COALA – Correlation-Aware Active Learning of Link Specifications

Axel-Cyrille Ngonga Ngomo, Klaus Lyko, and Victor Christen

Department of Computer Science
AKSW Research Group
University of Leipzig, Germany
{ngonga,klaus.lyko,christen}@informatik.uni-leipzig.de

Abstract. Link Discovery plays a central role in the creation of knowledge bases that abide by the five Linked Data principles. Over the last years, several active learning approaches have been developed and used to facilitate the supervised learning of link specifications. Yet so far, these approaches have not taken the correlation between unlabeled examples into account when requiring labels from their user. In this paper, we address exactly this drawback by presenting the concept of the correlation-aware active learning of link specifications. We then present two generic approaches that implement this concept. The first approach is based on graph clustering and can make use of intra-class correlation. The second relies on the activation-spreading paradigm and can make use of both intra- and inter-class correlations. We evaluate the accuracy of these approaches and compare them against a state-of-the-art link specification learning approach in ten different settings. Our results show that our approaches outperform the state of the art by leading to specifications with higher F-scores.

Keywords: Active Learning, Link Discovery, Genetic Programming.

1 Introduction

The importance of the availability of links for a large number of tasks such as question answering [20] and keyword search [19] as well as federated queries has been pointed out often in literature (see, e.g., [1]). Two main problems arise when trying to discover links between data sets or even deduplicate data sets. First, naive solutions to Link Discovery (LD) display a quadratic time complexity [13]. Consequently, they cannot be used to discover links across large datasets such as DBpedia[1] or Yago[2]. Time-efficient algorithms such as PPJoin+ [21] and $\mathcal{HR}^3$ [11] have been developed to address the problem of the a-priori quadratic runtime of LD approaches. While these approaches achieve practicable runtimes even on large datasets, they do not guarantee the quality of the links that are returned by LD frameworks. Addressing this second problem of LD demands

[1] http://dbpedia.org
[2] http://www.mpi-inf.mpg.de/yago-naga/yago/

P. Cimiano et al. (Eds.): ESWC 2013, LNCS 7882, pp. 442–456, 2013.

the development of techniques that can compute accurate *link specifications* (i.e., aggregations of atomic similarity or distance measures and corresponding thresholds) for deciding whether two resources should be linked. This problem is commonly addressed within the setting of machine learning. While both supervised (e.g., [15]) and unsupervised machine-learning approaches (e.g., [17]) have been proposed to achieve this goal, we focus on supervised machine learning.

One of the main drawbacks of supervised machine learning for LD lies in the large number of links necessary to achieve both a high precision and a high recall. This intrinsic problem of supervised machine learning has been addressed by relying on active learning [18]. The idea here is to rely on *curious classifiers*. These are supervised approaches that begin with a small number of labeled links and then inquire labels for data items that promise to improve their accuracy. Several approaches that combine genetic programming and active learning have been developed over the course of the last couple of years and shown to achieve high F-measures on the deduplication (see e.g., [4]) and LD (see e.g., [15]) problems. Yet, so far, none of these approaches has made use of the correlation between the unlabeled data items while computing the set of most informative items. In this paper, we address exactly this drawback.

The basic intuition behind this work is that we can provide a better approximation of the real information content of unlabeled data items by taking the similarity of unlabeled items into account. We call this paradigm the correlation-aware active learning of link specifications and dub it COALA. A better approximation should ensure that curious classifiers converge faster. Consequently, we should be able to reduce the number of data items that the user has to label manually. We thus present and evaluate two generic approaches that implement this intuition. Overall, our contributions are as follows:

1. We describe the correlation-aware active learning of link specifications.
2. We present the first two generic approaches that implement this concept. The first is based on graph clustering while the second implements the spreading activation principle.
3. We combine these approaches with the EAGLE algorithm [15] and show in ten different settings that our approaches improve EAGLE's performance with respect to both F-score and standard deviation.

The approaches presented herein were included in the LIMES framework[3]. A demo of the approach can be accessed by using the SAIM interface [4]. The rest of this paper is structured as follows: We first present some of the formal notation necessary to understand this work. In addition, we give some insights into why the inclusion of correlation information can potentially improve the behavior of a curious classifier. Thereafter, we present two approaches that implement the paradigm of including correlation information into the computation of the most informative link candidates. We compare the two approaches with the state of

[3] `http://limes.sf.net`
[4] `http://saim.aksw.org`

the art in ten different settings and show that we achieve faster convergence and even a better overall performance in some cases. We finally present some related work and conclude.

2 Preliminaries

In this section, we present the core of the formal notation used throughout this paper. We begin by giving a brief definition of the problem we address. Then, we present the concept of active learning.

2.1 Link Discovery

The formal definition of LD adopted herein is similar to that proposed in [12]. Given a relation R and two sets of instances S and T, the goal of LD is to find the set $M \subseteq S \times T$ of instance pairs (s, t) for which $R(s, t)$ holds. In most cases, finding an explicit way to compute whether $R(s, t)$ holds for a given pair (s, t) is a difficult endeavor. Consequently, most LD frameworks compute an approximation of M by computing a set $\tilde{M} = \{(s, t) : \sigma(s, t) \geq \theta\}$, where σ is a (complex) similarity function and θ is a distance threshold. The computation of an accurate (i.e., of high precision and recall) similarity function σ can be a very complex task [6]. To achieve this goal, machine-learning approaches are often employed. The idea here is to regard the computation of σ and θ as the computation of a classifier $\mathcal{C} : S \times T \to [-1, +1]$. This classifier assigns pairs (s, t) to the class -1 when $\sigma(s, t) < \theta$. All other pairs are assigned the class $+1$. The similarity function σ and the threshold θ are derived from the decision boundary of $\mathcal{C}$.

2.2 Active Learning of Link Specifications

Learning approaches based on genetic programming have been most frequently used to learn link specifications [5,15,17]. Supervised batch learning approaches for learning such classifiers must rely on large amounts of labeled data to achieve a high accuracy. For example, the genetic programming approach used in [7] has been shown to achieve high accuracies when supplied with more than 1000 positive examples. Recent work has addressed this drawback by relying on active learning, which was shown in [15] to reduce the amount of labeled data needed for learning link specifications. The idea behind active learners (also called *curious classifiers* [18]) is to query for the labels of chosen pairs (s, t) (called *link candidates*) iteratively. We denote the count of iterations with t. The function $label : S \times T \to \{\oplus, \ominus, \otimes\}$ stands for the labeling function and encodes whether a pair (s, t) is (1) known be a positive example for a link (in which case $label(s, t) = \oplus$), (2) known to be a negative example (in which case $label(s, t) = \ominus$) or (3) is unclassified (in which case $label(s, t) = \otimes$). We denote classifiers, similarity functions, thresholds and sets at iteration t by using a superscript notation. For example, the classifier at iteration t is denoted

$\mathcal{C}^t$ while $label^t$ stands for the labeling function at iteration t. We call the set $\mathcal{P}^t = \{(s,t) \in S \times T : (label(s,t) = \otimes) \wedge (\mathcal{C}^t(s,t) = +1)\}$ the set *presumed positives*. The set $\mathcal{N}^t$ of *presumed negatives* is defined analogously. If $label(s,t) = \otimes$, then we call the class assigned by $\mathcal{C}$ to (s,t) the *presumed class* of (s,t). When the class of a pair (s,t) is explicit known, we simply use the expression (s,t)*'s class*. The set $\mathcal{C}^{+t} = \{(s,t) : \mathcal{C}^t(s,t) = +1\}$ is called the set of *positive link candidates* while the set $\mathcal{C}^{-t} = \{(s,t) : \mathcal{C}^t(s,t) = -1\}$ is called the set of *negative link candidates*. The query for labeled data is carried out by selecting a subset of $\mathcal{P}^t$ with the magnitude k^+ (resp. a subset of $\mathcal{N}^t$ with the magnitude k^-). In the following, we will assume $k = k^+ = k^-$. The selection of the k elements from $\mathcal{P}^t$ and $\mathcal{N}^t$ is carried out by using a function ifm : $S \times T \to \mathbb{R}$ that can compute how informative a pair (s,t) is for the $\mathcal{C}^t$, i.e., how well the pair would presumably further the accuracy of $\mathcal{C}^t$. We call $\mathcal{I}^{+t} \subseteq \mathcal{P}^t$ (resp. $\mathcal{I}^{-t} \subseteq \mathcal{N}^t$) the set of *most informative positive* (resp. *most informative negative*) link candidates. In this setting, the information content of a pair (s,t) is usually inverse to its distance from the boundary of $\mathcal{C}^t$.

Active learning approaches based on genetic programming adopt a *comittee*-based setting to active learning. Here, the idea is to learn m classifiers $\mathcal{C}_1, \ldots, \mathcal{C}_m$ concurrently and to have the m classifiers select the sets $\mathcal{I}^-$and $\mathcal{I}^+$. This is usually carried out by selecting the k unlabeled pairs (s,t) with positive (resp. negative) presumed class which lead to the highest disagreement amongst the classifiers. Several informativeness functions ifm have been used in literature to measure the disagreement. For example,the authors of [15] use the pairs which maximize

$$\text{ifm}(s,t) = (m - pos(s,t))(m - neg(s,t)), \tag{1}$$

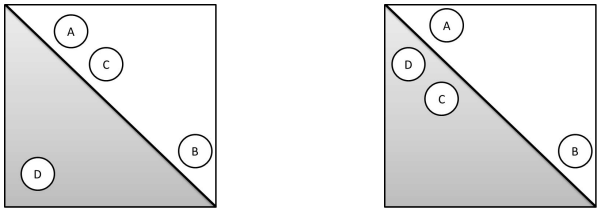
where $pos(s,t)$ stands for the number of classifiers which assign (s,t) the presumed class $+1$, while $neg(s,t)$ stands for the number of classifiers which assign (s,t) the class -1. The authors of [7] on the other hand rely on pairs (s,t) which maximize the entropy score

$$\text{ifm}(s,t) = H\left(\frac{pos(s,t)}{m}\right) \text{ where } H(x) = -x\log(x) - (1-x)\log(1-x). \tag{2}$$

Note that these functions do not take the correlation between the different link candidates into consideration.

3 Correlation-Aware Active Learning of Link Specifications

The basic insight behind this paper is that the correlation between the features of the elements of $\mathcal{N}$ and $\mathcal{P}$ should play a role when computing the sets $\mathcal{I}^+$ and $\mathcal{I}^-$. In particular, two main factors affect the information content of a link candidate: its similarity to elements of its presumed class and to elements of the other class. For the sake of simplicity, we will assume that the presumed class of the link candidate of interest is $+1$. Our insights yet hold symmetrically for link candidates whose presumed class is -1.



(a) Intra-correlation (b) Inter-correlation

Fig. 1. Examples of correlations within classes and between classes. In each subfigure, the gray surface represent $\mathcal{N}$ while the white surface stands for $\mathcal{P}$. The oblique line is $\mathcal{C}$'s boundary.

Let $A = (s_A, t_A), B = (s_B, t_B) \in \mathcal{P}$ to be two link candidates which are equidistant from $\mathcal{C}$'s boundary. Consider Figure 1a, where $\mathcal{P}= \{A, B, C\}$ and $\mathcal{N}= \{D\}$. The link candidate B is on on average most distant from any other elements of $\mathcal{P}$. Thus, it is more likely to be a statistical outlier than A. Hence, making a classification error on B should not have the same impact as an erroneous classification of link candidate A, which is close to another presumably positive link candidate, C. Consequently, B should be considered less informative than A. Approaches that make use of this information are said to exploit the *intra-class correlation*. Now, consider Figure 1b, where $\mathcal{P}= \{A, B\}$ and $\mathcal{N}= \{C, D\}$. While the probability of A being an outlier is the same as B's, A is still to be considered more informative than B as it is located closer to elements of $\mathcal{N}$ and can thus provide more information on where to set the classifier boundary. This information is dubbed *inter-class correlation*.

4 Approaches

Several approaches that make use of these two types of correlations can be envisaged. In the following, we present two approaches for these purposes. The first makes use of intra-class correlations and relies on graph clustering. The second approach relies on the spreading activation principle in combination with weight decay. We assume that the complex similarity function σ underlying $\mathcal{C}$ is computed by combining n atomic similarity functions $\sigma_1, \ldots, \sigma_n$. This combination is most commonly carried out by using metric operators such as min, max or linear combinations.[5] Consequently, each link candidate (s, t) can be described by a vector $(\sigma_1(s, t), \ldots, \sigma_n(s, t)) \in [0, 1]^n$. We define the *similarity of link candidates* $sim : (S \times T)^2 \to [0, 1]$ to be the inverse of the Euclidean distance in the space spawned by the similarities σ_1 to σ_n. Hence, the similarity of two link candidates (s, t) and (s', t') is given by:

[5] See [12] for a more complete description of a grammar for link specifications.

$$sim((s,t),(s',t')) = \frac{1}{1+\sqrt{\sum_{i=1}^{n}(\sigma_i(s,t)-\sigma_i(s',t'))^2}}. \quad (3)$$

Note that we added 1 to the denominator to prevent divisions by 0.

4.1 Graph Clustering

The basic intuition behind using clustering for COALA is that groups of very similar link candidates can be represented by a single link candidate. Consequently, once a representative of a group has been chosen, all other elements of the group become less informative. An example that illustrates this intuition is given in Figure 2. We implemented COALA based on clustering as shown in Algorithm 1. In each iteration, we begin by first selecting two sets $\mathcal{S}^+ \subseteq \mathcal{P}$ resp. $\mathcal{S}^- \subseteq \mathcal{N}$ that contain the positive resp. negative link candidates that are most informative for the classifier at hand. Formally, $\mathcal{S}^+$ fulfills

$$\forall x \in \mathcal{S}^+ \; \forall y \in \mathcal{P}, y \notin \mathcal{S}^+ \rightarrow \text{ifm}(y) \leq \text{ifm}(x). \quad (4)$$

The analogous equation holds for $\mathcal{S}^-$. In the following, we will explain the further steps of the algorithm for $\mathcal{S}^+$. The same steps are carried out for $\mathcal{S}^-$. First, we

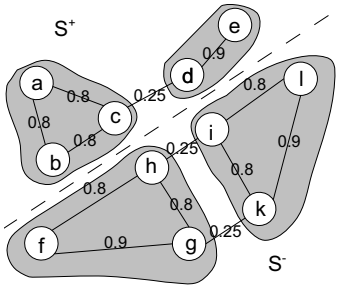


Fig. 2. Example of clustering. One of the most informative single link candidate is selected from each cluster. For example, d is selected from the cluster $\{d, e\}$.

compute the similarity of all elements of $\mathcal{S}^+$ by using the similarity function shown in Equation 3. In the resulting similarity matrix, we set all elements of the diagonal to 0. Then, for each $x \in \mathcal{S}^+$, we only retain a fixed number ec of highest similarity values and set all others to 0. The resulting similarity matrix is regarded as the adjacency matrix of an undirected weighted graph $G = (V, E, sim)$. G's set of nodes V is equal to $\mathcal{S}^+$. The set of edges E is a set of 2-sets[6] of link candidates. Finally, the weighted function is the similarity

[6] A n-set is a set of magnitude n.

function *sim*. Note that *ec* is the minimal degree of nodes in G. In a second step, we use the graph G as input for a graph clustering approach. The resulting clustering is assumed to be a partition $\mathcal{V}$ of the set V of vertices of G. The informativeness of partition $V_i \in \mathcal{V}$ is set to $\max_{x \in V_i} \text{ifm}(x)$. The final step of our approach consists of selecting the most informative node from each of the k most informative partitions. These are merged to generate $\mathcal{I}^+$, which is sent as query to the oracle. The computation of $\mathcal{I}^-$ is carried out analogously. Note that this approach is generic in the sense that it can be combined with any graph clustering algorithm that can process weighted graphs as well as with any informativeness function ifm. Here, we use BorderFlow [16] as clustering algorithm because (1) it has been used successfully in several other applications [9,10] and (2) it is parameter-free and does not require any tuning.

Algorithm 1. COALA based on Clustering

input : mappingSet set of mappings, exampleCount number of examples, edgesPerNode maximal number of edges per node
output: list of mappings for the oracle oracleList

$\mathcal{S}^-$:=get closest negative mapppings(*mappingSet*)
$\mathcal{S}^+$:= get closest positive mapppings(*mappingSet*)
clusterSet:= $\emptyset$
for set $\in \{\mathcal{S}^-, \mathcal{S}^+\}$ **do**
  G := buildGraph(set,edgesPerNode)
  clusterSet $\leftarrow$ *clustering*(G)
  visitedClusters := $\emptyset$, *addedElements* :=0
  sortedMappingList := sortingByDistanceToClassfier(mappingSet)
  repeat
  (s, t):= next(*sortedMappingList*)
  partition:=getPartition((s, t))
  if *partition* $\notin$ *visitedClusters* **then**
    oracleList:=add((s, t))
    addedElements:=+1
    visitedClusters:=addCluster(*partition*)
  until *addedElements* = exampleCount

4.2 Spreading Activation with Weight Decay

The idea behind spreading activation with weight decay (WD) is to combine the intra- and inter-class correlation to determine the informativeness of each link candidate. Here, we begin by computing the set $\mathcal{S} = \mathcal{S}^+ \cup \mathcal{S}^-$, where $\mathcal{S}^+$ and $\mathcal{S}^-$ are described as above. Let $\mathfrak{s}_i$ and $\mathfrak{s}_j$ be the i^{th} and j^{th} elements of $\mathcal{S}$. We then compute the quadratic similarity matrix $\mathcal{M}$ with entries $m_{ij} = sim(\mathfrak{s}_i, \mathfrak{s}_j)$ for $i \neq j$ and 0 else. Note that both negative and positive link candidates belong to $\mathcal{S}$. Thus, $\mathcal{M}$ encodes both inter- and intra-class correlation. In addition to $\mathcal{M}$,

we compute the activation vector $\mathcal{A}$ by setting its entries to $a_i = \text{ifm}(\mathfrak{s}_i)$. In the following, $\mathcal{A}$ is considered to be a column vector. The spreading of the activation with weight decay is then carried out as shown in Algorithm 2.

Algorithm 2. COALA based on Weight Decay

input : mappingSet set of mappings, r fix point exponent, exampleCount number of examples
output: oracleList list of mapping for the oracle
1 $\mathcal{M}$:= buildAdjacenceMatrix(mappingSet)
2 $\mathcal{A}$:= buildActivationVector(mappingSet)
3 **repeat**
4 $\mathcal{A} := \mathcal{A}/max_{\mathcal{A}}$
5 $\mathcal{A} := \mathcal{A} + \mathcal{M} \times \mathcal{A}$
6 $\mathcal{M} := (\forall m_{ij} \in \mathcal{M} : m_{ij} := m_{ij}^{\mathsf{r}})$
7 **until** $\forall m_{ij} \in \mathcal{M} | m_{ij} \neq 1 : m_{ij} \leq \epsilon$
8 oracleList:= getMostActivatedMapping($\mathcal{A}$,exampleCount)

In a first step, we normalize the activation vector $\mathcal{A}$ to ensure that the values contained therein do not grow indefinitely. Then, in a second step, we set $\mathcal{A} = \mathcal{A} + \mathcal{M} \times \mathcal{A}$. This has the effect of propagating the activation of each $\mathfrak{s}$ to all its neighbors according to the weights of the edges between $\mathfrak{s}$ and its neighbors. Note that elements of $\mathcal{S}^+$ that are close to elements of $\mathcal{S}^-$ get a higher activation than elements of $\mathcal{S}^+$ that are further away from $\mathcal{S}^-$ and vice-versa. Moreover, elements at the center of node clusters (i.e., elements that are probably no statistical outliers) also get a higher activation than elements that are probably outliers. The idea behind the weight decay step is to update the matrix by setting each m_{ij} to m_{ij}^r, where $r > 1$ is a fix exponent. This is the third step of the algorithm. Given that $\forall i \forall j\ m_{ij} \leq 1$, the entries in the matrix get smaller with time. By these means, the amount of activation transferred across long paths is reduced. We run this three-step procedure iteratively until all non-1 entries of the matrix are less or equal to a threshold $\epsilon = 10^{-2}$. The k elements of $\mathcal{S}^+$ resp. $\mathcal{S}^-$ with maximal activation are returned as $\mathcal{I}^+$resp. $\mathcal{I}^-$. In the example shown in Figure 3, while all nodes from $\mathcal{S}^+$ and $\mathcal{S}^-$ start with the same activation, two nodes get the highest activation after only 3 iterations.

5 Evaluation

The goal of our evaluation was to study the improvement in F-score achieved by integrating the approaches presented above with a correlation-unaware approach. We chose to use EAGLE [15], an approach based on genetic programming. We ran a preliminary experiment on one dataset to determine good parameter settings for the combination of EAGLE and clustering (CL) as well as the combination EAGLE and weight decay (WD). Thereafter, we compared the F-score achieved by EAGLE with that of CL and WD in ten different settings.

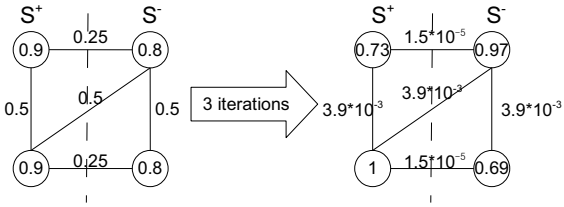


Fig. 3. Example of weight decay. Here r was set to 2. The left picture shows the initial activations and similarity scores while the right picture shows the results after 3 iterations. Note that for the sake of completeness the weights of the edges were not set to 0 when they reached ϵ.

5.1 Experimental Setup

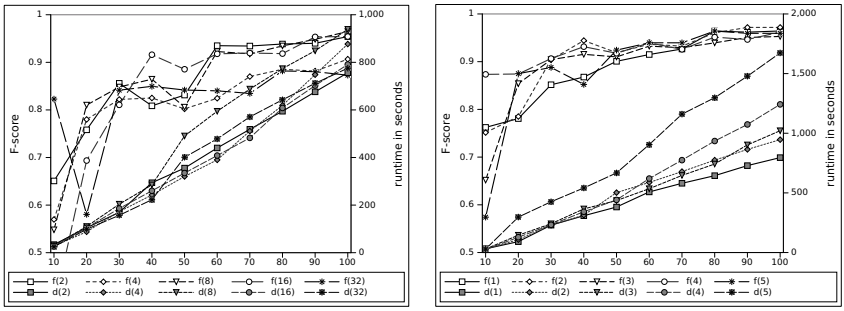
Throughout our experiments, we set both mutation and crossover rates to 0.6. Individuals were given a 70% chance to get selected for reproduction. The population sizes were set to 20 and 100. We set $k = 5$ and ran our experiments for 10 iterations, evolving the populations for 50 generations each iteration. We ran our experiments on two real-world datasets and three synthetic datasets. The synthetic datasets consisted of the datasets from the OAEI 2010 benchmark[7]. The real-world datasets consisted of the ACM-DBLP and Abt-Buy datasets, which were extracted from websites or databases [8] [8]. The ACM-DBLP dataset consists of 2,617 source and 2,295 target publications with 2,224 links between them. The Abt-Buy dataset holds 1,092 links between 1,081 resp. 1,092 products. Note that this particular dataset is both noisy and incomplete. All non-RDF datasets were transformed into RDF and all string properties were set to lower case. Given that genetic programming is non-deterministic, all results presented below are the means of 5 runs. Each experiment was ran on a single thread of a server running JDK1.7 on Ubuntu 10.0.4 and was allocated maximally 2GB of RAM. The processors were 2.0GHz Quadcore AMD Opterons.

5.2 Results

Parametrization of WD and CL. In a preliminary series of experiments we tested for a good parametrization of both WD and CL. For this purpose we ran both approaches on the DBLP-ACM dataset using 5 different values for the r exponent for weight decay and the clustering ec parameter. The tests were ran with a population of 20, $r = \{2, 4, 8, 16, 32\}$ and $ec = \{1, 2, 3, 4, 5\}$. Figures 4a and 4b show the results of achieved F-scores and runtimes. In both plots $f(p)$ and $d(p)$ denote the F-score and runtime of the particular method using the

[7] http://oaei.ontologymatching.org/2010/

[8] http://dbs.uni-leipzig.de/en/research/projects/object_matching/fever/benchmark_datasets_for_entity_resolution



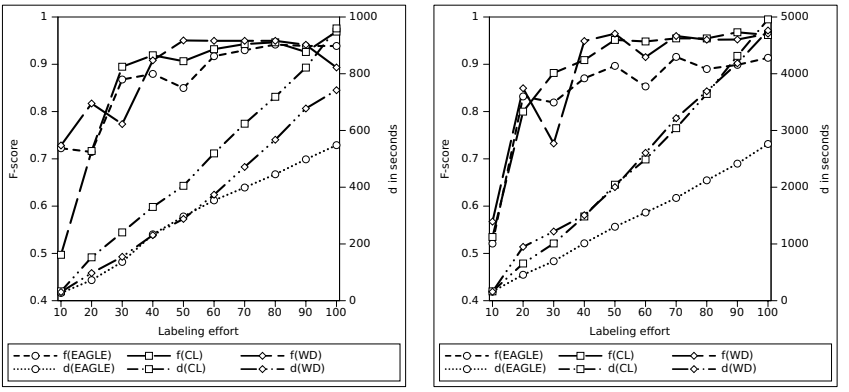
(a) r parameter of WD (b) ec for CL

Fig. 4. Testing different r and ec parameter for both approaches on the DBLP-ACM dataset. $f(p)$ denotes the F-score accieved with the method wusing the parameter p, while $d(p)$ denotes the reuired run time.

p parameter. Figure 4a suggests that $r = 2$ leads to a good accuracy (especially for later inquiries) while requiring moderate computation resources. Similarly, $r = 16$ promises fast convergence and led to better results in the fourth and fifth iterations. Still, we chose $r = 2$ for all experiments due to an overall better performance. The test for different ec parameters led us to use an edge limit of $ec = 3$. This value leads to good results with respect to both accuracy and runtime as Figure 4b suggests.

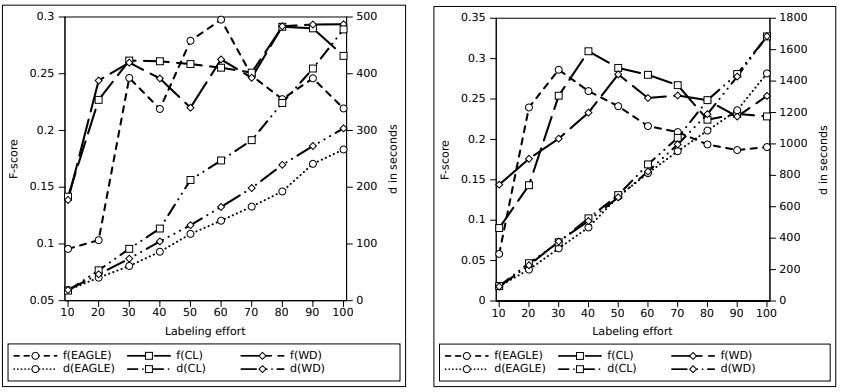
Runtime and F-Score. Figures 5 - 9 show the results of both our approaches in comparison to the EAGLE algorithm. And a summary of the results is given in Table 1. Most importantly, our results suggest that using correlation information can indeed improve the F-score achieved by curious classifiers. The average of the results achieved by the approaches throughout the learning process (left group of results in Table 1) shows that already in average our approaches outperform EAGLE in 9 from 10 settings. A look at the final F-scores achieved by the approaches show that one of the approaches WD and CL always outperform EAGLE both with respect to the average F-score and the standard deviation achieved across the 5 runs except on the Restaurant data set (100 popultion), where the results of CL and EAGLE are the same. This leads us to conclude that the intuition underlying this paper is indeed valid. Interestingly, the experiments presented herein do not allow declaring CL superior to WD or vice-versa. While CL performs better on the small population, WD catches up on larger populations and outperform CL in 3 of 5 settings. An explanation for this behavior could lie in WD taking more information into consideration and thus being more sensible to outliers than CL. A larger population size which reduces the number of outliers would then be better suited to WD. This explanation is yet still to be proven in larger series of experiments and in combination with other

link discovery approaches such as RAVEN. Running WD and CL is clearly more time-demanding than simply running EAGLE. Still the overhead remains within acceptable boundaries. For example, while EAGLE needs approx. 2.9s for 100 individuals on the Abt-Buy dataset while both WD and CL require 3.4s (i.e., 16.3% more time).



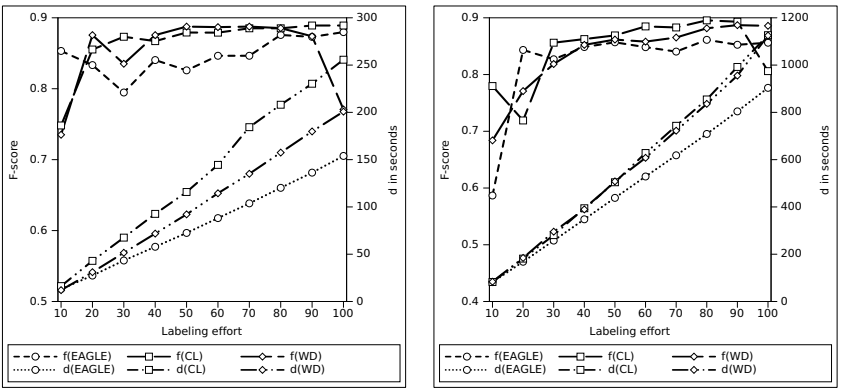
(a) Population = 20 (b) Population = 100

Fig. 5. F-score and runtime on the ACM-DBLP dataset. f(X) stands for the F-score achieved by algorithm X, while d(X) stands for the total duration required by the algorithm.



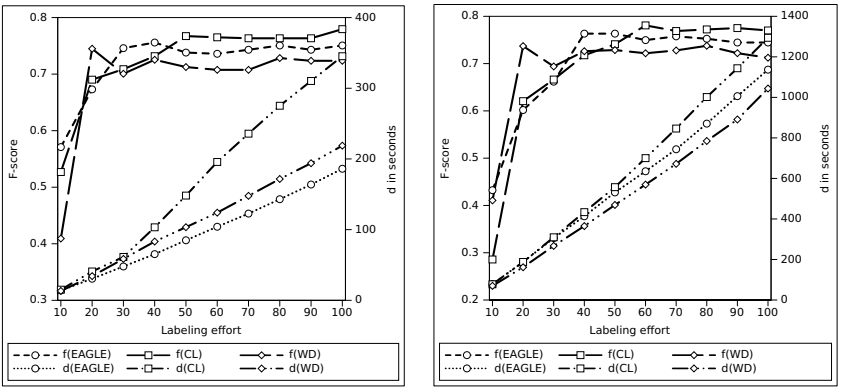
(a) Population = 20 (b) Population = 100

Fig. 6. F-score and runtime on the Abt-Buy dataset



(a) Population = 20 (b) Population = 100

Fig. 7. F-score and runtime on the OAEI 2010 Person1 dataset

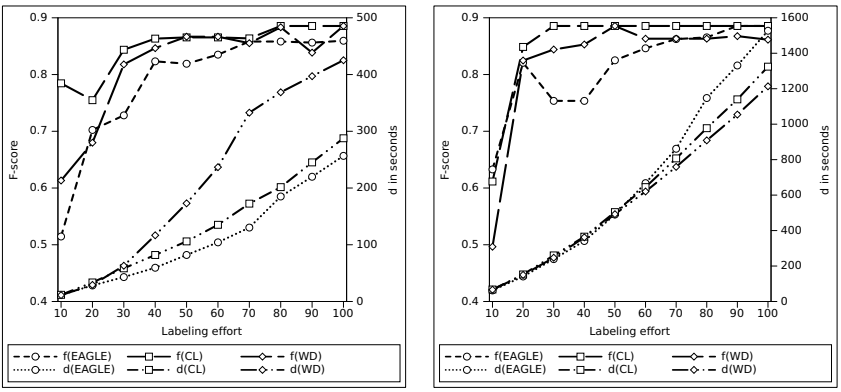


(a) Population = 20 (b) Population = 100

Fig. 8. F-score and runtime on the OAEI 2010 Person2 dataset

6 Related Work

The number of LD approaches has proliferated over the last years. Herein, we present a brief overview of existing approaches (see [11,7] for more extensive presentations of the state of the art). Overall, two main problems have been at the core of the research on LD. First, the time complexity of LD was addressed. In [13], an approach based on the Cauchy-Schwarz inequality was used to reduce the runtime of LD processes based on metrics. The approach $\mathcal{HR}^3$ [11] rely on space tiling in spaces with measures that can be split into independent measures across the dimensions of the problem at hand. Especially, $\mathcal{HR}^3$ was shown to be the first approach that can achieve a relative reduction ratio r' less or equal



(a) Population = 20 (b) Population = 100

Fig. 9. F-score and runtime on the OAEI 2010 Restaurant dataset

Table 1. Comparison of average F-scores achieved by EAGLE, WD and CL. The top section of the table shows the results for a population size of 20 while the bottom part shows the results for 100 individuals. Best scores are in bold font. Abt stands for Abt-Buy, DBLP for DBLP-ACM and Rest. for Restaurants.

	Average values			**Final values**		
DataSet	**EAGLE**	**WD**	**CL**	**EAGLE**	**WD**	**CL**
Abt	0.22± 0.06	**0.25 ± 0.07**	0.25 ± 0.08	0.22± 0.05	**0.29 ± 0.03**	0.27 ± 0.05
DBLP	0.87± 0.1	**0.89± 0.09**	0.87± 0.08	0.94± 0.02	0.89± 0.13	**0.97± 0.0**
Person1	0.85± 0.05	0.85± 0.06	**0.87± 0.03**	0.88± 0.02	0.77± 0.25	**0.89± 0.01**
Person2	0.72± 0.05	0.69± 0.11	**0.73± 0.08**	0.75± 0.02	0.72± 0.09	**0.78± 0.0**
Rest.	0.79± 0.13	0.82± 0.08	**0.85± 0.05**	0.51± 0.36	0.61± 0.28	**0.78± 0.01**
Abt	0.21 ± 0.06	0.23± 0.07	**0.23± 0.05**	0.19 ± 0.04	**0.25± 0.04**	0.23± 0.04
DBLP	0.87± 0.1	0.89± 0.09	**0.89± 0.08**	0.91± 0.03	**0.96± 0.01**	0.96± 0.02
Person1	0.82± 0.05	**0.84± 0.07**	**0.84± 0.07**	0.86± 0.02	**0.89± 0.01**	0.81± 0.18
Person2	**0.7± 0.09**	0.69± 0.1	0.69± 0.07	0.74± 0.03	0.71± 0.08	**0.77± 0.03**
Rest.	0.81± 0.11	0.82± 0.06	**0.85± 0.03**	**0.89± 0.0**	0.86± 0.02	**0.89± 0.0**

to any given relative reduction ratio $r > 1$. Concepts from the deduplication research field were also employed for LD. For example, standard blocking approaches were implemented in the first versions of SILK[9] and later replaced with MultiBlock [6], a lossless multi-dimensional blocking technique. KnoFuss [17] also implements blocking techniques to achieve acceptable runtimes. Moreover, time-efficient string comparison algorithms such as PPJoin+ [21] were integrated into the hybrid framework LIMES [12]. Other LD frameworks can be found in the results of the ontology alignment evaluation initiative [3]. The second problem

[9] `http://wifo5-03.informatik.uni-mannheim.de/bizer/silk/`

that was addressed is the complexity of link specifications. Although unsupervised techniques were newly developed (see, e.g., [17]), most of the approaches developed so far abide by the paradigm of supervised machine learning. For example, the approach presented in [5] relies on large amounts of training data to detect accurate link specification using genetic programming. RAVEN [14] is (to the best of our knowledge) the first active learning technique for LD. The approach was implemented for linear or Boolean classifiers and shown to require a small number of queries to achieve high accuracy. While the first active genetic programming approach was presented in [4], similar approaches for LD were developed later [7,15]. Still, none of the active learning approaches for LD presented in previous work made use of the similarity of unlabeled link candidates to improve the convergence of curious classifiers. Yet, works in other research areas have started considering the combination of active learning with graph algorithms (see e.g., [2]).

7 Conclusion

We presented the first generic LD approaches that make use of the correlation between positive and negative link candidates to achieve a better convergence. The first approach is based on clustering and only makes use of correlations within classes while the second algorithm makes use of both correlations within and between classes. We compared these approaches on 5 datasets and showed that we achieve better F-scores and standard deviations than the EAGLE algorithm. Thus, in future work, we will integrate our approach into other algorithms such as RAVEN. Moreover, we will measure the impact of the graph clustering algorithm utilized in the first approach on the convergence of the classifier. Our experimental results showed that each of the approaches we proposed has its pros and cons. We will thus explore combinations of WD and CL.

References

1. Auer, S., Lehmann, J., Ngonga Ngomo, A.-C.: Introduction to linked data and its lifecycle on the web. In: Polleres, A., d'Amato, C., Arenas, M., Handschuh, S., Kroner, P., Ossowski, S., Patel-Schneider, P. (eds.) Reasoning Web 2011. LNCS, vol. 6848, pp. 1–75. Springer, Heidelberg (2011)
2. Bodó, Z., Minier, Z., Csató, L.: Active learning with clustering. Journal of Machine Learning Research - Proceedings Track 16, 127–139 (2011)
3. Euzenat, J., Ferrara, A., van Hage, W.R., Hollink, L., Meilicke, C., Nikolov, A., Ritze, D., Scharffe, F., Shvaiko, P., Stuckenschmidt, H., Sváb-Zamazal, O., dos Santos, C.T.: Results of the ontology alignment evaluation initiative 2011. In: OM (2011)
4. de Freitas, J., Pappa, G., da Silva, A., Gonçalves, M., Moura, E., Veloso, A., Laender, A., de Carvalho, M.: Active learning genetic programming for record deduplication. In: 2010 IEEE Congress on Evolutionary Computation Evolutionary Computation (CEC), pp. 1–8 (2010)

5. Isele, R., Bizer, C.: Learning linkage rules using genetic programming. In: OM. CEUR Workshop Proceedings, vol. 814 (2011)
6. Isele, R., Jentzsch, A., Bizer, C.: Efficient multidimensional blocking for link discovery without losing recall. In: Marian, A., Vassalos, V. (eds.) WebDB (2011)
7. Isele, R., Jentzsch, A., Bizer, C.: Active learning of expressive linkage rules for the web of data. In: Brambilla, M., Tokuda, T., Tolksdorf, R. (eds.) ICWE 2012. LNCS, vol. 7387, pp. 411–418. Springer, Heidelberg (2012)
8. Köpcke, H., Thor, A., Rahm, E.: Comparative evaluation of entity resolution approaches with fever. Proc. VLDB Endow. 2(2), 1574–1577 (2009)
9. Morsey, M., Lehmann, J., Auer, S., Ngonga Ngomo, A.-C.: DBpedia SPARQL Benchmark – Performance Assessment with Real Queries on Real Data. In: Aroyo, L., Welty, C., Alani, H., Taylor, J., Bernstein, A., Kagal, L., Noy, N., Blomqvist, E. (eds.) ISWC 2011, Part I. LNCS, vol. 7031, pp. 454–469. Springer, Heidelberg (2011)
10. Ngonga Ngomo, A.C.: Parameter-free clustering of protein-protein interaction graphs. In: Proceedings of MLSB Symposium (2010)
11. Ngonga Ngomo, A.-C.: Link Discovery with Guaranteed Reduction Ratio in Affine Spaces with Minkowski Measures. In: Cudré-Mauroux, P., et al. (eds.) ISWC 2012, Part I. LNCS, vol. 7649, pp. 378–393. Springer, Heidelberg (2012)
12. Ngonga Ngomo, A.C.: On link discovery using a hybrid approach. Journal on Data Semantics 1, 203–217 (2012)
13. Ngonga Ngomo, A.C., Auer, S.: LIMES - A Time-Efficient Approach for Large-Scale Link Discovery on the Web of Data. In: Proceedings of IJCAI, pp. 2312–2317 (2011)
14. Ngonga Ngomo, A.C., Lehmann, J., Auer, S., Höffner, K.: RAVEN – Active Learning of Link Specifications. In: Proceedings of OM@ISWC (2011)
15. Ngonga Ngomo, A.-C., Lyko, K.: EAGLE: Efficient active learning of link specifications using genetic programming. In: Simperl, E., Cimiano, P., Polleres, A., Corcho, O., Presutti, V. (eds.) ESWC 2012. LNCS, vol. 7295, pp. 149–163. Springer, Heidelberg (2012)
16. Ngonga Ngomo, A.-C., Schumacher, F.: BorderFlow: A local graph clustering algorithm for natural language processing. In: Gelbukh, A. (ed.) CICLing 2009. LNCS, vol. 5449, pp. 547–558. Springer, Heidelberg (2009)
17. Nikolov, A., d'Aquin, M., Motta, E.: Unsupervised learning of link discovery configuration. In: Simperl, E., Cimiano, P., Polleres, A., Corcho, O., Presutti, V. (eds.) ESWC 2012. LNCS, vol. 7295, pp. 119–133. Springer, Heidelberg (2012)
18. Settles, B.: Active learning literature survey. Computer Sciences Technical Report 1648, University of Wisconsin–Madison (2009)
19. Shekarpour, S., Auer, S., Ngonga Ngomo, A.C., Gerber, D., Hellmann, S., Stadler, C.: Keyword-driven sparql query generation leveraging background knowledge. In: International Conference on Web Intelligence (2011)
20. Unger, C., Bühmann, L., Lehmann, J., Ngonga Ngomo, A.-C., Gerber, D., Cimiano, P.: Sparql template-based question answering. In: Proceedings of WWW (2012)
21. Xiao, C., Wang, W., Lin, X., Yu, J.X.: Efficient similarity joins for near duplicate detection. In: WWW, pp. 131–140 (2008)