Graph-Boosted Active Learning for Multi-source Entity Resolution

Anna Primpeli(✉) and Christian Bizer

Data and Web Science Group, University of Mannheim, Mannheim, Germany
{anna,chris}@informatik.uni-mannheim.de

Abstract. Supervised entity resolution methods rely on labeled record pairs for learning matching patterns between two or more data sources. Active learning minimizes the labeling effort by selecting informative pairs for labeling. The existing active learning methods for entity resolution all target two-source matching scenarios and ignore signals that only exist in multi-source settings, such as the Web of Data. In this paper, we propose ALMSER, a graph-boosted active learning method for multi-source entity resolution. To the best of our knowledge, ALMSER is the first active learning-based entity resolution method that is especially tailored to the multi-source setting. ALMSER exploits the rich correspondence graph that exists in multi-source settings for selecting informative record pairs. In addition, the correspondence graph is used to derive complementary training data. We evaluate our method using five multi-source matching tasks having different profiling characteristics. The experimental evaluation shows that leveraging graph signals leads to improved results over active learning methods using margin-based and committee-based query strategies in terms of F1 score on all tasks.

Keywords: Entity resolution · Link discovery · Multi-source entity matching · Active learning

1 Introduction

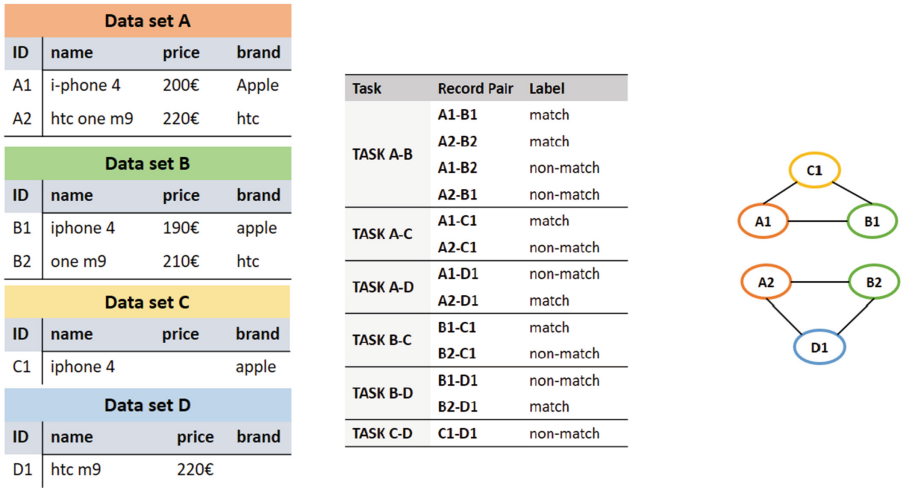
Entity resolution, also referred as entity matching or link discovery, aims to identify records in one or multiple data sources which describe the same real-world entity [4,5]. Supervised entity resolution methods treat entity matching as a classification problem and rely on a labeled set of matching and non-matching record pairs for training [5,7]. Active learning aims to minimize the labeling effort by involving the annotator in the learning loop and selecting only the most informative pairs for labeling [27].

There has been quite some research on active learning for entity resolution [3, 9,12,17,26]. However, to the best of our knowledge, all of these works focus on active learning methods for matching records between two data sources, while signals that only exist in multi-source settings are not exploited to further reduce the number of record pairs that need to be labeled by the annotator. Exploiting

A. Hotho et al. (Eds.): ISWC 2021, LNCS 12922, pp. 182–199, 2021.
https://doi.org/10.1007/978-3-030-88361-4_11

such signals is for example beneficial for link discovery [16] in the context of the Web of Data [8], as link discovery efforts often target multiple data sources.

We fill in this gap and propose an active learning method for entity resolution that exploits additional signals that only exist in multi-source settings. We consider the multi-source entity resolution task as a combination of multiple two-source matching tasks between pairs of data sources having the same schema but different underlying matching patterns. Figure 1 shows an example of a multi-source entity resolution task consisting of four data sources describing mobile phones (Fig. 1a). The pairwise combinations of the four data sources constitute a multi-source matching task comprising of six two-source tasks (Fig. 1b). For each of these tasks different attributes can be relevant for matching, e.g. *name* and *brand* for task A-C and *name* and *price* for task A-D. The goal of multi-source entity resolution is to learn a matcher that correctly identifies correspondences between the records of all sources. These correspondences can be viewed as a correspondence graph with all distinct records being the nodes of the graph connected by edges indicating matching record pairs (Fig. 1c). The discovered correspondences are used afterwards to fuse data from multiple sources or are published as RDF links on the Web of Data [8].



Data set A

ID	name	price	brand
A1	i-phone 4	200€	Apple
A2	htc one m9	220€	htc

Data set B

ID	name	price	brand
B1	iphone 4	190€	apple
B2	one m9	210€	htc

Data set C

ID	name	price	brand
C1	iphone 4		apple

Data set D

ID	name	price	brand
D1	htc m9	220€	

(a) Data sources

Task	Record Pair	Label
TASK A-B	A1-B1	match
	A2-B2	match
	A1-B2	non-match
	A2-B1	non-match
TASK A-C	A1-C1	match
	A2-C1	non-match
TASK A-D	A1-D1	non-match
	A2-D1	match
TASK B-C	B1-C1	match
	B2-C1	non-match
TASK B-D	B1-D1	non-match
	B2-D1	match
TASK C-D	C1-D1	non-match

(b) Two-source matching tasks of the multi-ER task

(c) Correspondence graph

Fig. 1. Example of a multi-source entity resolution task.

This paper proposes *ALMSER*, a graph-boosted **A**ctive **L**earning method for **M**ulti-**S**ource **E**ntity **R**esolution which exploits the rich correspondence graph that the multi-source setting offers in two ways: First, to pick informative query candidates and second to boost the training of the learner with additional train-

ing data. Most active learning methods apply either a query-by-committee strategy [3,26] or a margin-based strategy [13,14] for picking informative candidates and use only the labeled set at each iteration for training the learner. Query-by-committee strategies measure the informativeness of the query candidates as the degree of disagreement among the predictions of a classifier ensemble, while margin-based strategies pick the instances that are closer to the decision boundary of a classifier. In contrast, our query strategy exploits graph signals such as graph transitivity and minimum cuts to discover potentially false negative and false positive record pairs among the predictions of the learner. We assume that focusing on the errors of the learner to derive informative pairs for labeling can lead to the faster discovery of relevant matching patterns in comparison to uncertainty-based query strategies. For boosting the training of the learner, we derive likely matching and non-matching record pairs from the clean components of the graph which we use as additional training data.

We evaluate ALMSER using five multi-source entity resolution tasks having different profiling characteristics. Our evaluation shows that graph signals lead to an overall improved performance over baseline methods which use a state-of-the-art committee-based query strategy and a margin-based query strategy.

The contributions of our work are summarized as follows:

- We are the first to tackle the problem of multi-source entity resolution with active learning.
- We propose an active learning method for multi-source entity resolution which uses the correspondence graph for query selection and training data augmentation.
- We evaluate our method on five multi-source entity resolution tasks and show that it consistently outperforms baseline methods that do not use graph signals in terms of F1 score. In terms of area under the F1 score curve, our method also performs better than methods that use graph signals for training data augmentation but not for query selection.

The remainder of the paper is organized as follows: Sect. 2 discusses related work on multi-source entity resolution and active learning. Section 3 explains our method. Section 4 presents the experimental setup and discusses the experimental results. Finally, Sect. 5 concludes the paper and summarizes our findings.

2 Related Work

Entity resolution is a central prerequisite for integrating data from multiple sources [4,5,19] as well as for setting RDF links in the context of the Web of Linked Data [8,16]. Entity resolution has been extensively studied over decades [6,16,19]. Although there exist works on supervised and unsupervised multi-source entity resolution [1,24,28] as well as on active learning methods for the two-source matching task [3,9,13], there has been no work on multi-source entity resolution with active learning.

Multi-source Entity Resolution: There are two lines of research on multi-source entity resolution which either focus on solving the scalability issues of integrating data from multiple sources [28] or use graph signals in supervised [1] or unsupervised matching settings [25]. The supervised SOCCER method proposed by Shen et al. [28] defines an efficient order of pair-wise matching tasks in large multi-source settings. In the work of Bellare et al. on knowledge base synthesis [1], a supervised classifier is applied during the matching step and the matching results are refined using the connected components of the correspondence graph, similar to our work. Saeedi et al. compare different clustering methods for multi-source entity resolution [24] and propose CLIP [25] a clustering approach which requires hand-written domain specific rules for calculating the weighted edges of the graph. The CLIP method assumes duplicate-free sources, an assumption which is not necessary for our proposed method. JointBERT [20] applies deep learning techniques for multi-source entity resolution and treats the matching problem in parallel as a binary and multi-class classification task.

Active Learning for Entity Resolution: Active learning aims to minimize the human labeling effort involved in supervised tasks [27]. Active learning approaches with a specific focus on RDF link discovery [16] include EAGLE [17] and ActiveGenLink [9]. Meduri et al. compare various active learning methods for entity resolution on multiple benchmark data sets for two-source matching and show that random forest classifiers with learner-aware committee-based query strategies achieve fast convergence and close to perfect entity matching quality [13]. However, it is reported that for some tasks margin-based query strategies can perform equally well. Therefore, we compare ALMSER to both committee-based and margin-based active learning baselines. There have been many active learning methods for entity resolution which use committee-based query strategies for selecting informative candidates [3,9,26]. In the work of Chen et al. [3] it is shown that using a committee of heterogeneous classifiers is more effective in comparison to committees consisting of the same model with different parametrisations. Recent works on active learning for entity resolution have turned the focus to deep learning based methods tailored for low-resource settings [10,15]. Such methods rely on transfer learning [10] or large randomly sampled sets [15] for initializing the deep learning models and require a pre-labeled development set for hyperparameter optimization [10,15]. In comparison to the existing deep active learning works for entity matching, our suggested approach involves less annotation effort as it leverages unsupervised matching for initialization and does not require an additional development set for model learning.

Active Learning with Graph Signals: Using graph signals for boosting the query strategy of active learning methods, has been explored in related work for different applications [2,18] and has inspired our work on graph-boosted active learning for multi-source entity resolution. Nguyen et al. develop an active learning method for image classification which uses the k-medoid algorithm for clustering the data [18]. Different signals of the graph structure, such as the cluster representativeness and density are used for refining an uncertainty sam-

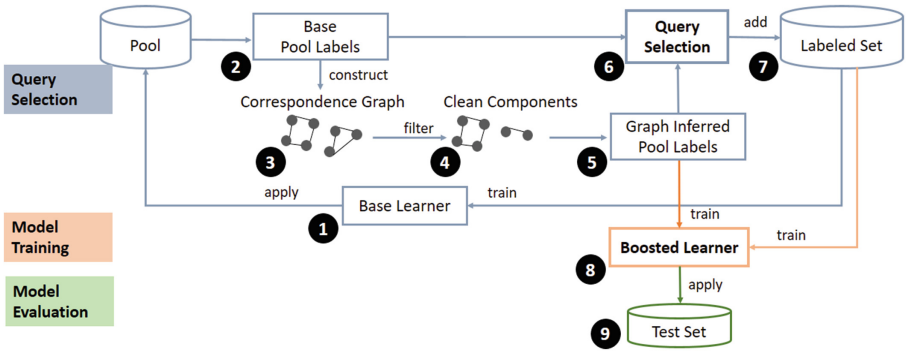


Fig. 2. Overview of the ALMSER algorithm.

pling query strategy. Similarly, Bilgic et al. propose an active learning method for multi-class classification which exploits graph signals for boosting an uncertainty sampling query selection strategy [2].

3 Methodology

In this section, we present our proposed active learning algorithm for multi-source entity resolution, which we abbreviate with ALMSER. This subsection summarizes the overall process that is executed by ALMSER. The following subsections detail each step in the process.

We consider a pool-based active learning setting in which a pool of unlabeled record pairs is available to the learner. This pool is typically the result of a preceding blocking step [5]. Figure 2 gives an overview of the ALMSER algorithm. We initialize ALMSER by bootstrapping the labeled set of record pairs (Artefact 7 in Fig. 2). After initialization, the following steps are executed: First, we train a base learner using the current labeled set (1) and get base predictions for all unlabeled record pairs of the pool (2) which together with the labeled set are used to construct a correspondence graph (3). Next, we derive the clean components of the graph (4) and assign graph-inferred labels to the record pairs of the pool which are part of the clean components (5). The query strategy picks the most informative record pair for labeling considering the disagreement between the predicted labels of the base learner and the graph-inferred labels (6). The selected record pair is annotated as *match* or *non-match* and is added to the labeled set (7). We exploit the graph-inferred labels to derive additional training data which together with the labeled set are used for training the boosted learner (8). In order to evaluate how the performance of the boosted learner develops during the active learning process, we apply the boosted learner to the test set after each iteration (9).

3.1 Initialization of ALMSER

The initialization of active learning is a non-trivial step which has been shown to suffer from the cold start problem [12,22]. This problem refers to the lack of labeled data from all classes in the early iterations which further hinders the training of the learner as well as of the classifiers used for query selection in the case of classification-based query strategies. To circumvent the cold start problem, we apply an unsupervised bootstrapping method which summarizes the feature vector of each record pair into an aggregated similarity score and selects as seeding pairs the ones with the lowest and highest scores. The details about this method are presented in [22]. We apply the approach for each two-source task of the multi-source setting and select two record pairs per task: one with the highest and one with the lowest aggregated score. Considering that in the very early active learning iterations the base model, which we use to construct the correspondence graph, is highly unstable, we perform the first 20 active learning iterations using the state-of-the-art committee based query-strategy HeALER [3]. Afterwards, we switch to the graph-based query strategy that we explain in Sect. 3.5.

3.2 Correspondence Graph Construction

After initializing the labeled set, the graph-boosted active learning cycle starts. In each active learning iteration we construct the correspondence graph of the multi-source task (steps 1–3 in Fig. 2) with the aim to obtain graph signals which we exploit in later steps of our methodology for query selection and model training. The correspondence graph contains all distinct records of the record pairs in the pool and the labeled set as nodes. We add an edge to the graph for every confirmed matching record pair in the labeled set, while we add no edge for every labeled non-matching pair.

Additionally, we use the pool predictions of a random forest classifier, which we refer to as base learner, for inferring potential matching pairs. More concretely, in each iteration the base learner is trained on all record pairs of the labeled set and applied to the record pairs of the pool. Each pool pair is assigned the predicted base label, *match* or *non-match* together with a confidence score, which is the predicted class probability of the base learner. We add an edge to the correspondence graph between the nodes of every pool record pair with a matching base label, while we add no edge if the base label is non-match.

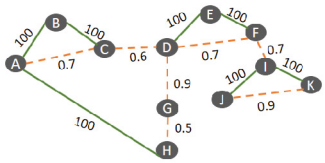
Finally, we assign weights to the edges of the correspondence graph. Every edge that derives from the labeled set and is therefore confirmed to be true, receives the weight 100. The edges deriving from the base learner matching predictions are weighted according to their confidence score.

3.3 Correspondence Graph Cleansing

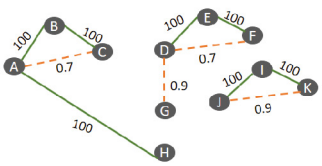
Exploiting the transitivity of the correspondence graph can lead to the discovery of false negative base learner predictions: e.g. given three record pairs (A-B), (B-C) and (A-C) which have been predicted by the base learner as *match*, *match*

Labeled Set	
A-B	match
B-C	match
D-E	match
D-F	match
D-H	mon-match
I-K	match
I-J	match

(a) Example labeled set of current iteration



(b) Cor. graph given labeled set and base-model predictions



(c) Cor. graph after removal of minimum cuts (D-C & G-H) and bridges (F-I)

Fig. 3. Exploiting the graph to detect false positives - an example.

and *non-match* respectively, we can infer using graph transitivity that (A-C) is also a matching pair and that it is likely a false negative prediction of the base learner.

However, given that the edges of the correspondence graph deriving from the matching base learner predictions are subject to some noise, a wrongly assigned edge can lead to a series of false positive record pairs. Therefore, we need to discover likely wrong edges and remove them from the correspondence graph. The example in Fig. 3 demonstrates this problem. Figure 3b shows an example graph with 11 nodes and weighted edges. The solid edges connect nodes of matching record pairs found in the labeled set of Fig. 3a and are therefore assigned a weight of 100 while the dotted edges represent the base labels and are assigned their corresponding confidence weights. The resulting graph is connected and forms one connected component, i.e. there is a path from any node to any other node in the graph, indicating that all nodes refer to the same real-world entity. However, this cannot be the case as there is a confirmed non-matching pair $(D - H)$ in the labeled set of Fig. 3a. Therefore, the path between the nodes D and H needs to be cut. Given the edge weights, we calculate the minimum cut of the graph. The edges which should be removed in order to cut the path between D and H are the following: $(D - C)$ and $(G - H)$ as their total edge weight is less than any other cut alternative. We can additionally observe that the edge $(F - I)$, forms a bridge between the two components (E, D, F, G) and (I, J, K) and is a possible false positive. Figure 3c shows the graph after minimum cuts and bridges removal which reveals three connected components.

We rely on these observations and remove edges between nodes of potentially false positive record pairs using a two step procedure. First, we iterate over all non-matching pairs in the labeled set and for each pair we check if there is a path between the two nodes-records in the graph. In case we find a path, we calculate the minimum cut of the graph considering the weights of the edges. For the calculation of the minimum cuts, we use the *networkx* implementation.[1] As second step, we identify and remove bridge edges from the graph. In order to

[1] https://networkx.org/documentation/stable/reference/algorithms/generated/networkx.algorithms.flow.minimum_cut.html.

ensure that there is no unnecessary increase of many small-sized components, we only remove the bridge edges connecting nodes having more than two neighbours each.

3.4 Clean Components Filtering

After cleansing the correspondence graph, we filter its clean components and assign graph-inferred labels to a subset of the pool record pairs (steps 4–5 in Fig. 2) with the aim to get more accurate graph signals that can both identify wrong base learner predictions and lead to clean augmented training data.

In order to derive the clean components of the correspondence graph, we first compute all connected components. Considering that smaller components are cleaner than larger ones, we assume a component to be clean if its size is equal or smaller than the amount of data sources to be matched. Although this heuristic comes natural for deduplicated sources, we show during evaluation that it is also a good approximation for discovering the clean components of the graph in multi-source matching tasks with non-deduplicated data sources.

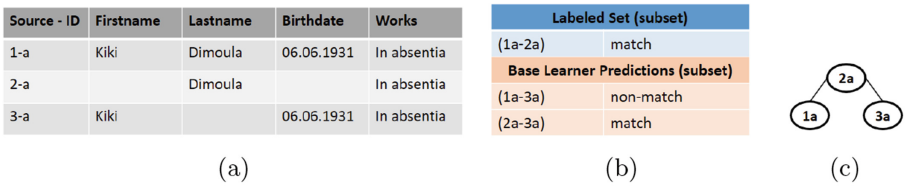
We use the correspondence graph and the clean connected components to assign graph-inferred labels to a subset of the pool record pairs. For record pairs belonging to the same clean component, we assign a matching graph-inferred label. If there is no path in the correspondence graph between the two records of the pair, then we assign a non-match graph-inferred label. Finally, for record pairs belonging to non-clean components, no graph-inferred label is assigned.

3.5 Query Selection

The query selection strategy of ALMSER evaluates as the most informative candidates the record pairs whose graph-inferred label is different from the base label and assigns binary informativeness scores to all record pairs in the pool: 1 if there is a conflict between the base and the graph-inferred labels otherwise 0 (Step 6 in Fig. 2). While margin-based and committee-based query strategies aim to select instances for which the learner or a committee of models produces non-confident predictions, our query strategy uses the clean components of the correspondence graph to pick instances that are most likely predicted wrong by the base learner. These disagreements between the graph and the base learner hint towards matching patterns that are not covered yet by the base learner and can occur under two conditions: First, if the record pair has been predicted by the base learner to be a non-match and due to graph transitivity the graph-inferred label is match. Second, if the record pair has been predicted as match by the base learner but the corresponding edge was found to be a bridge edge or was part of a minimum cut between confirmed non-matching pairs and therefore was removed during the cleansing step, as described in Sect. 3.3.

We illustrate the discovery of new matching patterns by graph transitivity with the simple example of Fig. 4 which presents three records from different data sources describing the same author (4a) and a subset of labeled pairs and base learner predictions (Fig. 4b) which are used to construct the correspondence

graph (Fig. 4c). Given the matching pair (1a-2a) of the labeled set, the base learner might be trained to capture matching patterns based on the similarity of the *Lastname* and the *Works* attributes. However, it might wrongly predict the pair (1a-3a) as non-matching as it has not learned yet the pattern that high similarity of *Birthdate* and *Firstname* together also indicate a match. Based on the graph transitivity, the pair (1a-3a) is assigned a matching graph-inferred label and therefore receives an informativeness score of 1. Selecting this pair as a query candidate supports the model in learning the relevance of the *Birthdate* and *Firstname* attribute combination for matching.



Source - ID	Firstname	Lastname	Birthdate	Works
1-a	Kiki	Dimoula	06.06.1931	In absentia
2-a		Dimoula		In absentia
3-a	Kiki		06.06.1931	In absentia

(a)

Labeled Set (subset)	
(1a-2a)	match
Base Learner Predictions (subset)	
(1a-3a)	non-match
(2a-3a)	match

(b)

(c)

Fig. 4. Graph-boosted query selection strategy - an example.

In order to ensure that the query strategy selects equally likely false positives, i.e. pairs with a non-match graph inferred label, and likely false negative pairs, i.e. pairs with a match graph-inferred label, we assign selection probability weights to all record pairs with an informativeness score of 1. For example, given 10 likely false negatives and 1 likely false positive, we assign the selection probability weights 0.1 for each false negative and 1.0 for the false positive pair. Finally, given the selection probability scores, we perform weighted random selection over the candidate record pairs with an informativeness score of 1 and select one pair which is annotated and added to the labeled set (Step 7 in Fig. 2).

3.6 Boosted Learner Training

In a real-world active learning setting, we would learn one boosted model at the very last active learning iteration, as the boosted learner does not affect the query selection, i.e. the query strategy of ALMSER is agnostic towards the boosted learner. However, in order to be able to evaluate the boosted model along each active learning iteration, we train it and apply it to the test set as final step of each iteration (Steps 8 and 9 in Fig. 2). We perform training data augmentation with the aim to improve the training of the boosted learner. Similarly to the base learner, we use a random forest classifier as the boosted learner, assuming that a random forest model with a large number of estimators can expand to fit the matching patterns of all matching tasks in a multi-source entity resolution setting. For training the boosted learner we use both the record pairs of the labeled set, which contains the records pairs selected during initialization as explained in Sect. 3.1 and the manually validated record pairs, and the subset of

Table 1. Profiling information of evaluation matching tasks.

Multi-source task	# Data sources	# Pairs (in K)		Schema Complex.	Range of sparsity	Corner cases
		Matches	Non-matches			
MusicBrainz	5	16.1	369.7	[3–5]	[0.05–0.12]	[0.08–0.42]
MusicBrainz_mut	5	16.1	369.7	[3–6]	[0.05–0.23]	[0.06–0.62]
Computers	4	4.8	69.6	[3–4]	[0–0.05]	[0.02–0.30]
computers_mut	4	4.8	69.6	[3–6]	[0–0.18]	[0.24–0.50]
Restaurants	4	11.2	56.5	[4–7]	[0–0.08]	[0.05–0.19]

the pool record pairs which have been assigned a graph-inferred label, i.e. record pairs deriving from clean components of the correspondence graph of the current iteration.

4 Experimental Evaluation

We evaluate ALMSER using five multi-source matching tasks having different profiling characteristics. In this section, we first present the evaluation tasks. Afterwards, Sect. 4.3 compares ALMSER to two baseline active learning methods that do not use graph signals. Section 4.4 evaluates the distinct components of ALMSER that exploit graph signals and compares them to baseline methods using the graph signal only for training data augmentation. All data sets and code used for experimental evaluation are available for public download.[2]

4.1 Multi-source Matching Tasks

We use five multi-source matching tasks for our experimental evaluation. The tasks cover the domains music, products, and restaurants. Table 1 contains profiling information about the five tasks, including the amount of sources to be matched as well as the amount of matching and non-matching pairs per task. In our previous work on profiling entity matching tasks [21], we have defined a set of profiling dimensions for assessing the difficulty of entity matching tasks. The last three columns in Table 1 show the value ranges of the profiling dimensions schema complexity, sparsity, and corner cases for the two-source matching tasks that make up each multi-source task. Schema complexity refers to the amount of attributes that contribute to solving the matching task. Sparsity indicates the ratio of missing attribute values. The dimension corner cases approximates the fraction of difficult to match pairs within each task [21].

The MusicBrainz multi-source task has been used for the evaluation in [24, 25]. The task is based on song records from the MusicBrainz dataset. Each data source is a modified version of the original dataset and therefore the two-source matching tasks that make up the multi-source task have different underlying

[2] https://github.com/wbsg-uni-mannheim/ALMSER-GB.

patterns: while in five of the ten two-source tasks the attributes *album*, *length* and *title* are most relevant for matching, for the rest of the tasks different attributes reveal the underlying matching patterns such as *title* and *song number* or *title* and *artist*. Additionally to the original MusicBrainz multi-source task, we curate a modified version of it, abbreviated with *MusicBrainz_mut*, by increasing the attribute sparsity up to 30% per data source and adding noise in 50% of the attribute values which further results in an increase of corner cases in comparison to the original MusicBrainz task.

We exploit the WDC Training Corpus for Large-scale Product Matching[3] [23] and derive a subset of computer product records published in four e-commerce websites, for curating the product-related multi-source matching task, which we abbreviate with *computers*. Similarly to the MusicBrainz task, we curate a modified version of the computers task which we abbreviate with *computers_mut* and contains an increased schema complexity, sparsity and amount of corner cases. While the underlying matching patterns of the original task focus mostly on the combination of the *title* and *part number* attributes, the mutated version of the tasks requires additional attributes to be solved such as *category*, *capacity* and *generation*.

The restaurant related multi-source task derives from the Magellan repository[4] which provides a large number of two-source matching tasks. We retrieve four of the restaurant data sources that have been crawled from large restaurant aggregators and use the phone number as weak supervision in order to establish the complete mappings between all data source pairs. While three of the six two-source matching tasks have a low containment of corner cases (<10%) and can be solved only with address related attributes, the rest of the two-source tasks require additional attributes such as *name*, *cuisine* and *website*.

We turn the records of all tasks into features vectors by calculating datatype specific similarity scores, similar to the Magellan entity matching system [11]. For string attributes, the following similarity scores are calculated: Levenshtein, Jaccard, Jaccard with inner Levenshtein, token overlap, and token containment. For numeric attributes the absolute difference is calculated and re-scaled to the range [0, 1]. In the case that a similarity score cannot be computed for an attribute combination because of missing values, we assign the out of range score −1. This allows any classifier to consider all record pairs without dropping or replacing the missing values.

The selected multi-source tasks cover two distinct scenarios: the first scenario includes matching tasks of duplicate free data sources and therefore their correspondence graph forms connected components of maximum size equal to the total amount of sources, which is the case of the MusicBrainz and MusicBrainz_mut tasks. The second scenario covers tasks of non-deduplicated data sources resulting in components that are larger than the total amount of sources, which happens for the computers, computers_mut, and restaurants tasks.

[3] http://webdatacommons.org/largescaleproductcorpus/v2/.
[4] https://sites.google.com/site/anhaidgroup/useful-stuff/data.

4.2 Experimental Setup

We split the multi-source tasks into two subsets: one for initializing the pool that is available for querying and one for testing. In order to ensure that there is no leakage by graph transitivity from the pool set to the test set, we split the record pairs to pool pairs and test pairs based on the connected components of the complete correspondence graph with a ratio 70%–30%.

We execute three runs for each active learning experiment and allow 200 iterations for each run. In each iteration, one record pair is selected for annotation, i.e. 200 record pairs have been labeled in total by the end of each experimental run. We report the mean $F_1 micro$ score per iteration as well as the standard deviation which measures the model stability among the different experimental runs. Additionally, we report the upper learning bound of passive learning for which all record pairs of the pool together with their respective labels are used for model learning. All experiments were run on a Linux server with Intel Xeon 2.4 GHz processors. Considering that ALMSER constructs in each iteration a correspondence graph, its runtime is larger in comparison to baseline methods which do not use graph signals, e.g. one baseline iteration for the computers task without graph signals takes 2.9–3.15 s while one ALMSER iteration takes 14.58–15.10 s.

4.3 Comparison to Baselines Without Graph Signals

We compare ALMSER to two baseline active learning methods using the two distinct types of classification-based query strategies: committee-based and margin-based [19] and no graph signals. The first baseline method, abbreviated with QHC, uses the state-of-the-art committee-based query strategy of the HeALER algorithm [3] which measures the informativeness of each candidate record pair as the disagreement of the predictions of a committee of heterogeneous classification models. Similar to their method, ALMSER also uses random forest classifiers as learners. The second baseline method which is a common margin-based baseline [13,14,27], abbreviated with MB is a learner agnostic method, i.e. the classification model used as part of the query strategy is different from the learner, and selects the query candidates with minimum distance to the decision hyperplane defined by a SVM classifier. The learners of the baseline methods do not use graph signals and therefore are trained only on the labeled set. In order to ensure a comparable start of the learning process for all methods, we apply the initialization step that we describe in Sect. 3.1 for all baseline methods.

Figure 5 shows the average F1 score curves of ALMSER and the two baseline methods for each multi-source matching task per iteration. Additionally, we show the standard deviation of the F1 scores per iteration with the light coloured area around the plotted curves and the upper learning bound of passive learning. We can observe that as the active learning process unfolds, ALMSER outperforms both baselines for all tasks. The sudden drops in F1 in the early iterations, e.g. iterations 25 to 50 for the setting computers_mut as shown in Fig. 5d, can be

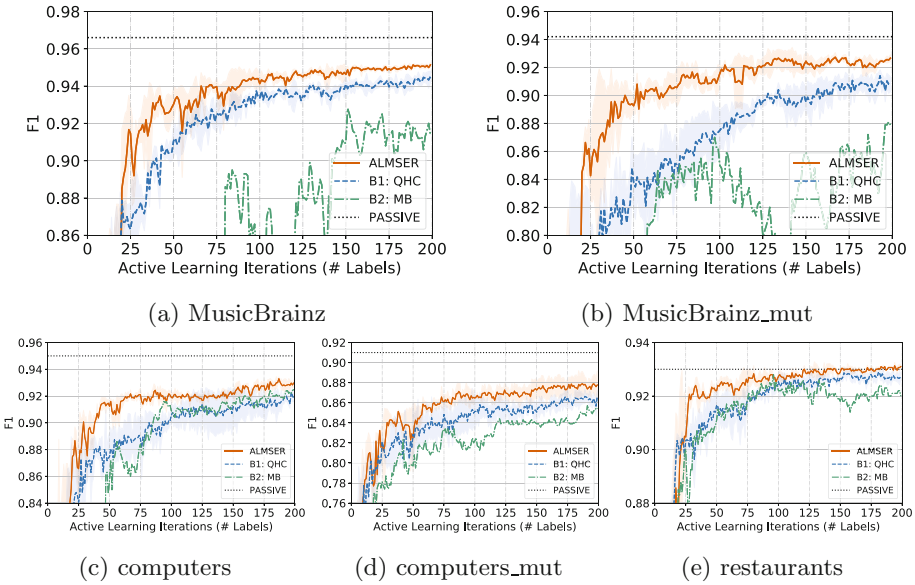


Fig. 5. Comparison of ALMSER to active learning baselines and passive learning.

attributed to the overfitting of the model on a small amount of clean data and is a common observation for active learning methods [3].

When 200 record pairs have been annotated, the ALMSER F1 scores for all tasks are by 0 to 0.032% points lower than the passive learning results that would be achieved by training a random forest classifier with all pairs from the pool as training data. The MB baseline underperforms the QHC baseline for all tasks while it fails to converge after 200 iterations for both the MusicBrainz and the MusicBrainz_mut tasks. Table 2 compares the F1 scores of the baseline methods MB and QHC to ALMSER at three points of the active learning process. We can observe that ALMSER achieves a quicker gain in F1 in the earlier iterations of the active learning process and outperforms the QHC and MB baselines by up to 5.5 and 13.4% points respectively at the 75th iteration. Although ALMSER outperforms the two baseline methods that use no graph signals even in the 200th active learning iteration, the gain in F1 is reduced to 1.9 and 4.8% points for the QHC and MB baselines respectively.

In order to evaluate in which kind of tasks ALMSER achieves the highest boost in comparison to the QHC baseline, which was shown to outperform the MB baseline, we measure the area between ALMSER's and QHC's F1 curves. The area between the F1 curves is the largest for the MusicBrainz_mut task which is the task having the largest containment in corner cases (up to 62%) as well as the highest sparsity (up to 23%) among all multi-source tasks used for the experiments. In contrast, the smallest area between ALMSER's and QHC's F1 curves is the one of the restaurants task, which contains the lowest amount of

corner cases (up to 19%) of all tasks. This indicates that ALMSER is especially fitted for more difficult multi-source matching tasks which require the matcher to deal with different matching patterns.

Table 2. Evaluation of baselines with no or partial graph signals - F_1 and AUC.

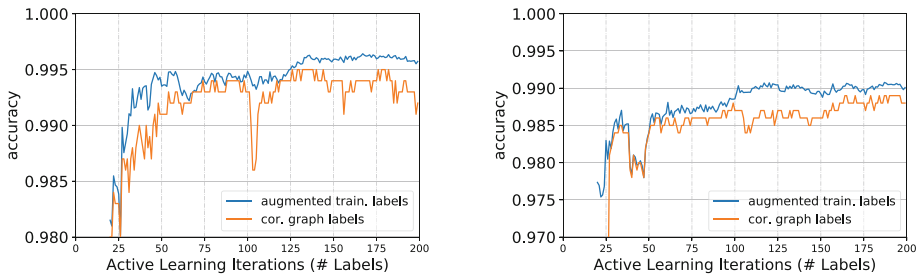
Iteration	AL method	MusicBrainz	MusicBrainz_mut	Computers	Computers_mut	Restaurants
F_1 @ 75th	MB	0.805	0.836	0.881	0.833	0.915
	MB_boost_learner	0.877	0.833	0.889	0.827	0.914
	QHC	0.921	0.851	0.891	0.841	0.917
	QHC_boost_learner	0.891	0.891	0.894	0.843	0.924
	ALMSER_qs	0.912	0.841	0.919	0.842	0.916
	ALMSER	**0.939**	**0.906**	**0.921**	**0.862**	**0.926**
F_1 @ 125th	MB	0.890	0.817	0.912	0.840	0.919
	MB_boost_learner	0.866	0.856	0.910	0.846	0.917
	QHC	0.932	0.893	0.909	0.854	0.925
	QHC_boost_learner	0.914	0.914	0.901	0.865	**0.930**
	ALMSER_qs	0.924	0.884	**0.927**	0.868	0.923
	ALMSER	**0.946**	**0.920**	0.918	**0.873**	0.929
F_1 @ 200th	MB	0.914	0.879	0.925	0.854	0.920
	MB_boost_learner	0.903	0.872	0.922	0.859	0.924
	QHC	0.945	0.908	0.918	0.866	0.927
	QHC_boost_learner	0.926	0.926	0.916	0.871	**0.932**
	ALMSER_qs	0.934	0.896	**0.938**	**0.884**	0.926
	ALMSER	**0.951**	**0.927**	0.930	0.878	0.931
F1-AUC 50th-200th	MB	128.81	123.84	135.68	124.82	138.02
	MB_boost_learner	138.28	131.74	136.56	127.21	138.52
	QHC	140.07	132.43	135.62	127.66	138.51
	QHC_boost_learner	136.39	136.39	134.93	128.82	138.35
	ALMSER_qs	138.37	131.38	138.96	128.95	138.21
	ALMSER	**141.57**	**137.51**	**139.19**	**130.13**	**139.18**

4.4 Evaluation of the Graph-Boosted Components

In this part of our experimental analysis, we evaluate the two graph-boosted components of ALMSER, i.e. the query strategy and the model learning. We evaluate the following three setups that use partial graph signals and compare them to ALMSER: 1. ALMSER_qs, a variation of ALMSER which utilizes the graph signal only as part of the query strategy but not for boosting the learner with additional training data, 2. QHC_boost_learner which applies the QHC query strategy for selecting candidates and uses the graph signal for augmenting the training data and boosting the learner as described in Sect. 3.6, and 3. MB_boost_learner uses the MB query strategy together with augmenting the training data.

We present the F1 scores at three snapshots of the active learning process for all methods that use graph signals for data augmentation in Table 2. Additionally, we report the area under the F1 curve (F1-AUC) from iteration 50 to iteration 200 for all methods. We observe that the best performing methods for all tasks and snapshots use partial, e.g. ALMSER_qs at iteration 125 for the computers, or full graph signals, e.g. ALMSER at iteration 200 for the MusicBrainz_mut task. Although ALMSER underperforms its individual graph-boosted components for five snapshot-task combinations as shown in Table 2, the area under the F1 curve of ALMSER is the largest for all tasks, indicating that ALMSER achieves overall better results between iterations 50 and 200. Comparing the AUC scores of the approaches that partially utilize graph signals, we see that none of them consistently outperforms the other. For some tasks, e.g. MusicBrainz_mut, using graph signals for boosting the learner in combination with a QHC query strategy performs better than exploiting the graph only for selecting query candidates. This observation is reversed for other tasks, e.g. computers_mut.

Finally, we report the size and correctness of the augmented training set which results from the clean components of the correspondence graph, as explained in Sect. 3.4. Figure 6 presents the accuracy of the augmented training set in comparison to the accuracy of the complete correspondence graph for the MusicBrainz_mut and the restaurants tasks, in each active learning iteration. We observe that our heuristic for filtering clean components extracts a cleaner part of the correspondence graph, as the accuracy of the augmented training set exceeds the one of the complete graph in each iteration for multi-source tasks with both duplicate free (MusicBrainz_mut) and non-duplicate free (restaurants) data sources.



(a) MusicBrainz_mut (b) restaurants

Fig. 6. Correctness of augmented training data vs graph labels.

Exploiting the record pairs from the clean components for training the boosted learner, results in large amounts of additional training pairs. However, only the subset of record pairs in the augmented training set with a graph-inferred label different from the base label can give additional matching information to the boosted learner. Table 3 shows the size of the augmented training set,

the amount of record pairs in the training set with a disagreement between the graph-inferred and the base labels, as well as the ratio of correct graph-inferred labels to all record pairs with a disagreement in three active learning snapshots. Although the size of the augmented training set is much larger in comparison to the clean labeled data, there is only a relatively small amount of disagreements between the predictions of the base learner trained on the labeled set and the graph-inferred labels. Considering that for the majority of those disagreements the graph-inferred label is correct (78.8–96.7% as shown in Table 3), we can conclude that the additional matching information that the boosted learner derives from the clean components of the graph, is subject only to a small amount of noise and can successfully support the discovery of additional matching patterns which are not covered yet by the record pairs in the labeled set.

Table 3. Augmented training data in three AL snapshots.

Iteration	Musicbrainz_mut			Restaurants		
	#Train pairs (K)	#Disagr.	% Correct graph	#Train pairs (K)	#Disagr.	% Correct graph
75th	256.5	1,476	0.966	42.8	73	0.821
125th	256.4	1,506	0.967	42.5	52	0.788
200th	254.7	1,009	0.874	42.6	27	0.814

5 Conclusion

This paper presented ALMSER, the first active learning method for multi-source entity resolution. ALMSER exploits the correspondence graph that is available in multi-source entity resolution settings to improve two components of the active learning workflow: the query strategy and the training of the learner. Our evaluation on five multi-source tasks showed that ALMSER outperforms two baseline active learning methods including the state-of-the art committee-based query strategy which use no graph signals in terms of F1 score on all tasks. We evaluated the distinct graph-boosted components of the ALMSER algorithm and showed that utilizing graph signals as part of both the query selection and the model training achieve an increased overall performance.

References

1. Bellare, K., Curino, C., Machanavajihala, A., Mika, P., Rahurkar, M., Sane, A.: WOO: a scalable and multi-tenant platform for continuous knowledge base synthesis. PVLDB **6**(11), 1114–1125 (2013)
2. Bilgic, M., Mihalkova, L., Getoor, L.: Active learning for networked data. In: Proceedings of ICML (2010)

3. Chen, X., Xu, Y., Broneske, D., Durand, G.C., Zoun, R., Saake, G.: Heterogeneous committee-based active learning for entity resolution (HeALER). In: Welzer, T., Eder, J., Podgorelec, V., Kamišalić Latifić, A. (eds.) ADBIS 2019. LNCS, vol. 11695, pp. 69–85. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-28730-6_5
4. Christen, P.: Data Matching: Concepts and Techniques for Record Linkage, Entity Resolution, and Duplicate Detection. Data-Centric Systems and Applications. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-31164-2
5. Christophides, V., Efthymiou, V., Palpanas, T., Papadakis, G., Stefanidis, K.: An overview of end-to-end entity resolution for big data. ACM Comput. Surv. (CSUR) **53**(6), 1–42 (2020)
6. Fellegi, I.P., Sunter, A.B.: A theory for record linkage. J. Am. Stat. Assoc. **64**(328), 1183–1210 (1969)
7. Halevy, A., Rajaraman, A., Ordille, J.: Data integration: the teenage years. In: Proc. VLDB, 9–16 (2006)
8. Heath, T., Bizer, C.: Linked Data: Evolving the Web into a Global Data Space. Synthesis Lectures on the Semantic Web. Morgan & Claypool Publishers (2011)
9. Isele, R., Bizer, C.: Active learning of expressive linkage rules using genetic programming. Web Semant. **23**, 2–15 (2013)
10. Kasai, J., Qian, K., Gurajada, S., Li, Y., Popa, L.: Low-resource deep entity resolution with transfer and active learning. In: Proceedings of ACL (2019)
11. Konda, P., et al.: Magellan: toward building entity matching management systems over data science stacks. PVLDB **9**(13), 1581–1584 (2016)
12. Konyushkova, K., Sznitman, R., Fua, P.: Learning active learning from data. In: Proceedings of Advances in Neural Information Processing Systems (2017)
13. Meduri, V., Popa, L., Sen, P., Sarwat, M.: A comprehensive benchmark framework for active learning methods in entity matching. In: Proceedings of SIGMOD (2020)
14. Mozafari, B., Sarkar, P., Franklin, M., Jordan, M., Madden, S.: Scaling up crowdsourcing to very large datasets: a case for active learning. PVLDB **8**(2), 125–136 (2014)
15. Nafa, Y., et al.: Active deep learning on entity resolution by risk sampling. arXiv preprint arXiv:2012.12960 (2020)
16. Nentwig, M., Hartung, M., Ngonga Ngomo, A.C., Rahm, E.: A survey of current link discovery frameworks. Semant. Web **8**(3), 419–436 (2017)
17. 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). https://doi.org/10.1007/978-3-642-30284-8_17
18. Nguyen, H.T., Smeulders, A.: Active learning using pre-clustering. In: Proceedings of ICML (2004)
19. Papadakis, G., Ioannou, E., Thanos, E., Palpanas, T.: The Four Generations of Entity Resolution. Synth. Lect. Data Manag. **16**(2), 1–170 (2021)
20. Peeters, R., Bizer, C.: Dual-objective fine-tuning of BERT for entity matching. PVLDB **14**(10) (2021)
21. Primpeli, A., Bizer, C.: Profiling entity matching benchmark tasks. In: Proceedings of CIKM (2020)
22. Primpeli, A., Bizer, C., Keuper, M.: Unsupervised bootstrapping of active learning for entity resolution. In: Harth, A., et al. (eds.) ESWC 2020. LNCS, vol. 12123, pp. 215–231. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-49461-2_13
23. Primpeli, A., Peeters, R., Bizer, C.: The WDC training dataset and gold standard for large-scale product matching. In: Companion Proceedings of WWW (2019)

24. Saeedi, A., Peukert, E., Rahm, E.: Comparative evaluation of distributed clustering schemes for multi-source entity resolution. In: Kirikova, M., Nørvåg, K., Papadopoulos, G.A. (eds.) ADBIS 2017. LNCS, vol. 10509, pp. 278–293. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-66917-5_19
25. Saeedi, A., Peukert, E., Rahm, E.: Using link features for entity clustering in knowledge graphs. In: Gangemi, A., et al. (eds.) ESWC 2018. LNCS, vol. 10843, pp. 576–592. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-93417-4_37
26. Sarawagi, S., Bhamidipaty, A.: Interactive deduplication using active learning. In: Proceedings of SIGKDD (2002)
27. Settles, B.: Active Learning: Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers (2012)
28. Shen, W., DeRose, P., Vu, L., Doan, A., Ramakrishnan, R.: Source-aware entity matching: a compositional approach. In: Proceedings of ICDE (2007)