Unsupervised Link Discovery Through Knowledge Base Repair

Axel-Cyrille Ngonga Ngomo, Mohamed Ahmed Sherif, and Klaus Lyko

Department of Computer Science
University of Leipzig
Augustusplatz 10, 04109 Leipzig
{ngonga|sherif|lyko}@informatik.uni-leipzig.de

Abstract. The Linked Data Web has developed into a compendium of partly very large datasets. Devising efficient approaches to compute links between these datasets is thus central to achieve the vision behind the Data Web. Unsupervised approaches to achieve this goal have emerged over the last few years. Yet, so far, none of these unsupervised approaches makes use of the replication of resources across several knowledge bases to improve the accuracy it achieves while linking. In this paper, we present COLIBRI, an iterative unsupervised approach for link discovery. COLIBRI allows the discovery of links between n datasets ($n \geq 2$) while improving the quality of the instance data in these datasets. To this end, COLIBRI combines error detection and correction with unsupervised link discovery. We evaluate our approach on five benchmark datasets with respect to the F-score it achieves. Our results suggest that COLIBRI can significantly improve the results of unsupervised machine-learning approaches for link discovery while correctly detecting erroneous resources.

1 Introduction

Over the last years, the Linked Open Data cloud has evolved from a mere 12 to more than 300 knowledge bases [1]. The basic architectural principles behind this data compendium are akin to those of the document Web and thus decentralized in nature.[1] This architectural choice has led to knowledge pertaining to the same domain being published by independent entities in the Linked Open Data cloud. For example, information on drugs can be found in Diseasome[2] as well as DBpedia[3] and Drugbank.[4] Moreover, certain datasets such as DBLP have been published by several bodies,[5] leading to duplicated content in the Data Web. With the growth of the number of independent data providers, the concurrent publication of datasets containing related information promises to become a phenomenon of increasing importance. Enabling the joint use of these datasets for tasks such as federated queries, cross-ontology question answering and data integration is most commonly tackled by creating links between the

[1] See http://www.w3.org/DesignIssues/LinkedData.html.
[2] http://wifo5-03.informatik.uni-mannheim.de/diseasome/
[3] http://dbpedia.org
[4] http://wifo5-03.informatik.uni-mannheim.de/drugbank/
[5] http://dblp.l3s.de/, http://datahub.io/dataset/fu-berlin-dblp and http://dblp.rkbexplorer.com/.

resources described in the datasets. Devising accurate link specifications (also called linkage rules [11]) to compute these links has yet been shown to be a difficult and time-consuming problem in previous works [11, 10, 19, 21].

The insight behind this work is that declarative link specifications (e.g., SILK and LIMES specifications) compare the property values of resources by using similarity functions to determine whether they should be linked. For example, imagine being given three knowledge bases K_1 that contains cities, K_2 that contains provinces and K_3 that contains countries as well as the `dbo:locatedIn` predicate[6] as relation. The specification that links K_1 to K_2 might compare province labels while the specifications that link K_1 and K_2 to K_3 might compare country labels. Imagine the city `Leipzig` in K_1 were linked to `Saxony` in K_2 and to `Germany` in K_3. In addition, imagine that `Saxony` were erroneously linked to `Prussia`. If we assume the first Linked Data principle (i.e., "Use URIs as names for things")[7], then the following holds: By virtue of the transitivity of `dbo:locatedIn` and of knowing that it is a many-to-1 relation,[8] we can deduce that one of the links in this constellation must be wrong. Note that this inference would hold both under open- and closed-world assumptions. Thus, if we knew the links between `Leipzig` and `Germany` as well as `Leipzig` and `Saxony` to be right, we could then repair the value of the properties of `Saxony` that led it to be linked to `Prussia` instead of `Germany` and therewith ensure that is linked correctly in subsequent link discovery processes.

We implement this intuition by presenting COLIBRI, a novel iterative and unsupervised approach for LD. COLIBRI uses link discovery results for *transitive many-to-1 relations* (e.g., `locatedIn` and `descendantSpeciesOf`) and *transitive 1-to-1 relations* (e.g., `owl:sameAs`) between instances in knowledge bases for the sake fo attempting to repair the instance knowledge in these knowledge bases and improve the overall quality of the links. In contrast to most of the current unsupervised LD approaches, COLIBRI takes an n-set[9] of set of resources $K_1, \ldots, K_n$ with $n \geq 2$ as input. In a *first step*, our approach applies an *unsupervised machine-learning* approach to each pair (K_i, K_j) of sets of resources (with $i \neq j$). By these means, COLIBRI generates $n(n-1)$ mappings. Current unsupervised approaches for LD would terminate after this step and would not make use of the information contained in some mappings to improve other mappings. The intuition behind COLIBRI is that using such information can help improve the overall accuracy of a link discovery process if the links are *many-to-1 and transitive* or *1-to-1 and transitive*. To implement this insight, all mappings resulting from the first step are forwarded to a *voting approach* in a *second step*. The goal of the voting approach is to detect possible errors within the mappings that were computed in the previous step (e.g., missing links). This information is subsequently used in the *third step* of COLIBRI, which is the *repair step*. Here, COLIBRI first detects the sources of errors in the mappings. These sources of errors can be wrong or missing property values

[6] The prefix `dbo:` stands for `http://dbpedia.org/ontology/`.

[7] http://www.w3.org/DesignIssues/LinkedData.html

[8] From this characteristic, we can infer that (1) a city cannot be located in two different provinces, (2) a city cannot be located in two different countries and (3) a province cannot be located in two different countries.

[9] An n-set is a set of magnitude n.

of the instances. Once these sources of errors have been eliminated, a new iteration is started. COLIBRI iterates until a termination condition (e.g., a fixpoint of its objective function) is met.

Overall, the main contributions of this work are as follows:

– We present the (to the best of our knowledge) the first unsupervised LD approach that attempts to repair instance data for improving the link discovery process.
– Our approach is the first unsupervised LD approach that can be applied to $n \geq 2$ knowledge bases and which makes use of the intrinsic topology of the Web of Data.
– We evaluate our approach on six datasets. Our evaluation shows that we can improve the results of state-of-the-art approaches w.r.t. the F-measure while reliably detecting and correcting errors in instance data.

We rely on EUCLID [18] as machine-learning approach and thus provide a fully deterministic approach. We chose EUCLID because it performs as well as non-deterministic approaches on the datasets used in our evaluation [18] while presenting the obvious advantage of always returning the same result for a given input and a given setting. Moreover, it is not tuned towards discovery exclusively `owl:sameAs` links [23]. Still, COLIBRI is independent of EUCLID and can be combined with any link specification learning approach. The approaches presented herein were implemented in LIMES.[10]

2 Preliminaries

In this section, we present some of the notation and concepts necessary to understand the rest of the paper. We use Figure 1 to exemplify our notation. The formalization of LD provided below is an extension of the formalization for 2 input knowledge bases presented in [17]. Given n knowledge bases $K_1 \ldots K_n$, LD aims to discover pairs $(s_i, s_j) \in K_i \times K_j$ that are such that a given relation $\mathcal{R}$ holds between s_i and s_j. The direct computation of the pairs for which $\mathcal{R}$ holds is commonly very tedious if at all possible. Thus, most frameworks for LD resort to approximating the set of pairs for which $\mathcal{R}$ holds by using *link specifications* (LS). A LS can be regarded as a classifier C_{ij} that maps each element of the Cartesian product $K_i \times K_j$ to one of the classes of $Y = \{+1, -1\}$, where K_i is called the *set of source instances* while K_j is the *set of target instances*. $(s, t) \in K_i \times K_j$ is considered by C_{ij} to be a correct link when $C_{ij}(s, t) = +1$. Otherwise, (s, t) is considered not to be a potential link. In our example, C_{12} returns +1 for $s = \texttt{ex1:JohnDoe}$ and $t = \texttt{ex2:JD}$.

We will assume that the classifier C_{ij} relies on comparing the value of complex similarity function $\sigma_{ij} : K_i \times K_j \to [0, 1]$ with a threshold θ_{ij}. If $\sigma_{ij}(s, t) \geq \theta_{ij}$, then the classifier returns $+1$ for the pair (s, t). In all other cases, it returns -1. The complex similarity function σ_{ij} consists of a combination of atomic similarity measures $\pi^l_{ij} : K_i \times K_j \to [0, 1]$. These atomic measures compare the value of a particular property of $s \in K_i$ (for example its `rdfs:label`) with the value of a particular property of $t \in K_j$ (for example its `:name`) and return a similarity score between 0 and 1. In our example, σ_{12} relies on the single atomic similarity function $trigrams(\texttt{:ssn}, \texttt{:ssn})$, which compares the social security number attributed to resources of K_1 and K_2.

[10] http://limes.sf.net

We call the set of all pairs $(s, t) \in K_i \times K_j$ that are considered to be valid links by C_{ij} a *mapping* . We will assume that the resources in each of the knowledge bases $K_1, \ldots, K_n$ can be ordered (e.g., by using the lexical ordering of their URI) and thus assigned an index. Then, a mapping between the knowledge bases K_i and K_j can be represented as a matrix M_{ij} of dimensions $|K_i| \times |K_j|$, where the entry in the xth row and yth column is denoted $M_{ij}(x, y)$. If the classifier maps (s, t) to -1, then $M_{ij}(x, y) = 0$ (where x is the index of s and y is the index of t). In all other cases, $M_{ij}(x, y) = \sigma(s, t)$. For the sake of understandability, we will sometimes write $M_{ij}(s_x, t_y)$ to signify $M_{ij}(x, y)$. In our example, C_{34} is a linear classifier, $\sigma_{34} = trigrams(\texttt{:id}, \texttt{:id})$ and $\theta_{34} = 1$. Thus, (`ex3:J36`, `ex4:Cat40_1`) is considered a link.

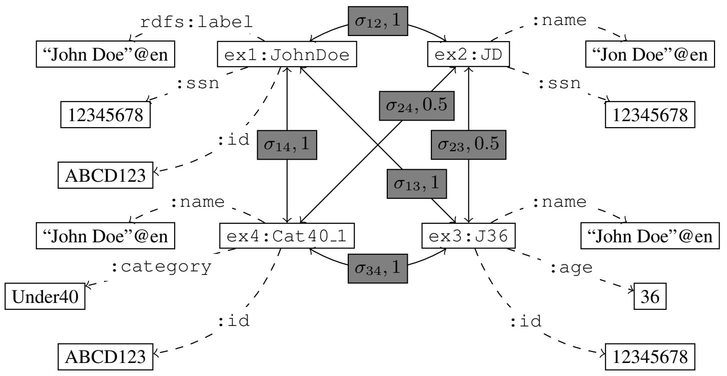


Fig. 1. Example of four linked resources from four different knowledge bases. The white nodes are resources or literals. Properties are represented by dashed labelled arrows. Links are represented by plain arrows. The grey boxes on the links show the names of the similarity measures used to link the resources they connect as well as the similarity value for each of these resource pairs. $\sigma_{12} = trigrams(\texttt{:ssn}, \texttt{:ssn})$, $\sigma_{13} = \sigma_{14} = trigrams(\texttt{:id}, \texttt{:id})$, $\sigma_{23} = \sigma_{24} = \sigma_{34} = dice(\texttt{:name}, \texttt{:name})$, $\sigma_{ij} = \sigma_{ji}$.

Supervised approaches to the computation of link specifications use labelled training data $L \subseteq K_i \times K_j \times Y$ to minimize the error rate of C_{ij}. COLIBRI relies on an unsupervised approach. The idea behind *unsupervised approaches* to learning link specifications is to refrain from using any training data (i.e., $L = \emptyset$). Instead, unsupervised approaches aim to optimize an *objective function*. The objective functions we consider herein approximate the value of the F-measure achieved by a specification and are thus dubbed pseudo-F-measures (short: PFMs) [21].

In this work, we extend the PFM definition presented in [18]. Like in [21, 23, 9], the basic assumption behind this PFM is that one-to-one links exist between the resources in S and T. We chose to extend this measure to ensure that it is symmetrical w.r.t. to the source and target datasets, i.e., PFM(S, T) = PFM(T, S). Our pseudo-precision $\mathcal{P}$ computes the fraction of links that stand for one-to-one links and is equivalent to the strength function presented in [9]. Let $links(K_i, M_{ij})$ be the subset of K_i whose elements are linked to at least one element of K_j. Then, $\mathcal{P}(M_{ij}) = \frac{|links(K_i,M_{ij})|+|links(K_j,M_{ij})|}{2|M_{ij}|}$. The pseudo-recall $\mathcal{R}$ computed the fraction of the total number of resources (i.e., $|K_i|+|K_j|$) from that are involved in at least one link: $\mathcal{R}(M_{ij}) = \frac{|links(K_i,M_{ij})|+|links(K_j,M_{ij})|}{|K_i|+|K_j|}$. Finally, the PFM $\mathcal{F}_\beta$, is defined as $\mathcal{F}_\beta = (1+\beta^2)\frac{\mathcal{PR}}{\beta^2\mathcal{P}+\mathcal{R}}$.

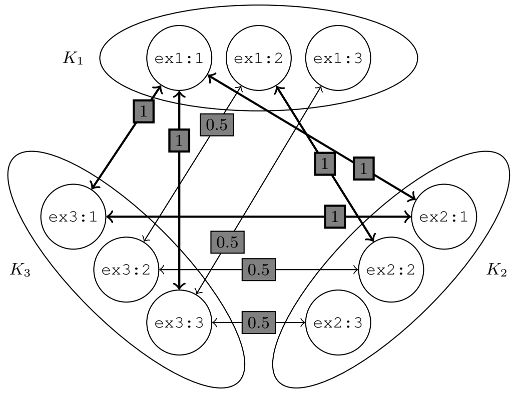


Fig. 2. Example of mappings between 3 sets of resources. K_1 has the namespace ex1, K_2 the namespace ex2 and K_3 the namespace ex3. Thick lines stand for links with the similarity value 1 while thin lines stand for links with the similarity value 0.5.

For the example in Figure 2, $\mathcal{P}(M_{12}) = 1$, $\mathcal{R}(M_{12}) = \frac{2}{3}$ and $\mathcal{F}_1 = \frac{4}{5}$. Our PFM works best if S and T are of comparable size and one-to-one links are to be detected. For example, EUCLID achieves 99.7% F-measure on the OAEI Persons1 dataset.[11] It even reaches 97.7% F-measure on the DBLP-ACM dataset, therewith outperforming the best supervised approach (FEBRL) reported in [15]. Yet, EUCLID achieves worse results compared to FEBRL on the Amazon-Google Products dataset with an F-measure of 43% against 53.8%, where $|T| \approx 3|S|$.

3 The COLIBRI Approach

In this section, we present the COLIBRI approach and its components in detail. We begin by giving an overview of the approach. Then, for the sake of completeness, we briefly present EUCLID, the unsupervised LD approach currently underlying COLIBRI. For more information about EUCLID, please see [18]. Note that COLIBRI can be combined with any unsupervised LD approach. After the overview of EUCLID, we present the voting approach with which COLIBRI attempts to detect erroneous or missing links. In a final step, we present how COLIBRI attempts to repair these sources of error.

[11] http://oaei.ontologymatching.org/

3.1 Overview

Most of the state-of-the-art approaches to LD assume scenarios where two sets of resources are to be linked. COLIBRI assumes that it is given n sets of resources $K_1, \ldots, K_n$. The approach begins by computing mappings M_{ij} between resources of pairs of sets of resources (K_i, K_j). To achieve this goal, it employs the EUCLID algorithm [18] described in the subsequent section. The approach then makes use of the transitivity of $\mathcal{R}$ by computing voting matrices V_{ij} that allow detecting erroneous as well as missing links. This information is finally used to detect resources that should be repaired. An overview of COLIBRI is given in Algorithm 1. In the following sections, we explain each step of the approach.

Algorithm 1 The COLIBRI approach. $\mathcal{M}$ stands for the set of all M_{ij} while $\tilde{\mathcal{V}}$ stands for the set of all $\tilde{V}_{ij}$. The $maxIterations$ parameter ensures that the approach terminates.

```
1: $F_{new} := 0, F_{old} := 0, iterations = 0$
2: while $F_{new} - F_{old} > 0$ do
3:     $F_{old} := F_{new}$
4:     $F_{new} := 0$
5:     for $i \in \{1, \ldots, n\}$ do
6:         for $j \in \{1, \ldots, n\}, j \neq i$ do
7:             $M_{ij}$ = EUCLID $(K_i, K_j)$
8:             $F_{new} := F_{new}$+PSEUDOF$(M_{ij})$
9:         end for
10:    end for
11:    $F_{new} := F_{new}/(n(n-1))$
12:    if $F_{new} - F_{old} > 0$ then
13:        for $i \in \{1, \ldots, n\}$ do
14:            for $j \in \{1, \ldots, n\}, j \neq i$ do
15:                $V_{ij}$ =COMPUTEVOTING$(M_{ij}, \mathcal{M})$
16:                $\tilde{V}_{ij}$ =POSTPROCESS$(V_{ij})$
17:            end for
18:        end for
19:        for $(a, b) \in$ GETWORSTLINKS$(\tilde{\mathcal{V}})$ do
20:            $(rs, rt)$ =GETREASON$(a, b)$
21:            REPAIR(rs,rt)
22:        end for
23:    end if
24: end while
```

3.2 EUCLID

Over the last years, non-deterministic approaches have been commonly used to detect highly accurate link specifications (see, e.g., [19, 21]). EUCLID (Line 8 of Algorithm 1) is a deterministic unsupervised approach for learning link specifications. The core idea underlying the approach is that link specifications of a given type (linear, conjunctive, disjunctive) can be regarded as points in a link specification space. Finding an accurate link specification is thus equivalent to searching through portions of this specification space. In the following, we will assume that EUCLID tries to learn a conjunctive classifier, i.e., a classifier which returns $+1$ for a pair $(s, t) \in K_i \times K_j$ when

$\bigwedge\limits_{l=1}^{m}(\pi_{ij}^l(s,t) \geq \theta_{ij}^l)$ holds. The same approach can be used to detect disjunctive and linear classifiers. EUCLID assumes that it is given a set of m atomic similarity functions π_{ij}^l with which it can compare $(s,t) \in K_i \times K_j$. The atomic functions π_{ij}^l build the basis of an m-dimensional space where each of the dimensions corresponds to exactly one of the π_{ij}^l. In this space, the specification $\bigwedge\limits_{l=1}^{m}(\pi_{ij}^l(s,t) \geq \theta_{ij}^l)$ has the coordinates $(\theta_{ij}^1, \ldots, \theta_{ij}^m)$. The core of EUCLID consists of a hierarchical grid search approach that aims to detect a link specification within a hypercube (short: cube) which maximizes the value of a given objective function $\mathcal{F}$. The hypercubes considered by EUCLID are such that their sides are all orthogonal to the axes of the space . Note that such a hypercube can be described entirely by two points $b = (b_1, \ldots, b_m)$ and $B = (B_1, \ldots, B_m)$ with $\forall i \in \{1, \ldots, m\}(b_i \leq B_i)$.

EUCLID begins by searching through the cube defined by $b = \underbrace{(0, \ldots, 0)}_{m\ times}$ and $B = \underbrace{(1, \ldots, 1)}_{m\ times}$ (i.e., the whole of the similarity space). A point w with coordinates $(w_1, \ldots, w_m)$ corresponds to the classifier with the specific function $\bigwedge_{l=1}^{m}(\pi_{ij}^l(s_i, s_j) \geq w_l)$. Let $\alpha \in \mathbb{N}, \alpha \geq 2$ be the granularity parameter of EUCLID. The search is carried out by generating a grid of $(\alpha+1)^m$ points g whose coordinates $g_i = \left(b_i + k_i \frac{(B_i - b_i)}{\alpha}\right)$, where $k_i \in \{0, \ldots, \alpha\}$. We call $\varDelta_i = \frac{(B_i - b_i)}{\alpha}$ the *width* of the grid in the ith dimension. EUCLID now computes the pseudo-F-measure $\mathcal{F}$ of the specification corresponding to each point on the grid. Let $g^{\max}$ be a point that maximizes $\mathcal{F}$. Then, EUCLID updates the search cube by updating the coordinates of the points b and B as follows: $b_i = (\max\{0, g_i^{\max} - \varDelta_i\})$ and $B_i = (\min\{1, g_i^{\max} + \varDelta_i\})$. Therewith, EUCLID defines a new and smaller search cube. The search is iterated until a stopping condition such as a given number of iterations is met.

3.3 Voting

The result of EUCLID is a set of $n(n-1)$ mappings M_{ij} which link the resource set K_i with the resource set K_j. The goal of the second step of a COLIBRI iteration is to determine the set of resources that might contain incomplete or erroneous information based on these mappings. The basic intuition behind the approach is to exploit the transitivity of the relation $\mathcal{R}$ is as follows: If the link $(s,t) \in K_i \times K_j$ is correct, then for all k with $1 \leq k \leq n$ with $k \neq i, j$, there should exist pairs of links (s,z) and (z,t) with $M_{ik}(s,z) > 0$ and $M_{kj}(z,t) > 0$. Should such pairs not exist or be weakly connected, then we can assume that some form of error was discovered.

Formally, we go about implementing this intuition as follows: We first define the voting matrices V_{ij} as $V_{ij} = \frac{1}{n}\left(M_{ij} + \sum\limits_{k=0, k\neq i,j}^{n} M_{ik}M_{kj}\right)$ (Line 15 of Algorithm 1). In the example shown in Figure 2, the mappings are

$$M_{12} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix}, M_{13} = \begin{pmatrix} 1 & 0 & 1 \\ 0 & 0.5 & 0 \\ 0 & 0 & 0.5 \end{pmatrix} \text{ and } M_{23} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0.5 & 0 \\ 0 & 0 & 0.5 \end{pmatrix}.$$

The corresponding voting matrices are thus

$$V_{12} = \begin{pmatrix} 1 & 0 & 0.25 \\ 0 & 0.625 & 0 \\ 0 & 0 & 0.125 \end{pmatrix}, V_{13} = \begin{pmatrix} 1 & 0 & 0.5 \\ 0 & 0.5 & 0 \\ 0 & 0 & 0.25 \end{pmatrix} \text{ and } V_{23} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0.5 & 0 \\ 0 & 0 & 0.25 \end{pmatrix}.$$

Each voting matrix V_{ij} encompasses the cumulative results of the linking between all pairs of resource sets with respect to the resources in (K_i, K_j). Computing V_{ij} as given above can lead to an explosion in the number of resources associated to s_i. In our example, the erroneous link between `ex1:1` and `ex3:3` leads to `ex1:1` being linked not only to `ex2:1` but also to `ex2:3` in V_{12}. We thus post-process each V_{ij} by only considering the best match for each $s \in K_i$ within V_{ij}, i.e., by removing each non-maximal entry from each row of V_{ij} (Line 16 of Algorithm 1). We label the resulting matrix $\tilde{V}_{ij}$. For our example, we get the following matrices:

$$\tilde{V}_{12} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0.625 & 0 \\ 0 & 0 & 0.125 \end{pmatrix}, \tilde{V}_{13} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0.5 & 0 \\ 0 & 0 & 0.25 \end{pmatrix} \text{ and } \tilde{V}_{23} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0.5 & 0 \\ 0 & 0 & 0.25 \end{pmatrix}.$$

COLIBRI now assumes that the links encoded in $\tilde{V}_{ij}$ are most probably correct. All entries of $\tilde{V}_{ij}$ being 1 are thus interpreted as all matrices agreeing on how to link the resources in (K_i, K_j). In the example in Figure 2, this is the case for $\tilde{V}_{12}(\texttt{ex1:1}, \texttt{ex2:1})$. Should this not be the case, then the disagreement between the matrices can result from the following reasons:

1. *Missing links*: This is the case in our example for the link $(\texttt{ex1:3}, \texttt{ex2:3})$ which is not contained in M_{12}. For this reason, $\tilde{V}_{12}(\texttt{ex1:3}, \texttt{ex2:3})$ is minimal.
2. *Weak links*: This is the case for the second-lowest entry in $\tilde{V}_{12}$, where the entry for $(\texttt{ex1:2}, \texttt{ex2:2})$ is due to $M_{13}(\texttt{ex1:2}, \texttt{ex3:2})$ and $M_{32}(\texttt{ex3:2}, \texttt{ex2:2})$ being 0.5.

COLIBRI now makes use of such disagreements to repair the entries in the knowledge bases with the aim of achieving a better linking. To this end, it selects a predetermined number of links (a, b) over all $\tilde{\mathcal{V}}_{ij}$ whose weight is minimal and smaller than 1 (GETWORSTLINKS in Algorithm 1). These links are forwarded to the instance repair.

3.4 Instance repair

For each of the links (a, b) selected by the voting approach, the instance repair routine of COLIBRI begins by computing why $\tilde{V}_{ij}(a, b) < 1$. To achieve this goal, it computes the *reason* $(rs, rt) \in \left(K_i \times \bigcup\limits_{k=1, k\neq i}^{n} K_k\right) \cup \left(\bigcup\limits_{k=1, k\neq j}^{n} K_k \times K_j\right)$ by detecting the smallest entry that went into computing $\tilde{V}_{ij}(a, b)$. Three possibilities occur:

1. $(rs, rt) \in K_i \times K_j$: In this case, the weak or missing link is due to the initial mapping M_{ij}.
2. $(rs, rt) \in K_i \times K_k$ with $k \neq i \wedge k \neq j$: In this case, the weak or missing link is due to the in-between mapping M_{ik}.
3. $(rs, rt) \in K_k \times K_j$ with $k \neq i \wedge k \neq j$: Similarly to the second case, the weak or missing link is due to the in-between mapping M_{kj}.

In all three cases, the repair approach now aims to improve the link by repairing the resource rs or rt that most probably contains erroneous or missing information. To achieve this goal, it makes use of the similarity measure σ used to generate (rs, rt). The value of this measure being low suggests that the property values p^l and q^l used across the similarity measures π^l are dissimilar. The idea of the repair is then to overwrite exclusively the values of $p^l(rs)$ with those of $q^l(rt)$ or vice-versa. The intuition behind deciding upon whether to update rs or rt is based on the *average similarity* $\bar{\sigma}(rs)$ resp. $\bar{\sigma}(rt)$ of the resources rs and rt to other resources. For a resource $s \in K_i$, this value is given by

$$\bar{\sigma}(s) = \frac{1}{n-1}\left(\sum_{k=1,k\neq i}^{n} \max_{t\in K_k} \sigma_{ik}(s,t)\right). \tag{1}$$

Here, the assumption is that the higher the value of $\bar{\sigma}$ for a given resource, the higher the probability that it does not contain erroneous information.

Let us consider anew the example given in Figure 2 and assume that the link that is to be repaired is $(\texttt{ex1:2}, \texttt{ex2:2})$. One reason for this link would be $rs = \texttt{ex1:2}$ and $rt = \texttt{ex3:2}$. Now $\bar{\sigma}(\texttt{ex1:2}) = 0.75$ while $\bar{\sigma}(\texttt{ex3:2}) = 0.5$. COLIBRI would thus choose to overwrite the values of `ex3:2` with those of `ex1:2`.

The overwriting in itself is carried out by overwriting the values of $q^l(rt)$ with those of $p^l(rs)$ if $\bar{\sigma}(rs) \geq \bar{\sigma}(rt)$ and vice-versa. This step terminates an iteration of COLIBRI, which iterates until a termination condition is reached, such as the average value of $\mathcal{F}$ for the mappings generated by EUCLID declining or a maximal number of iterations. The overall complexity of each iteration of COLIBRI is $O(n^2 \times E)$, where E is the complexity of the unsupervised learning algorithm employed to generate the mappings. Thank to the algorithms implemented in LIMES which have a complexity close to $O(m)$ where $m = \max\{|S|, |T|\}$ for each predicate, EUCLID has a complexity of $O(pm)$, where p is the number of predicates used to compare entities. Consequently, the overall complexity of each iteration of COLIBRI is $O(pmn^2)$ when it relies on EUCLID. While we observed a quick converge of the approach on real and synthetic datasets within our evaluation (maximally 10 iterations), the convergence speed of the approach may vary on the datasets used.

4 Evaluation

The aim of our evaluation was to measure whether COLIBRI can improve the F-measure of mappings generated by unsupervised link discovery approaches. To this end, we measured the increase in F-measure achieved by COLIBRI w.r.t to the number of iterations it carried out on a synthetic dataset generated out of both synthetic and real data. To the best of our knowledge, no benchmark dataset is currently available for link discovery across $n > 2$ knowledge bases. We thus followed the benchmark generation approach for instance matching presented in [6] to generate the evaluation data for COLIBRI.

4.1 Experimental Setup

We performed controlled experiments on data generated automatically from two synthetic and three real datasets. The synthetic datasets consisted of the Persons1 and

Restaurant datasets from the OAEI2010 benchmark data sets.[12] The real datasets consisted of the ACM-DBLP, Amazon-Google and Abt-Buy datasets.[13] We ran all experiments in this section on the source dataset of each of these benchmark datasets (e.g., ACM for ACM-DBLP). We omitted OAEI2010's Person2 because its source dataset is similar to Person1's. Given the lack of benchmark data for link discovery over several sources, we generated a synthetic benchmark as follows: Given the initial source dataset K_1, we first generated $n - 1$ copies of K_1. Each copy was altered by using a subset of the operators suggested in [6]. The alteration strategy consisted of randomly choosing a property of a randomly chosen resource and altering it. We implemented three syntactic operators to alter property values, i.e., misspellings, abbreviations and word permutations. The syntactic operator used for altering a resource was chosen randomly. We call the probability of a resource being chosen for alteration the *alteration probability* (ap). The goal of this series of experiments was to quantify (1) the gain in F-measure achieved by COLIBRI over EUCLID and (2) the influence of ap and of the number n of knowledge bases on COLIBRI's F-measure.

The F-measure of EUCLID and COLIBRI was the average F-measure they achieved over all pair (K_i, K_j) with $i \neq j$. To quantify the amount of resources that were altered by COLIBRI in the knowledge bases $K_1, \ldots, K_n$, we computed the average *error rate* in the knowledge bases after each iteration as follows:

$$errorrate = 1 - \frac{1}{n(n-1)} \sum_{i=1}^{n} \sum_{j=1, j \neq i}^{n} \frac{2|K_i \cap K_j|}{|K_i| + |K_j|}. \qquad (2)$$

The maximal number of COLIBRI iterations was set to 10. We present the average results but omit the standard deviations for the sake of legibility. For precision, the standard deviation was maximally 4%. The recall's standard deviation never exceeded 1% while it reached 2% for the F-measure.

4.2 Experimental Results

We varied the number of knowledge bases between 3 and 5. Moreover, we varied the alteration probability between 10% and 50% with 10% increments. We then measured the precision, recall, F-measure, runtime and number of repairs achieved by the batch version of COLIBRI over several iterations. Due to lack of space, we present portions of the results we obtained in Figure 3 and Table 1.[14] Table 1 shows an overview of the results we obtained across the different datasets. Our results show clearly that COLIBRI can improve the results of EUCLID significantly on all datasets. On the Restaurant dataset for example, COLIBRI is 6% better than EUCLID on average. On ACM, the average value lies by 4.8%. In the best case, COLIBRI improves the results of EUCLID from 0.85 to 0.99 (Amazon, $ap = 50\%$, KBs = 4). Moreover, COLIBRI never worsens the results of EUCLID. This result is of central importance as it suggests that our approach

[12] Available online at http://oaei.ontologymatching.org/2010/.

[13] Available online at http://dbs.uni-leipzig.de/en/research/projects/object_matching/fever/benchmark_datasets_for_entity_resolution.

[14] See http://limes.sf.net for more results.

can be used across the Linked Data Web for any combination of number of knowledge and error rates within the knowledge bases.

The results achieved on the Restaurant dataset are presented in more detail in Figure 3. Our results on this dataset (which were corroborated by the results we achieved on the other datasets) show that the results achieved by EUCLID alone depend directly on the probability of errors being introduced into the data sets. For example, EUCLID is able to achieve an F-measure of 0.94 when provided with data sets with an error rate of 30%. Yet, this F-measure sinks to 0.88 when the error rate is set to 50%. These results do suggest that EUCLID is robust against errors. This is due to the approach being able to give properties that contain a small error percentage a higher weight. Still, the COLIBRI results show clearly that COLIBRI can accurately repair the knowledge bases and thus achieve even better F-measures. On this particular data, the approach achieves an F-measure very close to 1 in most cases. Note that the number of iterations required to achieve this score depends directly on the number of knowledge bases and on the error probability.

One interesting observation is that the average F-measure achieved by EUCLID decreases with the number of knowledge bases used for linking. This is simply due to the overall larger number of errors generated by our evaluation framework when the number of knowledge bases is increased. While larger number also make the detection of errors more tedious, COLIBRI achieves significant increase of F-measure in this setting. In particular, the F-measure of EUCLID is improved upon by up to 12% absolute on the Restaurant dataset ($ap = 50\%$) as well as 7% absolute on Persons1 ($ap = 50\%$).

As expected, the runtime of our approach grows quadratically with the number of knowledge bases. This is simply due to EUCLID being run for each pair of knowledge bases.The runtimes achieved suggest that COLIBRI can be used in practical settings and on large datasets as long as the number of dimensions in EUCLID's search space remains small. In particular, one iteration of the approach on the DBLP data sets required less than 2 minutes per iteration for 3 knowledge bases, which corresponds to 3 EUCLID runs of which each checked 3125 link specifications. The worst runtimes were achieved on the Persons1 dataset, where COLIBRI required up to 11min/iteration. This was due to the large number of properties associated with each resource in the dataset, which forced EUCLID to evaluate more than 78,000 specifications per iteration.

5 Related Work

Most LD approaches for learning link specifications developed so far abide by the paradigm of supervised machine learning. One of the first approaches to target this goal was presented in [11]. While this approach achieves high F-measures, it also requires large amounts of training data. However, creating training data for link discovery is a very expensive process, especially given the size of current knowledge bases. Supervised LD approaches which try to reduce the amount training data required are most commonly based on active learning (see, e.g., [12, 19]). Still, these approaches are not guaranteed to require a small amount of training data to converge. In newer works, unsupervised techniques for learning LD specifications were developed [21, 18]. The main advantage of unsupervised learning techniques is that they do not require any training

Table 1. Average F-measure of EUCLID (F_E) and COLIBRI (F_C) after 10 iterations, runtime (R, in seconds) and number of repaired links L achieved across all experiments. KBs stands for the number of knowledge bases used in our experiments.

ap	10%				30%				50%			
Measures	F_E	F_C	R	L	F_E	F_C	R	L	F_E	F_C	R	L
KBs	Restaurant											
3	0.98	1.00	0.6	4	0.94	0.99	0.5	17	0.89	0.98	0.4	43
4	0.99	1.00	1.2	8	0.93	1.00	1.0	33	0.90	1.00	0.9	35
5	0.98	1.00	1.8	20	0.93	1.00	1.5	30	0.88	1.00	1.3	34
KBs	Persons1											
3	0.99	1.00	225.6	11	0.96	1.00	206.2	38	0.94	1.00	190.4	57
4	0.98	1.00	494.3	23	0.96	1.00	422.1	47	0.93	1.00	349.9	77
5	0.98	1.00	819.4	20	0.95	1.00	747.6	75	0.93	1.00	656.2	110
KBs	ACM											
3	0.95	0.96	85.7	220	0.89	0.96	69.3	301	0.84	0.95	66.5	484
4	0.94	0.94	168	12	0.88	0.88	140.4	36	0.83	0.96	131.1	261
5	0.94	0.94	271.7	30	0.87	0.94	240.9	821	0.82	0.84	202.8	348
KBs	DBLP											
3	0.94	0.98	135	220	0.85	0.97	117.2	828	0.77	0.82	111	2686
4	0.93	0.98	268.8	312	0.83	0.90	234.7	306	0.76	0.81	201.1	350
5	0.93	0.98	334.9	517	0.82	0.84	395.9	182	0.76	0.77	338.1	156
KBs	Amazon											
3	0.97	0.99	90.4	60	0.92	0.99	85.2	177	0.86	0.98	81.8	300
4	0.97	0.99	187.5	98	0.91	0.98	172.6	185	0.85	0.99	160.4	150
5	0.96	0.99	301.8	131	0.90	0.99	278.7	369	0.84	0.88	246.8	60

data to discover mappings. Moreover, the classifiers they generate can be used as initial classifiers for supervised LD approaches. In general, unsupervised approaches assume some knowledge about the type of links that are to be discovered. For example, unsupervised approaches for ontology alignment such as PARIS [23] aim to discover exclusively `owl:sameAs` links. To this end, PARIS relies on a probabilistic model and maps instances, properties and ontology elements. Similarly, the approach presented in [21] assumes that a 1-to-1 mapping is to be discovered. Here, the mappings are discovered by using a genetic programming approach whose fitness function is set to a PFM. The main inconvenient of this approach is that it is not deterministic. Thus, it provides no guarantee of finding a good specification. This problem was addressed by EUCLID [18].

While ontology-matching approaches that rely on more than 2 ontologies have existed for almost a decade [16, 5], LD approaches that aim to discover between n datasets have only started to emerge in newer literature. For instance, the approach proposed by [8] suggests a composition method for link discovery between n datasets. The approach is based on strategies for combining and filtering mappings between resources to generate links between knowledge bases. The framework introduced by [13] aims to predict links in multi-relational graph. To this end, it models the relations of the knowledge bases using set of description matrices and combines them using an additive model. The *Multi-Core Assignment Algorithm* presented by [3] automated the creation of `owl:sameAs` links across multiple knowledge bases in a globally consistent manner. The only drawback of this approach is that it requires a large amount of processing power. This problem was addressed in [3].

Approaches related to COLIBRI also include link predication approaches based on statistical relational learning (SRL). Examples of SRL approaches that can be used for predicate detection include CP and Tucker [14] as well as RESCAL [20], which all rely on tensor factorization. In general, approaches which rely on tensor factorization have a higher complexity that EUCLID. For example, CP's complexity is quadratic in the number of predicates. Related approaches that have been employed on Semantic Web data and ontologies include approaches related to Bayesian networks, inductive learning and kernel learning [4, 24, 2, 20, 22]. Due to the complexity of the models they rely on, most of these approaches are likely not to scale to very large datasets. LIMES (in which EUCLID is implemented) has yet been shown to scale well on large datasets [17]. An exact evaluation of the complexity and runtime of a combination of COLIBRI and SRL-based approaches remains future work. More details on SRL can be found in [7].

6 Conclusion and Future Work

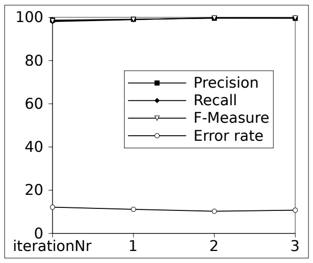
We presented COLIBRI, the first unsupervised LD approach which attempts to repair instance knowledge in n knowledge bases ($n \geq 2$) to improve its linking accuracy. Our evaluation suggests that our approach is robust and can be used by error rates up to 50% when provided with at least 3 knowledge bases. In addition, our results show that COLIBRI can improve the results of EUCLID by up to 14% F-measure. In future work, we plan to extend our evaluation further and analyse our performance on real data as well as on knowledge bases of different size. We plan to deploy our approach in interactive scenarios within which users are consulted before the knowledge bases are updated. The voting procedure implemented by COLIBRI can be used to provide users with a measure for the degree of confidence in a predicted link and in the need for a repair within an interactive learning scenario.

Acknowledgement

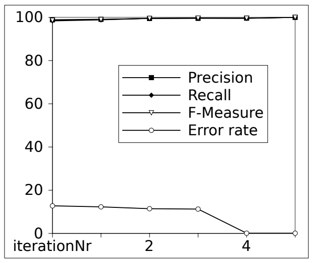
The work presented in this paper was partly financed by the FP7 project GeoKnow, GA 318159 and the DFG project LinkingLOD.

References

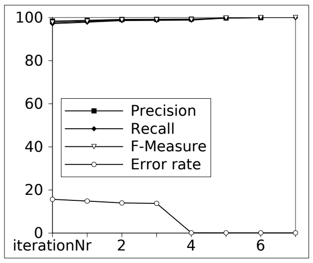
1. Sören Auer, Jens Lehmann, and Axel-Cyrille Ngonga Ngomo. Introduction to linked data and its lifecycle on the web. In *Reasoning Web*, pages 1–75, 2011.
2. Stephan Bloehdorn and York Sure. Kernel methods for mining instance data in ontologies. In *ASWC*, pages 58–71. Springer, 2007.



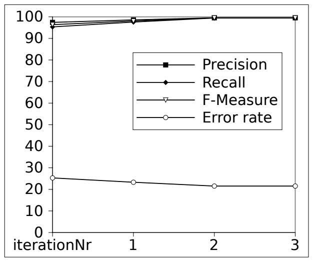
(a) ap=10%, KBs = 3



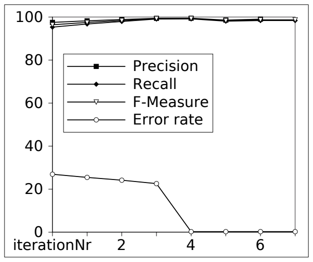
(b) ap=10%, KBs = 4



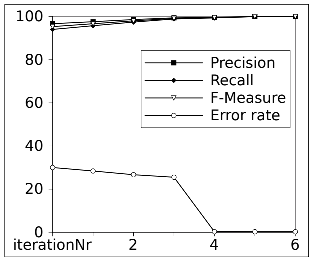
(c) ap=10%, KBs = 5



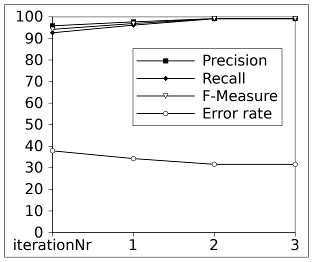
(d) ap=20%, KBs = 3



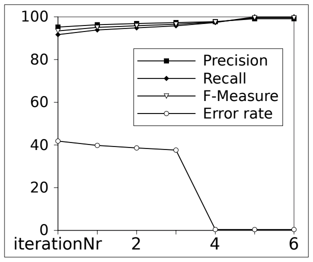
(e) ap=20%, KBs = 4



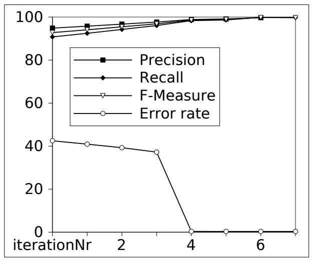
(f) ap=20%, KBs = 5



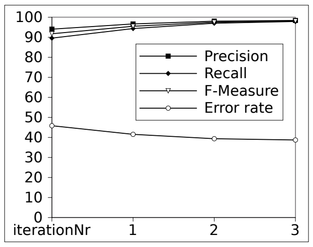
(g) ap=30%, KBs = 3



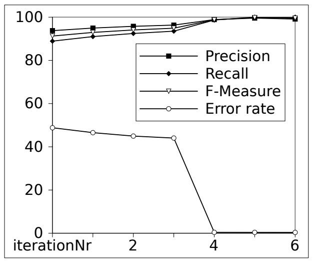
(h) ap=30%, KBs = 4



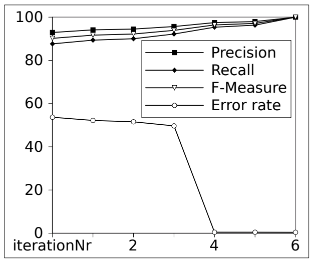
(i) ap=30%, KBs = 5



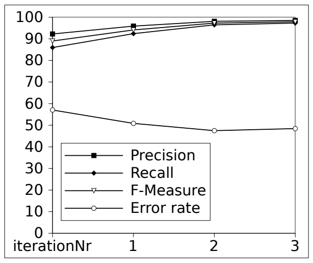
(j) ap=40%, KBs = 3



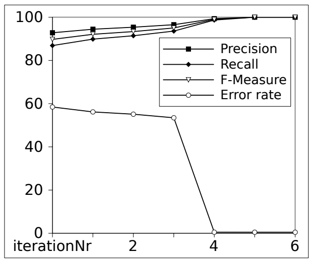
(k) ap=40%, KBs = 4



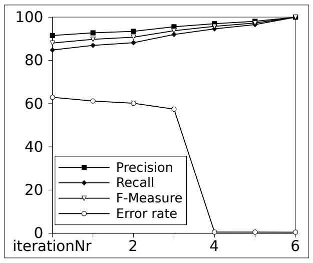
(l) ap=40%, KBs = 5



(m) ap=50%, KBs = 3



(n) ap=50%, KBs = 4



(o) ap=50%, KBs = 5

Fig. 3. Overview of the results on the Restaurants dataset.

3. Christoph Böhm, Gerard de Melo, Felix Naumann, and Gerhard Weikum. Linda: distributed web-of-data-scale entity matching. In *CIKM*, pages 2104–2108, 2012.
4. Claudia d'Amato, Nicola Fanizzi, and Floriana Esposito. Non-parametric statistical learning methods for inductive classifiers in semantic knowledge bases. In *ICSC*, pages 291–298, 2008.
5. Jérôme Euzenat. Algebras of ontology alignment relations. In *ISWC*, pages 387–402, 2008.
6. A. Ferrara, S. Montanelli, J. Noessner, and H. Stuckenschmidt. Benchmarking Matching Applications on the Semantic Web. In *ESWC*, 2011.
7. Lise Getoor and Ben Taskar. *Introduction to Statistical Relational Learning (Adaptive Computation and Machine Learning)*. The MIT Press, 2007.
8. Michael Hartung, Anika Groß, and Erhard Rahm. Composition methods for link discovery. In *BTW*, pages 261–277, 2013.
9. Oktie Hassanzadeh, Ken Q. Pu, Soheil Hassas Yeganeh, Renée J. Miller, Lucian Popa, Mauricio A. Hernández, and Howard Ho. Discovering linkage points over web data. *VLDB Endow.*, 6(6):445–456, April 2013.
10. R. Isele, A. Jentzsch, and C. Bizer. Efficient Multidimensional Blocking for Link Discovery without losing Recall. In *WebDB*, 2011.
11. Robert Isele and Christian Bizer. Learning linkage rules using genetic programming. In *OM Workshop*, 2011.
12. Robert Isele, Anja Jentzsch, and Christian Bizer. Active learning of expressive linkage rules for the web of data. In *ICWE*, pages 411–418, 2012.
13. Xueyan Jiang, Volker Tresp, Yi Huang, and Maximilian Nickel. Link prediction in multi-relational graphs using additive models. In *SeRSy*, pages 1–12, 2012.
14. Tamara G. Kolda and Brett W. Bader. Tensor decompositions and applications. *SIAM Rev.*, 51(3):455–500, August 2009.
15. Hanna Köpcke, Andreas Thor, and Erhard Rahm. Evaluation of entity resolution approaches on real-world match problems. *VLDB Endow.*, 3(1-2):484–493, 2010.
16. Jayant Madhavan and Alon Y. Halevy. Composing mappings among data sources. In *VLDB*, pages 572–583, 2003.
17. Axel-Cyrille Ngonga Ngomo. On link discovery using a hybrid approach. *Journal on Data Semantics*, 1:203 – 217, 2012.
18. Axel-Cyrille Ngonga Ngomo and Klaus Lyko. Unsupervised learning of link specifications: deterministic vs. non-deterministic. In *OM Workshop*, 2013.
19. Axel-Cyrille Ngonga Ngomo, Klaus Lyko, and Victor Christen. COALA - Correlation-Aware Active Learning of Link Specifications. In *Proceedings of ESWC*, 2013.
20. Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. Factorizing yago: Scalable machine learning for linked data. In *WWW*, pages 271–280, New York, NY, USA, 2012. ACM.
21. Andriy Nikolov, Mathieu D'Aquin, and Enrico Motta. Unsupervised learning of data linking configuration. In *Proceedings of ESWC*, 2012.
22. Cristina Prez-Sol and Jordi Herrera-Joancomart. Improving relational classification using link prediction techniques. In *ECML/PKDD (1)*, volume 8188, pages 590–605, 2013.
23. Fabian M. Suchanek, Serge Abiteboul, and Pierre Senellart. PARIS: Probabilistic Alignment of Relations, Instances, and Schema. *PVLDB*, 5(3):157–168, 2011.
24. Ilya Sutskever, Ruslan Salakhutdinov, and Joshua B. Tenenbaum. Modelling relational data using bayesian clustered tensor factorization. In *NIPS*, pages 1821–1828, 2009.