Efficient Worker Selection Through History-based Learning in Crowdsourcing

Tarek Awwad*†, Nadia Bennani*, Konstantin Ziegler†, Veronika Sonigo‡, Lionel Brunie*, Harald Kosch†
*University of Lyon, CNRS, INSA Lyon, LIRIS - France
†University of Passau, Department of Distributed and Multimedia Information Systems - Germany
‡FEMTO-ST Institute - CNRS - Université Bourgogne Franche-Comté - France
[first.last]@{*insa-lyon.fr, †uni-passau.de ‡univ-fcomte.fr}

***Abstract*—Crowdsourcing has emerged as a promising approach for obtaining services and data in a short time and at a reasonable budget. However, the quality of the output provided by the crowd is not guaranteed, and must be controlled. This quality control usually relies on worker screening or contribution reviewing at the cost of additional time and budget overheads. In this paper, we propose to reduce these overheads by leveraging the system history. We describe an offline learning algorithm that groups tasks from history into homogeneous clusters and learns for each cluster the worker features that optimize the contribution quality. These features are then used by the online targeting algorithm to select reliable workers for each incoming task. The proposed method is compared to the state of the art selection methods using real world datasets. Results show that we achieve comparable, and in some cases better, output quality for a smaller budget and shorter time.**

***Index Terms*—Crowdsourcing; Task clustering; Cost reduction; Offline learning; Worker selection.**

I. Introduction

A. Crowdsourcing

Crowdsourcing is an emergent technique which consists in harnessing the skills of the crowd in order to resolve problems that cannot be resolved by algorithms within satisfactory precision and in a reasonable time and budget [1]. Typically, a crowdsourcing system is composed of three main entities: the requester who is the task owner, the platform and the workers (a.k.a. the crowd). In practice, complex problems are subdivided by requesters into small Human Intelligence Tasks (HITs) which, in turn, are assigned through a web-based platform[1] [2], to the connected workers who contribute in exchange of a monetary reward. Contributions from different workers are then aggregated to infer the solution to the initial problem. Crowdsourcing tasks can range from basic multiple choice questions to semantic labeling and artifact creation.

B. Quality issues in Crowdsourcing

The crowd that produces the data in crowdsourcing is constituted of workers characterized by various demographical (age, gender...), educational (level and domain of study,...) and interest-related features. This guarantees a rich and diversified output of the process. However, being generated by anonymous actors, crowdsourced data are of un-guaranteed quality. Their correctness indeed depends on the reliability of the workers i.e., both, their ability to give the correct answer and their trustworthiness. Therefore, there is a tremendous need for verifying the quality of crowdsourced data. This is however a very challenging goal, notably because the reliability of a contributor is not the same over the various tasks she participates in.

Previous works [2] have shown that increasing the number of collected contributions per task improves the quality of the final output. Testing the workers before their participation or reviewing their contributions afterward [3] helps also increase the quality of the crowdsourced data. However, these methods require more assignments and thus, they suffer from additional budget and time overheads, which limits the efficiency of the crowdsourcing process especially in large scale applications such as learning-dataset labeling and time-sensitive applications like event- and rescue-related information collection. The main question we address in this work is: how to control the quality of the data, while minimizing the budget and time overheads and being agnostic from complementary knowledge such as gold standard and trust information about workers.

C. Contributions and paper organization

In this paper, we describe a novel approach to provide quality control (QC) in crowdsourcing campaigns while reducing the budget and the time overheads. Our contributions are as follows:

1) We propose an offline learning method which infers the worker features that optimizes workers' performance for each task type. These features are used at runtime to select, for an incoming task, the most reliable workers within the connected crowd. (Section IV)
2) We evaluate our method using real world datasets collected on CrowdFlower. Results show that: (a) for a given budget our method achieves a better output quality (up to 6% of improvement) compared to the state-of-the-art profile-based selection method and (b) to achieve the same quality, our method considerably reduces the needed time and budget. (Section V)

This article is structured as follows: in section II we describe and formalize the worker selection problem. In section III, we describe the state-of-the-art. In section IV we detail our approach. Our experiments are presented in section V. We finish by concluding the article and discussing the future work.

[1] www.mturk.com [2] www.crowdflower.com

II. Setup and problem formalization

In this section, the annotations related to tasks and workers are detailed and the worker-selection problem is formalized.

a) Tasks: A crowdsourcing task has two main states: *completed* and *running*. A *completed* task is a task for which all needed contributions have been gathered and no more contributions can be submitted. A *running* task is a submitted task that still needs more contributions. Every task can be characterized by a feature vector. All the possible task features constitute the task feature set of size q which we call TF. An instance of TF is $F_t =< F_t^{(1)}, ..., F_t^{(q)} >$. It denotes the feature vector of a task t. We consider, for simplicity, in the remainder of this paper Multiple Choice Question (MCQ) tasks[3]. A MCQ task has a set of $|L|$ options denoted $L = \{o_1, ..., o_L\}$, and one correct answer $r \in L$.

b) Workers: Similarly to a task, we consider that each worker w is characterized by a profile $P_w =< P_w^{(1)}, ..., P_w^{(p)} >$ where the features of P_w are parts of a worker feature set WF of size p. A worker profile can have three types: (i) a *declarative* profile built using information provided by the user, (ii) a *derived* profile computed from the user interaction in the system and (iii) a *hybrid* profile which combines both types of data. In our terminology we consider the first type hence, P_w denotes the **declarative** profile of w.

c) System history: Commercial crowdsourcing platforms store all data related to completed tasks, to workers as well as to their relative contributions. In this paper, we refer to these stored data as the *system history*. The sets of workers and of completed tasks in the system's history C are denoted W and T respectively. C is a $(|W| \times |T|)$-matrix of strings where $|W|$ and $|T|$ are the number of workers and completed tasks respectively. One coefficient C_{wt} of C refers to the answer given by a worker w for a task t. An empty string at C_{wt} means that the worker w did not contribute in the task t. A line C_w with only empty strings designates a worker who has never contributed to any task.

Beside its history, a system has a runtime configuration that consists of an incoming task τ and the set $W' \subseteq W$ of workers who are connected to the platform when the task is submitted.

d) The worker selection problem: Using the aforementioned setup terminology we can formalize the worker selection problem as follows:

Input : System history C, an incoming task $\tau \notin T$ and the connected crowd W'.

Output : The subset of workers in W' with the highest estimated accuracies for τ.

III. State of the art

Many methods have been proposed to perform QC in crowdsourcing. Most works have focused on optimizing the contribution aggregation process. Early works used majority voting (MV) to infer the correct answer of a given task. Furthermore, giving different weights to the different votes improves the quality of the MV aggregation. In [4], authors leverage different "accuracy features" such as graded and binary accuracy computed using the majority vote or expert labels to explicitly weight the contributions of each worker in a relevance-rating task. Other widely used techniques [3][5][6] rely on probabilistic data completion methods like the expectation maximization algorithm (EM) [7][8] to implicitly weigh the contributions and infer the correct answers. In these cases, the weights and the correct answers are simultaneously inferred by maximizing a likelihood model. The accuracy of the inference process depends on the campaign elements to be modeled (the worker, the task or both) and the parameters used to model them. Li *et al.* [5] use the worker accuracy and inaccuracy as weights for correct and wrong answers (respectively). In [6], a Generative model of Labels, Abilities, and Difficulties (GLAD) is proposed; it uses both the worker ability and the task difficulty as weights for the contributions in the aggregation process. In [9], the worker's reliability score is estimated using her participation behavior *e.g.*, time for completing a task, number of clicks,... (tracked in the platform interface using software plug-ins). This method, called fingerprinting, can eliminate spammy contributions but cannot assess the real quality of the contributions.

Optimizing the aggregation process is indeed an effective way for increasing the quality of the final output. However, it is not optimal since the quality of the results and the convergence of the method are sensitive to the initialization process and mainly to the amount of input data. Some methods propose to add more knowledge to this process using multiple stage crowdsourcing such as the produce/review workflow described in [3]. Adding a review stage increases the confidence of the aggregation process, however, it increases the time and the budget (since the reviewers are paid) needed to complete the campaign. Commercial crowdsourcing platforms control the input of the aggregation algorithm by screening the workers before allowing them to solve the actual tasks. Once they pass the screening, their contributions are no more controlled and hence, they have no obligation to submit accurate responses. To tackle this limitation, platforms like CrowdFlower, use a gold-based quality assurance [10] which consists in continuously measuring the accuracy of the worker using test tasks randomly injected in the workflow. A high error rate causes the rejection of the worker from the current campaign. Programmatic gold [11] is an extension of the gold-based QC where test tasks with incorrect answers are also used to train the workers against common errors. On one hand, those methods require a manual picking of test tasks which is not scalable. On the other hand, workers are paid for test tasks which is not optimal.

Another way to control the input of the aggregation process is by allowing only reliable workers to participate in the crowdsourcing campaign. Li *et al.* [5] propose a selection method which falls in this category. They describe an algorithm that finds, for each incoming task set, a group of most reliable workers. This is done by assigning, during the so called *probing stage*, a part of the tasks to the whole crowd

[3] Since it is possible to compute the accuracy of a worker in any achieved task, the proposed approach is independent from the task type. Thus this simplification does not affect the generality of the approach

in order to sample it and identify the reliable group for the remaining part. By targeting a specific group, one reduces the number of assignments and, as a result, decreases the budget. However because of the probing phase, the budget reduction is not optimal and the task completion time is increased. The method we propose in this paper belongs to this last category. It remedies to the aforementioned limitation by using knowledge about workers inferred from the system history.

IV. Our Approach

A. Overview

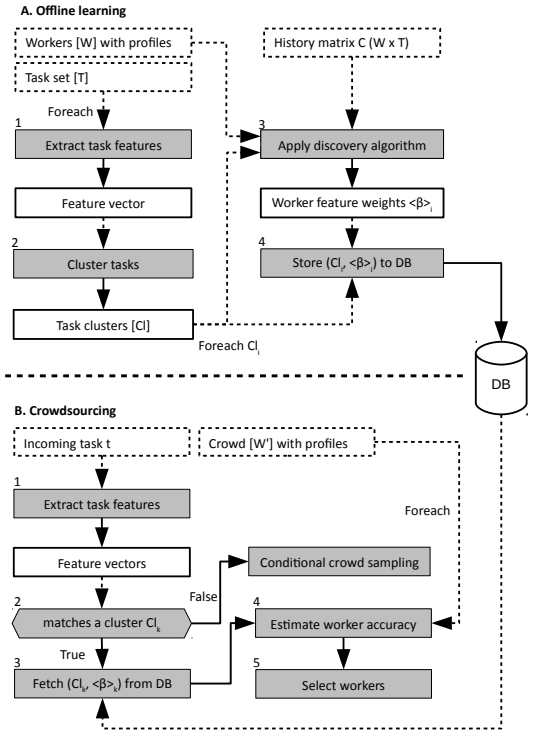

Fig. 1: An overview of our system showing two phases: A. the offline learning phase and B. The crowdsourcing phase.

Existing QC approaches are not optimal (in terms of time and budget needed to complete the task) as they deal with the particularity of each task in an extreme manner. Some ignore it and consider that all tasks are similar from a worker perspective which reduces the performance of the QC. Others consider each task to be unique which increases the budget and the time of completion. However, in practice tasks are not all similar, yet they share similar traits. Furthermore, a worker usually shows a stable performance in completing similar tasks. On the other hand, similar workers (that is, workers with similar profiles) tend to show similar performances when dealing with the same task. Our work aims at demonstrating how the QC process can take advantage of these facts.

We propose to use the system history to learn the correlation between the workers declarative profiles and the task types. This allows the *indirect* match between workers and tasks, which optimizes the task assignment process. Besides, using declarative profiles instead of derived ones helps eliminate any probing process. Optimizing the assignment and eliminating the probing phase (e.g. screening, sampling ...) reduces the time and the budget while achieving a high crowdsourcing quality.

The general workflow of the method is depicted in figure 1 and can be summarized as follows: in the *offline learning phase* (A), a feature vector is extracted for each task in the history (step A.1), then tasks are clustered based on these vectors (step A.2). For each cluster, the vector of worker features that maximizes the contribution quality is determined (step A.3) and stored (step A.4). In the *crowdsourcing phase* (B), the features of the incoming task are extracted (step B.1) and used to match the task to one of the existing types (step B.2). The feature vector associated to the found type during the learning phase is fetched (step B.3) and used to estimate the workers' accuracies for the current task (step B.4). Workers with higher accuracy are then selected to contribute to the task.

Our system is fully automatic. Indeed, crowdsourcing systems are highly dynamic in terms of tasks and workers arrivals and departures, which prevents the possibility of manually performing the task grouping or the selection process [12].

To deal with the cold start problem, we rely on the declarative profile - provided upon registration - to find and target the reliable workers.

B. Offline learning

1) Task clustering: A straightforward way of grouping the tasks into different types is to cluster them w.r.t. all of their features (e.g. length, language, reward ...) at once. Since this paper focuses on showing the impact of the offline learning on the overheads reduction, the clustering process and the task features are not detailed in this paper.

2) Discovery algorithm: The clustering process yields a set of task clusters, each of them defining a task type. The set of all clusters is denoted Cl. The next step is to infer the worker profile that maximizes the workers' performance for each type. For this purpose, an algorithm inspired by the discovery algorithm described in [5] is used. It consists of two steps. First, the workers' performance is computed in each task cluster. Second, a linear regression model is used to find the most significant worker features for this cluster. These features form the so called *perfect profile* for this task type. In contrast with the work proposed by Li *et. al* where the discovery algorithm is performed on every incoming task online, we apply it by cluster i.e., by task type and offline. Following are the details of both steps.

a) Performance inference: Estimating the performance of each worker can be achieved through multiple ways. In fact, since the offline learning deals with tasks that have been already completed, we can assume that the correct answers have been estimated upon the aggregation process. This correct answer is used to compute the worker's performance of which a straightforward representation is the worker's accuracy for the current task cluster. That is, her accuracy for all the tasks she completed in this cluster. Given a cluster cl and a worker

w, this accuracy, noted α_w, is equal to the ratio of correct answers given by w to the tasks in cl as shown in 1.

$$\alpha_w^{cl} = \frac{1}{N_w} \sum_{t \in cl} I(C_{wt} = r_t) \quad (1)$$

C is the system history, r_t is the correct answer of t and:

$$N_w = \sum_{t \in cl} I(C_{wt} \neq \varnothing) \quad \text{and} \quad I(X) = \begin{cases} 1 & if \quad X \\ 0 & else \end{cases} \quad (2)$$

Worker accuracies are used as targets in our learning model. Other effect model such as logit can also be used. However, since the goal here is to compare the overhead gain rather than the quality gain, the used model is not important as long as it is similar for the compared methods (Section V).

b) Beta vector inference: The second step of the discovery algorithm consists in determining the worker features that maximizes the workers' performance in a given task cluster. We use the linear regression model described in [5] and shown in equation 3. Let w be a worker, P_w her profile, α_w^{cl} her accuracy in a cluster cl and ϵ a Gaussian noise with mean 0:

$$\alpha_w^{cl} \sim \beta_0 + \beta_1 P_w^{(1)} + ... + \beta_p P_w^{(p)} + \epsilon, for\ all\ w \in W \quad (3)$$

Fitting the model of equation 3 yields the estimated values $\hat{\beta}_i$ of the feature weights β_i. Those weights reflect the relative importance of each profile feature in maximizing the worker performance. The set of weights computed for a given cluster cl form the *Beta-vector* of cl and is denoted $\hat{\beta}_{cl}$. The discovery algorithm is applied on every task cluster apart (See function *learn()* in algorithm (1)). Hence, the overall output of the offline learning phase is a set A of *(cluster, Beta vector)* couples. A is expressed as shown in equation 4.

$$\begin{aligned} A &= \{(cl, < \hat{\beta}_{cl} >), for\ all\ cl \in Cl\} \\ \text{where } &< \hat{\beta}_{cl} > = < \hat{\beta}_0, \hat{\beta}_1, ..., \hat{\beta}_p >_{cl} \end{aligned} \quad (4)$$

C. Online crowdsourcing

The online crowdsourcing phase is depicted in figure 1.B. For a given incoming task, selecting the reliable workers is done by extracting the features of the task, matching it to an existing type and selecting the top connected workers in terms of their estimated accuracy in completing the task using the model of equation 3, the profiles of the workers and the learned associations A for the matched type. Algorithm (2) details the targeting process. In this algorithm, λ is a *selection rate* that determines the portion of workers that should be selected. This parameter reflects the requester's needs in terms of budget and quality. Note that if matching the task to an existing cluster is not possible, an online sampling-based learning is launched.

V. Evaluation

In this section, we describe the datasets, the experiments and the results of the system evaluation. Our method is compared with Li *et. al*'s method since, to the best of our knowledge, it is the most robust method of QC that does not require manual interaction from the requester (like for gold-based [10] and programmatic gold-based QC [11]) while being independent from the used aggregation method (in contrast to aggregation optimization approaches [6]) and resilient to the cold start problem.

Algorithm 1: The learning functions

data : System history C, worker set W, task cluster cl

Function *learn*($cl \in Cl$) **:**
  $W_{cl} \leftarrow \{w \in W \quad | \quad \exists t \in cl, C_{wt} \neq \varnothing\}$
  foreach w *in* W_{cl} **do**
    $\alpha_w^{cl} \leftarrow computeAccuracy(C, w, cl)$ // equation 1
    $\alpha_{cl}.add(\alpha_w)$
  $< \hat{\beta}_{cl} > \leftarrow fit(\alpha_{cl}, W_{cl})$ // equation 3
  return $(cl, < \hat{\beta}_{cl} >)$

Algorithm 2: Targeting Algorithm

data : Incoming task τ, connected workers W', learned associations A, selection rate λ, cluster list Cl

Function *target*(τ, W', A, $\lambda \in [0, 1]$) **:**
  $F_\tau \leftarrow extractFeatures(\tau)$
  $cl_\tau \leftarrow argmin(distanc(Cl, F_\tau))$
  $< \hat{\beta}_{cl_\tau} > \leftarrow \{< \hat{\beta} > | (< \hat{\beta} >, cl) \in A\ and\ cl = cl_\tau\}$
  foreach w *in* W' **do**
    $\hat{\alpha}_w \leftarrow dotProduct(w.P, < \hat{\beta}_{cl_\tau} >)$
  $W'_{Sorted} \leftarrow W'.sortBy(\hat{\alpha})$
  for i *in* $0 \rightarrow round(|W'| \times \lambda)$ **do**
    $W_s[i] \leftarrow W'_{Sorted}[i]$
  return $[W_s]$

A. Evaluation metrics

Reducing the overhead of the quality control is the main goal of the method described in this paper IV-A. In this paragraph, we describe the metrics that we use to assess the ability of our method to achieve this goal.

1) Quality metrics: In crowdsourcing, the contributions are often aggregated and not used individually. Equation 5 is used to compute the quality obtained using an aggregation technique Agg. The accuracy of a group G of workers for a set S of tasks using Agg is equal to the ratio of correctly guessed answers over all guessed answers. Equivalent measures have been used in other works such as [5][13] and [14].

$$AccA_{Agg}^{S}(G) = \frac{1}{|S|} \sum_{t \in S} I(Agg(C_{Gt}) = r_t) \quad (5)$$

2) Overhead measurement: As detailed earlier, the budget and the time overheads, noted OH_{budget} and OH_{time}, are directly related to the number of additional assignments made to serve only the QC process e.g. the probing assignments. A straightforward method for computing the time and the budget overheads resulting from those contributions is to multiply their number by the time-per-assignment-per-worker TA and the reward-per-assignment-per-worker RA respectively. Equation 6 computes the overhead drop between two different QC methods i and j. Assume that $Assignment_i$

Dataset	Knowledge	Disambiguation
Type	3-option MCQ	3-option MCQ
Tasks/workers/contributions	60 / 140 / 8400	60 / 100 / 6000
Avg. worker accuracy	61.3%	39.1%
Avg. agreement* per task	61.8%	52.8%
EM/MV accuracy (All)	81.3% / 80%	45% / 43.3%

* Percentage of votes for the most selected option for a given task

TABLE I: Statistics of our datasets.

and $Assignment_j$ are the total number of assignments for the method i and j respectively. RA is a task dependent parameter, thus, it is similar for both methods when applied on the same task. TA, on the other hand, is a more complex parameter. It depends indeed on the task itself but also on the workers (number, preferences, ...) and on the other tasks (number, rewards, ...) available in the platform at a given moment. For a fixed reward it is possible to assume that TA is the same for 2 identical tasks when crowdsourced simultaneously.

$$\begin{aligned} OH_{time} &= TA \times (Assignment_i - Assignment_j) \\ OH_{budget} &= RA \times (Assignment_i - Assignment_j) \end{aligned} \quad (6)$$

B. Datasets

In order to assess the performance of the proposed method, we built two datasets with different task types. We submitted those tasks to the CrowdFlower platform and collected contributions without test questions (See gold-based assurance in sectionIII) in order to have a real representation of the crowd.

Knowledge related tasks: The first dataset consists of knowledge related questions distributed evenly over three different knowledge domains: sport, botany and technology.

Disambiguation tasks: The second data set consists of tasks such that the worker is given a sentence with a highlighted keyword then is asked to pick among 3 Wikipedia links the one that refers to the keyword in the context of the sentence. Tasks are related to either sport or technology.

Beside the task completion, workers who participated to the campaign were asked to answer ten questions related to their profiles. Table I summarizes statistics which describe the proposed datasets.

C. Experiment and results

	Knowledge		Disambiguation	
Method	**AUC***	**Avg. Acc****	**AUC***	**Avg. Acc****
Our	**0.242**	**80.6%**	**0.153**	**51.6%**
Li et. al	0.241	80.4%	0.145	47,8%
Random	0.231	78%	0.133	44%

TABLE II: Accuracy of EM on knowledge and disambiguation datasets for different selection technique.

We realized three experiments to evaluate the impact of the offline learning on the output quality (experiment **I**) and the overheads (experiments **II** and **III**). Since tasks are homogeneous per dataset, the history is not clustered in the experiments. It is considered to form only one cluster to which the incoming tasks matches every time. This does not impact the ability of evaluating the benefit of the offline learning.

Experiment I: The first experiment aims at evaluating the impact of the offline learning on the quality of the output. We randomly sampled 40 tasks to constitute the system history denoted $train_a$ used to train our model. The remaining set of 20 tasks forms the incoming task set. It is denoted $target$. To train Li *et. al*'s algorithm, a set $train_b$ of 10 tasks was sampled from $target$. We ran the training and targeting process multiple times and for different task splits. In each run the same percentage λ of workers is selected by both methods and by a random selection process for reference. For each selection method we computed the accuracy $AccA_{MV}^{target}(G_\lambda)$ and $AccA_{EM}^{target}(G_\lambda)$ where G_λ is the set of selected workers and $0.05 < \lambda < 0.35$. The results of multiple runs are averaged and the areas under the EM accuracy curves are shown in table II along with the average EM accuracy for different values of λ. Results show that our history based learning yields better final accuracy for EM than Li *et al*'s method. This is because the learning step is performed on a larger amount of tasks which allows a better estimation of the workers' accuracy and thus a better fitting of eq.3. This is particularly clear for low agreement tasks i.e. difficult tasks. Results for MV are similar; they are not shown in this article due to space limitation.

Experiment II: The second experiment aims at evaluating the impact of the offline learning on the overhead. We used the same setup as in **Experiment I** and we computed the time and budget overhead gain. For the knowledge dataset, we paid 0.4$ per worker for solving all of the tasks i.e., $RA =$ 0.0067$/assignment. Moreover, experimental measurements showed that $TA = 17$ sec/assignment. Using equation 6 and the measured values of TA and RA, we compute the overhead difference between our method and Li *et. al*'s method and the ratio of this difference over the total budget and time of the campaign. Figure 2a shows the overhead gain we made by eliminating the probing stage for $0.05 < \lambda < 0.35$.

The overhead gain that we achieve is optimal for lower selection rates (e.g. 90% for $\lambda = 0.05$) and decreases when the selection rate increases (50% for $\lambda = 0.35$). That is because the overhead gain relies mainly on removing the probing assignments. When the number of those assignments is important w.r.t. the targeted assignments i.e., for lower values of λ, the gain is high. When the number of targeted assignments grows, the relative cost of probing assignments (w.r.t. the whole campaign cost) is reduced.

Experiment III: The third experiment aims at evaluating the output accuracy that we can achieve for a fixed budget. We compared this accuracy with the accuracy achieved by Li *et. al* for the similar budget. We start by computing the number of assignments made during a given campaign. Let W' be the set of connected workers and T' the set of incoming tasks.

For Li *et. al*'s method : Let θ and γ be respectively the percentage splits of crowd and tasks used in the learning. Let λ_{li} be the selection rate determining the number of targeted workers. The number of assignments χ made during a campaign is the sum of probing and targeting assignments:

$$\chi = \theta \times |W'| \times \gamma \times |T'| + \lambda_{li} \times |W'| \times (1-\gamma) \times |T'| \quad (7)$$

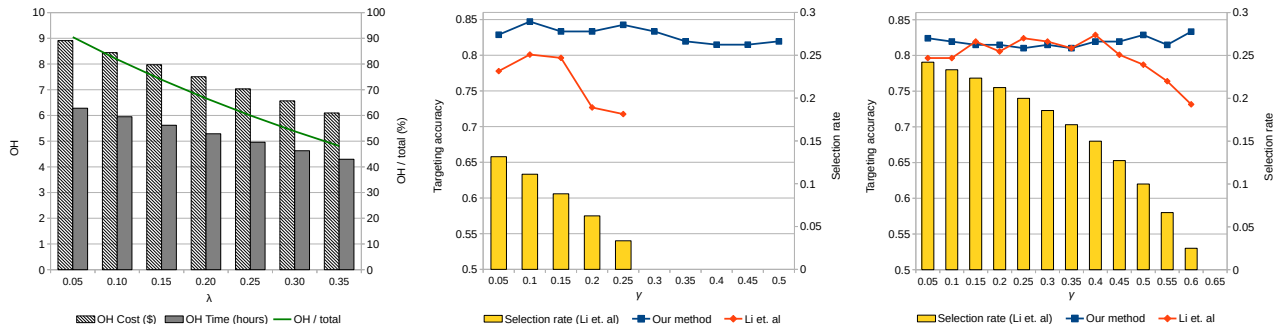

Fig. 2: (a) Overhead gain (Exp-II). EM accuracy for $\theta = 0.4$ and a fixed budget: (b) $\lambda = 0.15$ and (c) $\lambda = 0.25$ (Exp-III).

For our method : Let λ be the selection rate. The number of assignments φ made during a campaign is given by:

$$\varphi = \lambda \times |W'| \times |T'| \tag{8}$$

For a similar budget, the maximum number of assignments that can be done by both methods is identical. Hence :

$$\chi = \varphi \Leftrightarrow \theta \times \gamma + \lambda_{li} \times (1 - \gamma) = \lambda \tag{9}$$

We consider a first case where both methods select the same number of workers. In this case, $\lambda = \lambda_{li} = \theta$ (Eq. 9). For $\theta \in [0, 0.35]$, the crowd size used for learning is too small ($[0, 49]$ workers) and profile observations are not enough to fit a consistent model (eq 3). Hence, the accuracy of Li's selection is similar to the random selection's accuracy regardless the number of tasks used for learning.

In the second case we consider that λ and λ_{li} are not necessarily equal. We choose $\theta = 0.4$ to provide a sufficient number of workers for the probing stage. Figure 2b shows the results for $0.05 < \gamma < 0.5$ and a fixed budget determined by $\lambda = 0.15$. When a larger split of the tasks is used in the probing stage, a smaller number of workers can be targeted. For $\gamma = 0.3$ the whole budget is spent on probing the crowd. Hence, it is not possible to select workers. For $\gamma < 0.3$, the quality of the learning grows until the number of targeted workers becomes very low (for $\gamma > 0.2$, $\lambda_{li} < 0.06$ i.e., less than 7 workers are selected) and thus, the aggregation quality drops to 71%. Our method has an average accuracy of 82%.

Figure 2c shows the results for $\lambda = 0.25$. For this budget our method is as good as Li's method for $0.15 \leq \gamma \leq 0.4$, it has an average accuracy of 81%. Li *et. al*'s accuracy is low for $\gamma < 0.15$ because 3 tasks are not enough to probe the crowd and for $\gamma > 0.45$ because of the low selection rates. Being independent from any probing stage, our method uses the whole budget to target workers.

VI. Conclusion

In this paper we proposed an efficient profile-based worker selection method for crowdsourcing that aims at reducing the time and the budget overheads by substituting the online probing stage found in the state-of-the-art approaches by an offline learning process that learns the profile features which optimizes the worker's performance for each type of tasks. Results show that overheads are reduced for different aggregation techniques while at least maintaining their overall inference quality. We are currently building a larger dataset in order to study the impact of different task clustering techniques on the learning process. Besides, we are designing a new task feature taxonomy that optimizes the discovery of the correlation between the worker profiles and the task types.

References

[1] J. Howe, "The rise of crowdsourcing," *Wired magazine*, vol. 14, no. 6, pp. 1–4, 2006.

[2] V. C. Raykar, S. Yu, L. H. Zhao, A. Jerebko, C. Florin, G. H. Valadez, L. Bogoni, and L. Moy, "Supervised learning from multiple experts: Whom to trust when everyone lies a bit," in *ICML*, NY, USA, 2009, pp. 889–896.

[3] Y. Baba and H. Kashima, "Statistical quality estimation for general crowdsourcing tasks," in *ACM SIGKDD*, NY, USA, 2013, pp. 554–562.

[4] H. J. Jung and M. Lease, "Improving consensus accuracy via z-score and weighted voting." in *Human Computation*, 2011.

[5] H. Li, B. Zhao, and A. Fuxman, "The wisdom of minority: Discovering and targeting the right group of workers for crowdsourcing," in *WWW*, NY, USA, 2014, pp. 165–176.

[6] J. Whitehill, T. fan Wu, J. Bergsma, J. R. Movellan, and P. L. Ruvolo, "Whose vote should count more: Optimal integration of labels from labelers of unknown expertise," in *NIPS*, 2009, pp. 2035–2043.

[7] A. P. Dawid and A. M. Skene, "Maximum likelihood estimation of observer error-rates using the em algorithm," *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, vol. 28, no. 1, pp. pp. 20–28, 1979.

[8] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the em algorithm," *Journal of the royal statistical society. Series B (methodological)*, pp. 1–38, 1977.

[9] J. M. Rzeszotarski and A. Kittur, "Instrumenting the crowd: Using implicit behavioral measures to predict task performance," in *UIST*, NY, USA, 2011, pp. 13–22.

[10] J. Le, A. Edmonds, V. Hester, and L. Biewald, "Ensuring quality in crowdsourced search relevance evaluation: The effects of training question distribution," in *2010 workshop on crowdsourcing for search evaluation*, 2010, pp. 21–26.

[11] D. Oleson, A. Sorokin, G. P. Laughlin, V. Hester, J. Le, and L. Biewald, "Programmatic gold: Targeted and scalable quality assurance in crowdsourcing." *Human computation*, vol. 11, no. 11, 2011.

[12] D. Geiger and M. Schader, "Personalized task recommendation in crowdsourcing information systems current state of the art," *Decision Support Systems*, vol. 65, pp. 3 – 16, 2014.

[13] K. Mo, E. Zhong, and Q. Yang, "Cross-task crowdsourcing," in *SIGKDD*, NY, USA, 2013, pp. 677–685.

[14] Q. V. H. Nguyen, T. T. Nguyen, N. T. Lam, and K. Aberer, "Batc: a benchmark for aggregation techniques in crowdsourcing," in *ACM SIGIR*, 2013, pp. 1079–1080.