Select Your Questions Wisely: For Entity Resolution With Crowd Errors

Vijaya Krishna Yalavarthi
NTU Singapore
yalavarthi@ntu.edu.sg

Xiangyu Ke
NTU Singapore
xiangyu001@e.ntu.edu.sg

Arijit Khan
NTU Singapore
arijit.khan@ntu.edu.sg

ABSTRACT

Crowdsourcing is becoming increasingly important in entity resolution tasks due to their inherent complexity such as clustering of images and natural language processing. Humans can provide more insightful information for these difficult problems compared to machine-based automatic techniques. Nevertheless, human workers can make mistakes due to lack of domain expertise or seriousness, ambiguity, or even due to malicious intents. The bulk of literature usually deals with human errors via majority voting or by assigning a universal error rate over crowd workers. However, such approaches are incomplete, and often inconsistent, because the expertise of crowd workers are diverse with possible biases, thereby making it largely inappropriate to assume a universal error rate for all workers over all crowdsourcing tasks.

We mitigate the above challenges by considering an uncertain graph model, where the edge probability between two records A and B denotes the ratio of crowd workers who voted YES on the question if A and B are same entity. To reflect independence across different crowdsourcing tasks, we apply the notion of possible worlds, and develop parameter-free algorithms for both next crowdsourcing and entity resolution tasks. In particular, for next crowdsourcing, we identify the record pair that maximally increases the reliability of the current clustering. Since reliability takes into account the connected-ness inside and across all clusters, this metric is more effective in deciding next questions, in comparison with state-of-the-art works, which consider local features, such as individual edges, paths, or nodes to select next crowdsourcing questions. Based on detailed empirical analysis over real-world datasets, we find that our proposed solution, PERC (probabilistic entity resolution with imperfect crowd) improves the quality by 15% and reduces the overall cost by 50% for the crowdsourcing-based entity resolution.

1 INTRODUCTION

Entity Resolution (ER) is the task of disambiguating manifestations of real-world entities in various records by linking and clustering [7]. For example, there could be different ways of addressing the same person in text, or several photos of a particular object. Also known as Deduplication, this is a critical step in data cleaning and analytics, knowledge base construction, comparison shopping, health care, and law enforcement, among many others.

Although machine-based techniques exist for ER tasks, past studies have shown that crowdsourcing can produce higher quality results, especially for more complex jobs such as classification and clustering of images, video tagging, optical character recognition, and natural language processing [8]. Various crowdsourcing services, e.g., Amazon's Mechanical Turk (AMT) and CrowdFlower [17], allow individuals and commercial organizations to set up tasks that humans can perform for certain rewards. Since a crowd tasker does not work for free, bulk of the literature in this domain aims at minimizing the cost of crowdsourcing, while also maximizing the overall ER result quality [2, 20, 22, 25]. However, human workers can be error-prone due to lack of domain expertise, individual biases, task complexity and ambiguity, or simply because of tiredness, and malicious behavior [9, 19]. As an example, even considering answers from workers with high-accuracy statistics in AMT, we find that the average crowd error rate can be up to 25% (we define average crowd error rate in Section 5). State-of-the-art works elude this severe concern by majority voting [20, 22, 25], that is, to ask the same question to multiple people and consider the majority answer; or by assigning a universal error rate for crowd taskers [19]. Many other works bypass this as an orthogonal problem to crowdsourced ER, because there are various approaches to compute and reduce crowdsourcing biases and errors, including [4, 14, 16].

Challenges. Considering the quality assurance as an orthogonal problem to crowdsourced ER, however, is a substandard solution. Instead, approaching both these problems together improves the quality of ER, which is evident from recent works [9, 19, 23]. The majority voting is often unreliable because spammers and low-paid workers may collude to produce incorrect answers [16]. Besides, the tasker crowd is large, anonymous, transient, and it is usually difficult to establish a trust relationship with a specific worker [14]. Each batch of tasks is solved by a group of taskers who may be completely new, and one may not see them again, thereby making it unrealistic to assign a universal error rate for all workers over all crowdsourcing tasks.

The major contribution of our work is to develop an end-to-end pipeline for the crowdsourcing-based ER problem, taking into consideration potential crowd errors. While crowdsourcing a few questions might be sufficient for an initial clustering of records (e.g., one may crowdsource only $n-1$ record pairs so to construct a spanning tree with all n records), in order to improve the ER quality, specifically in the presence of crowd errors, crowdsourcing of more record pairs is necessary. Perhaps, asking the crowd about all $O(n^2)$ record pairs would provide a very good ER accuracy, but that is prohibitively expensive. Hence, the critical question that we investigate in this work is as follows. *Given the current clustering, what is the best*

CIKM'17 , November 6–10, 2017, Singapore, Singapore
© 2017 Association for Computing Machinery.
ACM ISBN 978-1-4503-4918-5/17/11...$15.00
https://doi.org/10.1145/3132847.3132876

datasets	*accuracy: F1-measure*	*# crowdsourced questions* MinMax [9]	DENSE [19]	PC-Pivot [23]	PERC [this work]	*% crowdsourcing cost reduction by* PERC *over* MinMax	DENSE	PC-Pivot
Allsports	0.9	13.6K	16.0K	21.7K	**11.7K**	13.97%	26.87%	46.08%
Gymnastics	0.9	1.3K	1.5K	1.8K	**0.8K**	38.46%	46.67%	55.56%
Landmarks	0.9	11.0K	8.0K	16K	**5.9K**	46.36%	26.25%	63.12%
Cora	0.8	22.5K	14.0K	✗	**7.2K**	68.00%	48.57%	✗

Table 1: Crowdsourcing cost reduction by PERC**: We present the number of crowdsourcing questions required to achieve a certain accuracy for various methods. For details, see Section 5.**

record pair to crowdsource next? Our objective is two-fold: The set of next crowdsourcing questions should be selected in a way that increases the ER accuracy as much as as possible, at the expenses of as few next crowdsourcing questions as possible.

Given its practical importance, not surprisingly, the problem of identifying the next question for crowdsourced ER, in the presence of crowd errors, has been studied recently: MinMax[9], PC-Pivot [23], and DENSE [19]. *These methods consider ad-hoc, local features to select next questions, such as individual paths (e.g.,* MinMax*), nodes (e.g.,* PC-Pivot*), or the set of either positive or negative edges (e.g.,* DENSE, shown in Appendix*). Hence, they generally fail to capture the strength of the entire clustering, resulting in higher crowdsourcing cost to achieve a reasonable ER accuracy.*

Our Contribution. As opposed to local metrics used in prior works, we select the next crowdsourcing question by considering the strength of the entire clustering. Our global metric, denoted as the *reliability*, follows the notion of connected-ness in an uncertain graph. Intuitively, reliability measures how well-connected a cluster is, and also how well-separated two clusters are. We then systematically identify the next crowdsourcing question, either from a weakly connected cluster, or across a pair of clusters that are weakly separated, thereby creating a balance between stronger and weaker components in the clustering. As a consequence, our reliability-based next crowdsourcing algorithm reduces the crowdsourcing cost significantly, which is evident in Table 1.

Our contributions can be summarized as follows.

- For the next crowdsourcing problem, we introduce a novel metric called "reliability" of a clustering, that measures connected -ness within and across clusters by following the notion of uncertain graphs (Section 3). This is more effective than local-feature-based next crowdsourcing approaches [9, 19, 23], as demonstrated with our running example (Section 3) and also verified in our experimental results (see Table 1).
- Using reliability-based next crowdsourcing, we develop an end-to-end solution, PERC, for crowdsourced ER (Section 4). Our algorithms are parameter-free in the sense that we do not require any user-defined threshold values, and no apriori information about the error rate of the crowd workers.
- We perform detailed experiments with four real-world datasets using Amazon's Mechanical Turk platform. The performance analysis illustrates the quality, cost, and efficiency improvements of our framework (Section 5).

Running Example. Consider a dataset of eight images shown in Figure 1. Records *A*, *B* belong to famous American actress and model, *Eva Mendes*; *C*, *D* to Bollywood star and lead actress of the American television series, Quantico, *Priyanka Chopra*; and *E*, *F*, *G*, *H* to Hollywood actor *Tom Cruise*. 80% of crowd workers voted YES that both records in each of the following pairs are

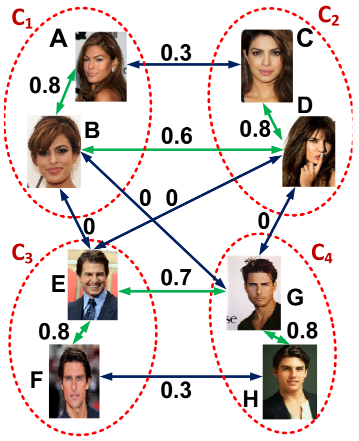


Figure 1: Running example: Edge probability denotes ratio of crowd workers voted YES for the respective records pair to be same entity.

same: $\langle A, B\rangle$, $\langle C, D\rangle$, $\langle E, F\rangle$, and $\langle G, H\rangle$. All crowd workers also answered NO for the edges between the following cluster pairs: $\langle \mathbb{C}_1, \mathbb{C}_3\rangle$, $\langle \mathbb{C}_2, \mathbb{C}_3\rangle$, $\langle \mathbb{C}_1, \mathbb{C}_4\rangle$, and $\langle \mathbb{C}_2, \mathbb{C}_4\rangle$. In this example, four clusters $\mathbb{C}_1, \mathbb{C}_2, \mathbb{C}_3$ and $\mathbb{C}_4$ are formed, as shown in Figure 1. Our objective is to identify the next question to crowdsource that maximizes the gain. It can be observed that asking a question between clusters $\mathbb{C}_3$ and $\mathbb{C}_4$ is more beneficial because all images in $\mathbb{C}_3$ and $\mathbb{C}_4$ belong to the same entity, and one more edge with probability greater than 0.5 helps in merging these two clusters.

2 PRELIMINARIES

2.1 Background

Entity Resolution (ER). An ER algorithm receives an input set of records $R = \{r_1, r_2, \ldots, r_n\}$ and a pairwise similarity function F, and it returns a set of matching pair of records: $\mathbb{C} = \{R_1, R_2, \ldots, R_m\}$, such that, $R_i \cap R_j = \phi$ for all i, j, and $\cup_i R_i = R$. We call each R_i a *cluster* of R, and each cluster represents a distinct real-world entity. The partition of R into a set of clusters is called a *clustering* $\mathbb{C}$ of R. If r_1 and r_2 are matching (non-matching), they are denoted by $r_1 = r_2$ ($r_1 \neq r_2$).

An ER algorithm generally obeys the two following relations.

Transitivity. Given three records r_1, r_2, and r_3, if $r_1 = r_2$ and $r_2 = r_3$, then we have $r_1 = r_3$.

Anti-transitivity. Given three records r_1, r_2, and r_3, if $r_1 = r_2$ and $r_2 \neq r_3$, then we have $r_1 \neq r_3$.

Thus, a clustering $\mathbb{C}$ of the input set R of records is transitively closed. One can derive the following theorem combinatorially. We omit the proof due to limitation of space.

THEOREM 1. *For n records, there can be* $(2^n - n)$ *different clusterings, where each cluster in some clustering must have between* $(1, n)$ *records.*

Crowdsourced ER. We use a crowdsourcing platform such as Amazon's Mechanical Turk (AMT), which provides APIs for conveniently using a large number of human workers to complete micro-tasks (also known as Human Intelligent Tasks (HITs)). To identify whether two records belong to the same entity, we create an HIT for the pair, and publish it to AMT with possible binary answers: A worker needs to submit 'YES' if she thinks that the record pair is matching, and 'NO' otherwise.

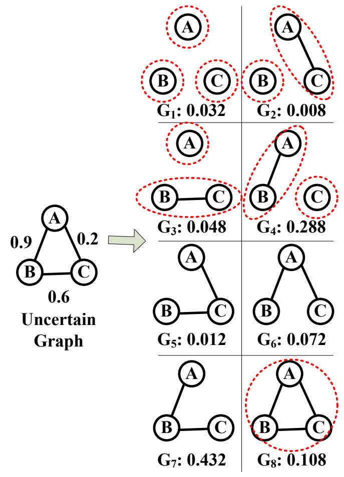


Figure 2: Possible worlds of an uncertain graph: Three possible worlds G_5, G_6, G_7 are not clusterings, as they are not transitively closed. For example, in G_5, $A = C$ and $C = B$, but $A \neq B$, thus violating transitivity.

For mitigating crowd errors, we allow multiple workers to perform the same HIT. We then assign an edge with probability $p(r_i, r_j)$ between two records r_i and r_j, where $p(r_i, r_j)$ is the ratio of crowd workers who voted YES on the question if r_i and r_j are same entity.

Uncertain Graph. Every HIT creates an uncertain, undirected edge between the respective record pair, thereby generating an uncertain, undirected graph $\mathcal{G} = (R, E, p)$, as depicted previously in Figure 1. Each record $r_i \in R$ denotes a node in the graph, $E \subseteq R \times R$ represents the set of edges between the record pairs that were crowdsourced, and $p(e) \in (0, 1)$ is the probability of the edge $e \in E$ as derived earlier. In our context, it is important to note that $p(e) = 0$ (i.e., all crowd workers voted non-matching) is *not* equivalent to the edge e being absent in $\mathcal{G}$ (i.e., the pair is not crowdsourced yet).

To reflect independence across different crowdsourcing tasks (i.e., each HIT can be performed by a different set of workers), we employ the well-established notion of possible world, together with the assumption that each edge can be matching or non-matching, independent of other edges [12]. Hence, the uncertain graph $\mathcal{G}$ yields $2^{|E|}$ deterministic graphs (or, possible worlds) $G \sqsubseteq \mathcal{G}$, where each G is a pair (R, E_G), with $E_G \subseteq E$ are matching edges, and its probability of being observed is given in Equation 1.

$$P(G) = \prod_{e \in E_G} p(e) \prod_{e \in E \setminus E_G} (1 - p(e)) \tag{1}$$

Next, we have the following observation.

LEMMA 1. *Every clustering of the input record set R corresponds to some possible world of the uncertain graph $\mathcal{G} = (R, E, p)$. However, every possible world of $\mathcal{G}$ might not be a clustering of R.*

The first part of the lemma is trivial (i.e., follows from the definition of a possible world), whereas the second part holds since every possible world is not transitively closed. We demonstrate this fact with an example in Figure 2, where three possible worlds G_5, G_6, and G_7 of the given uncertain graph are not clusterings.

Since every clustering corresponds to some possible world, we define the likelihood of a clustering as the probability of the respective possible world being observed. In Figure 2, the likelihood of the clustering $\{(A, B), (C)\}$ is same as $P(G_4)$, which is 0.288.

2.2 Entity Resolution Problem

Given $R, \mathcal{G}$, let us consider a clustering $\mathbb{C} = \{R_1, R_2, \dots, R_m\}$ of R. We define the likelihood of $\mathbb{C}$ as the probability that (1) all edges inside every cluster R_i exist, and (2) all edges across every pair of clusters R_j, R_k do not exist. Since an edge can exist independent of others, we compute the likelihood $L(\mathbb{C})$ as follows.

$$L(\mathbb{C}) = \prod_{R_i \in \mathbb{C}} \left[\prod_{e \in E \cap (R_i \times R_i)} p(e) \right] \times \prod_{\substack{R_j, R_k \in \mathbb{C} \\ j < k}} \left[\prod_{e \in E \cap (R_j \times R_k)} (1 - p(e)) \right] \tag{2}$$

We formally introduce the ER problem below.

PROBLEM 1 (ENTITY RESOLUTION). *Given the set R of records and an uncertain graph $\mathcal{G} = (R, E, p)$, find the (transitively closed) clustering $\mathbb{C}$ of R having the highest likelihood $L(\mathbb{C})$.*

The problem of finding the most-likely clustering (also referred to as the maximum-likelihood clustering), however, is **NP**-hard, which can be verified by a polynomial-time reduction from the **NP**-hard correlation clustering problem [19].

THEOREM 2. *Given an uncertain graph $\mathcal{G} = (R, E, p)$ over records set R, finding the maximum-likelihood clustering of R is* **NP**-*hard.*

Correlation clustering is the most natural setting for clustering a set of records that are connected by both positive and negative edges [10]. Many approximate and heuristic algorithms were proposed for correlation clustering [6, 19]. Indeed, all prior works such as [9, 19, 23] in the domain of crowdsourced ER, that incorporated human error, also employed correlation clustering. Therefore, in our PERC framework, we apply correlation clustering for the ER problem. Details about our clustering algorithm will be given in Section 4. We shall first introduce our next crowdsourcing algorithm in the following, which is the key contribution of this work.

3 NEXT CROWDSOURCING

We discuss our algorithm for selecting the next crowdsourcing question. We assume that an initial (maximum-likelihood) clustering $\mathbb{C}$ is already constructed from the records set R and the uncertain graph $\mathcal{G} = (R, E, p)$, and now we want to identify the best entity pair $\langle r_i, r_j \rangle \notin E$ to crowdsource next.

3.1 Reliability of a Clustering

Intuitively, our objective is to identify a pair $\langle r_i, r_j \rangle \notin E$ that can improve the quality of the given clustering as much as possible. To this end, we identify the two following "connected-ness"-based criteria that determine the quality of a clustering $\mathbb{C}$. Let us denote $\mathbb{C} = \{R_1, R_2, ..., R_m\}$, where each R_i is a cluster and represents a distinct real-world entity.

- How well each cluster R_i is connected?
- How well every pair of clusters R_j, R_k $(j < k)$ is disconnected?

Given a clustering $\mathbb{C} = \{R_1, R_2, ..., R_m\}$ and the uncertain graph $\mathcal{G} = (R, E, p)$, all edges inside a cluster are called YES edges, whereas the edges across two clusters are referred to as NO edges. If $e \in E$ is an YES edge, we define its existence probability $p_Y(e) = p(e)$. On the other hand, if $e \in E$ is a NO edge, we compute its existence probability as $p_N(e) = 1 - p(e)$. We derive an YES-NO

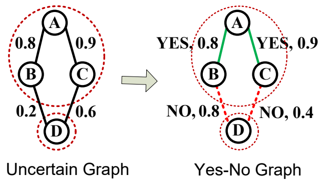


Figure 3: Reliability of a clustering

graph $\mathcal{G}_{Y|N} = (R, E, p_{Y|N}, L)$ from the uncertain graph $\mathcal{G}$ as follows. $\mathcal{G}_{Y|N}$ has the same set of nodes and edges as $\mathcal{G}$, but each edge e in $\mathcal{G}_{Y|N}$ has a binary label $L(e)$, which can be either YES or No, as defined above. For a YES edge e, its probability $p_{Y|N}(e) = p_Y(e)$. For a NO edge e, its probability $p_{Y|N}(e) = p_N(e)$. Next, we formalize the notion of connectivity and disconnectivity.

Definition 1 (Connectivity). *Given a cluster R_i and the YES-NO graph $\mathcal{G}_{Y|N}$, the connectivity of R_i is defined as the sum of the probability of those possible worlds of $\mathcal{G}_{Y|N}$ where all records in R_i are connected by YES edges. Formally,*

$$Connect(R_i) = \sum_{G \sqsubseteq \mathcal{G}_{Y|N}} [I(R_i, G) \times P(G)] \tag{3}$$

In the above equation, $I(R_i, G)$ is an indicator function over a possible deterministic graph $G \sqsubseteq \mathcal{G}_{Y|N}$ taking value 1 if records in R_i are all connected (by YES edges) in G, and 0 otherwise.

Definition 2 (Disconnectivity). *Given a pair of clusters R_j, R_k $(j < k)$ and the YES-NO graph $\mathcal{G}_{Y|N}$, the disconnectivity between R_j, R_k is defined as the sum of the probability of those possible worlds of $\mathcal{G}_{Y|N}$ where at least one NO edge exists between R_j and R_k. Formally,*

$$Disconnect(R_j, R_k) \tag{4}$$
$$= \begin{cases} 0 & ; if\ (R_j \times R_k) \cap E = \phi \\ 1 - \prod_{(r_i, r_l) \in (R_j \times R_k) \cap E} (1 - p_N(r_i, r_l)) & ; otherwise \end{cases}$$

Based on the above definition, we observe that for all $i, j, k; j < k$, the following events are independent. (1) A cluster R_i is connected, and (2) a pair of clusters R_j, R_k are disconnected. Therefore, one can multiply the probability of these events to measure the overall quality of a clustering $\mathbb{C}$. For practical reasons, we avoid multiplying fractions, and instead compute summation over logarithms (Equation 5). Thus, if either of $Connect(R_i)$ or $Disconnect(R_j, R_k)$ is zero, we substitute it by a very small positive fraction. Formally, we denote this metric as the *reliability* of a clustering.

Definition 3 (Reliability). *Given a clustering $\mathbb{C} = \{R_1, R_2, \ldots, R_m\}$ and the YES-NO graph $\mathcal{G}_{Y|N}$, the reliability of $\mathbb{C}$ is defined as the probability that every cluster R_i is connected and every pair of clusters R_j, R_k $(j < k)$ is disconnected, i.e.,*

$$Rel(\mathbb{C}) = \sum_i \log(Connect(R_i)) + \sum_{j<k} \log\left(Disconnect\left(R_j, R_k\right)\right) \tag{5}$$

Example 1. *In Figure 3, we compute the reliability of the clustering $\mathbb{C} = \{(A, B, C), (D)\}$. We first construct the YES-NO graph on the right. Then, we have: $Connect(A, B, C) = 0.72$, $Connect(D) = 1.0$, and $Disconnect((A, B, C), (D)) = 1 - (1 - 0.8)(1 - 0.4) = 0.88$. Hence, $Rel(\mathbb{C}) = \log 0.72 + \log 1 + \log 0.88 \approx -0.20$.*

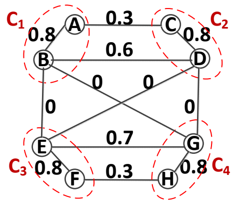


Figure 4: Reliability-based next crowdsourcing: running example

3.2 Next Crowdsourcing Problem

We derive, for every record pair $\langle r_i, r_j \rangle \notin E$, the improvement in reliability of the already computed clustering $\mathbb{C}$, if one crowdsources the pair, and thereby assigns the edge probability $p(r_i, r_j)$. However, one does not know $p(r_i, r_j)$ apriori. Therefore, we consider an optimistic scenario, that is, for all possible values of $p(r_i, r_j) \in (0, 1)$, we derive what will be the maximum possible increment in $Rel(\mathbb{C})$ by crowdsourcing $\langle r_i, r_j \rangle$. We select the record pair that maximally increases $Rel(\mathbb{C})$, under such optimistic assumption.

Our formulation has several desirable features, such as monotonicity and improving weaker components, as stated next.

Lemma 2. *For any new edge e that we crowdsourced, $Rel(\mathbb{C})$ will increase maximally when $p_{Y|N}(e) = 1$.*

In other words, if the new edge e is inside a cluster (i.e., YES edge), then its probability requires to be $p(e) = 1$, which means that all workers agreed on the record pair as matching. On the other hand, if the new edge e is across two clusters (i.e., NO edge), then its probability must be $p(e) = 0$, which implies that all workers agreed on the record pair as non-matching. To put it simply, if the next crowdsourcing result is fully consistent with our previous clustering, then the quality of the clustering improves maximally.

Lemma 3. *By adding a new edge e, the reliability of $\mathbb{C}$ remains the same when $p_{Y|N}(e) = 0$. It increases monotonically as we have larger values of $p_{Y|N}(e)$.*

Generally speaking, the more is the ratio of workers who agree with the previous clustering, the higher is the improvement in the clustering quality.

Lemma 4. *For any new edge e that we included, if $p_{Y|N}(e) >$ 0.5, the maximum-likelihood clustering, as defined in Problem 1, remains the same for the updated graph.*

This implies that if the majority of the crowd workers agree with our previous clustering, there is no need to change the clustering.

Below, we formally introduce the next crowdsourcing problem.

Problem 2 (Next Crowdsourcing). *Given the set R of records, an uncertain graph $\mathcal{G} = (R, E, p)$, and a clustering $\mathbb{C}$, find the record pair $\langle r_i, r_j \rangle \notin E$, such that adding an edge (r_i, r_j), with $p_{Y|N}(r_i, r_j) = 1$, maximally increases the reliability of $\mathbb{C}$.*

3.3 Demonstration with Running Example

We now demonstrate how our reliability-based next crowdsourcing technique deals with the running example in Figure 1.

Example 2. *Figure 4 is the abstract version of our running example in Figure 1. The clustering algorithm identifies four clusters: $\mathbb{C}_1 = \{A, B\}$, $\mathbb{C}_2 = \{C, D\}$, $\mathbb{C}_3 = \{E, F\}$, and $\mathbb{C}_4 = \{G, H\}$. Each cluster has connectivity 0.8. The disconnectivity values across these* *clusters are as follows. Disconnect*$(\mathbb{C}_1,\mathbb{C}_2)$ *= 0.82, Disconnect*$(\mathbb{C}_2,\mathbb{C}_3)$ *= 1, Disconnect*$(\mathbb{C}_1,\mathbb{C}_3)$ *= 1, and Disconnect*$(\mathbb{C}_3,\mathbb{C}_4)$ *= 0.79. As Disconnect*$(\mathbb{C}_3,\mathbb{C}_4)$ *is the least among all others, our algorithm priorities crowdsourcing an edge across* $\mathbb{C}_3,\mathbb{C}_4$*. Intuitively, the separation between* $\mathbb{C}_3,\mathbb{C}_4$ *is the weakest, thus we require to ask more questions about this separation. The reliability gain by adding a new edge e between* $\mathbb{C}_3,\mathbb{C}_4$*, having probability* $p_{Y|N}(e) = 1$*, is* $\log 1 - \log 0.79$*=0.10; whereas, the reliability gain by adding a new edge e between* $\mathbb{C}_1,\mathbb{C}_2$*, with probability* $p_{Y|N}(e) = 1$*, is* $\log 1 - \log 0.82$*=0.08. Hence, for next crowdsourcing, our algorithm selects an edge across* $\mathbb{C}_3,\mathbb{C}_4$*. Indeed, one more edge with probability greater than 0.5 across* $\mathbb{C}_3,\mathbb{C}_4$ *helps in merging these two clusters, while an edge with probability less than 0.5 will make their separation stronger. This is consistent with our running example that asking a question across clusters* $\mathbb{C}_3$ *and* $\mathbb{C}_4$ *is more beneficial.*

Remarks. As demonstrated with the running example, our next crowdsourcing method usually prioritizes the weaker components and improves their quality, thereby creating a balance between the quality of stronger and weaker components in the clustering. This is evident if we consider two pairs of clusters such that $Disconnect(R_1, R_2) < Disconnect(R_3, R_4)$, then our method will always prioritize a pair $\langle r_1, r_2\rangle \in R_1 \times R_2$ over any other pair $\langle r_3, r_4\rangle \in R_3 \times R_4$, for the next crowdsourcing. For brevity, let us denote by $d_1 = Disconnect(R_1, R_2)$ and $d_2 = Disconnect(R_3, R_4)$. In the first case, we consider an edge (r_1, r_2) with $p_N(r_1, r_2) = 1$, i.e., $p(r_1, r_2) = 0$. Hence, the increase in reliability, following Equation 5, is $\log(1/d_1)$. Analogously, in the second case, the increase in reliability is $\log(1/d_2)$. Since $d_1 < d_2$, the pair $\langle r_1, r_2\rangle$ is preferred over $\langle r_3, r_4\rangle$.

In case of connectivity of individual clusters, in general no such relationship exists. However, if the connectivity of one cluster is significantly smaller than that of the other, e.g., $Connect(R_1) << Connect(R_2)$, it is very likely that our method will select a pair from R_1 for the next crowdsourcing problem. Let $c_1 = Connect(R_1)$ and $c_2 = Connect(R_2)$. Also, assume that δ_1 is the maximum increase in c_1 if we add an edge e of probability $p_Y(e) = 1$ (i.e., $p(e) = 1$) in R_1. Similarly, let δ_2 be the maximum increase in c_2 if we add an edge e' of probability $p_Y(e') = 1$ (i.e., $p(e') = 1$) in R_2. Hence, in the first case, the increase in reliability is $\log(1 + \delta_1/c_1)$, whereas in the second case, the increase in reliability is $\log(1 + \delta_2/c_2)$. Since $c_1 << c_2$, it is very likely that $\delta_1/c_1 > \delta_2/c_2$. Therefore, in such cases, our method will prioritize a specific record pair from R_1 over all pairs from R_2, for the next crowdsourcing problem.

3.4 Next Crowdsourcing Algorithm

Difficulties. A naïve algorithm to find the best record pair for next crowdsourcing would be inefficient due to the following challenges.

- Computing the connectivity of a cluster, also known as the *all-terminal-reliability problem in device networks*, is #**P**-hard [12]. Hence, finding the exact connectivity value, even for a modest size cluster, is almost infeasible.
- At each round of crowdsourcing, we identify the best record pair not in E. Usually, the uncertain graph $\mathcal{G}$ is sparse, that is, $|E| << O(|R|^2)$. Therefore, at every round, one needs to compare $O(|R|^2)$ pairs in order to identify the best one for next crowdsourcing.

Monte Carlo Sampling. Due to its intrinsic hardness, we tackle the connectivity estimation problem from an approximation viewpoint. We use the answer computed by Monte Carlo (MC) sampling as a proxy. This is a reasonable choice as MC-sampling is an unbiased estimator, thus by running it for a sufficiently large number of times, its answer is expected to converge to the real answer with a high probability. In particular, we first sample t possible graphs, $G_1, G_2, \ldots, G_t$ of a subgraph of $\mathcal{G}_{Y|N}$ induced by the nodes in some cluster R_i, according to (YES) edge probability $p_{Y|N} = p_Y$. We then compute the ratio of possible graphs which are connected, out of t possible graphs that were generated. This gives the MC-estimation of connectivity for cluster R_i. To speed up the sampling process, we combine MC-sampling with a breadth first search (BFS) from one of the nodes in R_i [12]. If the maximum numbers of nodes and edges in a cluster are n_{max} and e_{max}, respectively, then the time complexity of MC-based connectivity estimation is given by $O\left(t\left(n_{max} + e_{max}\right)\right)$. Based on empirical results over our datasets, we observed that the MC-estimator converges with a number of samples $t \approx 1000$. This is roughly the same number observed in [12] for MC-sampling based reliability estimation over other real-world uncertain graphs.

Algorithm. The complete method for next crowdsourcing is given in Algorithm 1. Let us denote by *priority* of a pair $\langle r_i, r_j\rangle \notin E$ as the increase in reliability of the existing clustering, when the edge (r_i, r_j) is included with probability $p_{Y|N}(r_i, r_j) = 1$ (lines 7 and 14, Algorithm 1). At every round, we crowdsource the record pair with the highest priority. However, priority computation for all pairs at every round would be expensive. We discuss below how one can minimize the required number of priority computations.

We note that for a specific round, the priority of all the following record pairs $\langle r_k, r_l\rangle \in (R_i \times R_j) \setminus E$, for a certain R_i and R_j, are the same. Therefore, we compute the priority of only one record pair across every pair of clusters (lines 11-16, Algorithm 1). Finally, if an edge was inserted in some cluster R_i in the last round and there is no change in the previous clustering (lines 18-24, Algorithm 1), then the priority of the pairs inside other clusters, as well as those across two clusters, will not change. Similarly, if an edge was inserted between two clusters R_i, R_j in the last round and there is no change in the earlier clustering (lines 25-30, Algorithm 1), the priority of the pairs inside all clusters, as well as those across other cluster pairs, will not change. All of these reduce the priority re-computation necessary for at most $O(n_{max}^2)$ pairs at every round, if there is no change in the previous clustering.

In reality, n_{max} is small, around 30~350 records, for the real-world datasets that we have considered (and also used by state-of-the-art approaches [9, 19, 23]). Thus, overall time complexity of our next crowdsourcing algorithm is $O\left(n_{max}^2\left(t\left(n_{max} + e_{max}\right)\right)\right)$. In fact, the priority of each record pair inside a cluster can be computed in parallel, and/or one may sample a selected number of record pairs, uniformly at random, from the cluster; thereby further reducing the time required to select the next crowdsourcing question.

Asking Next Questions in Batches. Algorithm 1 selects a single question to ask next to the crowd workers. Instead, one may consider a batch version to issue multiple high-quality questions. For a batch size k (k is a tunable input parameter), we select the k record

Algorithm 1 Next Crowdsourcing Algorithm

```
Require: Records set $R$, uncertain graph $\mathcal{G} = (R, E, p)$, clustering $\mathbb{C}$
Ensure: Record pair $\langle r_i, r_j \rangle \notin E$ to be crowdsourced next
 1: Let $\mathbb{C} = \{R_1, R_2, \ldots, R_m\}$
 2: if Clustering updated last round then
 3:     priority queue $Q = \phi$
 4:     for all $R_i$ do
 5:         for all $\langle r_j, r_k \rangle \in (R_i \times R_i) \setminus E$ do
 6:             Form $\mathcal{G}'$ by adding $(r_j, r_k)$ in $\mathcal{G}$, with $p_Y(r_j, r_k) = 1$
 7:             $prio(r_j, r_k) = Rel_{\mathcal{G}'}(\mathbb{C}) - Rel_{\mathcal{G}}(\mathbb{C})$
 8:             Insert $(\langle r_j, r_k \rangle, prio(r_j, r_k))$ into $Q$
 9:         end for
10:     end for
11:     for all $(R_j \times R_k), j < k$ do
12:         Find one $\langle r_i, r_l \rangle \in (R_j \times R_k) \setminus E$
13:         Form $\mathcal{G}'$ by adding $(r_i, r_l)$ in $\mathcal{G}$, with $p_N(r_i, r_l) = 1$
14:         $prio(r_i, r_l) = Rel_{\mathcal{G}'}(\mathbb{C}) - Rel_{\mathcal{G}}(\mathbb{C})$
15:         Insert $(\langle r_i, r_l \rangle, prio(r_i, r_l))$ into $Q$
16:     end for
        /* Clustering not Updated in last round */
17: else
18:     if last edge was inserted in $R_i$ then
19:         for all $\langle r_j, r_k \rangle \in (R_i \times R_i) \setminus E$ do
20:             Form $\mathcal{G}'$ by adding $(r_j, r_k)$ in $\mathcal{G}$, with $p_Y(r_j, r_k) = 1$
21:             $prio(r_j, r_k) = Rel_{\mathcal{G}'}(\mathbb{C}) - Rel_{\mathcal{G}}(\mathbb{C})$
22:             Update $(\langle r_j, r_k \rangle, prio(r_j, r_k))$ into $Q$
23:         end for
24:     end if
25:     if last edge was inserted between $R_j$ and $R_k$, $j < k$ then
26:         Find one $\langle r_i, r_l \rangle \in (R_j \times R_k) \setminus E$
27:         Form $\mathcal{G}'$ by adding $(r_i, r_l)$ in $\mathcal{G}$, with $p_N(r_i, r_l) = 1$
28:         $prio(r_i, r_l) = Rel_{\mathcal{G}'}(\mathbb{C}) - Rel_{\mathcal{G}}(\mathbb{C})$
29:         Insert $(\langle r_i, r_l \rangle, prio(r_i, r_l))$ into $Q$
30:     end if
31: end if
32: $\langle r_i, r_j \rangle = Q.pop()$
33: return $\langle r_i, r_j \rangle$
```

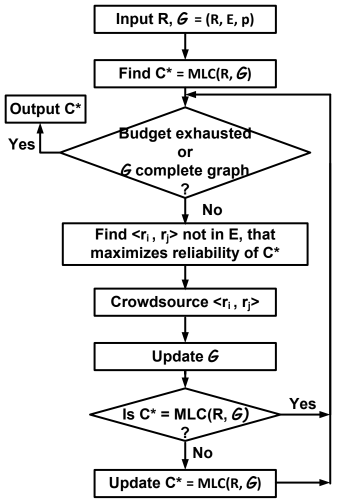
pairs having the highest priority. It is expected that by issuing multiple questions in batches, the overall quality would decrease, because one does not know the corresponding edge probabilities apriori; and therefore, we compute the priority of a record pair in an optimistic manner. However, asking questions in batches helps in reducing the running time of crowdsourced ER, because many crowd workers would be able to answer the questions in a batch in parallel.

4 THE PERC FRAMEWORK

The reliability-based next crowdsourcing method (Section 3) forms the crux of our PERC framework. Clearly, given a set of records and their similarity values obtained via next crowdsourcing, one requires to cluster these records. We discuss our clustering technique in Section 4.1, and then provide in Section 4.2 the complete pipeline that combines our next crowdsourcing and clustering algorithms.

4.1 Clustering Algorithm

Given the records set R and an uncertain graph $\mathcal{G} = (R, E, p)$, we use correlation clustering to find the maximum-likelihood clustering of R (Problem 1). We recall that all prior works in crowdsourced ER, e.g., [9, 19, 23], which incorporated human error, also employed



Input R, $\mathcal{G}$ = (R, E, p)
Find C* = MLC(R, $\mathcal{G}$)
Output C*
Yes
Budget exhausted or $\mathcal{G}$ complete graph ?
No
Find <r$_i$, r$_j$> not in E, that maximizes reliability of C*
Crowdsource <r$_i$, r$_j$>
Update $\mathcal{G}$
Is C* = MLC(R, $\mathcal{G}$) ?
Yes
No
Update C* = MLC(R, $\mathcal{G}$)

Figure 5: Overview of our PERC **framework**

correlation clustering. Since correlation clustering is **NP**-hard, several approximate and heuristic algorithms exist [6]. We empirically compare them, and find the *Spectral-Connected-Components* (SCC) technique to be the most effective one. This is also the same clustering method used in DENSE entity resolution [19].

Spectral-Connected-Components (SCC). This algorithm starts from the record pair having the highest probability of being the same entity, given the answers for these two records. If this probability is higher than 0.5, SCC merges the two records into one cluster. In each successive step, the algorithm finds the clusters with the highest probability of being the same entity, given the answers between them. If this probability is higher than 0.5, the two clusters are merged into one cluster. Otherwise, SCC stops merging clusters, and returns as output the current set of clusters.

Given two clusters R_i and R_j, SCC computes the probability $Pr(R_i, R_j)$ of merging them as given in Equation 6.

$$Pr(R_i, R_j) = \frac{\prod_{(r_k, r_l) \in (R_i \times R_j) \cap E} p(r_k, r_l)}{\prod_{(r_k, r_l) \in (R_i \times R_j) \cap E} p(r_k, r_l) + \prod_{(r_k, r_l) \in (R_i \times R_j) \cap E} (1 - p(r_k, r_l))} \tag{6}$$

Let the numbers of nodes in the uncertain graph $\mathcal{G}$ be n. Then, the time complexity of SCC clustering is $O(n^2)$.

EXAMPLE 3. *We demonstrate SCC clustering with our running example in Figure 4. The algorithm identifies record pairs containing the maximum edge weight (i.e., 0.8). We initially clusters any of A, B; C, D; E, F; or G, H. Later, we continue to cluster another three record pairs as they have the same maximum edge weight. Once the four clusters $\mathbb{C}_1 = \{A, B\}, \mathbb{C}_2 = \{C, D\}$, $\mathbb{C}_3 = \{E, F\}$, and $\mathbb{C}_4 = \{G, H\}$ are identified, we verify the edge weights across these cluster. SCC merges two clusters only if the benefit of merging (Equation 6) is more than 0.5. Let us consider the merging of clusters $\mathbb{C}_1$ and $\mathbb{C}_2$. Their probability of merging is $\frac{0.3 \times 0.6}{0.3 \times 0.6 + (1-0.3) \times (1-0.6)}$ = 0.39. In fact, none of the cluster pairs qualify for merging, and SCC reports $\mathbb{C}_1, \mathbb{C}_2, \mathbb{C}_3$, and $\mathbb{C}_4$ as the four clusters.*

4.2 Putting Everything Together

we provide the entire pipeline of our PERC framework in Figure 5. Given an input set R of records, and the initial uncertain graph $\mathcal{G}$ (which might have no edges in the beginning, or only a few edges based on initial crowdsourcing), we find the most-likely clustering (MLC) $\mathbb{C}$ of R, with SCC algorithm. Next, we iteratively find the best record pair $\langle r_i, r_j \rangle$ and crowdsource it, until our budget is exhausted, or we already find a complete (uncertain) graph over R. After every crowdsourcing task, we add an uncertain edge between the respective record pair, thereby updating $\mathcal{G}$.

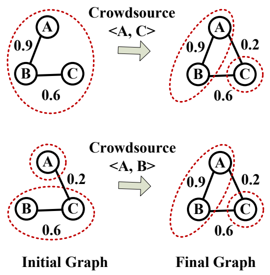


Figure 6: Most-likely clustering (MLC) changes due to addition of edges. Above: crowdsourcing result of $\langle A, C \rangle$ changes MLC from $\{(A, B, C)\}$ to $\{(A, B), (C)\}$. Below: crowdsourcing result of $\langle A, B \rangle$ changes MLC from $\{(A), (B, C)\}$ to $\{(A, B), (C)\}$.

An interesting feature of our framework is that at the end of every round, we check if the previous MLC $\mathbb{C}$ still remains the MLC for the updated graph. This can be quickly verified based on Lemma 4, that is, if the majority of the crowd workers agree with our previous clustering, there is no need to change the clustering. Otherwise, we recompute the new MLC and proceed to identify the best record pair to crowdsource for this new MLC. *Such re-clustering enables us to rectify mistakes that might have been incurred at earlier rounds due to incomplete information and crowd errors, thereby quickly converging to a high-quality solution.* We illustrate this feature of our framework with two examples in Figure 6. As one may observe, in both cases with the additional crowdsourcing evidences, the new MLC is more promising than the earlier one.

While such updates in the MLC clustering are quite effective, we empirically found that these updates happen only 20~25% of the times after next crowdsourcing. This illustrates that while updating the previous clustering is critical to improve the ER quality, it does not significantly impact the total computation time.

Dataset	# Records	# Entities	# Record-Pairs Crowdsourced	Crowd Error Rate
All Sports	267	86	35 511	5.67% (10 ques. / pair)
Gymnastics	94	12	4 371	10.65 % (5 ques. / pair)
Landmarks	529	15	30 070	4.82 % (5 ques. / pair)
Cora	949	165	29 281	27.77 % (5 ques. / pair)

Table 2: Properties of datasets

5 EXPERIMENTAL RESULTS

We present empirical results with four real-world, benchmark datasets (three image datasets and one text dataset). We evaluate entity resolution (ER) accuracy, efficiency, and crowdsourcing cost of PERC under various initial conditions, and by asking the next crowdsourcing questions one at a time and also in batches, with different crowd errors. We compare PERC with four state-of-the-art crowdsourced ER approaches: transitive closure (TC) clustering [20, 22, 25], MinMax [9], PC-Pivot [23], DENSE and bDENSE [19].

5.1 Environment Setup

The code is implemented in Python and we perform experiments on a single core of a 32GB, 2.40GHz Xeon server. All results are averaged over 10 runs. We present our results with spectral-connected-components (SCC) based correlation clustering, as it performs the best compared other correlation clustering methods [6, 19].

• **Datasets.** We use four benchmark, real-world datasets (Table 2) from the literature of crowdsourced ER.

AllSports: The *AllSports* dataset [19] consists of athlete images from different sports, with each image showing a single athlete.

Gymnastics: The *Gymnastics* dataset [19] contains athlete images, but only from gymnastics, and it is more difficult to distinguish the face of an athlete in this dataset, e.g., the athlete may be upside down on uneven bars.

Landmarks: The *Landmarks* dataset [9] has images from 9 cities. We consider a subset of the original dataset, consisting 529 images of 15 different Landmarks.

Cora: This is a text dataset containing references of scientific publications [23]. *Cora* is one of the largest datasets considered in the literature of crowdsourced ER, thus we use this dataset for demonstrating scalability.

We use Amazon's Mechanical Turk for crowdsourcing, and follow the same setting that was employed by Verroios and Molina in [19], e.g., considering answers from workers with high-accuracy statistics. We omit the details due to lack of space. In particular, for *AllSports*, we engage 10 workers for each task, whereas 5 workers are employed for each task in the other datasets [9, 19, 23]. For *AllSports* and *Gymnastics*, due to their smaller sizes, we crowdsource all record pairs. On the contrary, for *Landmarks* and *Cora* datasets, we crowdsource about 22% and 7%, respectively, of all record pairs, based on next crowdsourcing questions.

All these datasets come with the ground truth clustering results, which we refer to as the *gold standard clustering*. If a worker answered the record pair wrongly, then it is considered an error (Table 2). As an example, out of 10 workers, if 8 workers answered correct and 2 answered wrong, then the error in answering that particular records pair is 20%. Crowd error is measured as the average of all such errors over all crowdsourced record pairs.

• **Evaluation Metrics.**

Accuracy: After a certain number of answers are collected by the next crowdsourcing method, we apply ER algorithm for clustering. To measure accuracy, we compare the output to the gold standard clustering. Specifically, we employ precision (p) and recall (r), defined as follows.

$$p = \frac{\text{\#record-pairs correctly reported as matching}}{\text{\#record-pairs reported as matching}} \tag{7}$$

$$r = \frac{\text{\#record-pairs correctly reported as matching}}{\text{\#matching record-pairs in gold clustering}} \tag{8}$$

Finally, we compute F1-measure, which is defined below.

$$F1\text{-measure} = 2pr/(p + r) \tag{9}$$

Following previous works [9, 19], we use F1-measure to demonstrate the accuracy of PERC and other competitors.

Crowdsourcing Cost: The crowdsourcing cost denotes the total number of distinct record pairs being crowdsourced.

Efficiency: We report the average computation time required to select the next batch of crowdsourcing questions. Clearly, this is the runtime of the algorithm to select the next batch of questions, and

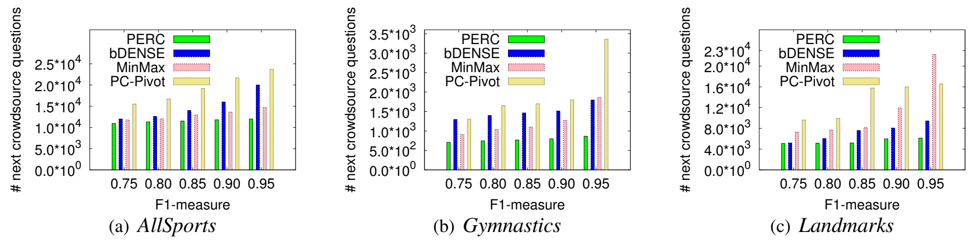


Figure 7: Cost improvement: # next crowdsourcing questions required to reach a certain accuracy (F1-measure)

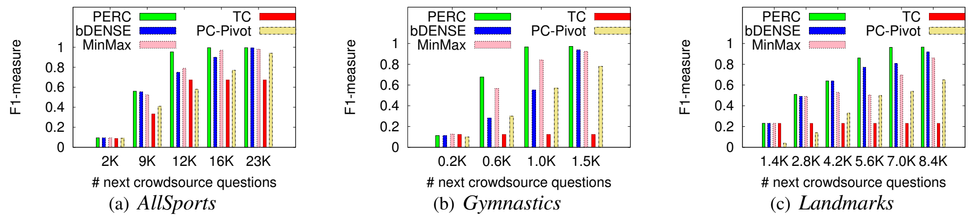


Figure 8: Accuracy improvement (F1-measure) for next crowdsourcing

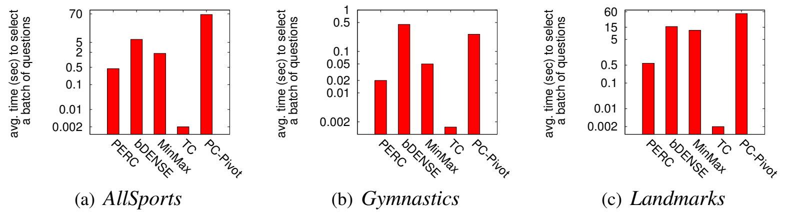


Figure 9: Efficiency improvement: Computation time required to select a batch of next crowdsourcing questions

it excludes the crowdsourcing time (which would be similar across different algorithms, for a given batch size).

• **Compared Algorithms.**

Transitive Closure (TC): This method selects, uniformly at random, one of those record pairs for which the matching/ non-matching relationship cannot be inferred (via transitivity and anti-transitivity) from the existing edges. Following [20, 25], we consider majority voting while deciding on the next crowdsourcing results. TC-clustering never reaches F1-measure ≥ 0.75 over our datasets, which is because this method does not consider conflicting evidences.

DENSE and bDENSE: DENSE [19] considers only either the set of positive edges, or the set of negative edges between two disjoint record sets for calculating the strength of evidences, denoted as the ρ-ratio (for details, see Introduction). bDENSE is a batch version of DENSE, that selects multiple questions (having higher ρ-ratios) to ask next, thereby allowing many crowd taskers to answer those questions in parallel. For ER, these methods apply SCC-clustering.

The authors in [19] considered majority voting to decide on the next crowdsourcing results. Moreover, they also assigned a fixed human accuracy of 0.9 (i.e., error rate = 0.1) on those answers.

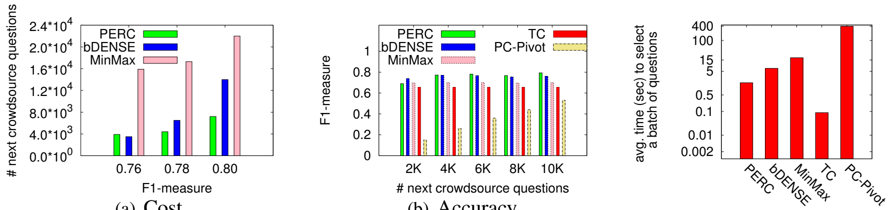
MinMax: For ER, [9] finds all positive and negative paths between a record pair. The weight of a path is determined by the smallest edge weight on that path. Finally, the algorithm selects the maximum-weight path to decide whether the records are matching or not. For next crowdsourcing, the authors proposed a hybrid strategy that prefers either a more certain matching pair, or a less certain non-matching pair. As (1) MinMax only considers the maximum-weight path (and ignores all other paths) between a record pair for both ER and next crowdsourcing, and (2) it does not consider the length of a path (intuitively, the error accumulated across a short-length path would be less than that through a longer path), the method can easily produce less effective results.

PC-Pivot: For ER, [23] uses pivot-based correlation clustering. The clustering refinement phase consists of either splitting, where nodes are removed from clusters; or merging, where two clusters are combined. The problem is that every node is considered individually, and edges connecting to that node are used to calculate the respective benefit. Hence, the method may fail to capture the strength of the entire clustering, resulting in higher crowdsourcing cost in order to achieve a reasonable ER accuracy.

5.2 Next Crowdsourcing Results

We started with different numbers of initial edges and batch sizes based on the size of our datasets. In particular, we had about 2K, 0.2K, and 1.4K initial crowdsourced edges, respectively, for *AllSports*, *Gymnastics*, and *Landmarks* datasets. We set the batch size as 320, 40, and 120 questions, respectively, over these datasets.

5.2.1 Crowdsourcing Cost Improvement. In Figure 7, we show the number of next crowdsourcing questions required to reach a certain accuracy. We consider F1-measure of 0.75 and above, because higher accuracy results are more important in real-world applications. We do not show TC-clustering, because it did not achieve an accuracy over 0.75 in all our datasets. We find that the number of crowdsourcing questions required to obtain a higher accuracy is much less — often by a margin of 50% — for PERC, in comparison to bDENSE, PC-Pivot, and MinMax. For example, to achieve F1-measure of 0.95 in the *Gymnastics* dataset, PERC, bDENSE, PC-Pivot, and MinMax require 863, 1792, 3360, and 1866 next crowdsourcing questions, respectively. These results demonstrate the effectiveness of PERC in reducing crowdsourcing cost.



(a) Cost (b) Accuracy (c) Efficiency

Figure 10: Accuracy, cost, and efficiency improvements over *Cora* dataset

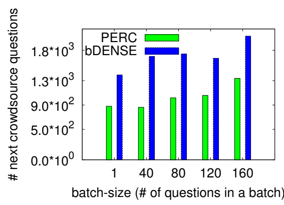


Figure 11: Varying batch-sizes: # next crowdsourcing questions required to reach F1-measure=0.95, *Gymnastics*

5.2.2 Accuracy Improvement. In Figure 8, we illustrate accuracy improvements of PERC over state-of-the-art approaches. We observed that the F1-measure of PERC increases at a higher rate and quickly reaches around 0.95 with less number of next crowdsourcing questions, compared to other methods, in all our datasets. As an example, with about 12K next crowdsourcing questions over *AllSports*, the F1-measure of PERC is 0.95, whereas for bDENSE, MinMax, PC-Pivot, and TC-clustering, the F1-measures are 0.75, 0.79, 0.58, and 0.67, respectively. These results demonstrate the accuracy improvements of PERC next crowdsourcing algorithm.

5.2.3 Efficiency Improvement. We compare the average computation time required to select a batch of next crowdsourcing questions, which is computed as follows. We first measure the computation time to select all next crowdsourcing questions in order to reach a certain accuracy, e.g., F1-measure of 0.9 for PERC, bDENSE, PC-Pivot, and MinMax. One may recall that PERC next crowdsourcing might trigger an update of the previous maximum-likelihood clustering. We empirically found that these updates happen 20~25% of the times after next crowdsourcing, and the times consumed for such re-clusterings are also added in the total time required for PERC. Since TC-clustering does not achieve such a high accuracy, we instead consider the time required to obtain the highest possible accuracy via TC-clustering. Next, we divide this time by the total number of batches issued to crowd workers, and report this value as the average computation time to select one batch of next crowdsourcing questions for the respective methods.

Figure 9 shows that the average time for one batch selection is at least an order of magnitude faster in case of PERC, compared to that of bDENSE, PC-Pivot, and MinMax. We note that the Y-axis is logarithmic in these figures. For example, with the *Landmarks* dataset, the average time to select one batch (with 120 questions) using PERC is only 0.5 sec, whereas it requires about 15 sec, 12 sec, and 51 sec, respectively, to select a batch of same size using bDENSE, MinMax, and PC-Pivot. Thus, our empirical results illustrate that PERC is at least an order of magnitude faster compared to bDENSE, MinMax, and PC-Pivot, in terms of selecting the next crowdsourcing questions.

5.2.4 Results with Cora Dataset. We present next crowdsourcing results over the larger *Cora* dataset in Figure 10. We started with 2K initial crowdsourced edges, and we set the batch size as 300 questions. Figures 10(a) and 10(b) demonstrate the cost and accuracy improvements of PERC. For example, to achieve F1-measure = 0.8, PERC requires about 7.2K questions, whereas bDENSE and MinMax require around 14K and 22K questions, respectively. The maximum F1-measure reached by PC-Pivot over *Cora* is 0.74. In Figure 10(c), we compare the average computation times required to select a batch of 300 next crowdsourcing questions over *Cora* dataset. The Y-axis is logarithmic. As earlier, PERC is 5~15 times faster than both bDENSE and MinMax, e.g., PERC requires 1.5 sec to select a batch of 300 questions, whereas bDENSE and MinMax consume 7 sec and 20 sec, respectively, for the same.

5.2.5 Varying Batch Sizes. We analyze the impact of varying batch sizes on crowdsourcing cost and accuracy (Figure 11). Smaller batch sizes help in improving the accuracy and to reduce the crowdsourcing cost. This is because we do not know the corresponding edge probabilities apriori; and hence, by issuing multiple questions in batches, the overall quality would decrease. However, asking questions in batches reduces the *overall* running time (i.e., next batch selection time + crowdsourcing time), since many crowd workers would be able to answer the questions in a batch in parallel. In Figure 11, we show the number of next crowdsourcing questions required to reach F1-measure=0.95 for PERC and bDENSE. We present our results over *Gymnastics* dataset. As expected, this crowdsourcing cost decreases with smaller batch sizes, for both these methods. We also observed that PERC outperforms bDENSE in terms of crowdsourcing cost under all batch sizes.

6 RELATED WORK

• **Crowdsourcing in Data Management.** Recently, crowdsourcing has been adopted in video and image annotations, search relevance, and natural language processing [1, 8]. Several systems have been developed to incorporate human work into a database/mobile system, e.g., CrowdDB, Deco, CrowdSearch, CDAS, CrowdForge, TurKit, and Qurk [16, 18]. There are also studies on leveraging crowd's ability to improve data management tasks, e.g., selection, sort, skyline, join, mining, classification, and max/top-k [3, 17].

• **Crowdsourced Entity Resolution (ER).** An important problem in crowdsourced ER is to reduce the number of questions asked to workers, e.g., a clustering-based method [21] where each question is a group of records and asks workers to classify the records into different clusters. Demartini et. al. [5] and Jeffrey et. al. [11] designed crowdsourcing systems based on a probabilistic framework, but does not employ transitivity to reduce the crowdsourcing cost. Wang et. al. [22] and Vesdapunt et. al. [20] utilized transitivity to reduce the number of questions. Various models to select high-quality questions were developed in [24, 25]. The most recent work [2] used a partial order approach, which additionally requires each entities having multiple attributes. More importantly, all these works assume no crowd error, or employ majority voting.

Recently, MinMax, PC-Pivot, and DENSE [9, 19, 23] directly incorporated crowd errors in ER tasks. However, as we stated earlier, these methods consider ad-hoc, local features to select next questions, such as individual paths, nodes, or the set of either positive or negative edges. Hence, they generally fail to capture the strength of the entire clustering, resulting in higher crowdsourcing cost in order to achieve a reasonable ER accuracy.

• **Dealing with Crowdsourcing Errors.** Quality control is critical in crowdsourcing [16]. Machine learning techniques have been employed to determine the quality of the crowd, e.g., [4, 13, 14]. Orthogonal to these works, our proposed solution incorporates crowd errors while performing next crowdsourcing and ER tasks.

• **Entity Resolution Algorithms.** Entity resolution (ER), also known as entity reconciliation, deduplication, or record linkage, is well studied in data cleaning and integration. Many ER algorithms have been proposed based on different input settings, e.g., single-pass clustering, star clustering, cut clustering, correlation clustering, and Markov clustering [7, 15]. We used correlation clustering because this is the most natural setting for clustering a set of records that are connected by both positive and negative edges [6]. Besides, our contribution — reliability-based next crowdsourcing question selection is orthogonal to the specific ER method employed.

7 CONCLUSIONS

We studied crowdsourced entity resolution together with erroneous crowd answers. Our solution PERC does not require any user-defined threshold values, and no apriori information about the error rate of crowd workers. We formulated the problem considering an uncertain graph model and using possible world semantics with edge independence. We employed the notion of reliability in uncertain graphs to identify the most effective next crowdsourcing questions. Based on detailed empirical results with four real-world datasets, PERC improves the accuracy by 15%, reduces the crowdsourcing cost by 50%, and also decreases the next question selection time by an order of magnitude compared to state-of-the-art approaches.

8 ACKNOWLEDGEMENT

Research was supported by MOE Tier-1 M401020000 and NTU M4081678. Any opinions, findings, and conclusions in this publication are those of the authors, and do not necessarily reflect the views of the funding agencies.

REFERENCES

[1] O. Alonso, D. E. Rose, and B. Stewart. 2008. Crowdsourcing for Relevance Evaluation. *SIGIR Forum* 42, 2 (2008), 9–15.
[2] C. Chai, G. Li, J. Li, D. Deng, and J. Feng. 2016. Cost-Effective Crowdsourced Entity Resolution: A Partial-Order Approach. In *SIGMOD*.
[3] L. Chen, D. Lee, and T. Milo. 2015. Data-driven Crowdsourcing: Management, Mining, and Applications. In *ICDE*.
[4] O. Dekel and O. Shamir. 2009. Vox Populi: Collecting High-Quality Labels from a Crowd. In *COLT*.
[5] G. Demartini, D. E. Difallah, and P. C.-Mauroux. 2012. ZenCrowd: Leveraging Probabilistic Reasoning and Crowdsourcing Techniques for Large-scale Entity Linking. In *WWW*.
[6] M. Elsner and W. Schudy. 2009. Bounding and Comparing Methods for Correlation Clustering Beyond ILP. In *ILP*.
[7] L. Getoor and A. Machanavajjhala. 2013. Entity Resolution for Big Data. In *KDD*.
[8] R. Gomes, P. Welinder, A. Krause, and P. Perona. 2011. Crowdclustering. In *NIPS*.
[9] A. Gruenheid, D. Kossmann, S. Ramesh, and F. Widmer. 2012. *Crowdsourcing Entity Resolution*. Technical Report. ETH Zurich.
[10] O. Hassanzadeh, F. Chiang, H. C. Lee, and R. J. Miller. 2009. Framework for Evaluating Clustering Algorithms in Duplicate Detection. In *VLDB*.
[11] S. R. Jeffery, M. J. Franklin, and A. Y. Halevy. 2008. Pay-as-you-go User Feedback for Dataspace Systems. In *SIGMOD*.
[12] R. Jin, L. Liu, B. Ding, and H. Wang. 2011. Distance-Constraint Reachability Computation in Uncertain Graphs. In *VLDB*.
[13] M. Joglekar, H. G.-Molina, and A. G. Parameswaran. 2013. Evaluating the Crowd with Confidence. In *KDD*.
[14] D. R. Karger, S. Oh, and D. Shah. 2011. Iterative Learning for Reliable Crowdsourcing Systems. In *NIPS*.
[15] N. Koudas, S. Sarawagi, and D. Srivastava. 2006. Record Linkage: Similarity Measures and Algorithms. In *SIGMOD*.
[16] X. Liu, M. Lu, B. C. Ooi, Y. Shen, S. Wu, and M. Zhang. 2012. CDAS: A Crowdsourcing Data Analytics System. In *VLDB*.
[17] A. Marcus and A. G. Parameswaran. 2015. Crowdsourced Data Management: Industry and Academic Perspectives. *Foundations and Trends in Databases* 6, 1-2 (2015), 1–161.
[18] A. Marcus, E. Wu, D. R. Karger, S. Madden, and R. C. Miller. 2011. Demonstration of Qurk: A Query Processor for Human Operators. In *SIGMOD*.
[19] V. Verroios and H. G.-Molina. 2015. Entity Resolution with Crowd Errors. In *ICDE*.
[20] N. Vesdapunt, K. Bellare, and N. Dalvi. 2014. Crowdsourcing Algorithms for Entity Resolution. In *VLDB*.
[21] J. Wang, T. Kraska, M. J. Franklin, and J. Feng. 2012. CrowdER: Crowdsourcing Entity Resolution. In *VLDB*.
[22] J. Wang, G. Li, T. Kraska, M. J. Franklin, and J. Feng. 2013. Leveraging Transitive Relations for Crowdsourced Joins. In *SIGMOD*.
[23] S. Wang, X. Xiao, and C.-H. Lee. 2015. Crowd-Based Deduplication: An Adaptive Approach. In *SIGMOD*.
[24] F. L. Wauthier, N. Jojic, and M. I. Jordan. 2012. Active Spectral Clustering via Iterative Uncertainty Reduction. In *KDD*.
[25] S. E. Whang, P. Lofgren, and H. G.-Molina. 2013. Question Selection for Crowd Entity Resolution. In *VLDB*.

APPENDIX

Limitation of DENSE [19] with running example. The Dense considers only either the set of positive edges (i.e., edges with majority YES votes), or the set of negative edges (i.e., edges having majority NO votes) between two disjoint record sets for calculating the strength of evidences. A metric ρ-ratio is defined, which finds the lack of strong evidences for clustering, and DENSE selects a pair to crowdsource that has the maximum ρ-ratio. In particular, ρ-ratio between sets A and B is calculated as follows.

$$\frac{P'_{Y1} \times P'_{Y2}}{P_{Y1} \times P_{Y2}} \times \min\{\frac{P'_N}{P_N}, \frac{P'_Y}{P_Y}\} \tag{10}$$

Here, $Y1$ is the set of positive edges between A and $R \setminus B$, $Y2$ the set of positive edges between B and $R \setminus B$, N the set of negative edges across A and B, and Y the set of positive edges across A and B. The set of all records are denoted by R. Let the probability for an edge $a \in \{Y1 \bigcup Y2 \bigcup Y \bigcup N\}$ being correct be $p(a)$, then we compute:

$$P_{Y1} = \prod_{a \in Y1} p(a); \quad P_{Y2} = \prod_{a \in Y2} p(a); \quad P_Y = \prod_{a \in Y} p(a);$$
$$P'_{Y1} = \prod_{a \in Y1} (1 - p(a)); \quad P'_{Y2} = \prod_{a \in Y2} (1 - p(a)); \quad P'_Y = \prod_{a \in Y} (1 - p(a));$$
$$P_N = \prod_{a \in N} p(a); \quad P'_N = \prod_{a \in N} (1 - p(a)) \tag{11}$$

Since ρ-ratios between the clusters $\langle \mathbb{C}_1, \mathbb{C}_2 \rangle$ and $\langle \mathbb{C}_3, \mathbb{C}_4 \rangle$ have the same value, which is due to the weaker negative edges, i.e. $\frac{P'_N}{P_N} = \frac{0.3}{0.7}$, DENSE assumes that asking a question across $\langle \mathbb{C}_1, \mathbb{C}_2 \rangle$ or $\langle \mathbb{C}_3, \mathbb{C}_4 \rangle$ is equivalent. However, in reality, asking a question between clusters $\mathbb{C}_3$ and $\mathbb{C}_4$ is more beneficial.