Received June 11, 2019, accepted June 20, 2019, date of publication June 26, 2019, date of current version July 24, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2924979

Quality Management of Workers in an In-House Crowdsourcing-Based Framework for Deduplication of Organizations' Databases

MORTEZA SABERI[1], OMAR KHADEER HUSSAINZ[2], AND ELIZABETH CHANG[2]
[1]School of Information, Systems and Modelling, University of Technology Sydney, Sydney, NSW, Australia
[2]School of Business, University of New South Wales, Canberra, ACT, Australia

Corresponding author: Morteza Saberi (morteza.saberi@uts.edu.au)

This work was supported by the UNSW Ph.D. Scholarship.

ABSTRACT While organizations in the current era of big data are generating massive volumes of data, they also need to ensure that its quality is maintained for it to be useful in decision-making purposes. The problem of dirty data plagues every organization. One aspect of dirty data is the presence of duplicate data records that negatively impact the organization's operations in many ways. Many existing approaches attempt to address this problem by using traditional data cleansing methods. In this paper, we address this problem by using an in-house crowdsourcing-based framework, namely, DedupCrowd. One of the main obstacles of crowdsourcing-based approaches is to monitor the performance of the crowd, by which the integrity of the whole process is maintained. In this paper, a statistical quality control-based technique is proposed to regulate the performance of the crowd. We apply our proposed framework in the context of a contact center, where the Customer Service Representatives are used as the crowd to assist in the process of deduplicating detection. By using comprehensive working examples, we show how the different modules of the DedupCrowd work not only to monitor the performance of the crowd but also to assist in duplicate detection.

INDEX TERMS Quality management, quality control, data quality, duplicate detection, in-house crowdsourcing.

I. INTRODUCTION

In today's era of Big Data, organizations constantly rely on evidence-based decision-making in their operations. Such a decision making approach aims to process the available underlying data to synthesize evidence that either ascertains or justifies the decision to be taken [1]. Apart from having available data, other key prerequisites for such a decision model to work are to have the underlying data of the right type and quality. Our focus in this paper is on improving the quality aspect of data. This is a serious issue for organizations as experts spend more than 50% of their time in finding errors and cleaning it [2]. Other figures suggest that about 47% of newly-created data records suffer from at least one error that results in non-value-add costs, inaccurate decisions, unhappy customers and the like. To address this issue, a lot of work has been undertaken in the area of data cleansing [3], [4]. Data cleansing, also known as data scrubbing or data munging is an important but tedious process of first ascertaining which data entries violate the integrity of the database and then taking action to rectify them to ensure the quality of the data is well maintained. In the era of big data, as is with the huge increase in the size of data, the many ways by which the integrity of the database can be violated too are many. Some examples include missing values, misspellings, mis-fielded values, duplicate records and so forth [5]. Our focus in this paper is on the presence of duplicate records in the database. Having such data negatively impacts an organization's performance in many ways such as lack of a single customer view, negative impact on a company's brand, poor customer service, inaccurate reporting and so forth [6]. Therefore, addressing it is very important.

True to its importance, addressing the presence of duplicate records in databases has been given appropriate attention in the literature. Similarity techniques based on characters, tokens, phonetic and numeric similarities are used to match fields of data based on probabilistic, supervised or semi-supervised based approaches to ascertain if a particular data record is a duplicate or not. These techniques, while effective, have start-up costs associated with them to initialize

The associate editor coordinating the review of this manuscript and approving it for publication was Ashish Mahajan.

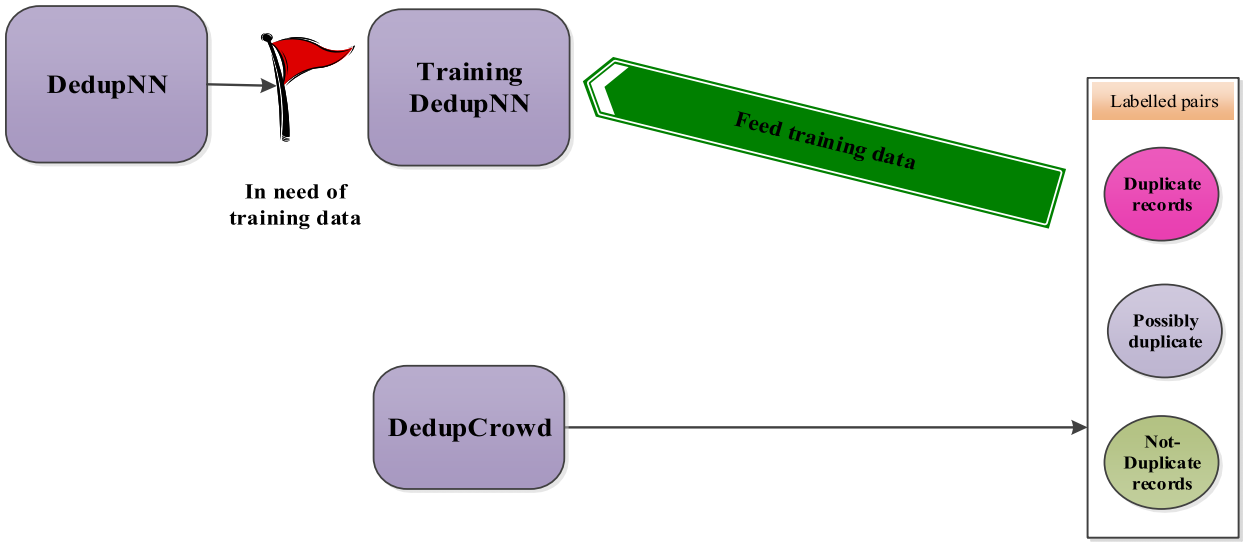

FIGURE 1. Working of *DedupCrowd* and how it provides training data for automated duplication detection.

with various look up tables based on which the duplicate detection process will be carried out [7]. While on the one hand, it can be argued that such required costs are usual to have the appropriate learning dataset based on which the process of duplicate records detection will be semi-automated; on the other hand in today's competitive times it also raises organizations' issues related to privacy and security. Data is now considered a crucial business asset and the start-up process, in most cases, requires a knowledgeable data analyst who is external to the organization. In scenarios where the organization deals with sensitive data this leads to privacy and security issues with a preference to keep this process in-house and secure [8]. Recent advances in human computing, namely crowdsourcing, has opened ways to address the problem of deduplication and also keep this process in-house. Human computing dates back to 1950 when Turing stated that digital computers aim to accomplish tasks which could be done by a human [9]. Crowdsourcing is one of well-known fields of human computing that is currently thriving which was proposed by Jeff Howe and Mark Robinson [10]. Crowdsourcing integrates the power of humans and machines to solve issues that are generally hard and/or laborious for a machine to address by itself [11], [12]. Crowdsourcing has been utilized in accomplishing tasks such as sentiment analysis, image processing [10], [13]–[16]. It has recently been used in the database community too for duplicate detection which show it is capable of improving the duplicate detection process [11], [16]–[23].

In this paper, we propose a human computing-based crowdsourcing approach termed DedupCrowd to assist in duplicate detection in a Contact Centre's (CC's) database. CCs are the new version of a call centre, offering multiple channels through which the customer may communicate with the organization including telephone, email and live online chat. While on the one hand, having access to such diverse communication channels offers customers the flexibility to communicate with the organization, it also increases the entry points from where inconsistent or incorrect data may enter the CC's database [25], [26]. A Customer Service Representative (CSR) is the organization's personnel responsible for answering the customer's queries. Having dirty data poses a significant challenge to CSRs in performing their task of effectively answering customers' queries and thus should be addressed [27], [28]. DedupCrowd utilizes a human computing-based approach by having CSRs as the crowd to assist in duplicate data detection. Shown in Figure 1, the output of DedupCrowd labels pairs of customer data as either duplicate, possibly duplicate or not duplicate. This knowledge not only provides an output to the process of deduplication by an in-house expert but also collects data that is used to train an automated model of deduplication (termed as DedupNN). This is important as in the real-world manual deduplication of a database is impractical and an automated version of it is needed based on the data that has been annotated by DedupCrowd using the process of crowdsourcing.

One of the main obstacles in a human based computing approach is the poor performance of the crowd (CSRs or workers) and how it can be addressed [16]. The researchers in the area of machine learning [17], and statistics and databases [18] contributed significantly in developing techniques to approximate the error of workers. However a practical framework for the online evaluation and eviction of poor workers from the crowdsourcing process is lacking [29]. DedupCrowd addresses this shortage and makes use of the workers' estimated errors through a statistical quality control (SQC) approach to evict poorly performing workers from the crowdsourcing process. Statistical quality control (SQC) has been used successfully in the area of manufacturing process quality control [30]. The output of this module provides the status of the crowdsourcing workers, which determines whether a given worker can continue to participate as part of the crowd or not.

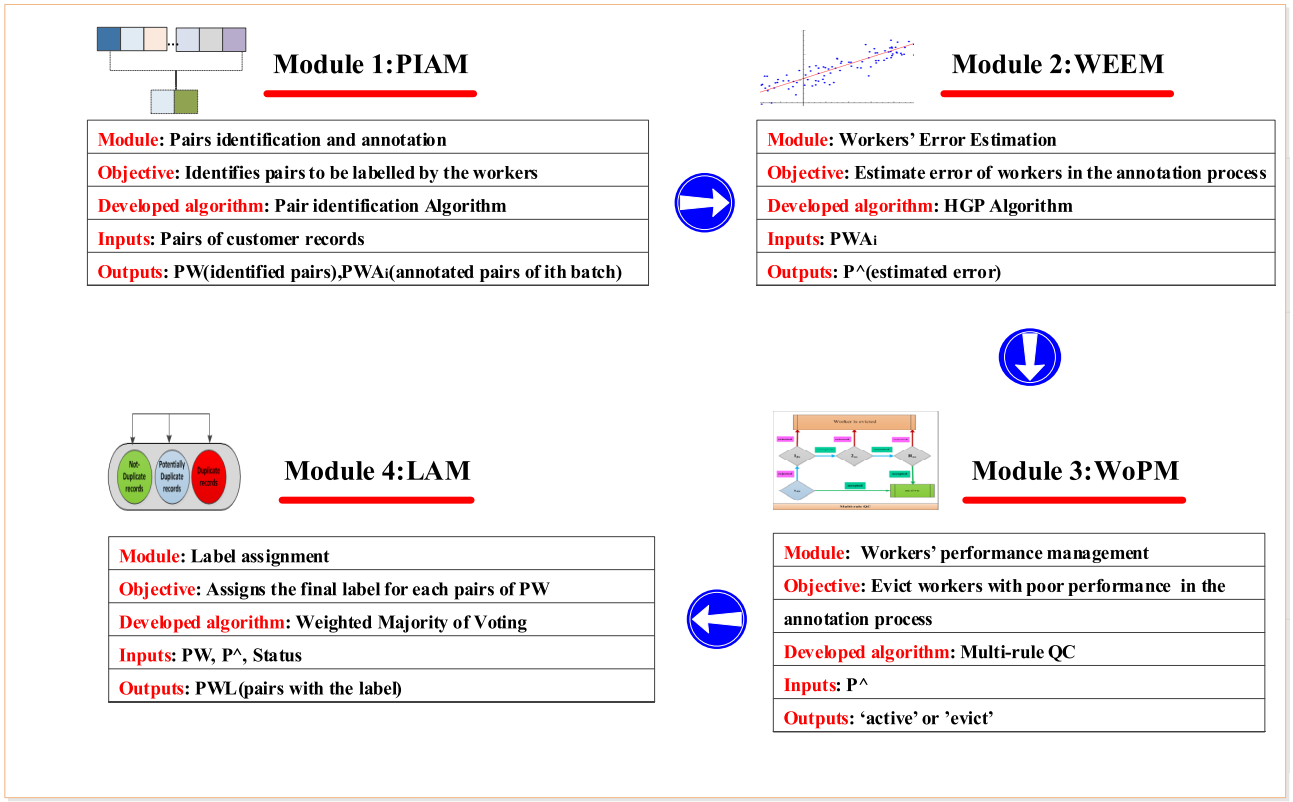

FIGURE 2. ***DedupCrowd*** **modules.**

The rest of the paper is organized as follows. Section 2 introduces the preliminaries and notation used in this paper. Section 3 presents the DedupCrowd framework with a description of its four integrated modules, namely, Pair Identification and Annotation, Workers' Error Estimation, Workers' Performance Management and Label Assignment modules. These four integrated modules are explained in detail in Sections 4 to 7. In Section 8, we show working examples how these four modules work together to assist CSRs in the deduplication of the CC's DB. Section 9 presents related work and Section 10 presents the conclusion and future work.

II. PRELIMINARIES AND NOTATION

The following terms from existing crowdsourcing literature are used in this paper:

Workers: CSRs who participate in the crowdsourcing-based platform for deduplication. $w_i$denotes the i^{th} worker of the given crowdsourcing platform. s_w denotes the set of workers who are active in crowdsourcing process.

Human Intelligence Task (HIT): A task that should be completed by workers, as they are too difficult or expensive for computers to execute by comparison. One important type of HIT is the multiple-choice HIT, the output of which is categorized into k categories. In our approach, we consider that HITs have three possible options to choose from. In other words, the CSR should annotate a pair of customer profiles as duplicate, possibly duplicate or not duplicate.

Work: A collection of HITs that are posted to the crowd for annotation.

Batch of HITs: A subset of tasks used by our statistical algorithm to estimate the error of workers individually.

Next, we define new terms that are used in our approach for crowdsourcing-based data deduplication.

Gold HITs: Set of HITs whose answers are known in advance [31].

Current error set: A set of performance error values computed at the completion of each batch. ce_{ij} denotes these values after completion of batch b_{jk} : $ce_{ij} = \left\{p_{ix}^{\curlywedge} | 1 \leq x \leq j\right\}$.

Past error: The arithmetic average of a worker wi's error values over a past period. It is represented by $past_{ik}$ and is computed as $past_{ik} = \frac{\sum_{m \in ind_{ik}} \overline{p}_{im}}{|ind_{ik}|}$ where $\overline{p}_{im} = \frac{\sum_{j=1}^{S_m} p_{ijm}^{\curlywedge}}{s_m}$.

In the next section, we discuss our proposed DedupCrowd framework for crowdsourcing-based deduplication of CC's databases.

III. DEDUPCROWD

As previously discussed, DedupCrowd is the proposed approach for deduplication by the crowd to provide high quality training data for automated deduplication and for measuring the workers' ability to successfully undertake the deduplication process. As shown in Figure 2, the proposed framework for DedupCrowd relies on four integrated modules, Pair Identification and Annotation, Workers' Error Estimation, Workers' Performance Management and Label Assignment.

- Pair Identification and Annotation Module (PIAM): identifies the HITs for posting to the workers for annotation as either being duplicate, possibly duplicate or not duplicate. The reason we do not send all the pairs of records is that the percentage of duplicate profiles is usually not high. Also the majority of plausible pairs are not very similar, hence they are clearly not duplicates [32]. The procedure followed in this module is explained in Section 3.1.
- Workers' Error Estimation Module (WEEM): Based on the annotated output of workers from PIAM, this module estimates the performance error of workers in annotating the given HITs. The error is estimated on a scale of 0 to 1 with the value 0 representing no error and 1 representing otherwise in annotating all HITs of the given batch. The Hybrid Gold-Plurality (HGP) algorithm estimates worker error. The WEEM's output will be used for workers' performance monitoring.
- Workers' Performance Management Module (WoPM): This module monitors the workers' performance by considering their errors and determines a status for them. It also evicts workers who perform poorly. A multi-rule QC system is used that takes both the current and past errors of a worker and ascertains the worker's status either as active or evict at the end of each batch. Active means that the worker can continue to make annotations in the next batch, whereas evict indicates the worker is evicted due to their poor performance in the crowdsourcing process. A detailed description is given in Section 6.
- Label Assignment Module (LAM): This module assigns the final label for each HIT identified in PIAM based on the Weighted Majority of Voting (Weighted MV) method. At the end of this process, the final label for the HITs is recorded as duplicate, possibly duplicate or not duplicate. The details of this module are given in Section 7.

The four integrated modules of DedupCrowd are represented in Figure 2. This figure shows the objective, the developed algorithm, and the inputs and outputs of each of these four modules. The sequence of the activities undertaken in these four integrated modules is explained in the next sub-section.

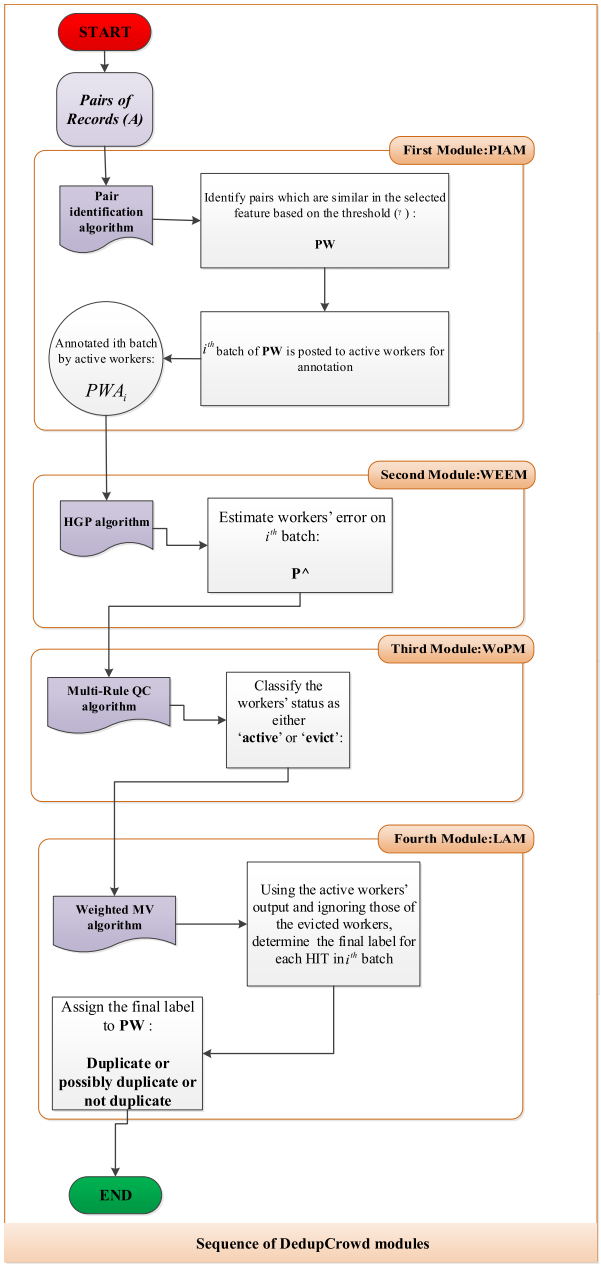

FIGURE 3. Flow of control between the different modules in *DedupCrowd*.

A. SEQUENCE OF ACTIVITIES AMONG DIFFERENT DEDUPCROWD MODULES

DedupCrowd has four integrated modules by which it obtains a label by a CSR for record (A). Workers who are involved in DedupCrowd annotate batches of HITs that are provided to them by PIAM. Figure 3 shows the sequence of activities in DedupCrowd for the ith batch and is explained below by categorizing each module.

- Pair Identification and Annotation Module (PIAM): DedupCrowd starts with pairs of customer records (A) that are given to PIAM. From these pairs (A), PIAM selects some pairs (PW) for the workers to annotate. PW is also partitioned into batches of HITs to be annotated by the workers. The annotation of workers in the ith batch is denoted by PWAi. Thus, PW and PWAi are two outputs of the first module, which are used by the next modules.
- Workers' Error Estimation Module (WEEM): At the end of each batch, the performance errors of the participating workers are determined using WEEM. WEEM receives PWAi and using the proposed Hybrid Gold Plurality (HGP) algorithm estimates the error of the participating workers. Thus, the output of WEEM is the estimated error of the participating workers in the ith batch (p^).
- Workers' Performance Management Module (WoPM): At the end of the annotation of each batch, the performance evaluation of each worker is done by considering

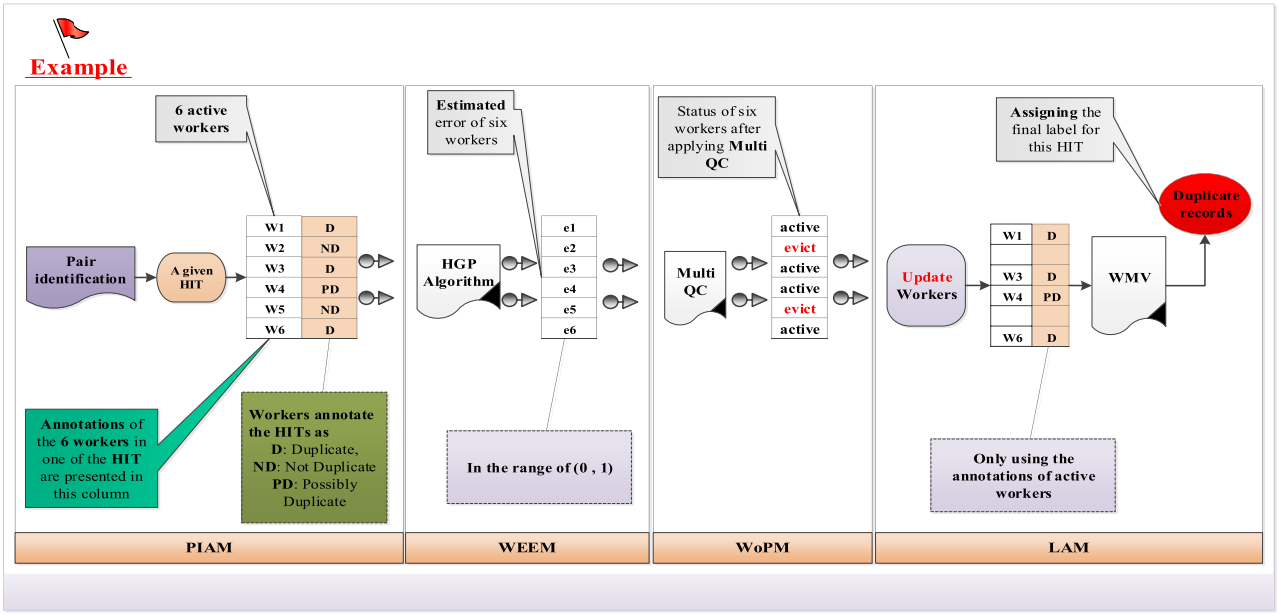

FIGURE 4. The process of assigning the final label of a given HIT in *DedupCrowd*.

their performance in the current batch, their error in previous batches and their error history. A multi-rule QC is used to make a decision as to whether the participating workers in this batch can continue in the next batch (i.e. their status is active) or not (i.e. their status is evict).

- Label Assignment Module (LAM): In this module, the final label for each HIT selected in PIAM in the ith batch is determined using the Weighted MV algorithm. It should be noted that Weighted MV only considers the annotation (votes) of workers whose status is active to identify the final label of the HITs.

Figure 4 illustrates the working sequence of DedupCrowd through an example. PIAM module has two stages: identifying pairs of records that should be given to the workers and annotating them using workers. Figure 4 shows how a given HIT is annotated by six workers as either D, PD or ND. In WEEM, the estimated performance error (error) of each worker for a batch of HITs is determined as e1-e6 in the range of 0 to 1. WoPM determines the status of each worker as either active or evict. In Figure 4, the second and the fifth workers are evicted due to their poor performance and using Weighted MV, LAM assigns the duplicate records label for this HIT.

Figure 4 illustrates the working sequence of DedupCrowd through an example. PIAM module has two stages: identifying pairs of records that should be given to the workers and annotating them using workers. Figure 4 shows how a given HIT is annotated by six workers as either D, PD or ND. In WEEM, the estimated performance error (error) of each worker for a batch of HITs is determined as e1-e6 in the range of 0 to 1. WoPM determines the status of each worker as either active or evict. In Figure 4, the second and the fifth workers are evicted due to their poor performance and using Weighted MV, LAM assigns the duplicate records label for this HIT.

TABLE 1. DedupCrowd nomenclature.

A	Customer Data
K	Size of batch
PW	Identified pairs for workers' annotation
PWA	Annotated PW by workers
$\mathbf{P}_{\text{-Gold}}$	Performance of workers in Gold HITs
$p_{ij}^{\wedge}$	Estimated error of i^{th} worker at j^{th} batch
$cm_{ij}^{\wedge}$	Control measurement value of i^{th} worker at j^{th} batch
PWL	Final label of PW
sw	Set of workers that are active in the current work
past	Workers' past error
θ	Dissimilar percent
γ	Identification threshold
status	Status of workers

B. DEDUPCROWD NOMENCLATURE

In this section, the nomenclature of the important variables used in DedupCrowd and its algorithms are presented in Table 1.

IV. MODULE 1: PAIR IDENTIFICATION AND ANNOTATION MODULE (PIAM)

As previously mentioned, PIAM identifies the pairs to be posted to the workers for deduplication and records their annotations. It is not necessary to post all the possible pairs to workers if they are not similar. Thus, the number of selected pairs is less than $\frac{n(n-1)}{2}$ pairs by considering n as the number of tuples in the CC's DB. This module filters dissimilar pairs and posts similar pairs to workers. To find dissimilar pairs, four metrics are proposed which are explained in the next sub-section.

A. PROPOSED METRICS IN PIAM

To find the pairs to be posted to the workers, metrics are proposed that are used to find similar records. It is important to use suitable metrics to calculate similarities between the features in the records as data base's features have different types of information. simdate, simlevenshtein, simnum and simphone are the four similarity functions that customized to this end are shown in equation 1, 2, 3 and 4, respectively.

$$sim_{date}(a,b) \leftarrow \left\{ \begin{array}{cc} 1-\left(\frac{|a-b|}{d_{max}}\right) & |a-b| < d_{max}, \\ 0 & else \end{array} \right\} \quad (1)$$

$$sim_{levenshtein}(a,b) \leftarrow 1-\left(\frac{distance_{levenshtein}(a,b)}{\max(|a|,|b|)}\right) \quad (2)$$

$$sim_{phone}(a,b) \leftarrow 1-\frac{\sum_i dis\,(a_i,b_i)}{|a|} \quad that\ \mathrm{dis}\,(a_i,b_i) = \left\{ \begin{array}{cc} 0.5; & |a_i-b_i|=1 \\ 0; & a_i=b_i \\ 1; & else \end{array} \right\} \quad (3)$$

The similarity between two customers is calculated by using equation (4):

$$\mathbf{Similarity}(\mathbf{A},\mathbf{B}) \leftarrow \text{average} \times \left[\begin{pmatrix} sim_{name}\,(A_{name},B_{name}), sim_{address}\,(A_{add},B_{add}), \\ sim_{date}\,(A_{dob},B_{dob}), sim_{phone}\,(A_{phone},B_{phone}) \end{pmatrix} \right] \quad (4)$$

Such that:

$$sim_{name}\,(\mathbf{A_{name}},\mathbf{B_{name}}) \leftarrow \left(sim_{levenshtein}\left(A_{firstname},B_{firstname}\right), sim_{levenshtein}\,(A_{lastname},B_{lastname})\right)$$

$$sim_{address}\,(\mathbf{A_{add}},\mathbf{B_{add}}) \leftarrow (sim_{levenshtein}\,(A_{street},B_{street}), sim_{num}\,(A_{streetnumber},B_{streetnumber}), sim_{levenshtein}\,(A_{suburb},B_{suburb}))$$

$$\mathbf{A}_{add} \leftarrow (A_{street},A_{streetnumber},A_{suburb})$$

Value costomer (A)

$$\leftarrow \left(A_{firstname},B_{firstname}A_{street},\ A_{streetnumber},\ A_{suburb},A_{dob},A_{phone}\right)$$

Using these equations, similar records are identified to be forwarded to the workers for their annotations by constructing an array PW as discussed next.

B. PIAM PSEUDO-CODE

Algorithm 1 demonstrates how PIAM constructs PW and forwards it to the workers in batches of HITs. After calculating the similarity between two pairs using equation (5), pairs whose similarity is higher than the threshold (γ) are identified for further checking by workers (lines 3-7). Since the workers' output is used for training the automated ANN deduplication models, a proportion (θ) of pairs with a low similarity is selected for annotation through crowdsourcing (lines 8 & 9). These two types of pairs form the array PW. These pairs are also used as a part of the gold HITs to test the performance of workers during the annotation process in combination with another method, which is explained later. Algorithm 2 is invoked from line 1 of Algorithm 1 and identifies the pairs (PW) to be annotated by the workers. Algorithm 3 is invoked from line 4 of Algorithm 1 and it records the annotated output of workers on the identified pairs (PW), building array PWA.

Algorithm 1 PIA(A, k,γ, θ, *sw*)

Input: Customer data: $\boldsymbol{A}$; dissimilar percent: $\boldsymbol{\theta}$; identification threshold: $\boldsymbol{\gamma}$; size of batch: **k;**
Output: Pairs for crowd labelling: $\boldsymbol{PW}$;

1 $PW \leftarrow$ **Pair Identification (A, k,$\boldsymbol{\gamma}$, $\boldsymbol{\theta}$, *sw*)**
2 $s \leftarrow \frac{|PW|}{k}$
3 for j = 1:s
4 $PWA_j \leftarrow$ **Annotation(PW, k, sw, j)**
5 End
6 **Return PW& PWA;**

Algorithm 2 Pair Identification (A, k,γ, θ, *sw*)

Input: Customer data: $\boldsymbol{A}$; dissimilar percent: $\boldsymbol{\theta}$; identification threshold: $\boldsymbol{\gamma}$;
Output: Pairs for crowd annotation: $\boldsymbol{PW}$;

1 $B \leftarrow Pairfunction(A)$
2 $r, x \leftarrow 0$
3 ***for*** $i = 1 : \binom{|A|}{2}$
4 $s \leftarrow Similarity(b_{i1}, b_{i2})$
5 ***if*** $s > \gamma r = r+1; PW(r) \leftarrow B_i$
Else $x = x+1; PWN(r) \leftarrow B_i$
6 ***end***
7 ***End***
8 $m \leftarrow round(\theta * |PWN|)$
9 $T \leftarrow random(m, 1, |PWN|)$
10 $PW \leftarrow PW + PWN(T)$
11 **Return PW;**

V. MODULE 2: WORKERS' ERROR ESTIMATION MODULE (WEEM)

This module determines the performance error of workers in the deduplication process. Two main crowd-control mechanisms are integrated to this end: the gold-standard performance method and plurality answer agreement in determining the error of workers (P^). By combing these two strategies a Hybrid Gold-Plurality (HGP) method is presented.

- Gold-standard performance method: The true performance of workers is measured by posting Gold HITs to them. It should be noted that these Gold HITs are

Algorithm 3 Annotation(PW, k, sw, j)

Input: Pairs for crowd labelling: ***PW***; **Status**;
function that returns workers' (sw) annotation: ***Worker-scrowd***;
function that returns error of active workers in completing Gold HIT: ***Gold test***;
Output: workers' annotation for j[th] batch: $\boldsymbol{PWA_j}$;

1 $s \leftarrow \frac{|PW|}{k}$
2 $num \leftarrow 0$
3 **If** j<s
4 $PW_j \leftarrow PW(1+(i-1)*k, k+i*k)$
5 **else**
6 $PW_s \leftarrow PW(1+(s-1)*k, |PW|)$
Endif
7 **For** j = 1:|sw|
8 ***if*** $(status_j == 1)$
9 num = num+1;
10 $PWA_j(:, num) \leftarrow$ Workerscrowd$(PW_i, w_{sw(j)})$
11 **End if**
12 $P_{-Gold} \leftarrow Goldtest(sw)$
13 **End**
14 **Return** $\mathbf{P_{-Gold}}$, $\mathbf{PWA_j}$;

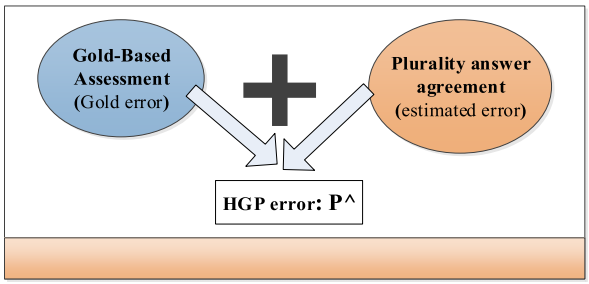

FIGURE 5. Two main components of the HGP algorithm.

different from the HITs (PW) determined by PIAM from which the estimated error of the workers will be determined. It should be noted that the true labels of Gold HITs are available for the requester which allow him to compare the true labels with the worker's results to find their Gold error. Algorithm 3 (Line 12) gives the steps in finding this error, which is denoted by P-Gold.

- Plurality answer agreement: The performance of workers is estimated by comparing their labels with those of other workers on PW. In our approach, the algorithm by Joglekar *et al.* is used to determine the plurality answer agreement mechanism [33].

In the proposed Hybrid Gold Plurality (HGP) algorithm, Joglekar *et al.* [33] algorithm's output (which gives the plurality answer agreement) is combined with the gold-standard mechanism's error. HGP algorithm's two parts are illustrated in Figure 5. To demonstrate how the HGP algorithm works, the next sub-section gives an explanation based on three active workers before presenting Algorithm 4 to show the different steps of the HGP algorithm for the error estimation process when there are more than two workers involved in the annotation process.

Algorithm 4 HGP($\mathrm{PWA_j}$,$\mathbf{P_{-Gold}}$,sw)

Input:
$\mathrm{w_i}$ error rate derived by Gold test: $\boldsymbol{p_{i-Gold}}$;
function that produces all plausible binary partitions of sw_k: **Partition;**
function that deletes zero and complex numbers from its array input: **Delete**;
function that calculates agreement rate between w_i and S/T: **Agreement.rate** ;
Output: $p_{ij}^{\wedge}$

1 Sc, Tc $\leftarrow$ partition($\mathrm{sw_k}-\{\mathrm{i}\}$)
2 k $\leftarrow$ size(partition($\mathrm{sw_k}-\{\mathrm{i}\}$))
3 **for r = 1 : k**
4 S $\leftarrow$ Sc(r), T$\leftarrow$ Tc(r)
5 $\mathrm{p_{S\_Gold}} \leftarrow$ Average $\left(\mathrm{p_{i\_Gold}} \mid \mathrm{i} \in \mathrm{S}\right)$
6 $\mathrm{a_{iS\_Gold}} \leftarrow \mathrm{p_{i\_Gold}} \times \mathrm{p_{S\_Gold}} + (1-\mathrm{p_{i\_Gold}}) \times (1-\mathrm{p_{S\_Gold}})$
7 $[a_{iS}, a_{iT}] \leftarrow$ agreement.rate(PWA_x, S, T)
8 $\mathrm{a_{iS\_Com}} \leftarrow$ average($\mathrm{a_{iS\_Gold}}$, $\mathrm{a_{iS}}$)
9

$$PE\,(r) \leftarrow \frac{1}{2} - \sqrt{\frac{(a_{iS\_Com}-\frac{1}{2})(a_{iT\_Com}-\frac{1}{2})}{2(a_{ST\_Com}-\frac{1}{2})}}$$

10 **end for**
11 $Pe \leftarrow$ **Delete**(**PE**)
12 $p_{ij}^{\wedge} \leftarrow$ ***Average***(PE)
13 **Return** p_{ij}

A. HGP ALGORITHM'S WORKING EXAMPLE

The working of the HGP algorithm is explained by assuming that three workers are involved with the annotation process, namely w1, w2, and w3. The workers error is estimated by using Equation 6 which has been derived based on Bernoulli's distribution and gold estimation [18]. Equation 7 determine hybrid gold plurality error.

$$p_1^{\wedge} \leftarrow \frac{1}{2} - \sqrt{\frac{\left(a_{12com}-\frac{1}{2}\right)\left(a_{13com}-\frac{1}{2}\right)}{2\left(a_{23com}-\frac{1}{2}\right)}} \tag{5}$$

$$a_{ij\_Gold} \leftarrow p_{i_{Gold}} \times p_{j\_Gold} + \left(1-p_{i_{Gold}}\right) \times \left(1-p_{j_{Gold}}\right) \tag{6}$$

$$a_{ij_{com}} \leftarrow average\left(a_{ij_{Gold}}, a_{ij}\right) \tag{7}$$

where
p_{i-Gold} is w_i error determined in the gold test
a_{ij} is the level of agreement between w_i & w_j
a_{ijcom} is the average of a_{ij} & a_{ij_Gold}
$p_1^{\wedge}$ is the error estimated for i[th] *worker*

a_{ij} is calculated by *Equation* 8.

$$a_{ij} \leftarrow \frac{agreement_{ij}}{m} \tag{8}$$

where $agreement_{ij}$ is the number of times w_i and w_j perform HITs in the same way and m is the size of the batch.

Numerical example: In this example, w1, w2, and w3 label 20 HITs for the b1, first batch. The agreement rates and the confusion matrix.[1] between these three workers have been reported in Table 2 and Table 3 respectively. Also, the Gold error of these workers are depicted in Table 4.

TABLE 2. Agreement rate between workers.

	W_1	W_2	W_3
W_1	-	15	13
W_2	15	-	16
W_3	13	16	-

TABLE 3. Confusion matrix.

$a_{ij\_Confusion}$	$a_{12\_Confusion}$	$a_{13\_Confusion}$	$a_{23\_Confusion}$
Value	$\frac{15}{20} = 0.75$	$\frac{13}{20} = 0.65$	$\frac{16}{20} = 0.8$
a_{ij_Gold}	a_{12_Gold}	a_{13_Gold}	a_{23_Gold}
Value	0.64	0.62	0.71

TABLE 4. Gold test error estimation.

	W_1	W_2	W_3
pGold	0.3	0.15	0.2

Using Table 3, Joglekar *et al.*'s algorithm results are as follows:

$$p_1^{\wedge} = \frac{1}{2} - \sqrt{\frac{\left(a_{12\_confusion} - \frac{1}{2}\right)\left(a_{13\_confusion} - \frac{1}{2}\right)}{2\left(a_{23\_Confusion} - \frac{1}{2}\right)}}$$

$$p_1^{\wedge} = \frac{1}{2} - \sqrt{\frac{\left(\frac{15}{20} - \frac{1}{2}\right)\left(\frac{13}{20} - \frac{1}{2}\right)}{2\left(\frac{16}{20} - \frac{1}{2}\right)}} \frac{1}{2} - \frac{1}{4} = \frac{1}{4} = 0.25$$

Using Table 3 and Table 4, HGP algorithm measures the errors of these three workers as follow:

$$p_1^{\wedge} = \frac{1}{2} - \sqrt{\frac{\left(a_{12-com} - \frac{1}{2}\right)\left(a_{13-com} - \frac{1}{2}\right)}{2\left(a_{23\_com} - \frac{1}{2}\right)}}$$

$$p_1^{\wedge} = \frac{1}{2} - \sqrt{\frac{(0.695 - 0.5)(0.635 - 0.5)}{2(0.755 - 0.5)}} = 0.272$$

Table 5 compares the determined error using the HGP and Joglekar et al.'s algorithm for the three workers. It should be noted that $p_1^{\wedge}$ denotes the estimated error of w1 in the first batch. Table 5 shows that the average of the estimated error using the HGP algorithm is higher than that given by Joglekar *et al.*'s algorithm. The reason for this higher estimation is that the error of w1 is determined by combining the results of Joglekar *et al.*'s algorithm and Gold HITs. Thus, it leads to a higher value of error when the HGP algorithm is used. However, the error determined by the HGP algorithm is closer to that being represented by the Gold HITs in comparison with the error that is estimated using Joglekar *et al.*'s algorithm. Thus, the HGP algorithm gives a more realistic measure of the error in comparison with Joglekar *et al.*'s algorithm.

TABLE 5. Comparison of Joglekar *et al.* and HGP algorithms.

	Joglekar et al. Algorithm	HGP Algorithm
$p_1^{\wedge}$	0.25	0.272
$p_2^{\wedge}$	0	0.07
$p_3^{\wedge}$	0.2	0.202

In this running example, the error estimation process is explained by considering three workers. In real situations, the number of workers may be higher than three. Thus, in the following sub-section, the full version of the HGP algorithm is presented which shows how it estimates workers' error when the number of active workers is more than three.

B. HGP ALGORITHM PSEUDO CODE

The HGP Algorithm procedure has been explained simply by assuming three workers are active in annotating process. When the number of workers is more than three the strategy is same with equation 6 by dividing peer workers into two disjoint peer groups, S and T. We use exhaustive strategy to select these two disjoint groups. Algorithm 4 shows how the HGP is working.

Note: S(n,k) is calculated using the following formula:

$$S(n, k) = \frac{1}{k!}\sum_{j=0}^{k}(-1)^{k-j}\binom{k}{j}j^n \tag{9}$$

VI. MODULE 3: WORKERS' PERFORMANCE MANAGEMENT MODULE (WoPM)

This module evicts workers with poor performance by monitors their performance. The workers errors, both current and past, are considered to this end by using a multi-rule QC system [35]. Multi-rule QC systems have been successfully applied in the area of statistical quality control and in DedupCrowd, we customize it in the area of crowdsourcing to monitor workers' performance. The four rules are proposed and utilized in conjunction with each other to build such a multi-rule QC system which monitors the performance of the workers. The multi-rule QC system considers the performance of workers by a control measurement (cm) value. The cm is a value determined for each worker by aggregating his or her previous and current errors in the annotation process (explained in Section 6.1). At the end of each batch, the cm

[1]This is a well-known matrix which is used to describe classification models' performance [34] Eduardo P. Costa, Ana C. Lorena, Andre C. P. L. F. Carvalho, and Alex A. Freitas, "A review of performance evaluation measures for hierarchical classifiers," in *Evaluation Methods for Machine Learning II: papers from the AAAI-2007 Workshop*, Vancouver, 2007, pp. 1-6

Algorithm 5 Control measurement ($p_{ij}^{\curlywedge}$, $p_{i.past}$)

Input: w_i past error: $\boldsymbol{p_{i.past}}$;
w_i current error: $\boldsymbol{p_{ij}^{\curlywedge}}$;

1 **Output:** $cm_{ij}^{\curlywedge}$
2 $w_{past} \leftarrow \frac{\frac{1}{e^j}+\frac{1}{j+1}}{2}$
3 **for i = 1 : j**
4 $w_{ij} \leftarrow \frac{1-w_{past}}{j}$
5 $cm_{ij}^{\curlywedge} \leftarrow \left[\left(w_{past} * past_i\right) + \sum_{t=1}^{j} w_{it} * p_{it}^{\curlywedge} \right]$
6 **end for**
7 **Return** $cm_{ij}^{\curlywedge}$

value is used by the multi-rule QC to determine the status of each worker. The value one is assigned to the status of workers to mark them as active workers. The following subsections explain the construction of the cm value.

A. CALCULATING THE CONTROL MEASUREMENT VALUE TO REPRESENT THE AGGREGATED PERFORMANCE OF WORKERS

The moving average technique is used to construct the cm variable to determine the aggregated performance of wi. As shown in Line 4 of algorithm 5, the cm variable is determined as the weighted average of two parameters; current and past errors. The current and past errors are determined by the HGP algorithm at different times. The current error of a worker is the set of errors in the different batches of the current work. For example, in batch b3, the current error of wi is the following set: $\{p_{i1}^{\curlywedge}, p_{i2}^{\curlywedge}, p_{i3}^{\curlywedge}\}$ Past error is the error of the worker in past work. Different weights are associated with each error value of each type in the calculation of cm. Algorithm 5 determines the control measurement value of the wi in bj. The effect (weight) of the worker's past error history is decreased in the algorithm, compared to their current history, to ensure better monitoring of their current performance of wi.

B. MULTI-RULE QC SYSTEM FOR WORKERS' PERFORMANCE MONITORING

The Multi-rule QC is customized in this paper to monitor the performance of workers. Using the multi-rule system instead of using a single rule enables the false worker eviction error to decrease. Also, instead of using in-control and out-of-control, active and evict labels are used respectively which fits better in the crowdsourcing domain. Figure 6 depicts the four proposed rules and have been explained as follows:

- Rule 1αs: A worker is rejected when the value of its submitted batch of HITs cmij is beyond the interval, which is of length of α sigma and centre of cmij. Sigma is the standard deviation of cm values and past errors, as shown in equation 10.
- Rule 1 βs: A worker is rejected when the cmij of its submitted batch of HITs is beyond the interval with a length of β sigma and centre of cmij. ($\beta > \alpha$)

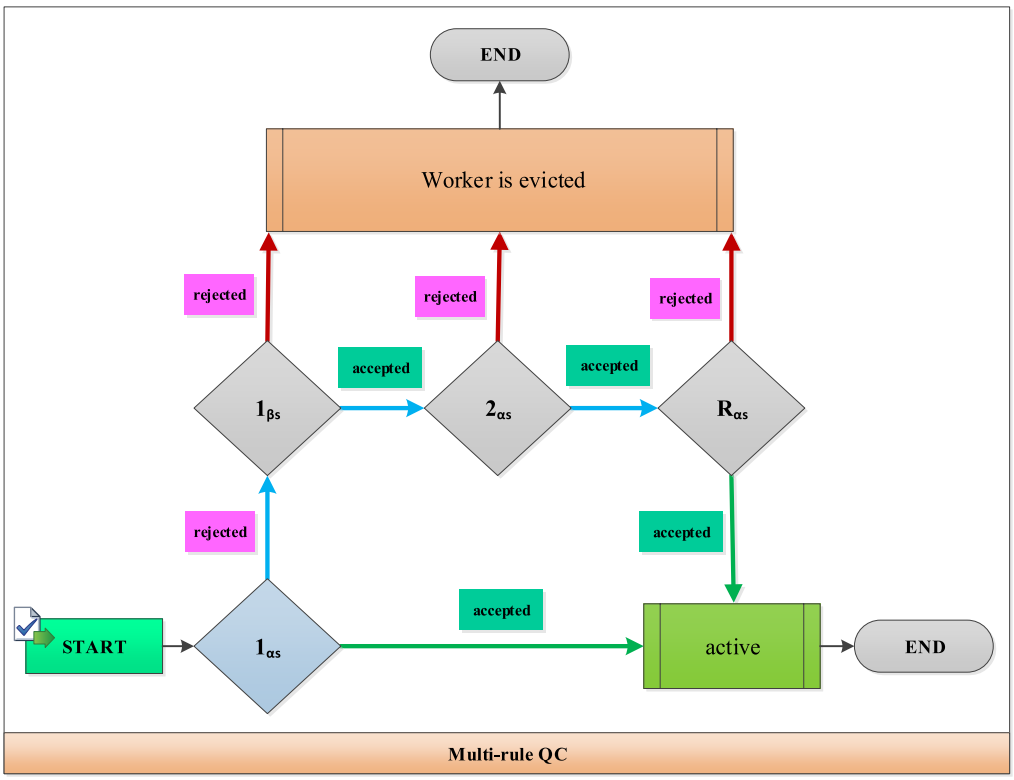

FIGURE 6. Multi-rule QC flowchart.

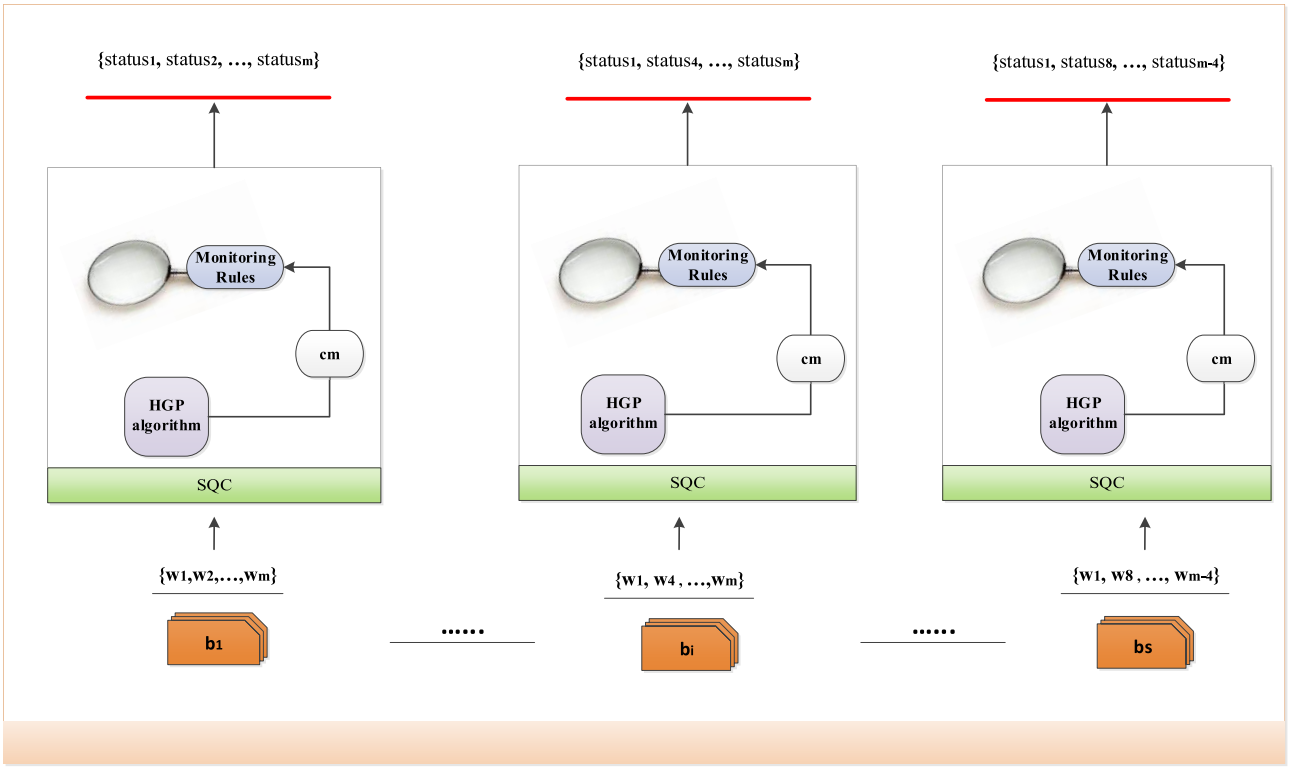

FIGURE 7. Online performance management using WoPM.

- Rule 2αs: A worker is rejected when cmij-1 and cmij of the HITs falls outside the interval with a length of α sigma and centre of cmij.
- Rule Rαs: A worker is rejected when the cmij-1 and cmij of the HITs has a distance more than α sigma and centre of cmij.

Figure 6 shows how the status of a worker is determined as either active or evict using the Multi-Rule QC system. The status of a worker is considered active when Rule 1αs is accepted. When this rule is rejected, the other three rules are tested and they must be accepted to allow the worker to continue in the DedupCrowd system. The rejection of any rule in the sequence Rule 1βs to Rule Rαs results in the eviction of the worker from the DedupCrowd system.

C. MULTI-RULE QC PSEUDO-CODE

As previously mentioned, the Multi-rule QC system assists the WoPM in determining whether a worker (wi) can continue on the platform or whether they should be evicted due to their poor performance. In order to determine the status of wi, the cm value of wi for bj (cmij) and standard deviation (σ) is fed into the multi-rule QC algorithm as shown in Figure 7. The multi-rule QC determines the status of wi by initiating its rules. The process of decision making by Rule 1αs, Rule 1βs, Rule 2αs and Rule Rαs is depicted in lines 1 to 4 of Algorithm 6, respectively. Should Rule 1αs be accepted, the value one is assigned to the status of wi. If Rule1αs is violated, worker (wi) is not evicted and Rule1βs, Rule2αs and

Algorithm 6 Multi-Rule QC ($p_{ij}^{\curlywedge}$, $cm_{ij}^{\curlywedge}$, $\sigma_{ij}^{\curlywedge}$)

Input: w_i *Control measurement* value at j^{th} batch: $\boldsymbol{cm_{ij}^{\curlywedge}}$; $w_i$standard deviation at j^{th} batch: $\boldsymbol{\sigma_{ij}^{\curlywedge}}$; w_i current error: $\boldsymbol{p_{ij}^{\curlywedge}}$;
Output: $status_{ij}$

1 **if** $p_{ij}^{\curlywedge} \in (cm_{ij}^{\curlywedge} - \frac{\alpha}{2}\sigma_{ij}, cm_{ij} + \frac{\alpha}{2}\sigma_{ij})$ **then** $status_{ij} \leftarrow 1$
2 **else if** $p_{ij}^{\curlywedge} \notin (cm_{ij}^{\curlywedge} - \frac{\beta}{2}\sigma_{ij}, cm_{ij} + \frac{\beta}{2}\sigma_{ij})$ **then** $status_{ij} \leftarrow 0$
3 **else if** $p_{i(j-1)}^{\curlywedge} \notin (cm_{ij}^{\curlywedge} - \frac{\alpha}{2}\sigma_{ij}, cm_{ij}^{\curlywedge} + \frac{\alpha}{2}\sigma_{ij})$ **then** $status_{ij} \leftarrow 0$
4 **else if** $\left|p_{ij}^{\curlywedge} - p_{i(j-1)}^{\curlywedge}\right| > \alpha\sigma_{ij}$ **then** $status_{ij} \leftarrow 0$
5 **else** $status_{ij} \leftarrow 1$
6 **end if**
7 **return** $status_{ij}$

Rule Rαs are run consecutively. If any of these three rules are violated, the worker is evicted and the value zero is assigned to the status of wi. The value one is assigned to wi's status when all these three rules are accepted. The pseudo-code of the Multi-rule QC is given in Algorithm 6.

D. RE-ACTIVATING WORKERS IN THE CASE OF TOO MANY WORKER SUSPENSIONS

It is possible that the multi-rule QC system evicts many workers. However, when the number of workers is lower than three, the HGP algorithm is no longer able to estimate the

error of the remaining workers. Also, when too many workers are evicted, the concept of crowdsourcing is no longer valid. To deal with this issue, a mechanism is proposed which allows DedupCrowd to continue its work by reactivating the status of some workers. In other words, if the percent of workers with an active status falls under the determined threshold (υ, i.e. 30%), some inactive workers are reactivated to ensure the percent of active members remains above the defined threshold, υ Lines 5-10 of Algorithm 7 depict this procedure. stat0 variable is the sorted array of status based on the workers' estimated error from minimum to maximum when their status value is zero. Algorithm 7 re-activates those workers whose errors were fewer than all the evicted workers.

Algorithm 7 Re-Activating (status)

Input: w_i *Control measurement* value at j^{th} batch: $\boldsymbol{cm_{ij}^{\wedge}}$;
w_i standard deviation at j^{th} batch: $\boldsymbol{\sigma_{ij}}$;
Output: $status_{ij}$

1 **if** $status_{ij} = 0$ $past_i \leftarrow \frac{sump_i + \sum_{t=1}^{j} \frac{1}{j} * p_{it}^{\wedge}}{2}$
$sump_i \leftarrow sump_i + \sum_{t=1}^{j} \frac{1}{j} * p_{it}^{\wedge}$
2 **else if** $j = s_k$ $past_i \leftarrow \frac{sump_i + \sum_{t=1}^{s_k} \frac{1}{s_k} * p_{it}^{\wedge}}{2}$
$sump_i \leftarrow sump_i + \sum_{t=1}^{s_k} \frac{1}{s_k} * p_{it}^{\wedge}$
3 **end if**
4 **end for**
5 **if** $(\frac{sum(status_j)}{|sw|} < \upsilon)$ *then* $(\alpha \leftarrow \upsilon - \frac{sum(status_j)}{|sw|})$
6 $z \leftarrow integer(\alpha * |sw|)$
7 **for** $k = 1: z$
8 $i \leftarrow stat0(k)$
9 $status_{ij} \leftarrow 1$
10 **end for**
11 **end if**
12 **return** $status_{\cdot} \propto past_{\cdot}$;

E. WoPM PSEUDO-CODE

Algorithm 8 demonstrates how WoPM monitors the performance of wi and determines whether this worker can continue in the crowdsourcing system (status ←1) or should be evicted (status ←0) due to their poor performance. First, WoPM uses Algorithm 5 to calculate the value of the cm (Line 2). This value gives the basis for WoPM to determine whether a worker (wi) can continue in the current work or not (Line 4). Line 5 of Algorithm 8 shows how some evicted workers are reactivated when the number of active workers in the crowdsourcing system drops below the defined threshold (υ). Figure 7 illustrates how the performance of workers is continually monitored during the annotation process from batch b1 to bs using the HGP algorithm and WoPM. It also enables the reader to gain a better understanding of the monitoring process which combines the WEEM and WoPM modules, resulting in a customized SQC in the context of crowdsourcing. The output of this combination is the status of each worker at the end of each batch. For example, batch b1 starts

Algorithm 8 WoPM ($P^{\wedge}$, *past*, *sw*)

Input: w_i control measurement: $\boldsymbol{cm_{ij}^{\wedge}}$;
w_i past error: $\mathbf{past_i}$;
w_i error rate derived by Gold test: $\boldsymbol{p_{i-Gold}}$;
w_i error estimated by HGP algorithm: $\boldsymbol{p_{ij}^{\wedge}}$;
Output: w_i updated past error: $\mathbf{past_i}$;
w_i status: $\boldsymbol{status_{ij}}$;
Start
1 **for** $i \in sw$
2 $cm_{ij} \leftarrow$ **Control measurement**$(past_i, p_{ij}^{\wedge})$
3 $\sigma_{ij} \leftarrow 3 \times \sqrt{\frac{cm_{ij}^{\wedge} \times (1 - cm_{ij}^{\wedge})}{n}}$
4 $status_{ij} \leftarrow$ **Algorithm 3**$(p_{ijk}^{\wedge}, cm_{ij}^{\wedge}, \sigma_{ij})$ /Multi-rule QC
5 $status, past \leftarrow$ **Re-Activating (stat)**
6 **end**
7 **return** $status_{\cdot} \propto past_{\cdot}$;

with workers 1 to m, but in batch bi, workers 2 and 3 (and maybe others) are evicted from the annotation process due to their poor performance in previous batches which is reported by the HGP algorithm and confirmed by the Multi-rule QC. This process continues until batch bs thereby ensuring that workers with an acceptable level of performance quality are used in the crowdsourcing system.

VII. MODULE 4: LABEL ASSIGNMENT MODULE (LAM)

In this module, each pair's final label considered in PIAM is identified by utilizing the weighted majority of voting (Weighted MV) method. Weighted MV is an aggregation approach which selects the most frequent label by considering the weighted voting as its input [37] and assigns greater weight to labels from active workers whose estimated error in the annotation process is much lower than other active workers. These weights are shown in lines 5-7 of Algorithm 9, which shows how PWL_j is calculated using the weighted MV. Line 3 of Algorithm 9 shows how the votes (annotations) of active workers are considered in this stage. The votes for the three prospective labels are counted in lines 4-8 of Algorithm 9 and the final label is assigned through lines 10-13 of the algorithm. In the next sub-section, the working of LAM is explained using an example.

VIII. WORKING OF DEDUPCROWD WITH EXAMPLES

In this section, we show working examples of how DedupCrowd and its four integrated modules incorporate the crowdsourcing approach to assist the CSR in the process of deduplication.

A. PIAM NUMERICAL EXAMPLE

To demonstrate the working of PIAM, we assume that the size of set A is 100. Thus, the number of plausible pairs is $\binom{100}{2}$ which is equal to 4950. We also assume that based on the calculation of Algorithm 5.2, the size of PW is 950 in

Algorithm 9 Weighted MV Algorithm (PWA_j, $P^{\wedge}$, $cm^{\wedge}$)

Input: set of identified pairs: ***PW***; set of active workers: **sw**,
W workers annotation at j^{th} batch: $\boldsymbol{PWA_j}$;
Output: PW's labels: $\mathbf{PWL_j}$

1 **START**
2 $s \leftarrow |\mathbf{PWA}_j|$
3 $\mathbf{sw} \leftarrow \{\mathbf{w}_i \wr \text{status}_i == 1\}$
4 **for** i = 1: s
5 $n(i) \leftarrow (\sum_{z \in sw} \left(1 - p^{\wedge}_{sw_{ij}}\right) | \mathbf{PWA_j}\ (sw_z, i) == 0)$
6 $pd(i) \leftarrow (\sum_{z \in sw} \left(1 - p^{\wedge}_{sw_{ij}}\right) | \mathbf{PWA_j}\ (sw_z, i) == 0.5)$
7 $d(i) \leftarrow (\sum_{z \in sw} \left(1 - p^{\wedge}_{sw_{ij}}\right) | \mathbf{PWA_j}\ (sw_z, i) == 1)$
8 **end**
9 **for** i = 1: s
10 $y \leftarrow maxarg(n\,(i)\,, pd\,(i)\,, d\,(i))$
11 **If** y == 1 $PWL_j(i) \leftarrow 0$
12 elseif y == 2 $PWL_j(i) \leftarrow 0.5$
13 else $PWL_j(i) \leftarrow 1$
14 **end**
15 **End**
16 **Return** $\mathbf{PWL_j}$

Line 5. Also, assume that θ (Line 8 of Algorithm 2) is equal to 0.5. Then the value of m (Line 8 of Algorithm 2) is equal to 200 and, finally, after updating the set PW, its size is equal to 1150 which is sent to the workers for annotation. Once the annotation process is over, the next step in DedupCrowd is to estimate the error of the utilized workers. This is done by WEEM, as explained in the next section.

B. WEEM NUMERICAL EXAMPLE

To demonstrate how WEEM estimates the error of workers when there are more than three workers, an example of the crowdsourcing system is simulated with the following parameters:

- Number of workers: n: 7
- Number of tasks: m: 10

Table 6 depicts the labeling of seven workers for ten HITs. It shows that w1 for HIT1 annotates it as D which indicates duplicate profiles and HIT4 is annotated ND, which indicates not duplicate profiles. Once the process is complete, the true error of workers can be calculated since it is assumed that the true label of HITs is known for the purposes of explanation of the example. As an example, a comparison of the first worker's annotation (w1) with the true labels in the last row of Table 6 shows that this worker made three errors in their annotations out of ten, thus their true error is 0.3. The annotations of the other six workers are used to evaluate the quality of the work of the given worker, using the HGP algorithm. There are $S(6,2)^2$ partitions as the candidate for sets S and T. Table 7 reports the estimated value for the first worker using the HGP algorithm (Algorithm 4) and Joglekar *et al.*'s algorithm. By accessing the true label, the true error for each worker is determined.

Comparing the results of the HGP algorithm and Joglekar *et al.*'s algorithm, the HGP algorithm's estimation is better since it is closer to the true error. Once WEEM estimates the error of the active workers, the next step in DedupCrowd is to categorize them as either active or evict, according to their performance in the annotation process. As explained in the next section, this is done by WoPM.

C. WoPM NUMERICAL EXAMPLE

Let us assume a worker (wi) in our crowdsourcing setting performs according to the reported value in Table 8 from the first to the fifth batch. Their performance in five batches (current error) along with their error history (past errors) is depicted in Table 8. In Table 9, cm associated calculations are reported along with value of standard deviation in each batch and α and β values.

Control measurement: As shown in Table 9, the value of cm calculated from Algorithm 5 after the completion of the first batch is 0.287. As shown in Table 8, this value is higher than the estimated error value (0.2) in the first batch using the HGP algorithm. This is because the worker has a relatively

$^2 31$

TABLE 6. Crowd labeling example.

Workers	Tasks										True Error
	$HIT_1{}^1$	HIT_2	HIT_3	HIT_4	HIT_5	HIT_6	HIT_7	HIT_8	HIT_9	HIT_{10}	
$w_1{}^1$	D	D	D	ND	ND	D	ND	ND	D	D	0.3
w_2	ND	D	D	ND	D	ND	ND	ND	D	D	0.3
w_3	D	D	ND	ND	ND	D	ND	ND	D	D	0.4
w_4	D	ND	ND	ND	D	D	ND	ND	D	D	0.1
w_5	D	ND	ND	ND	D	D	ND	ND	D	D	0.1
w_6	D	D	ND	ND	D	ND	ND	ND	ND	D	0.2
w_7	ND	D	ND	ND	D	ND	D	ND	D	D	0.2
True Labels	D	D	ND	ND	D	ND	ND	ND	D	D	

TABLE 7. Comparison of heuristic approach and Joglekar *et al.*'s algorithm estimations.

	HGP algorithm	Joglekar et al. Algorithm	True Error
$p_1^{\wedge}$	0.229	0.135	0.3

TABLE 8. Current and past error of a given worker (example, HGP algorithm output).

Past error	b_1 error (batch1)	b_2 error (batch2)	b_3 error (batch3)	b_4 error (batch4)	b_5 error (batch5)
0.4	0.2	0.25	0.3	0.4	0.47

TABLE 9. Control measurement calculations (example).

	b_1 (batch1)	b_2 (batch2)	b_3 (batch3)	b_4 (batch4)	b_5 (batch5)
w_{past}	0.434	0.234	0.150	0.109	0.087
w_{ij}	0.566	0.383	0.283	0.223	0.183
cm_{ij}	**0.287**	**0.266**	**0.272**	**0.300**	**0.331**
p_{ij}	0.2	0.25	0.3	0.4	0.47
Absolute difference	0.087	0.016	0.028	0.1	0.139
σ_{ij}	0.079	0.014	0.01	0.015	0.025
$\alpha \leftarrow 3$					
$\beta \leftarrow 5$					

TABLE 10. Multi-rule situation and the worker's status (example).

	b_1 (batch1)	b_2 (batch2)	b_3 (batch3)	b_4 (batch4)	b_5 (batch5)
$p_{ij}^{\wedge}$	0.2	0.25	0.3	0.4	0.47
Rule $1_{\alpha s}$ / output	(0.167,0.454)/ accept	(0.157,0.374)/ accept	(0.177,0.366)/ accept	(0.217,0.382)/ reject	(0.255,0.406)/ reject
Rule $1_{\beta s}$ / output				(0.163,0.436)/ accept	(0.205,0.456)/ reject
Rule $2_{\alpha s}$ / output				accept	
Rule $R_{\alpha s}$ / output				accept	
Status	***active***	***active***	***active***	***active***	***evict***

high past error value (0.4). This leads to an increased value of cm in the first batch, making it higher than its estimated error in b1. As explained in Section 6.1, cm considers the past error along with the current error when determining its value. The effect of past error is decreased when the worker annotates more batches. From Table 8, the current error of the worker in b3 is less in the determination of cmij. The cm variable utilizes two inputs, past error and current error, in determining the worker's performance rather than only considering their performance in the current batch to ascertain their number of errors.

Weight: As explained in Section 6.1, the weights of the current batch, previous batch and past batches of the worker are important in calculating the cm value. To explain with an example, according to Table 9, the weights of these factors in the first batch are: 0.434 (past) and 0.566 (b1) so their summation is equal to one. In the second batch, the weights are: 0.234 (past), 0.383(b1), and 0.383(b2). As previously mentioned in Section 6.1, for monitoring the current performance of wi, effect of the worker's past error history is decreased compared to their current history. The values in the first row of Table 9 show the decrease in the weight for the past error from 0.434 to 0.087.

Multi-rules QC: Table 10 shows the detail of the Multi-rules QC process. In the first batch, the status of the worker is calculated as active since the value of their estimates (0.2) falls inside the associated interval of Rule 1αs (0.167, 0.454). This procedure is repeated for the second and third batch and the other three rules are not called. In the fourth batch, the first rule (Rule 1αs) is rejected and the other three rules are called. Since these three rules are not violated, the status of the worker is still considered active. In the fifth batch, the worker's performance degrades in comparison with the fourth and since the first two rules are violated, the worker is evicted.

Once the worker's performance management process using WoPM is complete, the next step in DedupCrowd is to determine the label for each member of PW. This is done by LAM, as explained in the next section.

Note: The value of σij is calculated using Formula 10. For example, its value in the first and second batch is calculated as shown in the following:

$$\sigma_{i1} = stdv\left(past_i, c\, m_{i1}^{\wedge}\right) = stdv(0.4, 0.287) = 0.079 \tag{10}$$

$$\sigma_{i2} = stdv\left(past_i, c\, m_{i1}^{\wedge}, cm_{i2}^{\wedge}\right) = stdv(0.4, 0.28, 0.26) = 0.014 \tag{11}$$

TABLE 11. Crowd labeling example.

Workers	Tasks										True Error	HGP Algorithm
	HIT 1	HIT 2	HIT 3	HIT 4	HIT 5	HIT 6	HIT 7	HIT 8	HIT 9	HIT 10		
w1	D	ND	D	D	ND	PD	PD	ND	PD	D	0.3	0.324
w2	D	ND	ND	D	D	ND	D	ND	PD	D	0.3	0.354
w3	ND	ND	D	D	ND	ND	PD	ND	PD	PD	0.4	0.388
w4	PD	PD	D	D	PD	ND	PD	ND	PD	D	0.5	0.438
w5	D	PD	PD	D	ND	ND	D	ND	PD	PD	0.4	0.47
w6	D	ND	D	D	ND	PD	D	ND	D	D	0.1	0.162
w7	PD	ND	PD	D	ND	D	PD	ND	D	D	0.4	0.326
True Labels	D	ND	D	D	ND	ND	D	ND	D	D		

TABLE 12. Workers' weight.

W_1	W_2	W_3	W_4	W_5	W_6	W_7
0.676 (1- 0.324)	0.646	0.612	0.562	0.53	0.838	0.674

TABLE 13. Calculating final labels using weighted MV.

Classes grades	HITs									
	HIT_1	HIT_2	HIT_3	HIT_4	HIT_5	HIT_6	HIT_7	HIT_8	HIT_9	HIT_{10}
Not-Duplicate	*0.61*	*3.45*	*0.65*	*0.00*	*2.65*	*3.02*	*0.00*	*4.54*	*0.00*	*0.00*
Possibly Duplicate	*1.24*	*1.09*	*1.20*	*0.00*	*0.56*	*1.51*	*2.52*	*0.00*	*3.03*	*1.14*
Duplicate	*2.69*	*0.00*	*2.69*	*4.54*	*0.65*	*0.00*	*2.01*	*0.00*	*1.51*	*3.40*
Final labels (Weighted MV)	D	ND	D	D	ND	ND	PD	ND	PD	D

D. LAM NUMERICAL EXAMPLE

The current example is simulated to show the working of LAM considering the following parameters:

- Number of workers: n: 7, number of HITs: m: 10

Table 11 depicts the labeling of seven workers for ten HITs. Assume the true errors of workers are known, as shown in Table 11. The labeling of the other six workers is used to estimate the error of the given worker using the HGP algorithm. The errors estimated using HGP are shown in the last column of Table 11.

Note 1: For simplicity, assume that all workers are considered active after applying the WoPM module.

Note 2: The weights of workers in assigning the final labels are calculated by considering their errors.

The calculation of the weighted MV algorithm is shown in Table 13 and the final label for each HIT is shown in the last row of this table. The calculations of Lines 5 to 7 of Algorithm 9 are depicted in the first, second and third rows of this table. The calculations for HIT_1 in Table 13 show that the final label is a duplicate since its score (2.69) is the highest in comparison with the other two labels. The final label for the other HITs is assigned analogously.

Note: To show how the value 2.69 of the duplicate label in HIT1 is calculated based on Line 7 of Algorithm 9, workers w_1, w_2, w_5 and w_6 annotate HIT1 as D. Thus, by using their opinion weights shown in Table 12, we have

$$d(1) = 0.676 + 0.646 + 0.53 + 0.838 = 2.69$$

IX. RELATED WORK

In this section, we briefly describe prior work in two categories that are related to this work. The aim of this discussion is to highlight the gaps in the existing literature and the contribution of the proposed method in addressing these gaps.

Privacy-preserving record linkage - One of the challenges in duplicate detection is data privacy and confidentiality as it deals with analyzing personal identification information of customers, such as individuals' names, addresses and dates of birth. Organizations' prefer to keep this information secure and not share it because: (a) their customers may not want

to share their personal information with other companies for obvious privacy reasons, and (b) customers are the bread and butter of any company and rival companies and organizations will obviously be thirsty for such data. To address this, there has been research effort focusing on privacy-preserving record linkage (PPRL) [38]. However, PPRL is still young and out of its identified 15 dimensions, most of them are focusing on the development of privacy-preserving approximate matching of strings. However, for de-duplication initial knowledge in terms of classifying the records is needed for the automated deduplication model to be scalable for practical applications. DedupCrowd aims to address that by using the technique of crowdsourcing and using CSRs as the crowd to address these concerns.

Crowdsourcing Quality Control – Utilizing the notion of Crowdsourcing for deduplication in DedupCrowd allow us to gather training data for the automated process of deduplication. While this addresses the privacy and security issues related to the customer data, it gives rise to checking and ensuring the quality of the crowd. Unlike areas such as psychometric personality test where an accuracy of 59% in the output of crowd data is acceptable [39], in deduplication we need an accuracy close to 100%. Hence, we need a system to monitor the quality of the crowd's work to ensure an acceptable accuracy of the automated process. The literature discusses many approaches that aim to address this problem. For example, Meek *et al.* [40] propose seven pillar to check the quality of the captured crowdsourced data and to confirm its accuracy. Lívia Castro *et al.* [41] propose a taxonomy of 11 methods which they propose to use in order to test the validity of crowdsourced geographic information. Hoß feld *et al.* [42] highlight the challenges in conducting web-based crowdsourcing and the need for subjective quality assessment to detect cheaters and outliers. Allahbakhsh *et al.* [43] list checking worker profiles, reputation, expertise as some of the mechanisms that can be used in crowdsourcing scenarios to ensure its quality control. For quality design approaches, effective task preparation and workers selection are listed as the two mechanism that can be used to have a good crowd for crowdsourcing. While these approaches are required in the public domain of crowdsourcing – where the crowd can literally be any one from the WWW, in our application of DedupCrowd the requirements are more stringent in terms of who from the local CSRs can be the crowd and having methods to check for the workers' quality over each task and evict them when they perform poorly. To the best of our knowledge, such mechanisms have not been developed in the literature in such a context. There is related work to estimate worker error in two main streams, namely Gold-standard performance method and Plurality answer agreement. In the first one, gold standard tasks are posted to workers to calculate their performance in the annotation process. However, this method cannot be relied upon at all times as first making gold standard task is not easy and second there will be scenarios where if selected gold standard tasks are used, the worker knows its answer. In the second stream, the output of a given worker is evaluated against the output of other workers. The problem with this is that there is a possibility of coalition between workers with an intend to cheat that will lead to the overall low quality of work. These drawbacks specified to in-house crowdsourcing are addressed by DedupCrowd in this paper.

X. CONCLUSION

In this paper, an explanation is given as to how customer data is prepared for training for the purpose of duplicate detection. DedupCrowd and its modules were explained, where the labels of each identified pair of customer data are annotated. An explanation is given as to how the statistical algorithm is utilized in DedupCrowd to estimate the performance of workers and how the SQC approach is used to monitor the workers' performance. Also explained was the use of the multi-rule QC to evict poor workers to ensure the final labels are trustworthy. Finally, we have shown through working examples how DedupCrowd and its four integrated modules are working. In our future work, we aim to utilize the gathered training data from DedupCrowd to train DedupNN, a neural network-based approach that automates the process of the duplicate detection.

REFERENCES

[1] K. J. Hammond, "The value of big data isn't the data," *Harvard Bus. Rev.*, vol. 1, 2013. [Online]. Available: https://hbr.org/2013/05/the-value-of-big-data-isnt-the

[2] T. C. Redman, "Data's credibility problem," *Harvard Bus. Rev.*, vol. 91, no. 12, pp. 84–88, 2013.

[3] K. A.-M. Sarpong and J. K. Arthur, "Analysis of data cleansing approaches regarding dirty data-a comparative study," *Int. J. Comput. Appl.*, vol. 76, no. 7, pp. 14–18, Jan. 2013.

[4] K. A.-M. Sarpong and J. K. Arthur, "A review of data cleansing concepts-achievable goals and limitations," *Int. J. Comput. Appl.*, vol. 76, no. 7, pp. 19–22, Jan. 2013.

[5] E. Rahm and H. H. Do, "Data cleaning: Problems and current approaches," *IEEE Data Eng. Bull.*, vol. 23, no. 4, pp. 1–11, Dec. 2000.

[6] Louise Emery. (2017). *10 Reasons Why Duplicate Data is Harming Your Business*. [Online]. Available: https://www.qgate.co.uk/blog/10-reasons-why-duplicate-data-is-harming-your-business/

[7] [Online]. Available: http://users.cecs.anu.edu.au/~Peter.Christen/Febrl/febrl-0.3/febrldoc-0.3/node7.html

[8] [Online]. Available: https://www.edq.com/blog/manual-vs.-automated-data-validation/

[9] A. M. Turing, "Computing machinery and intelligence," *Mind*, vol. 1, pp. 433–460, May 1950.

[10] D. C. Brabham, "Crowdsourcing as a model for problem solving: An introduction and cases," *Convergence*, vol. 14, no. 1, pp. 75–90, Feb. 2008.

[11] A. D. Sarma, A. Parameswaran, H. Garcia-Molina, and A. Halevy, "Finding with the crowd," Stanford InfoLab., Tech. Rep., [Online]. Available: http://ilpubs.stanford.edu:8090/1049/2012

[12] M. Bernstein, H. E. Chi, L. Chilton, B. Hartmann, A. Kittur, and C. R. Miller, "Crowdsourcing and human computation: Systems, studies and platforms," in *Proc. Extended Abstr. Hum. Factors Comput. Syst.*, Vancouver, BC, Canada, May 2011, pp. 53–56.

[13] J. Yi, R. Jin, A. K. Jain, and S. Jain, "Crowdclustering with sparse pairwise labels: A matrix completion approach," in *Proc. AAAI Workshop Human Comput.*, Toronto, Canada., May 2012, pp. 1–7.

[14] J. P. Bigham, C. Jayant, H. Ji, G. Little, A. Miller, R. C. Miller, R. Miller, A. Tatarowicz, B. White, S. White, and T. Yeh, "VizWiz: Nearly real-time answers to visual questions," in *Proc. 23nd Annu. ACM Symp. User Interface Softw. Technol.*, New York, NY, USA, 2010, pp. 333–342.

[15] A. Pak and P. Paroubek, "Twitter as a corpus for sentiment analysis and opinion mining," in *Proc. 7th Int. Conf. Lang. Resour. Eval.*, 2010, pp. 1320–1326.

[16] S. B. Davidson, S. Khanna, T. Milo, and S. Roy, "Using the crowd for top-k and group-by queries," in *Proc. 16th Int. Conf. Database Theory*, Genoa, Italy, 2013, pp. 225–236.
[17] S. E. Whang, P. Lofgren, and H. Garcia-Molina, "Question selection for crowd entity resolution," *Proc. VLDB Endowment*, vol. 6, no. 6, pp. 349–360, Apr. 2013.
[18] A. Doan, J. M. Franklin, D. Kossmann, and T. Kraska, "Crowdsourcing applications and platforms: A data management perspective," *Proc. VLDB Endowment*, vol. 4, no. 12, pp. 1508–1509, 2011.
[19] A. Feng, M. Franklin, D. Kossmann, T. Kraska, S. R. Madden, S. Ramesh, A. Wang, and R. Xin, "Crowddb: Query processing with the vldb crowd," *Proc. VLDB Endowment*, vol. 4, no. 12, pp. 125–136, 2011.
[20] C. Gokhale, S. Das, A. Doan, J. F. Naughton, N. Rampalli, J. Shavlik, and X. Zhu, "Corleone: Hands-off crowdsourcing for entity matching," in *Proc. SIGMOD*, vol. 2014, pp. 601–612.
[21] L. Jiang, Y. Wang, J. Hoffart, and G. Weikum, "Crowdsourced entity markup," in *Proc. 1st Int. Conf. Crowdsourcing Semantic Web*, 2013, pp. 1–10.
[22] G. Demartini, D. E. Difallah, and P. Cudré-Mauroux, "ZenCrowd: Leveraging probabilistic reasoning and crowdsourcing techniques for large-scale entity linking," in *Proc. 21st Int. Conf. World Wide Web*, Lyon, France, Apr. 2012, pp. 469–478, 2012.
[23] Y. Yang, P. Singh, J. Yao, C. M. A. Yeung, A. Zareian, X. Wang, Z. Cai, M. Salvadores, N. Gibbins, W. Hall, and N. Shadbolt, "Distributed human computation framework for linked data co-reference resolution," in *Proc. Semantic Web Res. Appl.*, 2011, pp. 32–46.
[24] B. Mozafari, P. Sarkar, M. J. Franklin, M. I. Jordan, and S. Madden, "Active learning for crowd-sourced databases," 2012, *arXiv:1209.3686*. [Online]. Available: https://arxiv.org/abs/1209.3686
[25] O. K. Hussain, E. Chang, V. Ramakonar, and T. S. Dillon, "A customer relationship management ecosystem that utilizes multiple sources and types of information conjointly," in *Proc. 6th IEEE Int. Conf. Digital Ecosyst. Technol. (DEST)*, Jun. 2012, pp. 1–6.
[26] S. L. Pan and J.-N. Lee, "Using e-CRM for a unified view of the customer," *Commun. ACM*, vol. 46, no. 4, pp. 95–99, Apr. 2003.
[27] A. Reid and M. Catterall, "Invisible data quality issues in a CRM implementation," *J. Database Marketing Customer Strategy Manage.*, vol. 12, no. 4, pp. 305–314, 2005.
[28] A. K. Elmagarmid, P. G. Ipeirotis, and V. S. Verykios, "Duplicate record detection: A survey," *IEEE Trans. Knowl. Data Eng.*, vol. 19, no. 1, pp. 1–16, Jan. 2007.
[29] M. Saberi, K. Omar Hussain, and E. Chang, "Statistical quality control framework for crowd-worker in ER-in-house crowdsourcing system," in *Proc. 20th Annu. Int. Conf. Inf. Qual.*, 2015, pp. 476–482.
[30] D. C. Montgomery, *Introduction to Statistical Quality Control*. Hoboken, NJ, USA: Wiley, 2007.
[31] I. Abraham, O. Alonso, V. Kandylas, R. Patel, S. Shelford, A. Slivkins, and H. Wu, "Crowdsourcing gold-HIT creation at scale: Challenges and adaptive exploration approaches," in *Proc. HCOMP Workshop*, 2013, pp. 1–9.
[32] J. Wang, T. Kraska, M. J. Franklin, and J. Feng, "Crowder: Crowdsourcing entity resolution," *Proc. VLDB Endowment*, vol. 5, no. 11, pp. 1483–1494, 2012.
[33] M. Joglekar, H. Garcia-Molina, and A. Parameswaran, "Evaluating the crowd with confidence," in *Proc. 19th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Chicago, Il, USA, 2013, pp. 1483–1494.
[34] P. E. Costa, C. A. Lorena, C. P. L. F. A. Carvalho, and A. A. Freitas, "A review of performance evaluation measures for hierarchical classifiers," in *Proc. AAA*, 2007, pp. 1–6.
[35] L. Hu and Z. Wang, "The application of Student's t-test in internal quality control of clinical laboratory," *Frontiers Lab. Med.*, vol. 1, no. 2, pp. 125–128, 2012.
[36] S. P. Caudill, R. L. Schleicher, and J. L. Pirkle, "Multi-rule quality control for the age-related eye disease study," *Statist. Med.*, vol. 27, no. 20, pp. 4094–4106, Sep. 2008.
[37] M. Lease and O. Alonso, "Crowdsourcing and human computation, introduction," in *Encyclopedia of Social Network Analysis and Mining*. New York, NY, USA: Springer, vol. 2014, pp. 304–315.
[38] M. Gladbach, Z. Sehili, T. Kudrass, P. Christen, and E. Rahm, "Distributed privacy-preserving record linkage using pivot-based filter techniques," in *Proc. IEEE 34th Int. Conf. Data Eng. Workshops (ICDEW)*, Apr. 2018, pp. 33–38.
[39] B. S. Loe, F. Smart, L. Firtova, C. Brauner, L. Lueneborg, and D. Stillwell, "Validating the quality of crowdsourced psychometric personality test item," in *Proc. 4th AAAI Conf. Hum. Comput. Crowdsourcing*, 2016, pp. 1–21.
[40] S. Meek, J. Mike Jackson, and G. Didier Leibovici, "A flexible framework for assessing the quality of crowdsourced data," in *Proc. 17th AGILE Conf. Geographic Inf. Sci.*, Jul. 2014, pp. 1–7.
[41] D. L. Castro, P. D. A. Jofo, S. R. R. Dos, and Z. Alexander, "A taxonomy of quality assessment methods for volunteered and crowdsourced geographic information," *Trans. GIS*, vol. 22, no. 2, pp. 542–560, 2018.
[42] T. Hoßfeld, M. Hirth, P. Korshunov, P. Hanhart, B. Gardlo, C. Keimel, and C. Timmerer, "Survey of web-based crowdsourcing frameworks for subjective quality assessment," in *Proc. IEEE 16th Int. Workshop Multimedia Signal Process. (MMSP)*, Sep. 2014, pp. 1–6.
[43] M. Allahbakhsh, B. Benatallah, A. Ignjatovic, H. R. Motahari-Nezhad, E. Bertino, and S. Dustdar, "Quality Control in Crowdsourcing Systems: Issues and Directions," *IEEE Internet Comput.*, vol. 17, no. 2, pp. 76–81, Aug. 2013.

MORTEZA SABERI was a Lecturer with the Department of Industrial Engineering, University of Tafresh. He is currently a Lecturer with UTS and has outstanding research records and significant capabilities in the area of business intelligence, data mining, and applied machine learning. He has published more than 140 papers in reputable academic journals and conference proceedings. His Google Scholar citations and h-index are 1400 and 18, respectively. He was a recipient of the 2006–2012 Best Researcher of Young Researcher Club, Islamic Azad University (Tafresh Branch). He was also a recipient of the National Eminent Researcher Award among Young Researcher Club, Islamic Azad University members.

OMAR KHADEER HUSSAINZ is currently a Senior Lecturer with the University of New South Wales, Canberra. His research interests include business intelligence, cloud computing, and logistics informatics. In these areas, his research work focusses on utilizing decision making techniques for facilitating smart achievement of business outcomes. His research work has been published in various top international journals, such as *Information Systems*, *The Computer Journal*, *Knowledge Based Systems*, and *Future Generation of Computer Systems*. He has received awards and funding from competitive bodies such as the Australian Research Council for his research.

ELIZABETH CHANG is currently a Professor and Canberra Fellow with UNSW, Australian Defence Force Academy (ADFA). She has 30 years of work experience in both Academia and Industry. She has been a Full Professor in IT, software engineering, and logistics informatics for 14 years. She had been in senior positions in commercial corporations for ten years, typically working on commercial grade large software development. Her key research strength is in large complex software development methodologies, requirement engineering, structure and unstructured database design and implementation, trust, security, risk, and privacy. In the 2012 edition of MIS Quarterly vol. 36 issue, four Special Issues on Business Research, he was listed fifth in the world for researchers in business intelligence.