Ligon – Link Discovery with Noisy Oracles

Mohamed Ahmed Sherif[1], Kevin Dreßler[2], and Axel-Cyrille Ngonga Ngomo[1,2]

[1] Paderborn University, Data Science Group, Technologiepark 6, 33100 Paderborn, Germany
E-mail: {firstname.lastname}@upb.de
[2] Department of Computer Science, University of Leipzig, 04109 Leipzig, Germany
E-mail: {lastname}@informatik.uni-leipzig.de

Abstract. Link discovery plays a key role in the integration and use of data across RDF knowledge graphs. Active learning approaches are a common family of solutions to address the problem of learning how to compute links from users. So far, only active learning from perfect oracles has been considered in the literature. However, real oracles are often far from perfect (e.g., in crowdsourcing). We hence study the problem of learning how to compute links across knowledge graphs from noisy oracles, i.e., oracles that are not guaranteed to return correct classification results. We present a novel approach for link discovery based on a probabilistic model, with which we estimate the joint odds of the oracles' guesses. We combine this approach with an iterative learning approach based on refinements. The resulting method, Ligon, is evaluated on 10 benchmark datasets. Our results suggest that Ligon configured with 10 iterations and 10 training examples per iteration achieves more than 95% of the F-measure achieved by state-of-the-art algorithms trained with a perfect oracle. Moreover, Ligon outperforms batch learning approaches devised to be trained with small amounts of training data by more than 40% F-measure on average.

1 Introduction

The provision of links between knowledge graphs in RDF[3] is of central importance for numerous tasks on the Semantic Web, including federated queries, question answering and data fusion. While links can be created manually for small knowledge bases, the sheer size and number of knowledge bases commonly used in modern applications (e.g., DBpedia with more than 3×10^6 resources) demands the use of automated link discovery mechanisms. In this work, we focus on active learning for link discovery. State-of-the-art approaches that rely on active learning [5, 14, 10] assume that the oracle they rely upon is *perfect*. Formally, this means that given an oracle ω, the probability of the oracle returning a wrong result (i.e., returning `false` when an example is to be classified as `true`) is exactly 0. While these approaches show pertinent results in evaluation scenarios, within which the need for a perfect oracle can be fulfilled, this need is difficult if not impossible to uphold in real-world settings (e.g., when crowdsourcing training data). No previous work has addressed link discovery based on oracles that are not perfect.

We address this research gap by presenting a novel approach for learning link specifications (LS) from *noisy oracles*, i.e., oracles that are not guaranteed to return correct classifications. This approach is motivated by the problem of learning LS using crowdsourcing. Previous works have shown that agents in real crowdsourcing scenarios are often not fully reliable (e.g., [24]). We model these agents as noisy oracles, which provides erroneous answers to questions with a fixed probability. We address the problem of learning from such oracles by using a probabilistic model, which

[3] See https://www.w3.org/RDF/.

approximates the odds of the answer of a set of oracles being correct. Our approach, dubbed LIGON, assumes that the underlying oracles are *independent*, i.e., that the probability distributions underlying oracles are pairwise independent. Moreover, we assume that the oracles have a *static behavior*, i.e., that the probability of them generating correct/incorrect answers is constant over time.

The contributions of this paper are as follows: (1) We present a formalization of the problem of learning LS from noisy oracles. We derive a probabilistic model for learning from such oracles. (2) We develop the first learning algorithm dedicated to learning LS from noisy data. The approach combines iterative operators for LS with an entropy-based approach for selecting most informative training examples. In addition, it uses cumulative evidence to approximate the probability distribution underlying the noisy oracles that provide it with training data. Finally, (3) we present a thorough evaluation of LIGON and show that it is robust against noise, scales well and converges with 10 learning iterations to more than 95% of the average F-measure achieved by WOMBAT—a state-of-the-art approach for learning LS—provided with a perfect oracle.

2 Preliminaries

Knowledge graphs (also called knowledge bases) in RDF are defined as sets of triples $K \subseteq (\mathcal{R} \cup \mathcal{B}) \times \mathcal{P} \times (\mathcal{R} \cup \mathcal{B} \cup \mathcal{L})$, where $\mathcal{R}$ is the set of all resources, i.e., of all objects in the domain of discourse (e.g., persons and publications); $\mathcal{P} \subseteq \mathcal{R}$ is the set of all predicates, i.e., of binary relations (e.g., author); $\mathcal{B}$ is the set of all blank nodes, which basically stand for resources whose existence is known but whose identity is not relevant to the model and $\mathcal{L}$ is the set of all literals, i.e., of values associated to datatypes (e.g., integers).[4] The elements of K are referred to as *facts* or *triples*. We call the elements of $\mathcal{R}$ *entities* or *resources*.

The *link discovery* task on RDF knowledge graphs is defined as follows: Let S and T be two sets of resources, i.e., $S \subseteq \mathcal{R}$ and $T \subseteq \mathcal{R}$. Moreover, let $r \in \mathcal{P}$ be a predicate. The aim of link discovery is to compute the set $M = \{(s,t) \in S \times T : r(s,t)\}$. We call M a mapping. In many cases, M cannot be computed directly and is thus approximated by a mapping M'. To find the set M', declarative link discovery frameworks rely on *link specifications* (LS), which describe the conditions under which $r(s,t)$ can be assumed to hold for a pair $(s,t) \in S \times T$. Several formal models have been used for describing LS in previous works [10]. We adopt a formal approach derived from [19] and first describe the syntax and then the semantics of LS.

LS consist of two types of atomic components: *similarity measures* m, which allow the comparing of property values of input resources and *operators* op, which can be used to combine LS to more complex LS. Without loss of generality, we define a similarity measure m as a function $m : S \times P \times T \times P \to [0,1]$. An example of a similarity measure is the edit similarity dubbed `edit`[5] which allows computing the similarity of a pair $(s,t) \in S \times T$ w.r.t. the values of a pair of properties (p_s, p_t) for s resp. t. An *atomic LS* is a pair (m, θ). A *complex LS* is the result of combining two LS L_1 and L_2 through an *operator* that allows merging the results of L_1 and L_2. Here, we use the operators $\sqcap$, $\sqcup$ and $\backslash$ as they are complete w.r.t. the Boolean algebra and frequently used to define LS. An example of a complex LS is given in Figure 1.

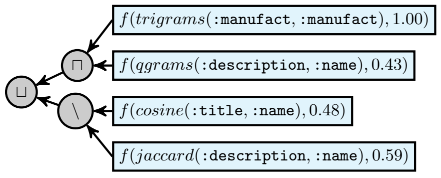
We define the semantics $[[L]]_\mu$ of a LS L w.r.t. a mapping μ as given in Table 1. The mapping $[[L]]$ of a LS L w.r.t. $S \times T$ contains the link candidates generated by L. A LS L is *subsumed* by L',

[4] See https://www.w3.org/RDF/ for more details.

[5] We define the edit similarity of two strings s and t as $(1 + lev(s,t))^{-1}$, where lev is the Levenshtein distance.

denoted by $L \sqsubseteq L'$, if for all mappings μ, we have $[[L]]_\mu \subseteq [[L']]_\mu$. Two LS are *equivalent*, denoted by $L \equiv L'$ iff $L \sqsubseteq L'$ and $L' \sqsubseteq L$. Subsumption ($\sqsubseteq$) is a partial order over the set of LS, denoted $\mathcal{L}$.

Fig. 1: Complex LS example. The filter nodes are rectangles while the operator nodes are circles.



$f(trigrams(\texttt{:manufact}, \texttt{:manufact}), 1.00)$
$\sqcap$
$f(qgrams(\texttt{:description}, \texttt{:name}), 0.43)$
$\sqcup$
$\setminus$
$f(cosine(\texttt{:title}, \texttt{:name}), 0.48)$
$f(jaccard(\texttt{:description}, \texttt{:name}), 0.59)$

Table 1: Link Specification Syntax and Semantics

LS	$[[LS]]_M$
$f(m, \theta)$	$\{(s,t) \mid (s,t) \in M \wedge m(s,t) \geq \theta\}$
$L_1 \sqcap L_2$	$\{(s,t) \mid (s,t) \in [[L_1]]_M \wedge (s,t) \in [[L_2]]_M\}$
$L_1 \sqcup L_2$	$\{(s,t) \mid (s,t) \in [[L_1]]_M \vee (s,t) \in [[L_2]]_M\}$
$L_1 \backslash L_2$	$\{(s,t) \mid (s,t) \in [[L_1]]_M \wedge (s,t) \notin [[L_2]]_M\}$

3 Noisy Oracles

We model *oracles* Ω for r as black boxes with a characteristic function $\omega : S \times T \rightarrow \{\texttt{true}, \texttt{false}\}$. The characteristic function ω_i of the oracle Ω_i returns `true` iff the oracle Ω_i assumes that $r(s,t)$ holds. Otherwise, it returns `false`. For ease of notation, we define $LC = S \times T$ and call the elements of LC *link candidates*. For $l \in LC$, we write $l \equiv \top$ to signify that $r(l)$ holds, i.e., $r(s,t)$ is true for $l = (s,t)$. Otherwise, we write $l \equiv \bot$. We now assume a learning situation typical for crowdsourcing, where n oracles are presented with a link candidate l and asked whether $l \equiv \top$ holds. We can describe each oracle Ω_i by the following four probabilities:

1. $p(\omega_i(l) = \texttt{true} \mid l \equiv \top)$, i.e., the probability of the oracle Ω_i generating true positives. This value is exactly 1 for a perfect oracle.
2. $p(\omega_i(l) = \texttt{false} \mid l \equiv \top)$, the probability of false negatives (0 for a perfect oracle).
3. $p(\omega_i(l) = \texttt{true} \mid l \equiv \bot)$, i.e., the probability of false positives, (0 for a perfect oracle).
4. $p(\omega_i(l) = \texttt{false} \mid l \equiv \bot)$, the probability of true negatives (1 for a perfect oracle).

Given that $p(A|B) + p(\neg A|B) = 1$, the sum of the first two and last two probabilities is always 1.

Example 1. A noisy oracle can have the following description: $p(\omega_i(l){=}\texttt{true}|l{\equiv}\top){=}0.7, p(\omega_i(l){=}\texttt{true}|l{\equiv}\bot){=}0.5, p(\omega_i(l){=}\texttt{false}|l{\equiv}\top){=}0.3, p(\omega_i(l){=}\texttt{false}|l{\equiv}\bot){=}0.5$.

For compactness, we use the following vector notation in the rest of the formal model: $\overrightarrow{\omega}$ refers to the vector of characteristic functions over all oracles. We write $\overrightarrow{\omega}(l) = \overrightarrow{x}$ to signify that the ith oracle returned the x_i for the link candidate l. Let us assume that the probabilities underlying all oracles Ω_i are known (we discuss ways to initialize and update these probabilities in the subsequent section). Recalling that we assume that our oracles are independent, we can now approximate the probability that $l \equiv y$ (with $y \in \{\top, \bot\}$) for any given link candidate l using the following Bayesian model:

$$p(l{=}y|\overrightarrow{w}{=}\overrightarrow{x}){=}\frac{\prod_{i=1}^{n} p(\omega_i{=}x_i|l{\equiv}y)}{\prod_{i=1}^{n} p(\omega_i{=}x_i)} p(l{\equiv}y) \quad (1)$$

Recall that the *odds* of an event A occurring are defined as $odds(A) = p(A)/P(\neg A)$. For example, the odds of any link candidate being a correct link (denoted o^+) are given by

$$o^+ = \frac{p(l \equiv \top)}{p(l \equiv \bot)} \text{ for any } l \in LC. \quad (2)$$

o^+ is independent of l and stands for the odds that an element of LC chosen randomly would be a link. Given feedback from our oracles, we can approximate the odds of a link candidate l being a correct link by computing the following:

$$odds(l{\equiv}\top|\overrightarrow{w}{=}\overrightarrow{x}){=}\frac{p(l{\equiv}\top|\overrightarrow{w}{=}\overrightarrow{x})}{p(l{\equiv}\bot|\overrightarrow{w}{=}\overrightarrow{x})}{=}\left(\prod_{i=1}^{n}\frac{p(\omega_i{=}x_i|l{\equiv}\top)}{p(\omega_i{=}x_i|l{\equiv}\bot)}\right)\frac{p(l{\equiv}\top)}{p(l{\equiv}\bot)}{=}\left(\prod_{i=1}^{n}\frac{p(\omega_i{=}x_i|l{\equiv}\top)}{p(\omega_i{=}x_i|l{\equiv}\bot)}\right)o^+. \quad (3)$$

A key idea behind our model is that a link candidate l can be considered to be a correct link if $odds(l \equiv \top|\overrightarrow{w} = \overrightarrow{x}) \geq k$ with $k > 1$. A link candidate is assumed to not be a link if $odds(l|\overrightarrow{w} = \overrightarrow{x}) \leq 1/k$. All other link candidates remain unclassified.

Computing the odds for a link now boils down to (1) approximating the four probabilities which characterize our oracles and (2) computing o^+.

4 Computing o^+

As known from previous works on probabilistic models [9], o^+ is hard to compute directly as it requires knowing the set of links M, which is exactly what we are trying to compute. Several strategies can be used to approximate o^+. In this work, we consider the following three:

1. *Ignore strategy:* We can assume the probabilities $p(l = \top)$ and $p(l = \bot)$ to be equally unknown and hence use $o^+ = 1$. This reduces Equation 3 to

$$odds(l{=}\top|\overrightarrow{w}{=}\overrightarrow{x}){=}\prod_{i=1}^{n}\frac{p(\omega_i{=}x_i|l{\equiv}\top)}{p(\omega_i{=}x_i|l{\equiv}\bot)}. \quad (4)$$

2. *Equivalence strategy:* If r is an equivalence relation (e.g., `owl:sameAs`), then the set of all possible candidates has the size $|S||T|$. There can be at most $\min(|S|,|T|)$ links between S and T as no two pairs (s,t) and (s,t') can be linked if $t \neq t'$ and vice-versa (see [15]). Hence,

$$o^+ \approx \frac{\min(|S|,|T|)}{|S||T| - \min(|S|,|T|)}. \quad (5)$$

3. *Approximate strategy:* We approximate o^+ by using our learning approach. We select the mapping $[[L^*]]$ computed using the best specification L^* learned by LIGON (see the subsequent Section) as our best current approximation of the mapping we are trying to learn. o^+ is then computed as follows:

$$o^+ \approx \frac{|[[L^*]]|}{|S||T| - |[[L^*]]|}. \quad (6)$$

We quantify the effect of these strategies on our learning algorithm in our experiments.

5 The LIGON approach

LIGON is an active learning algorithm designed to learn LS from noisy oracles. An overview of the approach is given in Algorithm 1 and explained in the sections below.

Algorithm 1: LIGON Learning Algorithm

```
Input: Set of positive examples $E_0 \subseteq LC$; Oracles $\Omega_1, \ldots, \Omega_n$; Odds parameter $k$
1  $j \longleftarrow 0$ ;
2  foreach oracle $\Omega_i$ do
3      Initialize confusion matrix $C_i$ with $\frac{1}{2}$;
4  repeat
5      foreach oracle $\Omega_i$ do
6          Gather $\omega_i(l)$ for each $l \in E_j$;
7          Update the confusion matrix $C_i$ ;
8          Update the characteristic matrix $D_i$;
9      Train ACTIVE LEARNER (WOMBAT by default) using $\bigcup_{i=0}^{j} E_i$;
10     Compute the set of the most informative unlabeled examples $E^*$;
11     foreach link candidate $l \in E^*$ do
12         Get the oracle result vector $\overrightarrow{x}$ for $l$;
13     Compute the set $E^+$ of positive examples with $odds(l = \top|\overrightarrow{w} = \overrightarrow{x}) \geq k$ ;
14     Compute the set $E^-$ of negative examples with $odds(l = \top|\overrightarrow{w} = \overrightarrow{x}) \leq \frac{1}{k}$ ;
15     $j \longleftarrow j + 1$ ;
16     $E_j \longleftarrow E^+ \cup E^-$;
17 until termination criterion holds;
18 return best link specification;
```

Confusion Matrices. We begin by assuming that we are given an initial set $E_0 \subseteq LC$ of positive and negative examples for links. In the first step, we aim to compute the initial approximations of the conditional probabilities which describe each of the oracles Ω_i. To this end, each oracle is assigned a confusion matrix C_i of dimension 2×2 (see lines 2-3 of Algorithm 1). Each entry of the matrix is initialized with $\frac{1}{2}$ to account for potential sampling biases due to high disparities between conditional probabilities. The first and second row of each C_i contains counts for links where the oracle returned `true` resp. `false`. The first and second column of C_i contain *counts* for positive resp. negative examples. Hence, C_{11} basically contains counts for positive examples that were rightly classified as $\top$ by the oracle. In each learning iteration, we update the confusion matrix by presenting the oracle with unseen link candidates and incrementing the entries of C (see lines 4-8 of Algorithm 1). We discuss the computation of the training examples in the subsequent section. Based on the confusion matrix, we can approximate all conditional probabilities necessary to describe the oracle by computing the 2×2 matrix D with $d_{ij} = c_{ij}/(c_{1j} + c_{2j})$. For example, $d_{11} \approx p(\omega_i(l) = \texttt{true}|l \equiv \top)$. We call D the *characteristic matrix* of Ω.

Example 2. Imagine an oracle were presented with a set of 5 positive and 5 negative training examples, of which he classified 4 resp. 3 correctly. We get
$\mathrm{C} = \begin{bmatrix} \frac{9}{2} & \frac{5}{2} \\ \frac{3}{2} & \frac{7}{2} \end{bmatrix}$ and $D = \begin{bmatrix} \frac{9}{12} & \frac{5}{12} \\ \frac{3}{12} & \frac{7}{12} \end{bmatrix}$.

Updating the Characteristic Matrices. Updating the probabilities is done via the confusion matrices. In each learning iteration, we present all oracles with the link candidates deemed to be most informative. Based on the answers of the oracles, we compute the odds for each of these link candidates. Link candidates l with odds in $[0, 1/k]$ and $[k, +\infty[$ are considered to be false respectively true. The new classifications are subsequently used to update the counts in the confusion matrices and therewith also the characteristic matrix of each of the oracles.

Active Learning Approach. So far, we have assumed the existence of an active learning solution for link discovery. Several active learning approaches have been developed over recent years [10]. Of these approaches, solely those based on genetic programming can generate specifications of arbitrary complexity. However, genetic programming approaches are not deterministic and are thus difficult to use in practical applications. Newer approaches based on iterative operators such as WOMBAT [19] have been shown to perform well in classical link discovery tasks. Therefore, we implemented a generic interface to apply LIGON to several active learning algorithms, where we used the WOMBAT algorithm as the default active learning algorithm for LIGON. See Section 6 for results of applying LIGON to other state-of-the-art active learning approaches.

Selecting the Most Informative Examples. Given an active learning algorithm, we denote the set of the m best LS generated in a given iteration i as B_i. The most informative examples are those link candidates l, which maximize the decision entropy across the elements of B_i [14]. Formally, let $[[B_i]]$ be the union of the set of link candidates generated by all LS $b \in B_i$. Then, the most informative link candidates are the $l \in B_i$ which maximize the entropy function $e(l, B_i)$, which is defined as follows: Let $p(l, B_i)$ be the probability that a link candidate belongs to $[[b]]$ for $b \in B_i$. Then, $e(l, B_i) = -p(l, B_i) \log_2 p(l, B_i)$.

Example 3. Let us assume $|B_i| = 4$. A link candidate l returned by two of the LS in B_i would have a probability $p(l, B_i) = 0.5$. Hence, it would have an entropy $e(l, B_i) = 0.5$.

Termination Criterion. LIGON terminates after a set number of iterations has been achieved or if a link specification learned by WOMBAT achieves an F-measure of 1 on the training data.

6 Experiments and Results

We aimed to answer 6 research questions with our experimental evaluation: Q_1. Which combination of strategies for computing odds and the threshold k leads to the best performance?, Q_2. How does LIGON behave when provided with an increasing number of noisy oracles?, Q_3. How well does LIGON learn from noisy oracles?, Q_4. How well does LIGON scale?, Q_5. How well does LIGON perform compare to batch learning approaches trained with a similar number of examples? and Q_6. How general is LIGON, i.e., can LIGON be applied to problems outside the link discovery domain? and does LIGON depend on the underlying active learning algorithm?

Hardware and Datasets. All experiments were carried out on a 64-core 2.3 GHz PC running *OpenJDK* 64-Bit Server 1.8.0_151 on *Ubuntu* 16.04.3 LTS. Each experiment was assigned 20 GB RAM. We evaluated LIGON using 8 benchmark datasets. Five of these benchmarks were real-world datasets [8] while three were synthetic from the OAEI 2010 benchmark.[6] An overview of the characteristics of the benchmark datasets is given in Table 2.

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

Table 2: Benchmark datasets overview.$|M^*|$ stands for the size of the reference mapping.

Datasets	$\|S\|$	$\|T\|$	$\|S\| \times \|T\|$	$\|M^*\|$
Persons 1	500	500	0.25×10^6	500
Persons 2	600	400	0.24×10^6	400
Restaurants	113	641	0.07×10^6	112
ABT–Buy	1081	1092	1.2×10^6	1,097
Amazon–GoogleProducts	1363	2226	4.4×10^6	1,300
DBLP–ACM	2616	2294	6×10^6	2,224
DBP–LinkedMDB	1056	1056	1.1×10^6	1,056
DBLP–GoogleScholar	2616	2616	168.1×10^6	5,347

Table 3: Average learning iteration runtime analysis. All runtimes are in seconds.

Dataset	Ligon	Wombat
Persons 1	2.415	2.412
Persons 2	0.946	0.942
Restaurants	0.261	0.258
ABT–Buy	4.277	4.273
Amazon–GoogleProducts	2.848	2.844
DBLP–ACM	4.277	4.273
DBpedia–LinkedMDB	6.158	6.154
DBLP–GoogleScholar	16.072	16.067

Measures. We used the paradigm proposed by [7] and measured the performance of algorithms using the best F-measure they achieved. As this measure fails to capture the average behaviour of algorithm over several iterations, we also report the normalized average area under the F-measure curve, which we denote AUC.

LIGON Setup. We initialized Ligon with 10 positive examples (ergo, $|E_0| = 10$). We fixed the number of the most informative examples to be labeled by the noisy oracles at each iteration to 10. For labeling the most informative examples, we use $n = 2, 4, 8$ and 16 noisy oracles which were all initialized with random confusion matrices. We set the size of B to 10. All experiments were repeated 10 times and we report average values.

Noisy Oracle Setup. The characteristic matrices C of our noisy oracles were generated at random. To this end we generated the true positive and true negative probabilities using a uniform distribution between 0.5 and 1, i.e. $p(\omega_i(l) = \texttt{true}|l \equiv \top) \in [0.5, 1]$ and $p(\omega_i(l) = \texttt{true}|l \equiv \top) \in [0.5, 1]$. The other probabilities were set accordingly, as they are complementary to the former two.

Parameter Estimation. Our first series of experiments aimed to answer Q_1. We ran Ligon with $k = 2, 4, 8$ and 16. These settings were used in combination with all three strategies for computing o^+ aforementioned. A first observation is that the AUC achieved by Ligon does not depend much on the value of k nor on the strategy used. This is a highly positive feature of our algorithm as it suggests that our approach is robust w.r.t. to how it is initialized. Interestingly, this experiment already suggests that Ligon achieves more than 95% of the performance of the original Wombat algorithm trained with a perfect oracle. For more detailed results see Figure 2. We chose to run the remainder of our experiments with the setting $k = 16$ combined with the *equivalent* strategy as this combination achieved the highest average F-measure of 0.86.

Comparison with Perfect Oracle. In our second set of experiments, we answered Q_2 by measuring how well Ligon performed when provided with an increasing number of oracles. In this series of experiments, we used 2, 4, 8, and 16 oracles which were initialized randomly. k was set to 16 and we used the *Equivalent* strategy. Once more, the robustness of our approach became evident as its performance was not majorly perturbed by a variation in the number of oracles. In all settings (i.e., with all combination of 2, 4 and 16 oracles and the three strategies), Ligon achieves an average

Fig. 2: AUC heatmap of Ligon combined by our strategies for o^+ approximation against the perfect oracle for benchmark datasets.

Dataset	k	Approximate	Equivalent	Ignore	Perfect
Person 1	2	0.942	0.934	0.942	0.986
	4	0.942	0.939	0.942	
	8	0.942	0.936	0.943	
	16	0.942	0.955	0.948	
Person 2	2	0.941	0.950	0.941	0.995
	4	0.941	0.946	0.941	
	8	0.941	0.941	0.941	
	16	0.941	0.961	0.941	
Restaurants	2	0.987	0.987	0.987	0.987
	4	0.987	0.987	0.987	
	8	0.987	0.987	0.981	
	16	0.987	0.987	0.987	
ABT–Buy	2	0.822	0.822	0.822	0.892
	4	0.822	0.824	0.821	
	8	0.822	0.824	0.823	
	16	0.822	0.827	0.822	
Amazon–GoogleProducts	2	0.707	0.709	0.709	0.708
	4	0.707	0.708	0.709	
	8	0.707	0.707	0.709	
	16	0.707	0.711	0.706	
DBLP–ACM	2	0.698	0.698	0.700	0.771
	4	0.698	0.701	0.700	
	8	0.698	0.702	0.698	
	16	0.698	0.694	0.654	
DBpedia–LinkedMDB	2	0.899	0.900	0.899	0.961
	4	0.899	0.900	0.898	
	8	0.899	0.906	0.899	
	16	0.899	0.908	0.900	
DBLP–GoogleScholar	2	0.813	0.817	0.815	0.892
	4	0.815	0.820	0.839	
	8	0.825	0.820	0.822	
	16	0.815	0.835	0.818	

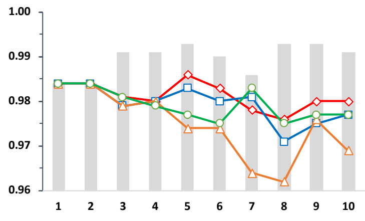
Fig. 3: Average AUC heatmap of Ligon using $2, 4, 8$ and 16 noisy oracles and the perfect oracle.

Dataset / # oracles	2	4	8	16	Perfect
Person 1	0.97	0.93	0.93	0.93	0.99
Person 2	0.98	0.98	0.98	0.98	0.99
Restaurants	0.97	0.97	0.97	0.97	0.97
ABT–Buy	0.89	0.88	0.88	0.88	0.97
Amazon–GoogleProducts	0.72	0.71	0.72	0.72	0.73
DBLP–ACM	0.70	0.71	0.70	0.71	0.76
DBpedia–LinkedMDB	0.89	0.88	0.88	0.88	0.97
DBLP–GoogleScholar	0.81	0.79	0.78	0.78	0.92
Average	0.86	0.86	0.85	0.86	0.91
Standard deviation	0.11	0.11	0.11	0.11	0.11

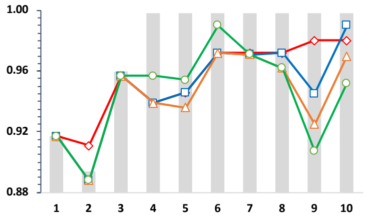
AUC close to 0.86 with no statistical difference. We can hence conclude that the performance of our approach depends mostly on the initial set of examples E_0 being accurate, which leads to our prior—i.e., the evaluation of the initial confusion matrix of the oracles—being sufficient. This sufficient approximation means that our Bayesian model is able to distinguish between erroneous classifications well enough to find most informative examples accurately and generalize over them. In other words, even a small balanced set containing 5 positive and 5 negative examples seems sufficient to approximate the confusion matrix of the oracles sufficiently well to detect positive and negative examples consistently in the subsequent steps. This answers Q_2. Figures 3 and 4) show the detailed results of running Ligon for 10 iterations for each of our 8 benchmark datasets.

To answer Q_3, we also ran our approach in combination with a *perfect oracle* (i.e., an oracle which knew and returned the perfect classification for all pairs from (S, T)). The detailed results are provided in Figures 4 and 3. Combining our approach with a perfect oracle can be regarded as providing an upper bound to our learning algorithm. Over all datasets, Ligon achieved 95% of the AUC achieved with the perfect oracle (min = 88% on *DBpedia-LMDB*, max = 100% on *Restaurants*) of the AUC achieved with the perfect oracle. This answers Q_3 and demonstrates that Ligon can learn LS with an accuracy close to that of an approach provided with perfect answers.

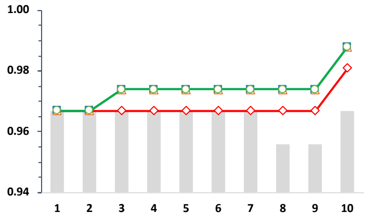
Runtime. In our third set of experiments, we were interested in knowing how well our approach scales. To this end, we measured the runtime of our algorithm while running the experiments carried



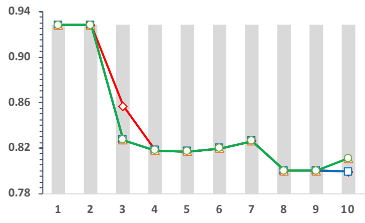
(a) Persons 1



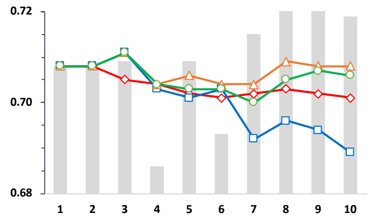
(b) Persons 2



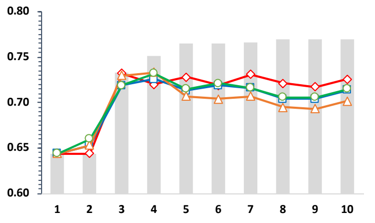
(c) Restaurants



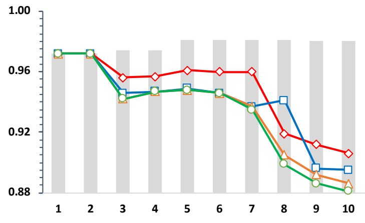
(d) ABT–Buy



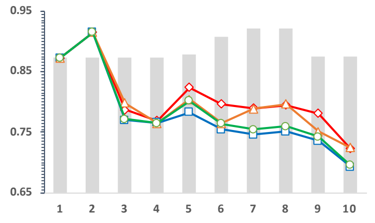
(e) Amazon–GoogleProducts



(f) DBLP–ACM



(g) DBpedia–LMDB



(h) DBLP–GoogleScholar

Fig. 4: F-measure results of LIGON with different numbers of noisy oracles vs. LIGON with a perfect oracle. x-axes show the iteration number while the y-axes show the F-measure. Note that, the y-axes show different value for better legibility. Gray bars represent the F-measure of the perfect oracle while the F-measure achieved by the 2, 4, 8 and 16 noisy oracles are represented by red, blue, orange and green lines respectively.

out to answer Q_2 and Q_3. In our experiments, WOMBAT, the machine learning approach used within LIGON, makes up for more than 99% of LIGON's runtime. for more detailed results see Table 3. This shows that the Bayesian framework used to re-evaluate the characteristic matrices of the oracles is clearly fast enough to be used in interactive scenarios, which answers Q_4. Our approach completes a learning iteration in less than 10 seconds on most datasets, which we consider acceptable even for interactive scenarios. The longer runtime on *DBLP-GoogleScholar* (roughly 16 seconds per iteration on average) is due to the large size of this dataset. Here, a parallel version of the WOMBAT algorithm would help improving the interaction with the user. The implementation of a parallel version of WOMBAT goes beyond the work presented here.

Comparison with Batch Learning. While active learning commonly requires a small number of training examples to achieve good F-measures, other techniques such as pessimistic and re-weighted batch learning have also been designed to achieve this goal [7]. In addition, the positive-only learning

Dataset	Pessimistic	Reweighted	Simple	Complete	Ligon
DBLP-ACM	0.93	**0.95**	0.94	0.94	0.73
Amazon-GP	0.39	0.43	0.53	0.45	**0.71**
Abt-Buy	0.36	0.37	0.37	0.36	**0.93**
Average	0.77	0.78	0.77	0.74	**0.89**

Table 4: Comparison of the F-Measure achieved by Ligon ($n = 16$, $k = 16$, strategy = equivalent, 10 iterations) against the approaches proposed in [7] and [19] on 3 benchmark datasets using 2% of the reference dataset as positive training examples.

algorithm Wombat has also been shown to perform well with a small number of training examples. In our final set of experiments, we compared the best F-measure achieved by Ligon when trained with 16 noisy oracles, $k = 16$ and the *equivalent* strategy with the pessimistic and re-weighted models proposed in [7] as well as the two versions of the Wombat approach [19]. All approaches were trained with 2% of the reference data (i.e., with a perfect oracle) as suggested by [7]. The results of these experiments are shown in Table 4. Note that we did not consider the datasets *Persons 1*, *Persons 2* and *Restaurant* because 2% of the training data accounts to less than 10 examples, which Ligon requires as initial training dataset E_0. Our results answer Q_5 clearly by showing that Ligon outperforms previous batch learning algorithms even when trained with noisy oracles. On average, Ligon is more than 40% better in F-measure. This clearly demonstrates that our active learning strategy for selecting training examples is superior to batch learning.

Generality of LIGON. While Ligon was designed with the link discovery domain in mind, it can certainly be applied to any binary classification problem in an active learning setting. To support this claim, we implemented a generalization of Ligon for arbitrary binary classification tasks. To this end, we used the active learning framework JCLAL [17] to wrap WEKA [4] classifiers and implemented our approach as a custom oracle. We selected three well known binary classification datasets (i.e., *Diabetes, breast-cancer* and *Ionosphere*) from the WEKA distribution on which we applied two state-of-the-art classification algorithms [25], namely GBDT [3] and Random Forests [1]. Based on our previous experiments, we used 4 noisy oracles. k was set to 16 and we used the *Ignore* strategy, since all the other strategies are specific to the link discovery domain. We executed two sets of experiments for noisy oracles with true positive/negative probabilities drawn from the two uniform distributions in $[0.5, 1]$ and $[0.75, 1]$. The results suggest that Ligon is robust to the errors induced by the noisy oracles. On average, Ligon achieves 75% and 89% of the learning accuracy for noisy oracles drawn from $[0.5, 1]$ and $[0.75, 1]$ respectively. See Table 5 for more detailed results. These results indicate that Ligon is not only applicable to problems outside the link discovery domain but also independent from the underlying active learning algorithm is able to achieve F-measures near to the ones scored using a perfect oracle, which answers Q_6.

7 Related Work

Link Discovery for RDF knowledge graphs has been an active research area for nearly a decade, with the first frameworks for link discovery [6, 11] appearing at the beginning of the decade. Raven [12] was the first active learning approach for link discovery and used perception learning to detect

Dataset	RF+P	RF+Li_1	RF+Li_2	GB+P	GB+Li_1	GB+Li_2
Diabetes	0.73	0.58	0.70	0.71	0.63	0.69
Breast-cancer	0.61	0.44	0.52	0.60	0.50	0.50
Ionosphere	0.85	0.52	0.81	0.75	0.48	0.59
Average	0.73	0.51	0.67	0.68	0.54	0.59

Table 5: Comparison of the F-Measure achieved by GBDT (GB) and Random Forests (RF) in combination with a perfect oracle (P) and LIGON (Li) on 3 WEKA benchmark datasets using 2% of the reference dataset as positive training examples. Li_1 and Li_2 denote LIGON on noisy oracles with true positive and true negative probabilities drawn from uniform distribution in $[0.5, 1]$ and $[0.75, 1]$, respectively.

accurate LS. Other approaches were subsequently developed to learn LS within the active learning setting [5, 13, 14]. Unsupervised learning approaches for monogamous relations [13–15] rely on different pseudo-F-measures to detect links without any training data. Positive-only learning algorithms [19] address the open-world characteristic of the Semantic Web by using generalization algorithms to detect LS. The work presented by [16] proposes an active learning approach for link prediction in knowledge graphs. LIGON differs from the state of the art in that it does not assume that it deals with perfect oracles. Rather, it uses several noisy oracles to achieve an F-measures close to those achieved with perfect oracles.

An *active learning approach with uncertain labeling knowledge* is proposed by [2], where the authors used diversity density to characterize the uncertainty of the knowledge. A probabilistic model of active learning with multiple noisy oracles was introduced by [22] to label the data based on the most perfect oracle. Another active learning framework was proposed by [20] to develop a query selection and evaluation approaches for pool-based active learning. For *Crowdsourcing* scenarios, [18] propose a supervised learning algorithm for multiple annotators (oracles), where the oracles' diverse reliabilities were treated as a latent variables. [21] introduce a dynamic estimation of oracle reliability for regression tasks. Another active learning approach from noisy labelers is proposed by [23]. [26] survey the truth inference from noisy labelers.

8 Conclusions and Future Work

We presented LIGON, an active learning approach designed to deal with noisy oracles, i.e., oracles that are not guaranteed to return correct classification results. LIGON relies on a probabilistic model to estimate the joint odds of link candidates based on the oracles' guesses. Our experiments showed that LIGON achieves 95% of the learning accuracy of approaches learning with perfect oracles in the link discovery setting. Moreover, we showed that LIGON is (1) not dependent on the underlying active learning algorithm and (2) able to deal with other classification problems.

In future work, we will evaluate LIGON within real crowdsourcing scenarios. A limitation of our approach is that it assumes that the confusion matrix of the oracles is static. While this assumption is valid with the small number of iterations necessary for our approach to converge, we will extend our model so as to deal with oracles which change dynamically. Furthermore, we will extend LIGON to handle n-ary classification problems and evaluate it on more stat-of-the-art approaches from the deep learning domain.

Acknowledgments

This work has been supported by the EU H2020 project KnowGraphs (GA no. 860801) as well as the BMVI projects LIMBO (GA no. 19F2029C) and OPAL (GA no. 19F2028A).

References

1. L. Breiman. Random forests. *Machine Learning*, 45(1):5–32, Oct 2001.
2. M. Fang and X. Zhu. Active learning with uncertain labeling knowledge. *Pattern Recognition Letters*, 43(Supplement C):98 – 108, 2014. ICPR2012 Awarded Papers.
3. J. H. Friedman. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pages 1189–1232, 2001.
4. M. A. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten. The WEKA data mining software: an update. *SIGKDD Explorations*, 2009.
5. R. Isele and C. Bizer. Active learning of expressive linkage rules using genetic programming. *J. Web Sem.*, 23:2–15, 2013.
6. A. Jentzsch, R. Isele, and C. Bizer. Silk - generating RDF links while publishing or consuming linked data. In *Proceedings of the ISWC Posters & Demons*, 2010.
7. M. Kejriwal and D. P. Miranker. Semi-supervised instance matching using boosted classifiers. In *The Semantic Web. Latest Advances and New Domains*. 2015.
8. H. Köpcke, A. Thor, and E. Rahm. Evaluation of entity resolution approaches on real-world match problems. *Proc. VLDB Endow.*, 3(1-2):484–493, Sept. 2010.
9. C. D. Manning, P. Raghavan, and H. Schütze. Introduction to information retrieval, 2008.
10. M. Nentwig, M. Hartung, A. N. Ngomo, and E. Rahm. A survey of current link discovery frameworks. *Semantic Web*, 8(3):419–436, 2017.
11. A. N. Ngomo and S. Auer. LIMES - A time-efficient approach for large-scale link discovery on the web of data. In *Proceedings of the International Joint Conference on Artificial Intelligence, Spain, 2011*.
12. A.-C. Ngonga Ngomo, J. Lehmann, S. Auer, and K. Höffner. RAVEN - active learning of link specifications. In *Proceedings of the 6th International Workshop on Ontology Matching, Germany, 2011*.
13. A.-C. Ngonga Ngomo and K. Lyko. Eagle: Efficient active learning of link specifications using genetic programming. In *Extended Semantic Web Conference*, pages 149–163. Springer, 2012.
14. A.-C. Ngonga Ngomo, K. Lyko, and V. Christen. Coala–correlation-aware active learning of link specifications. In *Extended Semantic Web Conference*, 2013.
15. A. Nikolov, M. d'Aquin, and E. Motta. Unsupervised learning of link discovery configuration. In *Extended Semantic Web Conference*. Springer, 2012.
16. N. Ostapuk, J. Yang, and P. Cudré-Mauroux. Activelink: deep active learning for link prediction in knowledge graphs. In *The World Wide Web Conference*, 2019.
17. O. G. R. Pupo, E. Pérez, M. del Carmen Rodríguez-Hernández, H. M. Fardoun, and S. Ventura. JCLAL: A java framework for active learning. *J. Mach. Learn. Res.*, 2016.
18. F. Rodrigues, F. Pereira, and B. Ribeiro. Learning from multiple annotators: Distinguishing good from random labelers. *Pattern Recognition Letters*, 2013.
19. M. Sherif, A.-C. Ngonga Ngomo, and J. Lehmann. WOMBAT - A Generalization Approach for Automatic Link Discovery. In *14th Extended Semantic Web Conference, Slovenia*. Springer, 2017.
20. Y. Sogawa, T. Ueno, Y. Kawahara, and T. Washio. Active learning for noisy oracle via density power divergence. *Neural Networks*, 46(Supplement C):133 – 143, 2013.
21. A. Tarasov, S. J. Delany, and B. M. Namee. Dynamic estimation of worker reliability in crowdsourcing for regression tasks: Making it work. *Expert Systems with Applications*, 41(14):6190 – 6210, 2014.
22. W. Wu, Y. Liu, M. Guo, C. Wang, and X. Liu. A probabilistic model of active learning with multiple noisy oracles. *Neurocomputing*, 2013.

23. Y. Yan, R. Rosales, G. Fung, and J. G. Dy. Active learning from crowds. In *ICML*, volume 11, pages 1161–1168, 2011.
24. H. Yu, Z. Shen, C. Miao, and B. An. Challenges and opportunities for trust management in crowdsourcing. In *Proceedings of the The 2012 IEEE/WIC/ACM WI-IAT '12*, USA, 2012.
25. C. Zhang, C. Liu, X. Zhang, and G. Almpanidis. An up-to-date comparison of state-of-the-art classification algorithms. *Expert Systems with Applications*, 2017.
26. Y. Zheng, G. Li, Y. Li, C. Shan, and R. Cheng. Truth inference in crowdsourcing: Is the problem solved? *Proceedings of the VLDB Endowment*, 10(5):541–552, 2017.