The Case for Multi-task Active Learning Entity Resolution

(Discussion Paper)

Giovanni Simonini, Henrique Saccani†, Luca Gagliardelli, Luca Zecchini, Domenico Beneventano and Sonia Bergamaschi

Università degli Studi di Modena e Reggio Emilia, Italy
<name.surname>@unimore.it
† *247673@studenti.unimore.it*

Abstract

Entity Resolution (ER) is a multi-task process that aims to detect different records in dirty datasets that refer to the same real-world entity. It is a building block for any data integration and cleaning framework. The state-of-the-art ER approaches heavily rely on supervised Machine Learning, hence on the availability of training data—which is notoriously hard to get in many contexts. To overcome this burden, many Active Learning (AL) strategies have been applied to every single task of ER to minimize the amount of training data by judiciously choosing which data to label.

Yet, no study has been conducted to apply AL to ER in a holistic way. This paper aims to fill this gap by experimentally studying on real-world datasets how to tackle ER as a *multi-task* AL problem and provides guidelines on how to balance AL on the different ER tasks to get better results.

Keywords

Entity Resolution, Active Learning, Data Integration

1. Introduction

1.1. Entity Resolution with Active Learning

Identifying records in datasets that refer to the same real-world entity is a fundamental task in data cleaning and data integration—known as *Entity Resolution* (ER).

State-of-the-art ER methods heavily rely on Machine Learning [1], hence on the availability of training data. One way to alleviate this problem is to employ Active Learning (AL), to judiciously choose the data to label so to minimize the human intervention, which is typically the bottleneck [2, 3, 4, 5]. As a matter of fact, AL is commonly employed for training binary classifiers to determine if a pair of records is a match or not. We call these classifiers *matchers*. The basic idea is that for pairs of records we can compute similarity/distance measures (e.g., string similarities of the attributes, such as Jaccard index) that allow to represent them as vectors of features. Then, some of these pairs are labeled by humans to provide a *training*

SEBD 2021: The 29th Italian Symposium on Advanced Database Systems, September 5-9, 2021, Pizzo Calabro (VV), Italy

0000-0002-3466-509X (G. Simonini); 0000-0001-5977-1078 (L. Gagliardelli); 0000-0002-4856-0838 (L. Zecchini); 0000-0001-6616-1753 (D. Beneventano); 0000-0001-8087-6587 (S. Bergamaschi)

CEUR Workshop Proceedings (CEUR-WS.org)

set for a classifier (e.g., Random Forest, SVM, etc.). AL aims to select the pairs that, once employed to train the learner, will increase its accuracy the most. The major shortcoming is that computing a feature vector for all possible pairs of records in a dataset is typically computationally unbearable, due to its inherently quadratic complexity. Thus, *blocking* is employed for discarding pairs of records that most likely will not match. The basic idea is to employ "cheap" similarity (*blocking*) functions that work as surrogates for the "expensive" one employed by the matchers. Also for blocking we can learn *blocking* functions employing standard Machine Learning classifiers [6]—which can be combined with AL.

1.2. Contribution

To date, employing AL-based matchers and blocking for the same ER task has always been studied as separate problems [7]. In this paper, we aim to investigate how a practitioner should allocate her "budget" for labeling data (i.e., a number of instances she is willing to label due to time constraints) between blocking and matching to achieve the best final result for her ER task at hand. Our preliminary results suggest that modeling ER as a *multi-task Active Learning* problem can actually lead to better-quality performance than simply splitting the budget between the two tasks. In Section 4 we demonstrate it on 4 real-world well-known datasets, showing that in some cases a multi-task AL approach can achieve results as good as a state-of-the-art Deep Learning ER algorithm trained with a significantly higher amount of labeled pairs.

2. Related Work

Entity Matching (EM) can be interpreted as a binary classification problem applied to record pairs, labeled as matches or non-matches. AL application to classification tasks has been covered in literature [8] with the goal of maximizing the classification accuracy while minimizing the required human labeling effort. However, state-of-the-art solutions for EM, like Magellan [9] or DeepMatcher [10], do not exploit AL, relying in some cases on other methods to achieve this result (e.g., transfer learning [11]).

If the oracle is a human annotator, AL can be considered as a case of a human-in-the-loop, situation discussed in EM literature [12]. Significant approaches to this problem move from the idea of using crowdsourcing for EM tasks [13, 12]: it is the case of Falcon [3], which proposes the idea of hands-off crowdsourcing (HOC), using the crowd in the major tasks of the EM process.

AL for EM is often faced as a problem of choosing the best combination of example selector and classifier [7]. In many cases, related work focuses either on matching (considering blocking as a separated pre-processing step) [14], or on blocking [15]; when both steps are considered (e.g., Falcon), the different tasks of the EM pipeline are still treated separately, lacking a holistic approach. In fact, even if multi-task AL has already been used for other tasks [16], at the best of our knowledge, it was never applied to data cleaning before our study.

3. Study Methodology

3.1. Active Learning ER pipeline

Figure 1 summarizes the setting of our study: we define two budgets, one for the blocking and one for the matching, which the user should use to label data in an ER pipeline. These two budgets represent the number of records that have to be labeled in each step to train a classifier that is used to discriminate whether a pair is a match or not. In the first *blocking phase*, we employ features that are cheap to be extracted (orders of magnitude less expensive than string similarities).

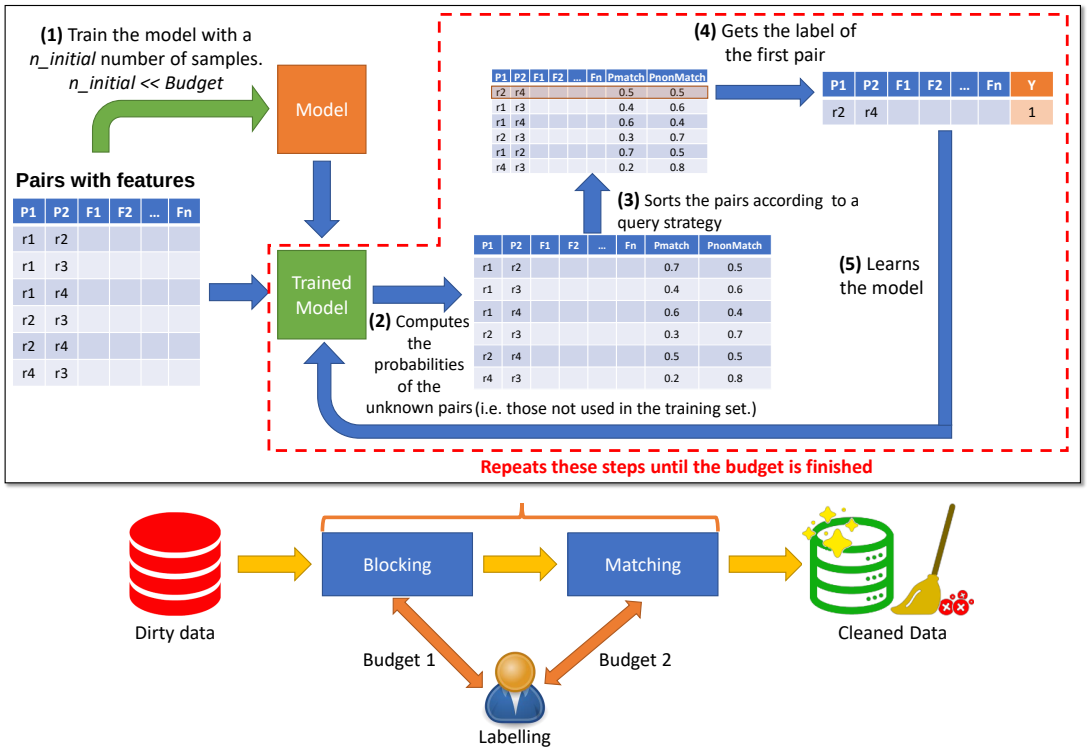


Figure 1: ER pipeline. The upper part of the figure represents the AL process that is executed once for *blocking* and once for *matching*.

Remember that the goal of blocking is indeed to avoid expensive comparisons. Thus, for each pair, we compute a set of features exploiting blocking functions—details below. These features are used to train a classifier with AL using *budget 1*: first, the classifier is trained with a minimal number of samples randomly chosen, then until the budget is consumed the classifier is trained by asking a user to label the most important pairs. At each iteration, the AL algorithm chooses the pairs to be labeled that are more informative, i.e., that will improve the most the accuracy of the classification model once labeled.

At the end of this process, only the pairs that are classified as candidates are kept. In the *matching phase* the same approach is applied, but using a different set of features that are based on expensive—more accurate—string similarity measures. Thus, a classifier is trained with AL

using *budget 2* and applied to all the non-labeled pairs to detect all the matching pairs. Of course, the *matcher* is trained also with the pairs labeled in the blocking phase since already available.

3.2. Blocking and Matching Framework

We define an *entity profile* p_i as a tuple composed of a unique identifier and a set of *name-value* pairs. We call *profile collection* P a set of profiles. Two profiles $p_i, p_j \in P$ are called *matches* if they refer to the same real-world object. Thus, Entity Resolution (ER) is the task of identifying all the matches, given P.

We employed a schema-agnostic redundant blocking technique (i.e. Token Blocking [17]) combined with meta-blocking [6]. We choose this setting because it has been proven to be an efficient and effective blocking technique, which does not require parameter tuning and works also when the schema alignment is not complete (or present at all). Given a profile collection P, Token Blocking generates an inverted index for each token appearing in the profiles; then, it considers each entry of that index as a block. This means that two profiles appear together in a block if they share at least one token, and they will share as many blocks as many are the tokens that they share. The set of resulting blocks is called block collection B.

We indicate with $|B|$ the number of blocks $b_k \in B$ and with $\|b_k\|$ the number of comparisons $\langle p_i, p_j \rangle$ involved by the block b_k. From B we extract the set of features $\mathcal{F}$ as described by Papadakis et al. in [6]. Computing these features from B is orders of magnitude faster than computing string-based similarities, hence being the ideal candidate method for generating the features in the blocking phase. Thus, for each comparison in B we extract a feature vector, which can be used for Active Learning. In particular, we employed the best feature set as reported in [6] and summarized in Table 1—notation in Table 2.

Name	Formula/Description
Node Degree (ND)	Number of non-redundant comparisons involving a profile p_i.
Jaccard Similarity (JS)	$JS(p_i, p_j) = \frac{\lvert B_{i,j} \rvert}{\lvert B_i \rvert + \lvert B_j \rvert - \lvert B_{i,j} \rvert}$
Co-occurrence Frequency-Inverse Block Frequency (CFIBF)	$CFIBF(p_i, p_j) = \lvert B_{i,j} \rvert \cdot log \frac{\lvert B \rvert}{\lvert B_i \rvert} \cdot log \frac{\lvert B \rvert}{\lvert B_j \rvert}$
Reciprocal Aggregate Cardinality of Common Blocks (RACCB)	$RACCB(p_i, p_j) = \sum_{b_k \in B_{i,j}} \frac{1}{\|b_k\|}$

Table 1
Features employed in the blocking phase.

Symbol	Description
P	Profile collection, a set of profiles.
p_i	An entity profile $p_i \in P$
B	Block collection obtained by appling Token Blocking on P
$\lvert B \rvert$	Number of blocks $b_k \in B$
$\langle p_i, p_j \rangle$	A comparison, i.e. two entity profiles that co-occurs in a block
$\|b_k\|$	Number of comparisons involved by the block b_k
B_i	Blocks in which is contained an entity profile p_i

Table 2
Important symbols used throughout the paper.

3.3. Matching

For the matching phase, we employed common similarity/distance functions for computing feature records of pairs of candidate matches. We employed simple heuristic that has been shown to work well on all the datasets: *(i)* for attributes with long string values (i.e., > 20 characters), we computed token-based measures (namely: Jaccard similarity, Dice similarity, Overlap coefficient, and Cosine similarity); *(ii)* for attributes with short string values (i.e., < 20 characters), we computed character-based measures (namely: Levenshtein distance and Needleman–Wunsch distance); *(iii)* for numeric attributes, we computed the ratio $\delta num = min(a, b)/max(a, b)$, for all non-null values and values different from 0.

4. Experiments

4.1. Datasets

In Table 3 we report the characteristics of the employed datasets—commonly used as ER benchmark datasets [6, 10, 13, 18].

We divided them into two main groups:

(a) *structured datasets*, where the attributes are completely aligned and the attribute values are mostly short, carrying specific information about a certain domain (e.g., Title of a journal, or Venue of a conference);

(b) *dirty datasets*, where some information is scattered among attributes (e.g., one may find the Brand of a product in the attribute Title) and also composed of log text describing in plain English some characteristics of the entity (e.g., Description of a product).

Name	#dataset1	#dataset2	#matches	#attributes	type
DblpAcm	2.6k	2.3k	2.2k	4	structured
ScholarDblp	2.5k	61.3k	2.3k	4	structured
AbtBuy	1.1k	1.1k	1.1k	3	dirty
AmazonGoogleProd.	1.4k	3.3k	1.3k	4	dirty

Table 3
Datasets characteristics.

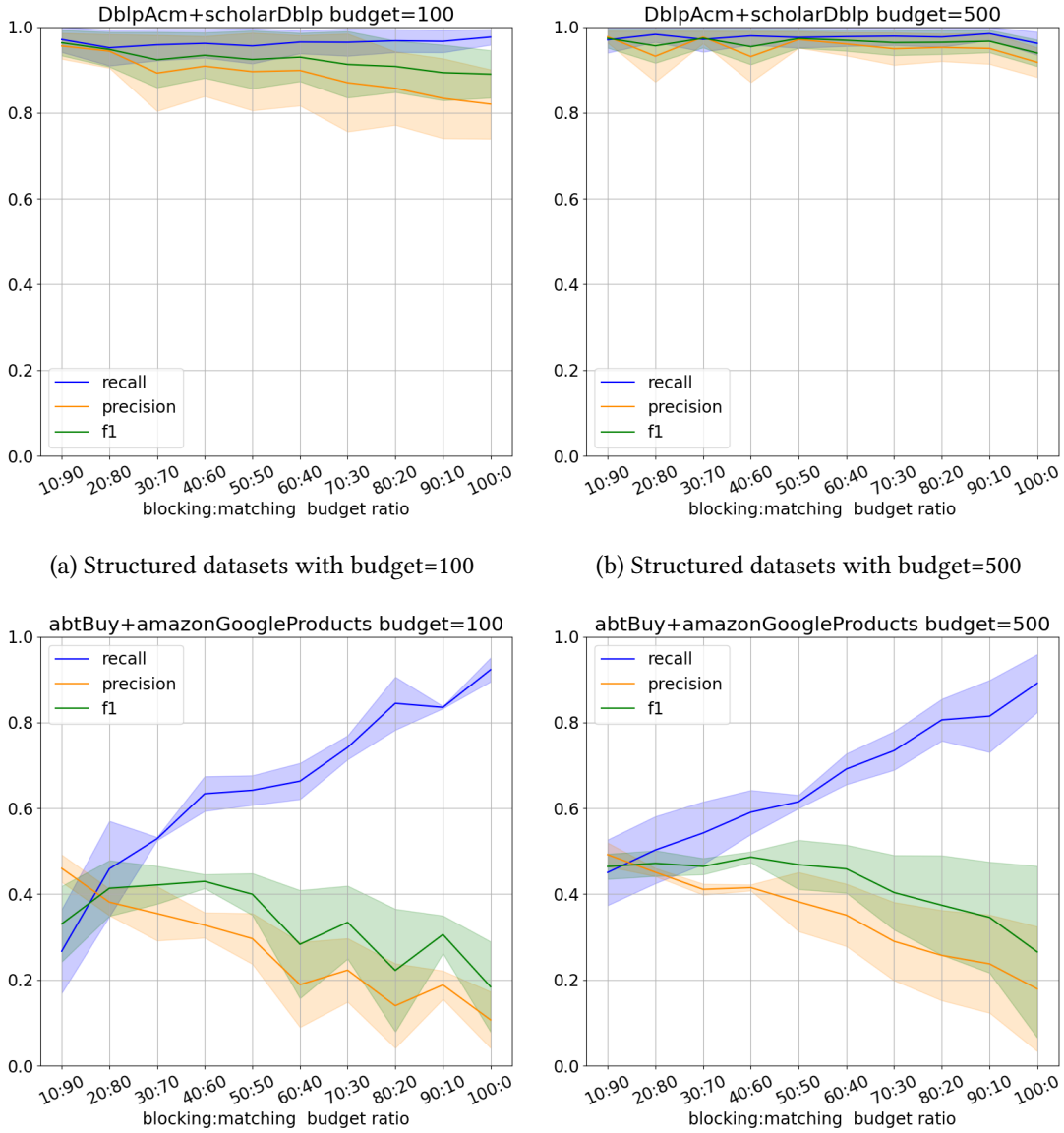
4.2. Setup

As binary classifier, we employ Random Forest[1], and uncertainty-based sampling as AL strategy[2], for both blocking and matching employing the features described in Section 3. We choose this Machine Learning (ML) algorithm for its interpretability and because it has been proven to work well for ER [9]. We consider two sets of budgets for labeling: 100 and 500. This means that for each experiment the AL for blocking and matching can ask to label 100 and 500 instances respectively, in total. These are representative of two typical scenarios: one where the practitioner has limited time to label instances and the other where she has more time (or more collaborators).

[1] https://scikit-learn.org/stable/modules/ensemble.html#forest
[2] https://modal-python.readthedocs.io/en/latest/

4.3. Results

In our experiment, we allocate a different share of AL budget for the blocking and the matching classifier by varying 10% at a time. The results are summarized in Figure 2, grouped by type of dataset (i.e., structured vs. dirty).



(a) Structured datasets with budget=100

(b) Structured datasets with budget=500

(c) Dirty datasets with budget=100

(d) Dirty datasets with budget=500

Figure 2: Results summary. Recall, Precision, and F1-score varying the percentage of the budget allocated to the AL algorithm for blocking and matching.

The results show how with just a little amount as 100 labels, AL can learn an effective ER pipeline for all the datasets: these results in some cases are almost as good as those that can be

obtained with a complex Deep Learning (DL) model and thousands of labeled data, and better than those that can be obtained with traditional ML methods[3].

4.3.1. Structured datasets.

For the structured datasets, with just 100 labels as budget for AL we can achieve almost the highest results published so far. Both DeepMatcher [10] and Magellan [9] (respectively DL and traditional ML state-of-the-art ER tools) are able to reach a 0.984 F-score on the structured DblpAcm dataset, but they require a significant amount of labeled data (more than 9k labeled pairs), while we can achieve a 0.981 F-score exploiting 100 labeled examples (0.990 if using 500). Similarly, for ScholarDblp, our AL reaches a 0.939 (0.972) F-score with a budget of 100 (500) labels, while DeepMatcher achieves a 0.947 F-score with more than 20k labeled pairs.

4.3.2. Dirty datasets.

For dirty datasets, our AL method manages to compete with state-of-the-art solutions relying on a budget of just 500 labeled examples: on AmazonGoogleProducts and AbtBuy, DeepMatcher yields a F-score of 0.693 and 0.628, respectively (trained with more than 6k labeled pairs), while AL achieves 0.463 and 0.521—which is 0.03 lower than Magellan for the former and 0.09 higher for the latter.

4.3.3. Takeaway.

Overall, the results in Figure 2 suggest also that the budget for blocking and matching should be split with a 40:60 ratio, which seems to work well for all the datasets.

5. Conclusions

We investigate the problem of ER as a multi-task Active Learning problem. Surprisingly, we find that AL can be employed to easily achieve state-of-the-art performance with basically no feature engineering and with a very common ML algorithm. Further, our preliminary results suggest that employing only 40% of the budget for labeling pairs for AL-based blocking and the rest of the budget for the AL-based matching seems to be the best configuration for all the datasets. We are currently studying how to combine different AL strategies, ML methods and blocking techniques (*involving loose schema information* [19]), to provide a more depth analysis of the problem.

References

[1] A. Doan, P. Konda, P. Suganthan G. C., Y. Govind, D. Paulsen, K. Chandrasekhar, P. Martinkus, M. Christie, Magellan: toward building ecosystems of entity matching solutions, Commun. ACM 63 (2020) 83–91.

[3] We refer to the results reported in [10].

[2] J. Wang, T. Kraska, M. J. Franklin, J. Feng, CrowdER: Crowdsourcing Entity Resolution, PVLDB 5 (2012) 1483–1494.
[3] S. Das, P. Suganthan G. C., A. Doan, J. F. Naughton, G. Krishnan, R. Deep, E. Arcaute, V. Raghavendra, Y. Park, Falcon: Scaling Up Hands-Off Crowdsourced Entity Matching to Build Cloud Services, in: SIGMOD, ACM, 2017, pp. 1431–1446.
[4] E. K. Rezig, L. Cao, G. Simonini, M. Schoemans, S. Madden, N. Tang, M. Ouzzani, M. Stonebraker, Dagger: A data (not code) debugger, in: CIDR, 2020.
[5] E. K. Rezig, L. Cao, M. Stonebraker, G. Simonini, W. Tao, S. Madden, M. Ouzzani, N. Tang, A. K. Elmagarmid, Data civilizer 2.0: A holistic framework for data preparation and analytics, PVLDB 12 (2019) 1954–1957.
[6] G. Papadakis, G. Papastefanatos, G. Koutrika, Supervised Meta-blocking, PVLDB 7 (2014) 1929–1940.
[7] V. V. Meduri, L. Popa, P. Sen, M. Sarwat, A Comprehensive Benchmark Framework for Active Learning Methods in Entity Matching, in: SIGMOD, ACM, 2020, pp. 1133–1147.
[8] B. Settles, Active Learning Literature Survey, University of Wisconsin-Madison Computer Sciences Technical Report 1648 (2009).
[9] P. Konda, S. Das, P. Suganthan G. C., A. Doan, A. Ardalan, J. R. Ballard, H. Li, F. Panahi, H. Zhang, J. F. Naughton, S. Prasad, G. Krishnan, R. Deep, V. Raghavendra, Magellan: Toward Building Entity Matching Management Systems, PVLDB 9 (2016) 1197–1208.
[10] S. Mudgal, H. Li, T. Rekatsinas, A. Doan, Y. Park, G. Krishnan, R. Deep, E. Arcaute, V. Raghavendra, Deep Learning for Entity Matching: A Design Space Exploration, in: SIGMOD, ACM, 2018, pp. 19–34.
[11] Y. Li, J. Li, Y. Suhara, A. Doan, W. Tan, Deep Entity Matching with Pre-Trained Language Models, PVLDB 14 (2020) 50–60.
[12] D. Firmani, B. Saha, D. Srivastava, Online Entity Resolution Using an Oracle, PVLDB 9 (2016) 384–395.
[13] B. Mozafari, P. Sarkar, M. J. Franklin, M. I. Jordan, S. Madden, Scaling Up Crowd-Sourcing to Very Large Datasets: A Case for Active Learning, PVLDB 8 (2014) 125–136.
[14] K. Qian, L. Popa, P. Sen, Active Learning for Large-Scale Entity Resolution, in: CIKM, ACM, 2017, pp. 1379–1388.
[15] G. Dal Bianco, M. A. Gonçalves, D. Duarte, BLOSS: Effective meta-blocking with almost no effort, Inf. Syst. 75 (2018) 75–89.
[16] Y. Zhang, Q. Yang, A Survey on Multi-Task Learning, CoRR abs/1707.08114 (2017). arXiv:1707.08114.
[17] G. Papadakis, G. M. Mandilaras, L. Gagliardelli, G. Simonini, E. Thanos, G. Giannakopoulos, S. Bergamaschi, T. Palpanas, M. Koubarakis, Three-dimensional Entity Resolution with JedAI, volume 93, 2020, p. 101565.
[18] L. Gagliardelli, S. Zhu, G. Simonini, S. Bergamaschi, Bigdedup: a big data integration toolkit for duplicate detection in industrial scenarios, in: 25th International Conference on Transdisciplinary Engineering (TE2018), volume 7, NLD, 2018, pp. 1015–1023.
[19] D. Beneventano, S. Bergamaschi, L. Gagliardelli, G. Simonini, *BLAST2*: An Efficient Technique for Loose Schema Information Extraction from Heterogeneous Big Data Sources, ACM JIDQ 12 (2020) 18:1–18:22.