frameworks. We will consider published performance evaluations for the considered tools including the outcome of instance-level benchmarks of the Ontology Evaluation Alignment Initiative (OAEI). We will try to assess the used evaluation criteria and comparability of the achieved results.

We expect the presented criteria and methodology to be useful to comparatively evaluate additional tools. We plan to continuously extend and update the tool comparison under `http://aksw.org/projects/linkinglod`.

2. Problem statement and requirements

2.1. Link Discovery Problem

The Link Discovery (LD) problem can be described as follows: Given two sets of resources S and T (for example about movies) and a relation $\mathcal{R}$ (e.g., `owl:sameAs` or `dbo:producer`), find all pairs $(s,t) \in S \times T$ such that $\mathcal{R}(s,t)$ holds. The result is represented as a set of links called a *mapping*: $M_{S,T} = \{(a_i, \mathcal{R}, b_j) | a_i \in A, b_j \in B\}$. Optionally a similarity score ($sim \in [0,1]$) computed by an LD tool can be added to the entries of mappings to express the confidence of a computed link. In this case, links can be represented as quadruples $(a_i, \mathcal{R}, b_j, sim(a_i, b_j))$.

Solving the LD problem is challenging due to the typically large volume and semantic heterogeneity of datasets making it difficult to meet major requirements such as high effectiveness and high efficiency. These and further requirements are part of the LD problem and will be discussed in the next subsection. LD has many similarities with the problem of entity resolution (also called deduplication, reference reconciliation or object matching) that has already been extensively addressed [9,25,7]. In particular, similar techniques for evaluating the similarity between objects and for improving the efficiency can be applied. Still there are significant differences between LD and entity resolution that have lead to the development of specific tools for LD. Most entity resolution approaches focus on homogeneous datasets of relatively simple, structured objects, described by a set of single-valued attributes. By contrast, the resources for LD can be heterogeneous and highly interrelated within the datasets. In particular, resources usually abide by an ontology, which describes the properties that resources of a certain type can have as well as the relations between the classes that the resources instantiate. Thus, the LD process usually involves an ontology and instance matching part (see general workflow in Figure 1). Furthermore, entity resolution techniques focus on finding semantically equivalent objects while LD aims at identifying diverse relations (including `owl:sameAs` and domain-specific relations).

2.2. Requirements

As mentioned before, supporting a high effectiveness and efficiency are two main requirements for a LD framework. In the following we pose further requirements and desiderata such as low manual effort for configuration and tuning, support for online LD as well as the provision of a powerful infrastructure.

Effectiveness: A LD tool should generate mappings of high-quality w.r.t. common measures such as precision, recall and F-measure. Hence, results should be precise, i.e., the links generated by a given framework should be correct (precision). A LD tool should also generate as many as possible links to ensure completeness. In summary, only links between resources that really belong together should be produced. This aim is usually achieved by a combination of different LD methods. Systems may support rather simple match techniques such as string similarity comparisons for labels (e.g., [40]) but also complex ones, e.g., by considering the semantic neighborhood of an resource or by

reusing already available links [21,37]. Furthermore, a LD tool should support different link types [54]. In our comparison we will evaluate which LD methods are supported by an LD tool and which effectiveness could be demonstrated in benchmark evaluations.

Efficiency: A LD tool should be fast and scalable to large datasets, e.g., with hundreds of thousands or millions of resources. A naive, non-scalable approach evaluates all possible pairs of resources (Cartesian product) resulting in a quadratic complexity. Hence, a main efficiency goal is to reduce the search space so that the evaluation of irrelevant pairs of resources is largely avoided. Another general optimization approach is parallel LD on multiple cores or multiple nodes in a cluster. This includes the utilization of modern hardware and infrastructures such as graphical processing units (GPU) or Hadoop-based clusters [33].

Low Configuration and Tuning Effort: Achieving a high effectiveness generally demands complex link specifications with the combined use of multiple similarity measures and adequate settings for configuration parameters such as similarity thresholds. Manually specifying such configurations is very difficult and time-consuming so that this effort should largely be reduced by automated approaches. This can be achieved by learning-based methods, e.g., by supervised approaches using training data of matching or non-matching pairs of resources. Alternatively, the LD framework can analyze the datasets, e.g., to select suitable similarity measures or properties to evaluate. In order to really reduce the manual configuration effort, the automated approaches should not introduce a significant extra configuration, e.g., for providing training data or specifying new tuning parameters.

Online and Offline LD: In addition to a classical offline execution of LD, applications such as mashups or on-demand query systems demand an online LD to integrate data from several data sources at runtime. Hence, a LD tool should support such a runtime or adhoc LD, e.g., by providing an appropriate API. Typically, the number of resources to be linked in this way is small thereby facilitating a sufficiently fast execution.

Powerful infrastructure: The support for LD discussed in the previous desiderata requires a set of powerful and easy-to-use tools. In particular, a LD tool should come with flexible libraries with different similarity functions, support different performance optimizations and provide different possibilities to access data sources for LD and a graphical user interface to display and configure the workflow. Furthermore, the specified LD workflow should be executable on different platforms, preferably with parallel processing. Furthermore, mechanisms for collaborative work in groups or crowd-sourcing should be provided to more easily overcome problems like labeling of training data or the generation of gold standards. Overall, a tool should be designed domain-independent but it should be possible to flexibly customize it for specific LD tasks, e.g., linking geographical resources or knowledge from the life sciences.

3. LD workflow

Current LD frameworks mostly apply workflows which consists of several steps to perform LD. In most cases, these workflows are instantiations of the generic workflow shown in Figure 1. This workflow is a generalization of the architecture given by analysing the later on compared LD frameworks (starting in section 4). The input of the workflow includes the two datasets (source, target) to be linked, configuration parameters and optional background knowledge resources. The input data may be provided in the form of RDF/OWL dumps or in the form of a SPARQL endpoint for query-based data access. Linking may be restricted to a subset of a data source, e.g., instances of a particular class, as for example a geographic data source contains settlements and there is no need to compare these with actors from a more generic data source such as DBpedia. The configuration input may either be a complete linking specification (e.g., rules for comparing resources) or selected parameters such as similarity thresholds. Training data required for learning-based linking is another kind of configuration input. Optionally, tools can make use of further knowledge resources such as dictionaries or previously determined mappings for reuse. The output of the workflow is the set of found links or correspondences representing a mapping between the source and target datasets.

The generic workflow itself has three main phases: preprocessing, matching (similarity computation) and postprocessing. Preprocessing in turn deals with two important tasks: finalizing the linking specification (configuration) and improving runtime efficiency, e.g., by reducing the search space for similarity computations in the main match phase. Preprocessing may also include preparatory steps to transform and clean the input data, e.g., to remove stop words or resolve abbreviations. While matching is completely automatic there may be user interaction for preprocessing, e.g., to la-

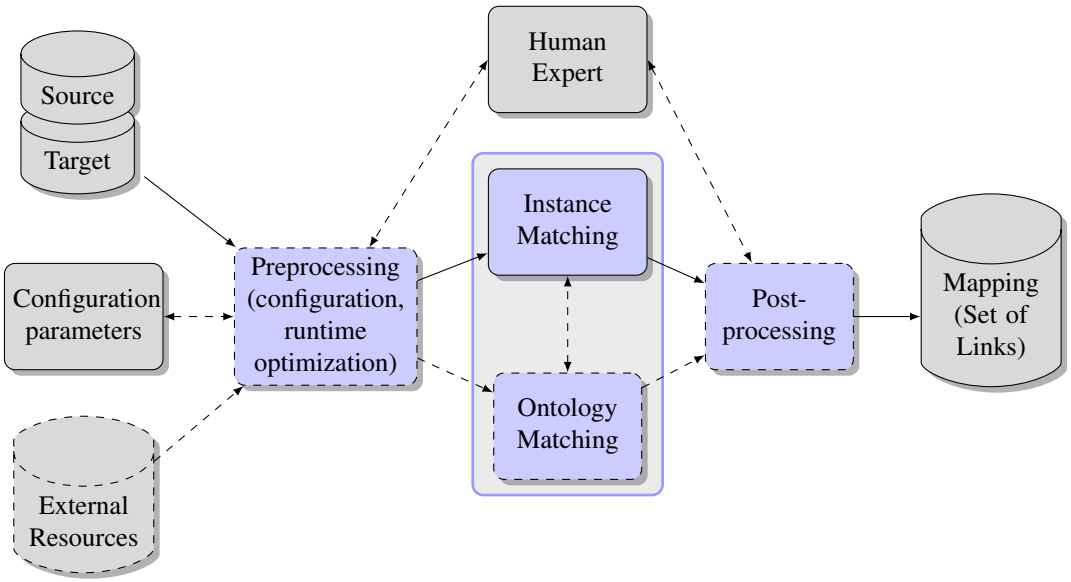

Figure 1. General workflow LD Frameworks (steps with dashed borders are optional)

bel training data for learning-based linking, and post-processing, e.g., to verify computed links with a lower confidence.

In the following we describe the two preprocessing steps about configuration and runtime optimization as well as the match and postprocessing phases and their implementation alternatives in more detail. In the tool evaluation we will study which of the different options are applied.

3.1. Configuration

LD is typically based on evaluating the similarity of resources according to one or several criteria. Each criterion is based on a specific similarity measure or similarity function and compares either properties or the semantic context of resources. For example two movies may be linked by a `owl:sameAs` property based on the similarity of their titles, their release years and the set of actors who starred in them. Specifying a linking configuration thus entails the specification of the elements (properties, context) to evaluate as well as the similarity measures to apply (e.g., a 3-gram string similarity, Jaccard similarity for sets or numerical difference) and a way to derive a combined linking decision from the individual similarity values, e.g., based on similarity thresholds to meet.

According to [25], different similarity values may be combined either numerically or using rule-based or workflow-based approaches. *Numerical approaches* aggregate different similarity values, e.g., by taking a weighted average, and apply a single similarity threshold to the aggregated value. *Rule-based approaches* use so-called *match rules* to derive a match or link decision. Such rules define logical combinations of conditions, e.g., 3-gram similarity for title > 0.9 and equal release year. *Workflow-based approaches* are less common and assume the iterative calculation of different similarity values during the match phase to determine a link decision. For example, one could first calculate the string similarity for a selected property and then apply a more expensive context-based similarity measure (e.g., for the set of movie actors) only for pairs of resources with a high similarity for the first criterion [52,31].

A manual definition of effective linking specifications such as match rules is difficult to achieve in many cases even for domain experts. Hence, it is desirable to automate at least some of the decisions such as selecting the properties or the similarity measures to evaluate. This is achieved by *adaptive LD approaches* that analyze characteristics of the input data to achieve a partially automated specification of the linking configuration [36,3,56,29].

Alternatively, *learning-based approaches* can be applied to semi-automatically or automatically derive a linking specification. The proposed learning approaches for this purpose are mostly supervised, i.e., they depend an suitable training data consisting of

pairs of resources which are labeled as matching (linking) or non-matching. The learned classification model may be based on different learning techniques such as decision trees, SVM or genetic algorithms. Labeling training data is often a manual step requiring the interaction of humans. The manual labeling effort may be kept feasible by crowdsourcing. Alternatively, the amount of training can be limited by *active learning* where user feedback is only requested for a smaller amount of controversial pairs where a similarity function cannot find a clear linking decision. Learning-based approaches may also be unsupervised, thereby avoiding the need for training data. However, these approaches may still require the specification of critical parameters such as suitable similarity or distance measures and threshold values [38,29].

3.2. Runtime optimization

The main approach to optimize the runtime for LD during preprocessing is a reduction of the search space to avoid that the Cartesian Product of the input datasets $S \times T$ needs to be evaluated. This is mainly supported by two complementary approaches called blocking and filtering. *Blocking* partitions the datasets into multiple partitions or blocks such that links are only determined between resources of the same partition. There are several approaches with disjoint or overlapping partitions for this, e.g., standard blocking (based on a predefined blocking key) or canopy clustering [4,7]. Furthermore, multiple blocking keys may be applied to partition the input according to several criteria so that the likelihood of finding all links is improved. The blocking key is commonly based on attribute or property values, e.g., one could partition movies according to the first three letters of the movie title or according to last name of the movie director. Resources can also be partitioned based on their associated ontology concepts if both data sources have comparable concepts, e.g., the genre of movies. We will call such an approach *concept-based blocking*.

Filtering utilizes details of the linking configuration, such as the similarity measure or similarity threshold, to filter pairs of records that cannot meet the similarity condition. For example, token-based string similarity measures such as the Jaccard or Dice similarity can only exceed a certain threshold if the input strings are of similar length and share a certain number of tokens [5]. Preprocessing can support the efficient execution of such filters in the match phase, e.g., by creating a token index.

Blocking and filtering can jointly be applied, e.g., to reduce the number of comparisons for partition-wise linking. Furthermore, both approaches can be utilized in combination with parallel LD [24,33].

3.3. Match approaches

The main phase of the LD workflow applies the linking specification and evaluates the specified similarity measures on the pairs of resources that still need to be considered according to the used blocking or filter methods. An LD tool typically has a library of different match techniques (or matchers) that apply a similarity measure on the resources to link with each other. These matchers have been categorized as either element- or structure-based [45,11] depending on whether they evaluate simple resource elements such as atomic property values (literals) or whether they consider the context of resources (e.g., related instances or the ontological context), Element-level matchers are most common and can be based on similarity measures for strings (n-gram, TF/IDF, edit distance, etc.) [6], numbers or domain-specific data types such as geographical coordinates. They are typically applied on matching of comparable properties of resources that have been specified as part of the linking specification (either manually or automatically). Similarity computation may also utilize different kinds of background knowledge such as general-purpose or domain-specific dictionaries and thesauri.

Structure- or context-based matchers are more sophisticated and aim at deriving the similarity of resources from the similarity of their context. There is a large spectrum of possible approaches depending on what context and which similarity computation is applied. For example, some approaches use so-called anchor links between highly similar resources as a seed to iteratively find matching entities in the sets of their related entities [21,18]. The search for matches can also be confined to instances of equivalent or related classes thereby utilizing the ontological context.

A promising LD approach is to utilize already existing links and mappings to find new links. Based on the transitivity of the equality relation one can compose several `owl:sameAs` links to derive new `owl:sameAs` links. Effective strategies for such a composition of mappings and links have been proposed and evaluated in [16]. Public mapping repositories such as BioPortal [43] or LinkLion [27] support the publication of links and thus their reuse for determining new links.

3.4. Postprocessing

In the final phase the results of the matchers need to be combined and the links need to be selected from the set of candidate links according to the linking specification, e.g., by applying a match rule or a learned classification model. The resulting links may be further refined or repaired to avoid inconsistencies, such as the violation of ontological or application-specific constraints. For example, one could request a 1:1 mapping so that each instance is linked with at most one instance of the other input dataset. Hence, postprocessing could enforce this restriction by selecting the best link per instance, e.g., with the highest computed confidence value. Human feedback is generally helpful during postprocessing to verify the correctness of computed links.

4. Functional Comparison

In this section, we provide a functional comparison of ten state-of-the-art frameworks for LD based on the requirements and the general LD workflow discussed in the previous sections. The selection of tools was furthermore driven by the following criteria:

- OAEI benchmark instance matching track participation and relatively good performance in the appropriate year or
- learning-based approach for LD and usage of OAEI instance matching track datasets.

Given this premise, Table 1 lists the considered frameworks with their originating organization, their first LD-related publication and further criteria that allows a rough grouping of the tools. Seven of the tools have participated in the instance matching contest of the OAEI. The remaining three frameworks (Silk, LIMES, KnoFuss) support among others learning-based approaches for determining linking specifications. A further criterion indicates that four of the seven tools of the first group have support for pure ontology matching in addition to instance matching. In fact, these frameworks (RiMOM, AgreementMaker, LogMap and CODI) mostly started with ontology matching and supported instance matching later. Due to the generality of the LD workflow and the given requirements the approach of the comparison can be easily used for other tools, too.

For the more detailed comparison of the tools we collect the main features in Tables 2 and 3 for the mentioned two groups of 7+3 systems. The considered criteria belong to the following categories largely following the steps of the introduced LD workflow:

- Supported input formats
- Configuration approach
- Runtime optimizations
- Match approaches
- Postprocessing
- Support for parallel processing
- User interface (GUI support) and interaction
- General availability

In the following subsections we will discuss these aspects for the different frameworks. Finally we will summarize our observations from the functional comparison.

4.1. Data input

Eight of the ten tools accept the input datasets in RDF file format while two frameworks (AgreementMaker, SERIMI) need to retrieve the data from SPARQL endpoints. While SPARQL endpoints support a flexible and dynamic data access they can cause availability and performance problems. In addition to RDF, CODI, LogMap and RiMOM additionally support OWL input files. Access to SPARQL endpoints is also supported by the learning-based tools Silk, LIMES and KnoFuss. Dynamic data access with SPARQL typically uses a restriction to certain classes (e.g., books, settlements) thereby limiting the data volume and search space for finding links.

Surprisingly, a large number of the considered frameworks does not seem to rely on external background knowledge such as dictionaries or already known links and mappings. This statement obviously does not hold for the framworks that rely on any form of supervised machine learning. This in strong contrast to ontology matching where virtually all current tools utilize dictionaries such as WordNet as background knowledge [44]. The tools RiMOM, AgreementMaker and LogMap also utilize such dictionaries for their ontology matching but apparently not for linking instance data.

4.2. Configuration

Most frameworks can only determine `owl:sameAs` links or equivalent instances. LIMES and Silk also support additional link types which need to be manually specified by the tool user.

System / initial publication	Institution	Learning-based	OAEI IM participation	Support for pure ontology matching
RiMOM [51]	Univ. of Tsinghua, China		✓	✓
KnoFuss [39]	Open Univ. Milton Keynes, UK	✓		
AgreementMaker [8]	Univ. of Illinois at Chicago, USA		✓	✓
Silk [54]	Univ. of Mannheim, Germany	✓		
CODI [37]	Univ. of Mannheim, Germany		✓	✓
LIMES [32]	Univ. of Leipzig, Germany	✓		
LogMap [21]	Univ. of Oxford, UK		✓	✓
SERIMI [3]	Delft Univ. of Techn., Netherlands		✓	
Zhishi.links [40]	Shanghai Jiao Tong Univ., China		✓	
SLINT+ [35]	Nat. Inst. of Informatics, Japan		✓	

Table 1
Considered LD tools (sorted by year of initial publication)

Four frameworks rely on a purely manually specified linking configuration (CODI, LogMap, AgreementMaker, Zhishi.links). In case of several matchers the resulting similarity values are combined according to a weighted average approach or a match rule. The learning-based tools KnoFuss, Silk, and LIMES also support manually specified match rules. Three tools (RiMOM, SERIMI, SLINT+) already follow a semi-automatic, adaptive linking specification by analyzing the datasets and identifying the most discriminating properties. For example, if publications have to be matched, the title will be more discriminating than the venue of the publication. SERIMI is limited to only a single property to be selected for matching. Further parameters such as similarity thresholds have to be manually specified.

Silk and LIMES both support supervised learning of a linking specification with genetic programming either with batch learning or active learning [19,28]. Genetic programming starts from a set of random link specifications and uses the evolutionary principles of selection and variation to evolve these specifications until a linking condition meets a predefined optimization criterion (fitness function) or a maximal number of iterations is reached. For supervised learning, manually labeled link candidates are used within the genetic algorithm to find link specifications that come close to the match decisions for the training data. Active learning aims at reducing the labeling effort for training data and applies an interactive labeling of automatically chosen link candidates [19]. Link candidates for active learning are selected to optimize criteria such as entropy or the similarity correlation to unlabeled instances [34].

KnoFuss and LIMES also implement an unsupervised learning of the linking specification [38,29]. The approaches also utilize genetic programming but try to iteratively optimize measures that evaluate indirect quality criteria such as high similarity values and closeness to a 1:1 mapping (assuming duplicate-free data sources) [38,30,29]. In KnoFuss, the candidate linking specifications aggregate the weighted similarity values for several string matchers and require the aggregated similarity value to exceed a certain threshold. The approach thus has to select the matchers, determine their weights, the aggregation function (e.g., average or max) and the similarity threshold.

4.3. Runtime optimization

Silk is the only framework implementing an explicit *blocking* to reduce the search space. They support the (manual) specification of multiple blocking keys, i.e., only instances sharing one of the blocking keys must be compared with each other. A multidimensional index is applied to implement this strategy [20]. An implicit blocking is achieved by preselecting in the input specification the classes to be processed but this does not allow to a-priori reduce the search space for the instances of a class which may be numerous.

	RiMOM	AgreementMaker	CODI	LogMap	SERIMI	Zhishi.links	SLINT+
Data Input	RDF, OWL	SPARQL	RDF, OWL	RDF, OWL	SPARQL	RDF	RDF
Supported linktypes	`owl:sameAs`	`owl:sameAs`	`owl:sameAs`	`owl:sameAs`	`owl:sameAs`	`owl:sameAs`	`owl:sameAs`
Configuration - matcher combination	adaptive weighted average	manual weighted combination	manual weighted average	manual weighted average	adaptive -	manual weighted combination	adaptive weighted average
Runtime optimization - Blocking - Filtering	 - indexing	 - indexing	 - -	 - indexing	 - -	 - indexing	 - indexing
String similarity measures	✓	✓	✓	✓	✓	✓	✓
Further similiarty measures	-	-	-	-	-	geographical coordinates	inverted disparity
Structure matcher	-	semantic similarity	iterative anchor-based mapping generation	iterative anchor-based mapping generation	-	semantic similarity	-
Use of - external dictionaries - existing mappings	 ?* -	 ?* -	 - -	 ?* -	 - -	 - -	 - -
Post-processing	-	-	Coherence checks	Inconsistency repair	-	-	-
Parallel processing	-	-	-	-	-	MapReduce	-
GUI/web interface	- / -	✓/ ?	- / -	✓/ ✓	- / -	- / ?	- / -
Download Tool/Source Open Source project	✓/ - -	-[1] / - -	✓/ ✓ ✓	✓/ ✓ ✓	✓/ ✓ ✓	✓/ - -	✓/ - -

Table 2

Characteristics of proposed LD frameworks ("-" means not existing, "?" unclear from publication, "*" supported in respectively ontology matching framework, [1] no answer on form submission)

	KnoFuss	Silk	LIMES
Data Input	RDF, SPARQL	RDF, SPARQL, CSV	RDF, SPARQL, CSV
Supported linktypes	`owl:sameAs`	`owl:sameAs`, user-specified others	`owl:sameAs`, user-specified others
Configuration	manual(match rules), unsupervised learning (genetic programming)	manual (match rules), supervised learning (genetic programming, active learning)	manual (match rules), supervised learning (genetic programming, active learning), unsupervised (genetic programming)
Runtime optimization - Blocking - Filtering	 - indexing	 multi-dimensional -	 - space tiling
String similarity measures	✓	✓	✓
Further similiarty measures	-	numeric, date equality	geographical coordinates, numeric, date equality
Structure matcher	-	-	-
Use of - external dictionaries - existing mappings	 - -	 - -	 - -
Post-processing	one-to-one mapping	-	Stable marriage, hospital-resident
Parallel Processing	-	MapReduce	(MapReduce)*
GUI/web interface	- / -	✓ / ✓	✓ / ✓
Download Tool/Source Open Source project	✓ / ✓ ✓	✓ / ✓ ✓	✓ / - -

Table 3
Characteristics of learning-based LD frameworks ("-" means not existing, "*" not in current release)

The main approach to improve runtime in the considered tools is filtering, especially by utilizing inverted index structures. This optimization focuses mostly on a specific property and similarity measure (matcher). For example token-based string similarity measures such as Jaccard require matching values to share several tokens. Hence all pairs without a common token can be excluded from the comparison. An inverted index allows to quickly determine the instances that still must be considered.

4.4. Matching strategies

All tools support element-level matchers on selected properties based on string similarity measures such as edit distance, n-gram, or Jaccard [6]. Only few tools (Zhishi.links, Silk, LIMES) also support built-in numerical similarity measures (e.g., Euclidean distance) or domain-specific measures such as for geographical coordinates. Except SERIMI, all frameworks can match on more than one property [2]. The similarity values of different matchers are combined according to

the linking specification (match rule, weighted average or according to a learned linking specification).

In addition to simple matching on property values four frameworks (CODI, LogMap, AgreementMaker, Zhishi.links) already apply a structural matching based on the ontology structure to find links. LogMap and CODI apply an iterative anchor-based matching approach. Within the instances of comparable concepts so-called anchor links are determined first between almost identical instances. Both LogMap and CODI then use information from the ontology to iteratively extend the existing mapping by evaluating the similarity of related instances, either utilizing object-property-assertions [18] or logical reasoning [23]. In LogMap the similarity computation is performed by an algorithm called ISUB [50] that combines three different metrics. CODI simply employs a threshold-based edit distance [41].

The structural matching in AgreementMaker is based on its approach used for ontology matching. Zhishi.links applies a two-step matching approach. Initially it determines property-based similarities. The results are filtered via a threshold and the similarities are then semantically refined based on the similarity of related resources in the ontological context [40].

4.5. Postprocessing

The main task of postprocessing is to select the links according to the linking specification, e.g., by applying a match rule taking into account the computed similarity values. Additional verification steps are applied by LogMap and CODI to avoid that inconsistent mappings are determined. These tools also support pure ontology matching where such postprocessing steps are quite common. Specifically, LogMap applies logical reasoning [22] and CODI utilizes logical coherence checks to identify links contradicting ontological restrictions [42]. Furthermore, KnoFuss and LIMES employ postprocessing strategies to ensure that every instance in the source can only have at most one corresponding instance in the target dataset [38,28].

4.6. Support for parallel LD

For high efficiency and scalability, support for parallel LD is beneficial. In addition to utilizing multiple processors of a single node parallel LD may also use several nodes in a cluster, e.g., running Apache Hadoop with MapReduce. Three of the ten frameworks already support a MapReduce implementation: LIMES [17], Zhishi.links [40] and Silk[3]. Another promising option is to use parallel processing on massively parallel graphic processors (GPUs) as already explored in [33]. While these optimizations have already been studied in the context of the mentioned tools they are not an integral part of the available tool versions as they require a specific infrastructure (Hadoop cluster or GPU).

4.7. User interface and interaction

User interfaces for the ten frameworks range from simple command line interfaces (with diverging sets of options) over stand-alone installations to web applications. Only four tools (LogMap, AgreementMaker, LIMES, Silk) support a GUI for convenient interactive use (Tables 2 and 3).

4.8. Availability for other researchers

As seen in Tables 2 and 3 all tools (except AgreementMaker) are publicly available; five tools even follow an Open Source strategy.

4.9. Observations

The considered tools provide a very good general availability providing a rich choice for interested users and researchers. Most tools already support advanced methods for semi-automatic configuration of linking specifications, in three cases based on learning approaches such as genetic programming. Efficiency is mainly addressed by filtering techniques for specific matchers rather than by more general blocking approaches to reduce the search space. Parallel processing has been investigated by some of the tools but is not generally available in the offered tool versions. Most tools only support rather simple property-based matchers; the more advanced structural match techniques are available in four tools. The high potential of utilizing already existing links and mappings as well as other data sources or dictionaries as background knowledge has not yet been exploited. GUI support is also limited to few tools. The three learning-based tools KnoFuss, Silk and LIMES provide the most options for linking configuration and runtime optimization; Silk and LIMES also provide a GUI and are not limited to finding `owl:sameAs` links only.

[3] `https://www.assembla.com/spaces/silk/wiki/Silk_MapReduce`

5. Comparison of evaluation results

In this section, we analyze the published evaluation results for the considered frameworks. Special emphasis is given to results for the Ontology Evaluation Alignment Initiative (OAEI)[4] in the instance matching track aiming on an evaluation of different systems under the same conditions.

Similarly to previous evaluation studies on entity resolution [7,25] we consider the following criteria:

- Format of input data (RDF, OWL, etc.).
- Determined link types,
- Real vs. artificial (synthetic) datasets: artificial datasets are typically created by systematically changing real instances to create create similar (matching) instances to identify by the evaluated approaches. This supports the generation of large datasets for scalability experiments.
- Considered data sources and domains.
- Effectiveness: achieved linking quality in terms of precision, recall and F-measure w.r.t. a perfect linking result (gold standard).
- Efficiency: runtime results and scalability to large data volumes.

In the following we first describe the results for OAEI instance matching benchmarks which provide the best possible comparability for the different tools so far. Afterwards we briefly discuss observations from additional evaluations and summarize the main findings.

5.1. OAEI benchmark tests

The Ontology Evaluation Alignment Initiative (OAEI) performs yearly contests since 2005 to comparatively evaluate current tools for ontology and instance matching. The original focus has been on ontology matching but since 2009 instance matching has also been a regular evaluation track. As already discussed in the previous section, seven of the ten tools have already participated in this track. Even the three other learning-based frameworks used some of the OAEI test cases for their evaluations. Despite this situation, the analysis of the results for the OAEI benchmark is made complicated because the tasks and the participating systems change every year.

Table 4 gives an overview over the OAEI instance matching tasks in five contests from 2010 until 2014. Most tasks have only been used in one year while others like IIMB have been changed in different years. Most tests are based on artificially changed datasets where values and the structural context of instances have been modified in a controlled way. The tests cover different domains (life sciences, people, geography, etc.) and LOD data sources (DBpedia, Freebase, GeoNames, NYTimes, etc.). Frequently the benchmarks consist of several match (linking) tasks to cover a certain spectrum of complexity. The number of instances is rather small in all tests with a maximal size of a data source of 9,958 or fewer instances. The evaluation focus has been solely on the effectiveness (e.g., F-Measure) while runtime efficiency has not been measured. Almost all tasks focus on identifying equivalent instances (`owl:sameAs` links).

We briefly characterize the different OAEI tasks as follows.

IIMB and Sandbox (SB) The IIMB benchmark has been part of the 2010, 2011 and 2012 contests and consists of 80 test cases using synthetically modified datasets derived from instances of 29 Freebase concepts. The tests and number of instances vary from year to year but the tests are generally of a very small size (e.g., at most 375 instances in 2012). The Sandbox (SB) benchmark from 2012 is very similar to IIMB but limited to 10 different test cases [1].

PR (Persons/Restaurant) This benchmark is based on real person and restaurant instance data which are artificially modified by adding duplicates and variations of property values. The dataset is relatively small with about 500-600 instances in the restaurant data source and even less in the person data source. [12]

DI-NYT (Data Interlinking - NYT) This 2011 benchmark includes seven tasks to link about 10,000 instances from the NYT data source to DBpedia, Freebase and GeoNames instances. The perfect match result contains about 31,000 `owl:sameAs` links to be identified [10].

RDFT This 2013 benchmark is also of small size (430 instances) and uses several tests with differently modified DBpedia data. For the first time in the OAEI instance matching track, no reference mapping is provided for the actual evaluation task. Instead, training data with an appropriate reference mapping is given for each test case thereby supporting frameworks relying on supervised learning [15].

[4] `http://www.ontologymatching.org`

	Name	Input Format	Type of problem	Domains	LOD Sources	Link Type	Max. # Resources	Tasks
2010	DI	RDF	real	life sciences	diseasome drugbank dailymed sider	equality	5,000	4
	IIMB	OWL	artificial	cross-domain	Freebase	equality	1,416	80
	PR	RDF, OWL	artificial	people geography	-	equality	864	3
2011	DI-NYT	RDF	real	people geography organizations	NYTimes DBpedia Freebase Geonames	equality	9,958	7
	IIMB	OWL	artificial	cross-domain	Freebase	equality	1,500	80
2012	SB	OWL	artificial	cross-domain	Freebase	equality	375	10
	IIMB	OWL	artificial	cross-domain	Freebase	equality	375	80
2013	RDFT	RDF	artificial	people	DBpedia	equality	430	5
2014	id-rec	OWL	artificial	publications	?	equality	2,649	1
	sim-rec	OWL	artificial	publications	?	similarity	173	1

Table 4
OAEI instance matching tasks over the years ("-" means not existing, "?" unclear from publication)

OAEI 2014 Two benchmark tasks have to be performed in 2014, the first one (id-rec) requiring the identification of the same real-world book entities (sameAs links). For this purpose, 1,330 book instances have to be matched with with 2,649 synthetically modifies instances in the target dataset. Data transformations include changes like the substitution of book titles and labels with keywords as well as language transformations. The second task (sim-rec) requires determining the similarity of pairs of instances which do not reflect the same real-world entities. This addresses common preprocessing tasks, e.g., to reduce the search space for LD. In 2014, the central evaluation platform SEALS [14] is used for instance matching, too. Therefore, a runtime evaluation can be conducted for participating tools. The sim-rec task is not further evaluated in this paper.

5.2. Evaluation results of OAEI tasks

Table 5 shows the participation of the considered tools in the different OAEI contests and benchmarks. Overall, many tools participated only once or twice (AgreementMaker, SERIMI, Zhishi.links, SLINT+) and several benchmarks have only been evaluated by one or two systems (IIMB 2010, 2011 and 2012, SB, DI 2010, id-rec 2014). The learning-based tools have used the PR and DI-NYT benchmarks but not within the contest so that a direct comparability is not given. This is because outside the contest tools could apply a

more intensive tuning and utilize additional information such as training data. Our comparison will thus focus on the benchmarks with most participants: PR, DI-NYT and RDFT.

Table 6 shows the reported F-Measure results for the PR benchmark tasks for matching people and restaurant records. The original reference mapping proved to be erroneous so that it was corrected after the OAEI contest making it difficult to compare the achieved results. Within the contest the RiMOM system could clearly outperform the CODI system. The evaluations outside the competition used the corrected reference mapping and show especially good results for KnoFuss. In general, the small size of the linking problems and the achievable F-Measure of 0.98-1.0 indicate hat the benchmark tasks are easy to solve.

The F-Measure results for the DI-NYT benchmark in Table 7) indicate a more diverse situation. From the three frameworks participating in the contest, Zhishi-links achieved the best results with consistent F-Measure values between 0.87 and 0.97 for the seven tasks. By contrast, AgreementMaker and SERIMI performed somewhat worse due to problems for one or two of the tasks. The results reported for the three systems that did not participate in the contest are generally better. The achievable F-measure results for all tasks are between 0.93 and 0.99 indicating that these tasks are also relatively easy to solve.

F-Measure results for RDFT benchmark from the OAEI 2013 contest are summarized in Table 8. Again, the different tasks could be solved to a large degree with maximal F-Measure values between 0.96 and 1.0. The overall best results are achieved by RiMOM followed by SLINT+ and LogMap. The 2014 id-rec task turned out to be much more challenging. From the two participants, RiMOM again outperformed LogMap with a F-Measure result of only 0.56 vs. 0.10.

FMeas.	LogMap	RiMOM2013	SLINT+
test01	0.80	**1.00**	0.98
test02	0.88	0.97	**1.00**
test03	0.84	**0.98**	0.92
test04	0.80	**0.96**	0.91
test05	0.74	**0.96**	0.88

Table 8
F-measure results for test cases of OAEI 2013 benchmark RDFT.

In summary, most of the OAEI instance benchmarks so far have been of small size and relatively easy to solve or attracted only few frameworks participating in the contest. RiMOM could outperform competing systems in three different benchmarks. The frameworks using OAEI benchmarks outside the contest achieved generally very good results that unfortunately are not directly comparable with the results for the frameworks participating in the OAEI contests. Runtime values and thus scalability have not yet been evaluated for OAEI instance matching.

5.3. Other evaluations

The learning-based frameworks KnoFuss, Silk and LIMES did not yet participate in the OAEI contest but evaluated their effectiveness and runtime efficiency with several evaluations using diverse data sources including DBpedia, DrugBank, GeoNames, DBLP and LinkedMDB. Unfortunately the studies typically used different test cases with specific configurations so that the results can hardly be compared with each other. For example, Silk [19] and LIMES [28] both evaluate a LinkedMDB-DBpedia dataset but use varying numbers of resources. Similarly, reported execution times strongly depend on the used hardware configuration so that they mainly serve to show the relative performance of the respective system w.r.t. different data sizes and other configuration parameters.

For genetic programming algorithms, efficiency largely depends on the number of needed iterations. As an example, Silk needed 2,558.8s for 25 iterations to link DrugBank with DBpedia but already 21,387.5s for 50 iterations [19]. The genetic algorithm also faces a quadratic complexity of the selection phase w.r.t. the data volume. Hence, random sampling is applied to reduce the number of possible candidates for the generation of the next population. Again, runtime and quality of the results compete with each other as shown in [38] where bigger sampling sizes help to achieve a good F-measure at the expense of increased execution times.

Instance-based linking is similar to entity resolution and the comparative evaluation of entity resolution frameworks faces similar challenges than the evaluation of LD frameworks. The study [26] evaluated several entity resolution tools on several real datasets on publications and product offers of e-commerce websites. While the publication-related match tasks were relatively easy to solve, the two e-commerce match tasks turned out to be especially challenging with a maximal F-Measure of only 60 and 71 % for the considered tools. These match tasks have also been used to

Task	AgreementMaker	SERIMI	CODI	Zhishi.links	LogMap	RiMOM	SLINT+	LIMES	Silk	KnoFuss
2010 PR			✓			✓		✓*	✓*	✓*
2010 IIMB			✓			✓				
2010 DI						✓				
2011 IIMB			✓							
2011 DI-NYT	✓	✓		✓			✓*		✓*	✓*
2012 SB					✓					
2012 IIMB					✓					
2013 RDFT					✓	✓	✓			
2014 id-rec					✓	✓				

Table 5

Tool participation in OAEI instance matching tracks over the years. "*" did not participate in OAEI contest

	RiMOM	CODI	KnoFuss *	Silk *	LIMES (unsupervised) *
Person1	1.00	0.91	1.00	-	1.00
Person2	0.97	0.36	0.99	-	0.94
Restaurant (OAEI)	0.81	0.72	0.78	-	-
Restaurant (fixed)	-	-	0.98	0.99	0.82

Table 6

F-Measure results of the OAEI 2010 benchmark PR (Person/Restaurant). "*" result was achieved outside the OAEI contest.

	AgrMaker	SERIMI	Zhishi.links	KnoFuss *	Silk *	Slint+ *
nyt-dbpedia-loc.	0.69	0.68	0.92	0.89	0.93	0.97
nyt-dbpedia-org.	0.74	0.88	0.91	0.92	-	0.95
nyt-dbpedia-peo.	0.88	0.94	0.97	0.97	-	0.99
nyt-freebase-loc.	0.85	0.91	0.88	0.93	-	0.95
nyt-freebase-org.	0.80	0.91	0.87	0.92	-	0.96
nyt-freebase-peo.	0.96	0.92	0.93	0.95	-	0.99
nyt-geonames	0.85	0.80	0.91	0.90	-	0.99
H-mean	0.82	0.85	0.91	0.93	-	0.97

Table 7

F-Measure results for OAEI 2011 benchmark DI-NYT [10]. H-mean is calculated manually from the single F-measure values of the appropriate publication. "*" result was achieved outside the OAEI contest.

evaluate further tools including LD frameworks such as LIMES, e.g., in [34,28]). Results in [34] confirm the difficulty of the e-commerce match tasks with achieved F-Measure values ranging below 35%.

5.4. Observations

Despite the laudable effort of the OAEI instance matching tracks the comparable evaluation of existing tools for LD is still a largely open challenge. This is mainly because the participation in the OAEI contest has been limited so far and using the OAEI tasks outside the competition limits the comparability of the achieved results as they are typically based on different prerequisites, e.g., the use of training data. Evaluation results on a single system or approach aim at showing their effectiveness and efficiency rather than provid-

ing a neutral comparative evaluation between systems. Given the general availability of LD tools it would be a worthwhile investigation to apply them under the same prerequisites on a set of LD tasks similar than in the entity resolution study [26]. Such a study could evaluate not only the effectiveness and efficiency but also the usability of tools.

6. Conclusion

We investigated ten LD frameworks and compared their functionality based on a common set of criteria. The criteria cover the main steps such as the configuration of linking specifications and methods for matching and runtime optimization. We also covered general aspects such as the supported input formats and link types, support for a GUI and software availability as open source. We observed that the considered tools already provide a rich functionality with support for semi-automatic configuration including advanced learning-based approaches such as unsupervised genetic programming or active learning. On the other side, we found that most tools still focus on simple property-based match techniques rather than using the ontological context within structural matchers. Furthermore, existing links and background knowledge are not yet exploited in the considered frameworks. More comprehensive support of efficiency techniques is also necessary such as the combined use of blocking, filtering and parallel processing.

We also analyzed comparative evaluations of the LD frameworks to assess their relative effectiveness and efficiency. In this respect the OAEI instance matching track is the most relevant effort and we thus analyzed its match tasks and the tool participation and results for the last years. Unfortunately, the participation has been rather low thereby preventing the comparative evaluation between most of the tools. Moreover, the focus of the contest has been on effectiveness so far while runtime efficiency has not yet been evaluated. To better assess the relative effectiveness and efficiency of LD tools it would be valuable to test them on a common set of benchmark tasks on the same hardware. Given the general availability of the tools and the existence of a considerable set of match task definitions and datasets this should be feasible with reasonable effort.

References

[1] José Luis Aguirre, Bernardo Cuenca Grau, Kai Eckert, Jérôme Euzenat, Alfio Ferrara, Robert Willem van Hague, Laura Hollink, Ernesto Jimenez-Ruiz, Christian Meilicke, Andriy Nikolov, et al. Results of the Ontology Alignment Evaluation Initiative 2012. In *Proc. 7th ISWC workshop on ontology matching (OM)*, pages 73–115, 2012.

[2] Samur Araujo, Arjen de Vries, and Daniel Schwabe. SERIMI Results for OAEI 2011. In *Proc. of the 6th International Workshop on Ontology Matching*, page 212, 2011.

[3] Samur Araújo, Jan Hidders, Daniel Schwabe, and Arjen P. de Vries. SERIMI - Resource Description Similarity, RDF Instance Matching and Interlinking. In *Proc. of the 6th International Workshop on Ontology Matching*, 2011.

[4] Rohan Baxter, Peter Christen, and Tim Churches. A Comparison of Fast Blocking Methods for Record Linkage. In *ACM SIGKDD*, volume 3, pages 25–27. Citeseer, 2003.

[5] Roberto J. Bayardo, Yiming Ma, and Ramakrishnan Srikant. Scaling Up All Pairs Similarity Search. In *Proceedings of the 16th International Conference on World Wide Web*, pages 131–140, 2007.

[6] Michelle Cheatham and Pascal Hitzler. String Similarity Metrics for Ontology Alignment. In Harith Alani, Lalana Kagal, Achille Fokoue, Paul Groth, Chris Biemann, JosianeXavier Parreira, Lora Aroyo, Natasha Noy, Chris Welty, and Krzysztof Janowicz, editors, *The Semantic Web – ISWC 2013*, volume 8219 of *Lecture Notes in Computer Science*, pages 294–309. Springer Berlin Heidelberg, 2013.

[7] Peter Christen. *Data Matching - Concepts and Techniques for Record Linkage, Entity Resolution, and Duplicate Detection*. Data-centric systems and applications. Springer, 2012.

[8] Isabel F Cruz, Flavio Palandri Antonelli, and Cosmin Stroe. AgreementMaker: efficient matching for large real-world schemas and ontologies. *Proceedings of the VLDB Endowment*, 2(2):1586–1589, 2009.

[9] Ahmed K. Elmagarmid, Panagiotis G. Ipeirotis, and Vassilios S. Verykios. Duplicate Record Detection: A Survey. *IEEE Transactions on Knowledge and Data Engineering*, 19(1):1–16, 2007.

[10] Jérôme Euzenat, Alfio Ferrara, Willem Robert van Hague, Laura Hollink, Christian Meilicke, Andriy Nikolov, François Scharffe, Pavel Shvaiko, Heiner Stuckenschmidt, Ondrej Sváb-Zamazal, et al. Final results of the Ontology Alignment Evaluation Initiative 2011. In *Proc. 6th ISWC workshop on ontology matching (OM)*, pages 85–110, 2011.

[11] Jérôme Euzenat, Pavel Shvaiko, et al. *Ontology matching*. Springer, 2007.

[12] Jérôme Euzenat, Alfio Ferrara, Christian Meilicke, Andriy Nikolov, Juan Pane, François Scharffe, Pavel Shvaiko, Heiner Stuckenschmidt, Ondrej Sváb-Zamazal, Vojtech Svátek, and Cássia Trojahn dos Santos. Results of the Ontology Alignment Evaluation Initiative 2010. In Pavel Shvaiko, Jérôme Euzenat, Fausto Giunchiglia, Heiner Stuckenschmidt, Ming Mao, and Isabel Cruz, editors, *Proc. 5th ISWC workshop on ontology matching (OM), Shanghai (CN)*, pages 85–117, 2010.

[13] Alfio Ferrara, Andriy Nikolov, and François Scharffe. Data linking for the semantic web. *International Journal on Semantic Web and Information Systems (IJSWIS)*, 7(3):46–76, 2011.

[14] Raúl García-Castro and Stuart N. Wrigley. SEALS Methodology for Evaluation Campaigns. Technical report, Seventh Framework Programme, 2011.

[15] Bernardo Cuenca Grau, Zlatan Dragisic, Kai Eckert, Jérôme Euzenat, Alfio Ferrara, Roger Granada, Valentina Ivanova, Ernesto Jiménez-Ruiz, Andreas Oskar Kempf, Patrick Lam-

brix, et al. Results of the Ontology Alignment Evaluation Initiative 2013. In *Proc. 8th ISWC workshop on ontology matching (OM)*, pages 61–100, 2013.

[16] Michael Hartung, Anika Groß, and Erhard Rahm. Composition Methods for Link Discovery. In *BTW*, 2013.

[17] Stanley Hillner and Axel-Cyrille Ngonga Ngomo. Parallelizing LIMES for large-scale link discovery. In *Proceedings of the 7th International Conference on Semantic Systems*, I-Semantics '11, pages 9–16, New York, NY, USA, 2011. ACM.

[18] Jakob Huber, Timo Sztyler, Jan Noessner, and Christian Meilicke. CODI: Combinatorial optimization for data integration - results for OAEI 2011. In *Proc. of the 6th International Workshop on Ontology Matching*, page 134, 2011.

[19] Robert Isele and Christian Bizer. Active Learning of Expressive Linkage Rules using Genetic Programming. *Journal of Web Semantics*, 2013.

[20] Robert Isele, Anja Jentzsch, and Christian Bizer. Efficient Multidimensional Blocking for Link Discovery without losing Recall. In *Fourteenth International Workshop on theWeb and Databases (WebDB 2011)*, 2011.

[21] Ernesto Jiménez-Ruiz and Bernardo Cuenca Grau. LogMap: Logic-Based and Scalable Ontology Matching. In *The Semantic Web–ISWC 2011*, pages 273–288. Springer, 2011.

[22] Ernesto Jiménez-Ruiz, Bernardo Cuenca Grau, and Ian Horrocks. LogMap and LogMapLt results for OAEI 2013. *Ontology Matching*, 2013.

[23] Ernesto Jiménez-Ruiz, Bernardo Cuenca Grau, Yujiao Zhou, and Ian Horrocks. Large-scale Interactive Ontology Matching: Algorithms and Implementation. In *ECAI*, pages 444–449, 2012.

[24] Lars Kolb, Andreas Thor, and Erhard Rahm. Dedoop: Efficient Deduplication with Hadoop. *Proceedings of the VLDB Endowment*, 5(12):1878–1881, 2012.

[25] Hanna Köpcke and Erhard Rahm. Frameworks for entity matching: A comparison. *Data & Knowledge Engineering*, 69(2):197 – 210, 2010.

[26] Hanna Köpcke, Andreas Thor, and Erhard Rahm. Evaluation of entity resolution approaches on real-world match problems. *Proc. VLDB Endow.t*, 3(1-2):484–493, September 2010.

[27] Markus Nentwig, Tommaso Soru, Axel-Cyrille Ngonga Ngomo, and Erhard Rahm. LinkLion: A Link Repository for the Web of Data. In *ESWC 2014 Posters & Demo session*, 2014.

[28] Axel-Cyrille Ngomo and Klaus Lyko. EAGLE: Efficient Active Learning of Link Specifications Using Genetic Programming. In *The Semantic Web: Research and Applications - 9th Extended Semantic Web Conference*, pages 149–163, 2012.

[29] Axel-Cyrille Ngonga Ngomo and Klaus Lyko. Unsupervised Learning of Link Specifications: Deterministic vs. Non-Deterministic. In *Proc. of the 8th International Workshop on Ontology Matching*, pages 25–36, 2013.

[30] Axel-Cyrille Ngonga Ngomo, Mohamed Ahmed Sherif, and Klaus Lyko. Unsupervised Link Discovery through Knowledge Base Repair. In *The Semantic Web: Trends and Challenges*, pages 380–394. Springer, 2014.

[31] Axel-Cyrille Ngonga Ngomo. HELIOS–Execution Optimization for Link Discovery. In *The Semantic Web - ISWC 2014 - 13th International Semantic Web Conference, Riva del Garda, Italy, October 19-23, 2014*, pages 17–32. Springer, 2014.

[32] Axel-Cyrille Ngonga Ngomo and Sören Auer. LIMES - A Time-Efficient Approach for Large-Scale Link Discovery on the Web of Data. In *Proceedings of the Twenty-Second international joint conference on Artificial Intelligence - Volume Volume Three*, IJCAI'11, pages 2312–2317. AAAI Press, 2011.

[33] Axel-Cyrille Ngonga Ngomo, Lars Kolb, Norman Heino, Michael Hartung, Sören Auer, and Erhard Rahm. When to Reach for the Cloud: Using Parallel Hardware for Link Discovery. In *The Semantic Web: Semantics and Big Data*, pages 275–289. Springer, 2013.

[34] Axel-Cyrille Ngonga Ngomo, Klaus Lyko, and Victor Christen. COALA–Correlation-Aware Active Learning of Link Specifications. In *The Semantic Web: Semantics and Big Data*, pages 442–456. Springer, 2013.

[35] Khai Nguyen, Ryutaro Ichise, and Bac Le. SLINT: A Schema-Independent Linked Data Interlinking System. In *Proc. of the 7th International Workshop on Ontology Matching*, 2012.

[36] Khai Nguyen, Ryutaro Ichise, and Bac Le. Interlinking Linked Data Sources Using a Domain-Independent System. In *Semantic Technology*, pages 113–128. Springer, 2013.

[37] M. Niepert, C. Meilicke, and H. Stuckenschmidt. A Probabilistic-Logical Framework for Ontology Matching. In *Proc. of the 24th AAAI Conference on Artificial Intelligence*, 2010.

[38] Andriy Nikolov, Mathieu d'Aquin, and Enrico Motta. Unsupervised learning of link discovery configuration. In *9th Extended Semantic Web Conference (ESWC 2012)*, 2012.

[39] Andriy Nikolov, Victoria Uren, and Enrico Motta. KnoFuss: A comprehensive architecture for knowledge fusion. In *Proceedings of the 4th international conference on Knowledge capture*, pages 185–186. ACM, 2007.

[40] Xing Niu, Shu Rong, Yunlong Zhang, and Haofen Wang. Zhishi.links results for OAEI 2011. In Pavel Shvaiko, Jérôme Euzenat, Tom Heath, Christoph Quix, Ming Mao, and Isabel F. Cruz, editors, *Proc. of the 6th International Workshop on Ontology Matching*, volume 814 of *CEUR Workshop Proceedings*, 2011.

[41] Jan Noessner and Mathias Niepert. CODI: combinatorial optimization for data integration: results for OAEI 2010. In *Proc. of the 5th International Workshop on Ontology Matching (OM-2010)*, page 142, 2010.

[42] Jan Noessner, Mathias Niepert, Christian Meilicke, and Heiner Stuckenschmidt. Leveraging Terminological Structure for Object Reconciliation. In *The Semantic Web: Research and Applications*, pages 334–348. Springer, 2010.

[43] Natalya F Noy, Nigam H Shah, Patricia L Whetzel, Benjamin Dai, Michael Dorf, Nicholas Griffith, Clement Jonquet, Daniel L Rubin, Margaret-Anne Storey, Christopher G Chute, et al. BioPortal: ontologies and integrated data resources at the click of a mouse. *Nucleic acids research*, 37:W170–W173, 2009.

[44] Erhard Rahm. Towards Large-Scale Schema and Ontology Matching. In Zohra Bellahsene, Angela Bonifati, and Erhard Rahm, editors, *Schema Matching and Mapping*, Data-Centric Systems and Applications, pages 3–27. Springer Berlin Heidelberg, 2011.

[45] Erhard Rahm and Philip A. Bernstein. A survey of approaches to automatic schema matching. *The VLDB Journal*, 10:334–350, 2001.

[46] Muhammad Saleem and Axel-Cyrille Ngonga Ngomo. HiBISCuS: Hypergraph-Based Source Selection for SPARQL Endpoint Federation. In *Proceedings of the 11th Extended Semantic Web nConference*, I-SEMANTICS '13. Springer, 2014.

[47] Muhammad Saleem, Shanmukha S. Padmanabhuni, Axel-Cyrille Ngonga Ngomo, Jonas S. Almeida, Stefan Decker, and Helena F. Deus. Linked Cancer Genome Atlas Database. In *Proceedings of the 9th International Conference on Semantic Systems*, I-SEMANTICS '13, pages 129–134, New York, NY, USA, 2013. ACM.

[48] Max Schmachtenberg, Christian Bizer, and Heiko Paulheim. Adoption of the linked data best practices in different topical domains. In *The Semantic Web–ISWC 2014*, pages 245–260. Springer, 2014.

[49] Saeedeh Shekarpour, Axel-Cyrille Ngonga Ngonga Ngomo, and Sören Auer. Question Answering on Interlinked Data. In *Proceedings of the 22Nd International Conference on World Wide Web*, WWW '13, pages 1145–1156, Republic and Canton of Geneva, Switzerland, 2013. International World Wide Web Conferences Steering Committee.

[50] Giorgos Stoilos, Giorgos Stamou, and Stefanos Kollias. A string metric for ontology alignment. In *The Semantic Web–ISWC 2005*, pages 624–637. Springer, 2005.

[51] Jie Tang, Bang-Yong Liang, Juanzi Li, and Kehong Wang. Risk Minimization Based Ontology Mapping. In Chi-Hung Chi and Kwok-Yan Lam, editors, *Content Computing*, volume 3309 of *Lecture Notes in Computer Science*, pages 469–480. Springer Berlin Heidelberg, 2004.

[52] Andreas Thor and Erhard Rahm. MOMA - A Mapping-based Object Matching System. In *CIDR*, pages 247–258, 2007.

[53] Christina Unger, Lorenz Bühmann, Jens Lehmann, Axel-Cyrille Ngonga Ngomo, Daniel Gerber, and Philipp Cimiano. Template-based question answering over RDF data. In *WWW*, pages 639–648, 2012.

[54] Julius Volz, Christian Bizer, Martin Gaedke, and Georgi Kobilarov. Discovering and Maintaining Links on the Web of Data. In *Proceedings of the 8th International Semantic Web Conference*, ISWC '09, pages 650–665, Berlin, Heidelberg, 2009. Springer-Verlag.

[55] Stephan Wölger, Katharina Siorpaes, Tobias Bürger, Elena Simperl, Stefan Thaler, and Christian Hofer. A Survey On Data Interlinking Methods. Technical report, STI Innsbruck, March 2011.

[56] Qian Zheng, Chao Shao, Juanzi Li, Zhichun Wang, and Linmei Hu. RiMOM2013 Results for OAEI 2013. In *Proceedings of the 8th International Workshop on Ontology Matching*, page 161, 2013.