Crowdsourced Entity Resolution with Control Queries (Discussion Paper)

Sainyam Galhotra[1], Donatella Firmani[2], Barna Saha[1], and Divesh Srivastava[3]

[1] UMass Amherst {sainyam;barna}@cs.umass.edu
[2] Roma Tre University, donatella.firmani@uniroma3.it
[3] AT&T Labs – Research, divesh@research.att.com

Abstract. Entity resolution (ER) seeks to identify which records in a data set refer to the same real-world entity. Given the diversity of ways in which entities can be represented, ER is known to be a challenging task for automated strategies, but relatively easier for expert humans. Nonetheless, also humans can make mistakes. Our contribution is an error correction toolkit that can be leveraged by a variety of hybrid human-machine ER algorithms, based on a formal way for selecting "control queries" for the human experts. We demonstrate empirically that less recent ER algorithms equipped with our tool can perform even better than most recent ER methods with built-in error correction.

1 Introduction

Entity resolution (ER) is the problem of identifying which records in a data set refer to the same underlying real-world entity [5, 10]. Examples include resolving profiles of people and businesses, or specifications of products and services, from websites and social media. ER can be very intricate, as entities are typically represented in a wide variety of ways that humans can match and distinguish based on domain knowledge, but would be challenging for automated strategies. For these reasons, many frameworks have been developed to leverage humans for performing entity resolution tasks [21, 12], by using, for instance, a crowd-sourcing platform such as Amazon Mechanical Turk. In these frameworks, humans are typically asked questions of the kind "do the record u represent the same entity than the record v?" However, certain record pairs can be challenging to resolve correctly even for humans: take for instance two used iPhones in a retail website, and try to verify whether they are the same model or they have some small differences, for instance in the hardware equipment. Cross-checking human answers can require significant effort, resulting in more money invested in the crowd-sourcing platform or low precision and recall in the computed result.

To this end, we design an error correction toolkit that operates outside the platform, and only requires a small amount of extra questions. Compared to

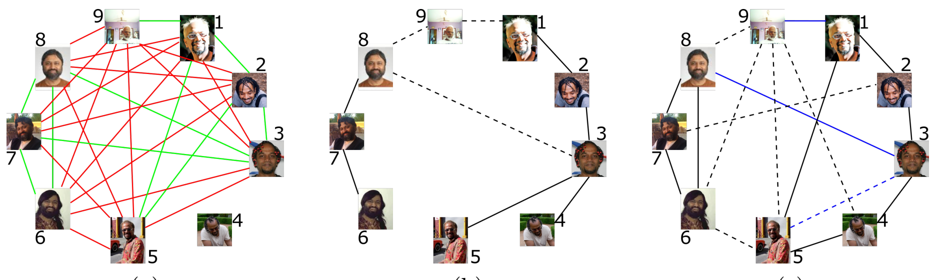


Fig. 1: (a) Example pictures from real web pages. Edges represent matching probabilities (above 0.5 in green). Picture 4 has no face detected. (b-c) Comparison of tree-based (b) with expander-based clusters (c). Positive (negative) answers are represented with solid (dashed) edges, and wrong answers in blue.

other error correction methods, our methodology can be applied to any ER algorithm, boosting its precision and recall and providing formal performance guarantees. In addition, it can be tuned (or even seamlessly turned off) trading off budget for accuracy and adapts to different ER applications.

Main intuition. In literature, there are two main ways to solve ER errors.

– Replicating the same question (i.e. about the same pair) and submitting the replicas to *multiple* humans, until enough evidence is collected for labeling the pair as matching or non-matching (e.g., with voting strategies).
– Using robust clustering techniques for partitioning records into entities, minimizing disagreements between answers (e.g., with correlation clustering).

While recent error-correction works such as [14, 18, 19] provide specific ER algorithms that incorporate both approaches at the same time, we decouple the pair-wise error correction layer from the robust clustering, and focus on which extra-answers would solve disagreements in a principled way, transparently to the underlying ER logic. At the pair level, we can leverage general purpose answer-quality mechanisms already described in the crowd-sourcing literature [2, 23] (that can take advantage of workers meta-data and incentives, to name a few, other than just replicating queries) or use simple majority voting, depending on the application domain. At the cluster level, we proceed as follows.

Expanders. ER strategies based on human experts (e.g., [21, 22, 20, 8]) typically build a *spanning forest* of positive answers for inferring entities via connectivity, based on the assumption that all answers are correct. However, the minimal connectivity of trees yields bad results even in presence of few errors. In a nutshell, we search for queries that can make the connectivity stronger by transforming each tree into an *expander graph*. Graphs with good expansion properties have indeed provably high connectivity and are cheap to build. Intuitively, every edge cut of a graph expander is β times larger than the smallest side of the cut, while every edge of a tree is a cut-edge. This difference is illustrated in Figure 1.

Contributions and outline. We show that the most recent error-correction algorithms can be outperformed by simpler strategies equipped with our expander layer. Section 2 contains an informal overview of our approach. Detailed description of our algorithms and their theoretical basis can be found in [9, 7]. Main experimental results of [9] are reported in Section 3. Finally, Sections 4 and 5 contain related work and our concluding remarks.

2 Overview

We model the pair-wise error correction layer as a black box – that is, a *noisy oracle* – returning already processed pair-wise answers and their corresponding confidence score, possibly labeling some queried pairs incorrectly. We formalize a robust version of the entity resolution problem based on such an oracle and consider *progressive F-measure* [8] as the metric to be maximized in this setting. If one plots a curve of F-measure as a function of the number of oracle queries, progressive F-measure is quantified as the area under this curve.

Problem 1 (Noisy Oracle Strategy) *Given a set of records V, a noisy oracle access to C, and a matching probability function p_m (possibly defined on a subset of $V \times V$), find the strategy that maximizes progressive F-measure.*

Random expansion. Our toolkit guarantees that every cluster – more precisely, the corresponding subgraph induced by positive answers – has good expansion properties. Specifically, every time an ER strategy such as [21, 22, 20, 8] asks a query involving two nodes – say (x, y) – we ask other queries incident to the corresponding clusters c_x and c_y, such as (x_1, y_1) and (x_2, y_2), with $x_1, x_2 \in c_x$ and $y_1, y_2 \in c_y$. Note that extra queries are *not* replicas of (x, y), and they are all distinct. If the answers are positive then the *degree* of nodes within $c_x \cup c_y$ increases, and if the degree gets high enough then we merge c_x and c_y together. We know that random *regular* graphs have good expansion properties [1] in practice, and hence choosing the queries randomly from $c_x \times c_y$ helps us to maintain the required expansion. We refer to our approach as *random expansion*. Technically, we consider *weighted* expanders, where the weight of and edge (x, y) is set to the probability $p_e(x, y)$ that the oracle answer to x, y is erroneous.

Example 1 *In Figure 1 we show the possible result of having an expander ($\beta = 1$) for data in Figure 1a, compared with spanning trees. Both connected components of the left figure (i.e., trees in the spanning forest) and expander graphs of right figure yield the same, correct, clustering* $\{1, 2, 3, 4, 5\}, \{6, 7, 8\}, \{9\}$. *While connected components only work in absence of errors, expander produces the correct clustering also in presence of plausible human errors such as* $(1, 9)$, $(8, 3)$ *(false positives) and* $(3, 5)$ *(false negative). Even though expansion requires more queries than building a spanning forest, it is far from being exhaustive: in the larger cluster out of* $\binom{5}{2} = 10$ *possible queries, only* 5 *are asked.*

Algorithm 1 High-level description on `adaptive`(β) pipeline.

```
1: run node ordering equipped with query_edges($\beta, \cdot, \cdot$) upon disagreement
2: boost_fscore($\beta$)
3: run edge ordering equipped with query_edges($\beta, \cdot, \cdot$) upon disagreement
4: boost_fscore($\beta$)
```

Expander Toolkit. Our toolkit consists of two methods: `query_edges`() and `boost_fscore`(), as described next. The input includes the parameter β, the matching probability function p_m, and the error probabilities p_e.

– Given a query (u, v) selected by an ER strategy, `query_edges`(β, u, v) provides an intermediate layer between the ER logic and the noisy oracle: instead of asking the selected query (u, v) as the considered strategy would do, `query_edges`(β, u, v) selects a bunch of random queries between the cluster of u and the cluster of v, and returns a YES answer only if the joint error probability of the cut is small.
– The `boost_fscore`(β) method instead provides functionalities for maintaining the expansion properties of subgraphs that were not considered during the execution of the ER process, by either adding new edges to weak cuts or splitting the clusters in expander subgraphs. Running `boost_fscore`(β) at a later phase of the ER process can fix premature decisions.

The parameter β trades off the number of queries for precision: smaller values of β correspond to sparser clusters, and therefore to less queries. $\beta = 0$ is equivalent to using the oracle strategy alone. Greater values of β correspond to denser clusters and to higher precision: we prove that its expected value is $1 - O(1/n^\beta)$.

Deployment of the toolkit. Consider a *perfect oracle* strategy such as those in [21, 22, 20, 8], assuming all the answers correct and thus asking a spanning forest. As mentioned, `query_edges`() can be used as a substitute of plain connectivity for deciding if two clusters refer to the same entity or not. For instance, in case of *node ordering* based strategies, such as [20], a singleton node is added to a cluster by querying a single edge with it. We can replace that querying step with `query_edges`() which will try to check if the singleton cluster containing the node refers to the same entity as the one it was supposed to be queried with. The *edge ordering* based strategies, such as [22], query the edges in decreasing order of probabilities to decide if the two endpoint clusters ought to be merged or not. Similarly, we can make the same decision by replacing this querying step with `query_edges`() of the two endpoint clusters. We note that the strategy in [8] combines node and edge ordering, and can be modified to apply random expansion similarly. Along with the incorporation of `query_edges`(), any strategy can call `boost_fscore`() towards the end for correcting the errors made in earlier stages. This shows that our toolkit is independent and easy to use.

Adaptive strategy. In the previous paragraph, we have described a simple way to merge the expansion toolkit with a perfect oracle strategies to minimize the effect of error in the noisy setting. One of the main advantages of the above described methods is that they maintain high precision throughout the

dataset	n	k	$\|C_1\|$	ref.	description
cora	1.9K	191	236	[16]	Title, author, venue, and date of scientific papers.
captcha*	244	69	8	[18]	CAPTCHA images showing four-digit numbers.
gym*	94	12	15	[18]	Images of gymnastics athletes in different positions.

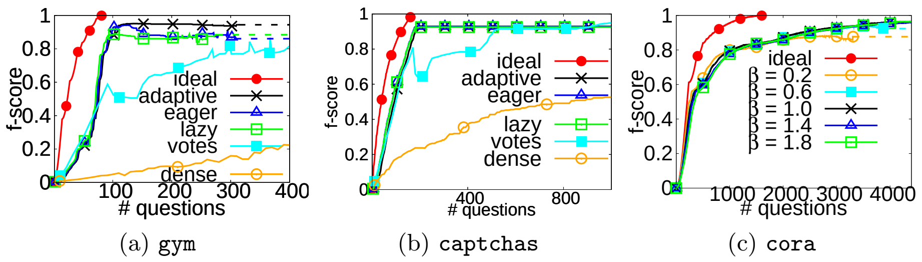
Table 1: Datasets used in our experiments. We report the number of records n, number of clusters k (i.e., entities), size of the largest cluster $|C_1|$, and the paper where they appeared first. Datasets with the symbol $\star$ come with a cache of answers from the AMT crowd. cora has synthetic noisy oracle answers.

resolution phase. This high precision is achieved at the cost of lower progressive F-score. A close look at the expansion toolkit shows that we can possibly improve this shortcoming as well. To this end, we also provide the adaptive() pipeline in Algorithm 1, building on the ideas of the hybrid() method in [8]. Let query_single_edge(u, v) refer to a pair-wise oracle query of the kind "do the record u represent the same entity than the record v?". The intuition of adaptive() is to switch between query_single_edge() and query_edges() depending on the current answer. Specifically, we call query_edges() only if the confidence score returned by the noisy oracle in "disagreement" with the matching probabilities (e.g., a positive answer in case of low matching probability). We make use of boost_fscore() between the node and edge ordering phases, for correcting early errors due to the adaptive nature of the strategy. Specifally, adaptive() allows temporary non-expander clusters, thus allowing spurious violations of our invariant. However, we observe in practice that such behaviour can provide high gains in progressive F-score without losing on precision.

3 Experiments

We compare our algorithms with the most recent approaches for crowdsourced ER [14, 18, 19]. As a frame of reference, we use the ideal curve with that achieves maximum progressive F-score by progressively growing the true clusters as in [8]. We ran experiments on a machine with two CPU Intel Xeon E5520 units with 16 cores each, running at 2.67GHz, with 32GB RAM. We consider both real and synthetic datasets, as summarized in Table 1. We show performances not only of adaptive(), but also of other variants considered in [9], and refer the reader to [9] for a more extensive experimental comparison.

In Figures 2a and 2b, we compare the progressive F-score of our and previous strategies. We set the x axis to the number of *distinct* queries in all the datasets. The plots show that our adaptive() pipeline achieves more than 90% F-score with less than 400 queries in case of gym. Similarly for captchas, the F-score achieved is more than 90% even with the simplest pipeline lazy(), which correspond to the ER strategy in [8] equipped with our boost_fscore() method at the end. This is due to the zero false positives in the oracle answers for captchas. In such scenario, lazy() has 100% precision. Benefits of using our expander pipelines become evident with more challenging dataset, such as gym.



(a) gym (b) captchas (c) cora

Fig. 2: Comparison of the strategies condidered in [9] over datasets of Table 1.

In order to investigate further the effect of oracle answers on `gym`, we set input probability 1 of a pair (u, v) if and only if (u, v) is matching, and 0 otherwise, and generate synthetic erroneous answers as the error rate for both false positives and negatives varies in the range $[0, 0.3]$. In this setting, we observed that performance of `adaptive`() is almost ideal up to answer error rate 0.2, which is higher than the error typically observed in real crowd sourcing platforms such as Amazon Mechanical Turk [2]. Figure 2c demonstrates the effect of changing the tuning parameter β on the performance of `adaptive`() (multiple queries are counted separately). It is evident that the progressive F-score improves as the β value is reduced. The downside of this is that it tries to grow a sparser graph, which has lower precision and the algorithm plateau's out at lower final F-score. On the other hand, higher values of β make the approach conservative to ask more queries and reduces the progressive F-score. The above described behavior is consistent with our theoretical bounds proven in [9].

Alternative forms of expansion. Although random expansion is analytically proven to require less *queries* than deterministic expansion [1], empirically one can make use of alternative forms of expansions for selecting "control queries". The table below compares the number of queries asked by `adaptive`() on `cora` with different expansion strategies, dubbed *high*, *low*, and *uncertain*, selecting queries with the closest matching probability to 1, 0, and 0.5 respectively.

F-score	p^+, p^-	random	high p_m	low p_m	uncertain p_m
0.9	0.1	2420	2481	2355	2417
0.85	0.2	3041	4683	4028	3744

Random expansion gets to the same F-score of deterministic alternatives with less or comparable number of queries. We observed similar results for all the considered datasets. Even though `cora` seems to select *uncertain* as the best deterministic alternative, a deeper look at the experimental data reveals that the 0.5 matching probability bucket is the one containing most edges in `cora`, and drawing edges from this bucket closely resembles random selection.

4 Related Works

ER has a long history, from the seminal paper by Fellegi and Sunter in 1969 [6], which proposed the use of a learning-based approach, to the recently proposed hybrid human-machine approaches. We focus on the latter line of work, which we refer to *crowdsourced ER*, where we can ask questions to humans about two records representing the same real-world entity.

Perfect oracle strategies. The strategies described in [20, 22, 8] abstract the crowd as a whole by an *oracle*, which can provide a correct YES/NO answer for a given pair of items. This allows for leveraging transitivity: when we know that u matches with v and v matches with w, the matching of u and w is inferred automatically. The strategies focus on different ER logics, which can be formally compared in terms of the number of questions asked for 100% recall [8, 15]. Unfortunately, the strategies do not apply to low quality of answers.

Error-correction methods. In order to deal with erroneous answers by humans, recent approaches typically ask the same query to individual crowd workers, rather than making use of control queries to a crowd abstraction such as the noisy oracle. The strategies in [14] submits the same query to multiple crowd workers, and every answer represents a *vote*. The goal is to achieve a clear majority of YES or NO answers for some pairs (u, v), and use those high confidence pairs for building clusters. The strategies described in [18] assume that every crowd worker has a known *error rate* p^E. In this setting, each YES/NO answer for a given pair of items can be interpreted as a matching probability. Such strategies start from an initial clustering and then refines the solution by asking to the crowd. Finally, the work in [19] uses a combination of k-*node* queries (for instance, with $k = 3$, "are u, v and z the same entity?") and pair-wise queries.

Other Related Works. Wang et al. [21] describe a hybrid human-machine framework CrowdER, that automatically detects pairs that have a high likelihood of matching, which are then verified by humans. Gokhale et al. [12] propose a hybrid approach for the end-to-end workflow, making effective use of active learning via human labeling. In [3] the authors devise a partial ordering based technique to resolve entities, which applies for records of text attributes. Progressive F-score has been discussed in [24, 17, 13]. Finally, estimating or improving crowd accuracy in general, as in [4, 11, 2], is outside the scope of this paper.

5 Conclusions

We address the problem of maximizing progressive F-measure for the crowdsourced ER task. We propose an efficient and cost-effective layer which can be applied on typical ER strategies to make them robust to crowd errors, and present different pipelines that use the expansion toolkit to provide high progressive F-score with provable guarantees. Our experiments over real and synthetic datasets confirm our theory and show the superiority of our pipelines over the techniques proposed in the literature. In our future work, we plan to apply our results to *temporal record linkage*, where error probability is higher as two records are farther in time and very small for nearby records of the same entity.

References

1. N. Alon and J. H. Spencer. *The probabilistic method.* John Wiley & Sons, 2004.
2. J. Bragg and D. S. Weld. Optimal testing for crowd workers. In *AAMAS*, 2016.
3. C. Chai, G. Li, J. Li, D. Deng, and J. Feng. Cost-effective crowdsourced entity resolution: A partial-order approach. In *SIGMOD*, pages 969–984, 2016.
4. N. Dalvi, A. Dasgupta, R. Kumar, and V. Rastogi. Aggregating crowdsourced binary ratings. In *WWW*, pages 285–294, 2013.
5. A. K. Elmagarmid, P. G. Ipeirotis, and V. S. Verykios. Duplicate record detection: A survey. *IEEE Trans. Knowl. Data Eng.*, 19(1):1–16, 2007.
6. I. P. Fellegi and A. B. Sunter. A theory for record linkage. *Journal of the American Statistical Association*, 64(328):1183–1210, 1969.
7. D. Firmani, S. Galhotra, B. Saha, and D. Srivastava. Robust entity resolution using a crowdoracle. *IEEE Data Eng. Bull.*, 41(2):91–103, 2018.
8. D. Firmani, B. Saha, and D. Srivastava. Online entity resolution using an oracle. *PVLDB*, 9(5):384–395, 2016.
9. S. Galhotra, D. Firmani, B. Saha, and D. Srivastava. Robust entity resolution using random graphs. In *SIGMOD*, pages 3–18, 2018.
10. L. Getoor and A. Machanavajjhala. Entity resolution: theory, practice & open challenges. *PVLDB*, 5(12):2018–2019, 2012.
11. A. Ghosh, S. Kale, and P. McAfee. Who moderates the moderators?: crowdsourcing abuse detection in user-generated content. In *EC*, pages 167–176, 2011.
12. C. Gokhale, S. Das, A. Doan, J. F. Naughton, N. Rampalli, J. Shavlik, and X. Zhu. Corleone: Hands-off crowdsourcing for entity matching. In *SIGMOD*, pages 601–612, 2014.
13. A. Gruenheid, X. L. Dong, and D. Srivastava. Incremental record linkage. *PVLDB*, 7(9):697–708, 2014.
14. A. Gruenheid, B. Nushi, T. Kraska, W. Gatterbauer, and D. Kossmann. Fault-tolerant entity resolution with the crowd. *arXiv preprint arXiv:1512.00537*, 2015.
15. A. Mazumdar and B. Saha. A theoretical analysis of first heuristics of crowdsourced entity resolution. *AAAI*, 2017.
16. A. McCallum, 2004. `https://people.cs.umass.edu/~mccallum/data.html`.
17. T. Papenbrock, A. Heise, and F. Naumann. Progressive duplicate detection. *Knowledge and Data Engineering, IEEE Transactions on*, 27(5):1316–1329, 2015.
18. V. Verroios and H. Garcia-Molina. Entity resolution with crowd errors. In *ICDE Conference*, pages 219–230, April 2015.
19. V. Verroios, H. Garcia-Molina, and Y. Papakonstantinou. Waldo: An adaptive human interface for crowd entity resolution. In *SIGMOD Conference*, 2017.
20. N. Vesdapunt, K. Bellare, and N. Dalvi. Crowdsourcing algorithms for entity resolution. *PVLDB*, 7(12):1071–1082, 2014.
21. J. Wang, T. Kraska, M. J. Franklin, and J. Feng. Crowder: Crowdsourcing entity resolution. *PVLDB*, 5(11):1483–1494, 2012.
22. J. Wang, G. Li, T. Kraska, M. J. Franklin, and J. Feng. Leveraging transitive relations for crowdsourced joins. In *SIGMOD Conference*, pages 229–240, 2013.
23. P. Welinder, S. Branson, S. Belongie, and P. Perona. The multidimensional wisdom of crowds. In *NIPS*, pages 2424–2432, USA, 2010. Curran Associates Inc.
24. S. E. Whang, D. Marmaros, and H. Garcia-Molina. Pay-as-you-go entity resolution. *Knowledge and Data Engineering, IEEE Transactions on*, 25(5):1111–1124, 2013.