A Demonstration of PERC: Probabilistic Entity Resolution With Crowd Errors

Xiangyu Ke, Michelle Teo, Arijit Khan, Vijaya Krishna Yalavarthi
Nanyang Technological University, Singapore
{xiangyu001,C140137}@e.ntu.edu.sg, {arijit.khan,yalavarthi}@ntu.edu.sg

ABSTRACT

This paper demonstrates PERC — our system for crowdsourced entity resolution with human errors. Entity Resolution (ER) is a critical step in data cleaning and analytics. Although many machine-based methods existed for ER task, crowdsourcing is becoming increasingly important since humans can provide more insightful information for complex tasks, e.g., clustering of images and natural language processing. However, human workers still make mistakes due to lack of domain expertise or seriousness, ambiguity, or even malicious intent. To this end, we present a system, called PERC (probabilistic entity resolution with crowd errors), which adopts an uncertain graph model to address the entity resolution problem with noisy crowd answers. Using our framework, the problem of ER becomes equivalent to finding the maximum-likelihood clustering. In particular, we propose a novel metric called "reliability" to measure the quality of a clustering, which takes into account both the connected-ness inside and across all clusters. PERC then automatically selects the next question to ask the crowd that maximally increases the "reliability" of the current clustering.

This demonstration highlights (1) a reliability-based next crowdsourcing framework for crowdsourced ER, which does not require any user-defined threshold, and no apriori information about the error rate of the crowd workers, (2) it improves the ER quality by 15% and reduces the crowdsourcing cost by 50% compared to state-of-the-art methods, and (3) its GUI can interact with users to help them compare different crowdsourced ER algorithms, their intermediate ER results as they progress, and their selected next crowdsourcing questions in a user-friendly manner. Our demonstration video is at: **https://www.youtube.com/watch?v=rQ7nu3b8zXY**.

PVLDB Reference Format:
Xiangyu Ke, Michelle Teo, Arijit Khan, and Vijaya Krishna Yalavarthi. A Demonstration of PERC: Probabilistic Entity Resolution With Crowd Errors. *PVLDB*, 11 (12): 1922-1925, 2018.
DOI: https://doi.org/10.14778/3229863.3236225

1. INTRODUCTION

When dealing with real-world datasets, there exist many different representations for the same entity, which are non-trivial to identify in many cases. For example, there could be different ways of addressing the same person in text, or several photos of a particular landmark. Entity Resolution (ER) is the task of disambiguating manifestations of such real-world entities in various records by linking and clustering [5], which has attracted a great deal of attention in the database research. While various machine-based techniques were proposed for ER, it is often difficult to design a uniform solution for all kinds of datasets (e.g., images, texts, videos), and their qualities can hardly be guaranteed when faced with more complex tasks such as image classification, video tagging, optical character recognition, and natural language processing. Crowdsourcing is thus a more general approach with human's insightful information for these difficult problems [6]. The existing crowdsourcing services, e.g., Amazon's Mechanical Turk (AMT) and CrowdFlower allow individuals and commercial organizations to set up tasks that humans can perform for certain rewards [10]. When the size of the dataset is large, it would be too expensive to ask the crowd about every pair of records, therefore bulk of the literature aims at minimizing the cost of crowdsourcing, while also maximizing the ER result quality.

However, human workers can still make mistakes due to lack of domain expertise, individual biases, task complexity and ambiguity, or simply because of tiredness, and malicious behaviors. 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% [15, 17]. Many works bypass this issue as an orthogonal problem to crowdsourced ER, because there are various approaches to compute and reduce crowdsourcing biases and errors, including [9, 8, 2]. Other works elude this severe concern by majority voting [16, 13, 14], or by assigning a universal error rate for crowd workers [12]. Unfortunately, the majority voting approach is often unreliable because spammers and low-paid workers may collude to produce incorrect answers [9], and it is unrealistic to determine the universal error rate precisely since the worker crowd is large-scale and highly transient. Recent works by Verroios et. al. [12], Gruenheid et. al. [7], and Wang et. al. [15] are the most similar to ours, since they assign edge probability (or weight) between a record pair as the ratio of crowd workers who voted Yes on the question if those two records are the same entity. However, these methods consider only local features, such as individual paths, nodes, or the set of positive and negative edges, to decide next crowdsourcing questions. Without capturing the strength of the entire clustering, their crowdsourcing cost becomes too high for achieving a reasonable ER accuracy.

In practise, given a current clustering, it is critical to identify *the best record pair to crowdsource next*. We propose a novel metric, called the "reliability", to measure the quality of a clustering, taking into account the connected-ness inside and across the clustering. Our method, PERC, selects the record pair to crowdsource next that

Proceedings of the VLDB Endowment, Vol. 11, No. 12
Copyright 2018 VLDB Endowment 2150-8097/18/8.
DOI: https://doi.org/10.14778/3229863.3236225

Table 1: Crowdsourcing cost reduction by PERC over state-of-the-art approaches: We present the number of crowdsourcing questions required to achieve a certain accuracy (0.8 F1-measure for *Cora* dataset, and 0.9 for others) for various methods. A detailed description about our datasets, accuracy measure and experiment setting is shown in our full paper [17].

datasets	*# crowdsourced questions*			*% crowdsourcing cost reduction by* PERC *over*	
	MinMax [7]	bDense [12]	PERC [this work]	MinMax	bDense
Allsports	13.6K	16.0K	**11.7K**	13.97%	26.87%
Gymnastics	1.3K	1.5K	**0.8K**	38.46%	46.67%
Landmarks	11.0K	8.0K	**5.9K**	46.36%	26.25%
Cora	22.5K	14.0K	**7.2K**	68.00%	48.57%

maximally increases the reliability of the current clustering. This significantly reduces the overall crowdsourcing cost — by 50%, as well as improves its quality by 15%, which is demonstrated in our empirical results (Table 1 and [17]).

To the best of our knowledge, this is the first demonstration proposal on *crowdsourced entity resolution with human errors,* aiming to present an uncertain graph-based solution, which is efficient, effective, and user-friendly in practice. Other recent demos in this area include NADEEF/ER [3], KATARA [1], and CrowdCleaner [11]. In particular, NADEEF/ER is a rule-based and interactive ER system without crowdsourcing, whereas both KATARA and CrowdCleaner are crowdsourced ER systems without explicitly considering crowd errors. Hence, these past demonstrations are different from ours.

2. FRAMEWORK OVERVIEW

Figure 1 presents PERC's architecture and the interaction between modules. The core modules of PERC — Entity Resolution (ER) and Next Crowdsourcing (NC) modules — are implemented in C++. The Displayer module, which provides the visualization of dynamic and interactive clustering results and next question selecting results, is developed with D3.js library (http://d3js.org/). We introduce the four modules of the PERC system in the following.

Interactor. The *Interactor* receives input datasets of various types (e.g., images, text, video), and pre-processes them to identify the records set R. It selects, uniformly at random, a small number record pairs. The *Record Pairs-HITs Transformer* component then publishes these pairs as HITs (Human Intelligent Tasks) on AMT platform. The *Interpreter* generates or updates the uncertain graph once the crowd answers are obtained. In particular, 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 submits 'Yes' if she thinks that the record pair is matching, and 'No' otherwise. 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 $p(r_i, r_j) \in (0, 1)$ denotes the ratio of crowd workers who voted Yes on the question if r_i and r_j are same entity. We denote by E the set of edges as formed above.

Entity Resolution Processor. On receiving an uncertain graph $\mathcal{G} = (R, E, p)$ as formed above, our *Entity Resolution processor* applies a greedy correlation clustering algorithm (e.g., the spectral-connected-components method [12]) to generate the maximum likelihood clustering (MLC) of the uncertain graph. After every crowdsourcing task, we add an uncertain edge between the respective record pair, thereby updating $\mathcal{G}$. Then, the *MLC Update Detector* verifies if the previous MLC needs to be updated or not.

Next Crowdsourcing Questions Selector. After the initial MLC generation, the *Next Crowdsourcing Questions Selector* iteratively finds the best record pair $\langle r_i, r_j \rangle$ and crowdsources it, until our budget is exhausted, or we already find a complete (uncertain) graph

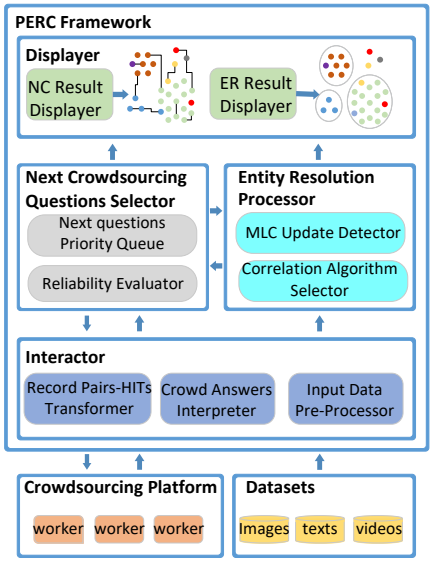

Figure 1: The architecture of PERC system

over the records set R. If MLC remains the same in successive iterations, the priority queue consisting of next crowdsourcing questions remains the same, and the question to ask in the current round is easy to decide by simply selecting the best (e.g., top-most) pair from the queue. Otherwise, the *Reliability Evaluator* re-computes the priority queue based on the metric called "reliability" of the new MLC. We shall formally introduce this novel criteria in Section 3.2.

Displayer. Our system also contains a user-friendly *Displayer* module for interactive visualization. The *ER Result Displayer* shows the dynamic changes of the graph and the clustering result based on the crowd answers obtained so far, and the *NC Result Displayer* presents the next questions to be published in the crowdsourcing platform. Users are allowed to rollback for visualizing every batch of questions asked previously and their corresponding ER result. If the ground-truth answer is provided, PERC will color the record nodes and presents it together with the real-time ER result.

3. CORE ALGORITHMS

Two primary modules of PERC rely on each other, as illustrated in Figure 2, and run their own algorithms.

3.1 Entity Resolution Processor

The task of *ER Processor* is to find the maximum likelihood (and transitively-closed) clustering for a given uncertain graph $\mathcal{G} = (R, E, p)$. The likelihood of a clustering $\mathbb{C} = \{R_1, R_2, \ldots, R_m\}$ is defined 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 (i.e., each HIT can be performed by a different set of workers), we compute the likelihood $L(\mathbb{C})$ as:

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

Therefore, the entity resolution problem is equivalent to finding the maximum likelihood clustering (MLC) over the uncertain graph constructed by the input records set. It can be performed by employing state-of-the-art *correlation clustering* algorithms [12,

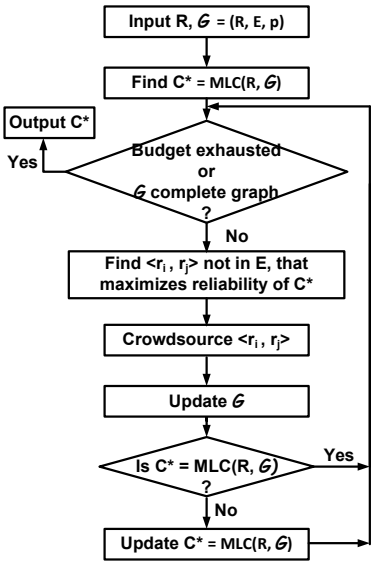

Figure 2: Workflow of PERC

4]. The users can select among the following algorithms: *Best*, *First*, *Pivot* and *Spectral-Connected-Components* (SCC). Usually we suggest SCC, since it performs the best in our empirical evaluation, compared to other approaches. It 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 and returns the current set of clusters as output. Given two clusters R_i and R_j, SCC computes the probability $Pr(R_i, R_j)$ of merging them as given below.

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

3.2 Next Crowdsourcing Processor

Once the MLC is constructed, the *Next Crowdsource processor* iteratively finds the the best entity pair $\langle r_i, r_j\rangle$ to crowdsource next.

3.2.1 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 MLC as much as possible. To this end, we identify the two following "connectedness"-based criteria that determine the quality of a clustering $\mathbb{C} = \{R_1, R_2, ..., R_m\}$. **(1)** How well each cluster R_i is connected? **(2)** 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 is an YES edge, we define its existence probability $p_Y(e) = p(e)$. On the other hand, if e is a NO edge, we compute its existence probability as $p_N(e) = 1 - p(e)$. We derive an YES-NO graph $\mathcal{G}_{Y|N}$ 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 $p_{Y|N}(e) = p_N(e)$.

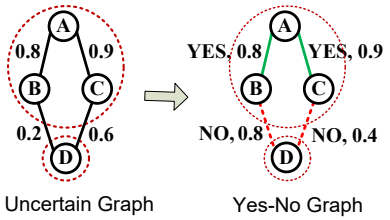

Figure 3: Reliability of a clustering

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.,

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

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

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

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.

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

3.2.2 Next Crowdsourcing Question Selection

The *Next Crowdsourcing Question Selector* computes, for every record pair $\langle r_i, r_j\rangle \notin E$, the improvement in reliability of the current MLC $\mathbb{C}^*$, if one crowdsources the pair, and thereby assigns the corresponding edge probability $p(r_i, r_j)$. Unfortunately, one does not know $p(r_i, r_j)$ apriori, therefore we consider an optimistic scenario. We study this in [17], and show that $Rel(\mathbb{C})$ *increases monotonically as we have larger value of* $p_{Y|N}(e)$, *where* e *is the newly added edge. Thus,* $p_{Y|N}(e) = 1$ *leads to the optimal gain.* In other words, we select the record pair that maximally increases $Rel(\mathbb{C}^*)$, under such optimistic assumption. Also, we found that *the priority of 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 in a specific round*, which helps us avoid redundant computation for pairs across the clusters. Finally, we develop efficient algorithms [17] based on the Monte-Carlo sampling, and omit the details here due to lack of space.

4. DEMOSTRATION

4.1 User Interface and Interactiveness

For demonstration, we will present a web app on a laptop (demonstration video: **https://www.youtube.com/watch?v=rQ7nu3b8zXY**). Figure 4 presents the user interface of PERC. The top panel is designed for user to provide the input dataset, ground-truth answer file (which is optional, and used for the visualization of accuracy), correlation clustering algorithm (default as SCC), and the next crowdsourcing algorithm (default as PERC).

The monitor window in the middle shows the real-time uncertain graph generated by crowd answers (gray edges). Users can tick at the top right boxes to view the intermediate ER clustering results and the next questions selected (red edges). Each node here represents a record in the input dataset. Those nodes being grouped

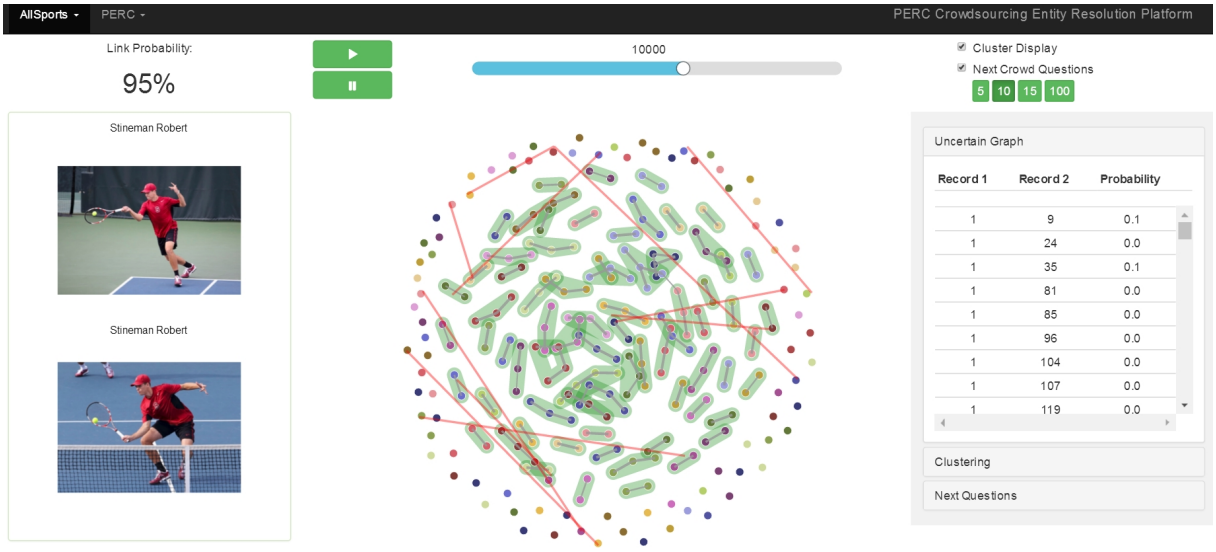

Figure 4: User interface of PERC

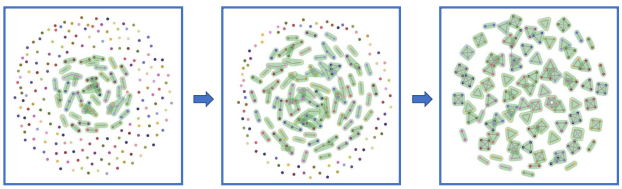

Figure 5: PERC demonstration scenario with AllSports dataset

together are considered as the same entity by PERC. The red edges are the batch of questions to be asked next, while the grey ones represent the questions that were already asked (for clarity, we only draw the edges with probability over 0.5 here. The higher the probability on the edge, the darker and the thicker it will be). If the ground-truth is given, the nodes will be colored (each color stands for an entity based on the ground-truth), thereby providing a better visualization of accuracy. When the cursor is hovered for a second, the details for the corresponding item will be displayed in the left panel. For nodes, it would be the detailed content, e.g. an image. For edges, it would be the two records that it connects and the corresponding probability of being the same (as shown in Figure 4). Users are allowed to rollback or forward the process bar at the top of the monitor to dynamically view the crowdsourcing process in an interactive manner and to terminate the process at any moment. Finally, the users can also employ our system to visualize the results of other existing crowdsourced ER techniques, e.g., MinMax [7] and bDense [12], for an effective comparison.

From the right panel, user can view the ER results, together with all the crowd answers, and the next crowdsourcing questions. Also, users may download them in files for further analysis.

4.2 Demonstration with AllSports Dataset

The *AllSports* dataset (`stanford.edu/~verroios/datasets/allsports.zip`) [12] consists of athlete images from different sports, with each image showing a single athlete. It contains 267 records, which can be divided into 86 distinct entities. We crowdsourced all the 35 511 record pairs, and each question is answered by 10 different human workers. For this dataset, the average crowd error rate is 5.67%.

The sub-figures in Figure 5 demonstrate an intuitive ER and Next Crowdsourcing evaluation procedure. From Figures 5(a) to 5(c), they correspond to 5K, 10K, and 15K record pairs, respectively, being crowdsourced. With more evidence added, more clusters are found. Figure 5(c) corresponds to the convergence state [17] in our algorithm. It can be observed here that many triangles and rectangles have been formed (in the ground-truth of *AllSports* dataset, most clusters have sizes of 2, 3, or 4).

PERC is a general framework for various kinds of data. Its performance on text data can be found in our full paper [17].

5. ACKNOWLEDGEMENT

Research was supported by NTU M4081678 and MOE Tier-1 RG83/16. Any opinions, findings, and conclusions in this publication are those of the authors, and do not necessarily reflect the views of the funding agencies.

6. REFERENCES

[1] X. Chu, J. Morcos, I. F. Ilyas, M. Ouzzani, P. Papotti, N. Tang, and Y. Ye. KATARA: Reliable Data Cleaning with Knowledge Bases and Crowdsourcing. *PVLDB*, 8(12):1952–1955, 2015.

[2] O. Dekel and O. Shamir. Vox Populi: Collecting High-Quality Labels from a Crowd. In *COLT*, 2009.

[3] A. Elmagarmid, I. F. Ilyas, M. Ouzzani, J.-A. Quiané-Ruiz, N. Tang, and S. Yin. NADEEF/ER: Generic and Interactive Entity Resolution. In *SIGMOD*, 2014.

[4] M. Elsner and W. Schudy. Bounding and Comparing Methods for Correlation Clustering Beyond ILP. In *ILP*, 2009.

[5] L. Getoor and A. Machanavajjhala. Entity Resolution for Big Data. In *KDD*, 2013.

[6] R. Gomes, P. Welinder, A. Krause, and P. Perona. Crowdclustering. In *NIPS*, 2011.

[7] A. Gruenheid, D. Kossmann, S. Ramesh, and F. Widmer. Crowdsourcing Entity Resolution. Technical report, Systems Group, Computer Sc., ETH Zurich, 2012.

[8] D. R. Karger, S. Oh, and D. Shah. Iterative Learning for Reliable Crowdsourcing Systems. In *NIPS*, 2011.

[9] X. Liu, M. Lu, B. C. Ooi, Y. Shen, S. Wu, and M. Zhang. CDAS: A Crowdsourcing Data Analytics System. *PVLDB*, 5(10):1040–1051, 2012.

[10] A. Marcus and A. G. Parameswaran. Crowdsourced Data Management: Industry and Academic Perspectives. *Foundations and Trends in Databases*, 6(1-2):1–161, 2015.

[11] Y. Tong, C. C. Cao, C. J. Zhang, Y. Li, and L. Chen. CrowdCleaner: Data Cleaning for Multi-Version Data on the Web via Crowdsourcing. In *ICDE*, 2014.

[12] V. Verroios and H. G.-Molina. Entity Resolution with Crowd Errors. In *ICDE*, 2015.

[13] N. Vesdapunt, K. Bellare, and N. Dalvi. Crowdsourcing Algorithms for Entity Resolution. *PVLDB*, 7(12):1071–1082, 2014.

[14] J. Wang, G. Li, T. Kraska, M. J. Franklin, and J. Feng. Leveraging Transitive Relations for Crowdsourced Joins. In *SIGMOD*, 2013.

[15] S. Wang, X. Xiao, and C.-H. Lee. Crowd-Based Deduplication: An Adaptive Approach. In *SIGMOD*, 2015.

[16] S. E. Whang, P. Lofgren, and H. G.-Molina. Question selection for crowd entity resolution. *PVLDB*, 6(6):349–360, 2013.

[17] V. K. Yalavarthi, X. Ke, and A. Khan. Select Your Questions Wisely: For Entity Resolution with Crowd Errors. In *CIKM*, 2017.