Crowdsourcing Task Assignment with Online Profile Learning

Silvana Castano(✉), Alfio Ferrara(✉), and Stefano Montanelli(✉)

DI, Università degli Studi di Milano, Via Comelico, 39, 20135 Milan, Italy
{silvana.castano,alfio.ferrara,stefano.montanelli}@unimi.it

Abstract. In this paper, we present a reference framework called Argo+ for worker-centric crowdsourcing where task assignment is characterized by feature-based representation of both tasks and workers and learning techniques are exploited to online predict the most appropriate task to execute for a requesting worker. On the task side, features are used to represent requirements expressed in terms of knowledge expertise that are asked to workers for being involved in the task execution. On the worker side, features are used to compose a profile, namely a structured description of the worker capabilities in executing tasks. Experimental results obtained on a real crowdsourcing campaign are discussed by comparing the performance of Argo+ against a baseline with conventional task assignment techniques.

1 Introduction

In the recent years, the crowdsourcing philosophy has gained a lot of attention and many crowdsourcing systems/platforms appeared on the web scene for satisfying the growing need of marketplaces where the offer of requesters providing jobs to execute can meet the work-force provided by the crowd. For requesters, the key point is to get the jobs completed as fast as possible by also considering the tradeoff between the quality of the obtained results and the expenses to sustain [11,14]. For workers, the key point is to find a system where the crowd motivation is triggered and encouraged, as well as the profits are attractive (in terms of experience, advance in human capital, remuneration) [1]. Most of the crowdsourcing platforms (e.g., CrowdFlower - https://www.crowdflower.com/, Amazon Mechanical Turk - https://www.mturk.com/, Crowdcrafting - https://crowdcrafting.org/) are *task-centric*, in that they are designed to support different types of tasks and mechanisms for task result evaluation, mostly characterized by consensus- and/or inference-based techniques [19]. On the opposite, mechanisms for worker evaluation and tools for supervising the worker accuracy/trustworthiness can (strongly) differ from one system to another. Mostly, the worker evaluation is enforced by exploiting the results on the executed tasks: the more a worker provides good/valid results, the more a worker is considered as trustworthy (see for instance [5,27,29,30]). The worker reliability is progressively revealed during task execution and it becomes known at the end of activities.

H. Panetto et al. (Eds.): OTM 2018 Conferences, LNCS 11229, pp. 226–242, 2018.
https://doi.org/10.1007/978-3-030-02610-3_13

However, humans have different knowledge and abilities, thus a crowd worker can be trustworthy on a certain task campaign that is coherent with her/his attitudes, as well as she/he can be inaccurate on another campaign with different topic requirements not compliant with her/his attitudes. In other words, there is the need to switch from a task-centric to a *worker-centric* design paradigm to leverage on the human factors of crowd workers and effectively enforce the crowdsourcing task assignment. A key capability of the worker-centric paradigm is the availability of techniques for learning the worker-profile features during task assignment and execution [2]. We argue that the capability to effectively discover and represent the profile of engaged crowd workers is a strategic asset of future crowdsourcing marketplaces. This way, it becomes possible for a system to predict the worker trustworthiness on considered topics and to selectively choose a qualified and motivated crowd to recruit/involve in a given campaign according to the required knowledge/abilities.

In this paper, we present Argo+, a reference framework for worker-centric crowdsourcing where task assignment is characterized by feature-based representation of both tasks and workers and learning techniques are exploited to online predict the most appropriate worker for a given task. On the task side, features are used to represent requirements expressed in terms of knowledge expertise that are asked to workers for being involved in the task execution. On the worker side, features are used to compose a profile, namely a structured description of the worker capabilities in executing tasks along multiple dimensions (i.e., the features). From the system point of view, the goal of Argo+ is to predict the tasks on which a worker is expected to provide successful results based on the specified task requirements, thus increasing the number of successful results while reducing the number of task executions at the same time. From the worker point of view, the goal of Argo+ is to receive tasks that are compatible with her/his profile, thus leveraging on worker interest, motivation, and finally satisfaction. A further distinguishing aspect of Argo+ is related to the *dynamic* and *adaptive* nature of the worker profiles that are continuously evolving based on the results of executed crowdsourcing activities. Online learning techniques are employed in Argo+ to capture the real worker capabilities by progressively adjusting the features of worker profiles to resemble the features of tasks that have been successfully executed by the worker. In the paper, we focus on discussing an essential implementation of the Argo+ framework characterized by the use of (i) *probabilistic topic modeling* for enforcing task assignment, and (ii) techniques inspired to the *Rocchio relevance feedback* for enforcing worker profile learning. Experimental results are finally provided to analyze the behavior of Argo+ on both a third-party dataset and a real crowdsourcing campaign, by comparing the performance of the proposed framework against a baseline with conventional task routing techniques.

The paper is organized as follows. Section 2 provides the related work. The proposed Argo+ framework is presented in Sect. 3. The Argo+ techniques and the related implementation based on probabilistic topic modeling are illustrated in Sect. 4. Experimental results are discussed in Sect. 5. Concluding remarks are finally provided in Sect. 6.

2 Related Work

The idea to enforce task assignment (a.k.a. *task routing*) by considering specific human factors according to a sort of worker-centric crowdsourcing model is becoming popular in the very-recent crowdsourcing literature [2,16]. In [8], HITs (Human Intelligent Tasks) to execute are assigned to crowd workers by matching keywords extracted from task descriptions and worker preferences extracted from worker profiles on social networks. Limitations still exist due to possible incompatibilities (i.e., use of different keywords) among task features and social-network preferences. A similar approach is presented in [1] where *relevance* and *diversity* measures are introduced to capture the workers that are most appropriate for assignment of a certain task. A *cross-task crowdsourcing* approach (CTC) characterized by a transfer learning method and a bayesian model is proposed in [22] based on the intuition that the history of executed tasks can be exploited to extract knowledge on workers abilities. In CTC, the application target is a large crowdsourcing platform where many tasks are executed by each worker so that learning can be effectively employed. Budget constraints are introduced in [11] to enforce task routing with an incentive-compatible approach. Issues about task routing discussed from the marketplace point-of-view are presented in [14] where a comparative analysis of a set of well-known crowdsourcing platforms is provided. The goal of [14] is to observe the crowdsourcing ecosystem as a whole and to provide insights about possible platform improvements on task design and worker understanding empowerments. An approach based on the specification of Service Level Agreements (SLAs) is proposed in [18] where the task assignment mechanism is dynamically adjusted to fulfill a set of declared requirements. The satisfaction of human skill requirements can be included in the SLA for being exploited by the task scheduler. However, a fixed list of skills is specified in the SLA and it cannot be extended at runtime to catch the real worker attitudes with respect to the executed tasks.

The idea to improve the crowdsourcing performances through learning of worker expertise has been also envisaged. In [10], implicit and explicit learning stages are enforced to capture the predict the degree of worker success in executing specific kinds of tasks. In [24], the worker profile is predicted by observing the results on the executions of *taste* tasks and a *taste-matching* function is proposed to adjust the prevision according to the correct task answer (presuming that such a correct task answer is available). In [9], a warm-up phase is presented for the iCrowd framework to estimate the worker accuracy through the execution of an initial set of tasks with known answer. After warm-up, the tasks assigned to a worker are adaptively chosen according to the estimated worker accuracy based on the quality of provided answers. Other similar approaches are aimed at capturing the worker abilities by relying on cognitive tests in a sort of psycho-analytics approach (e.g., [12]). Sometimes, the cognitive model is based on a predefined taxonomy/ontology of worker skills taken from target crowdsourcing applications or reference skill classifications [13,17]. Learning requires the specification of task types to consider and focused learning tasks to use for training. Online learning techniques have been also proposed for improving the

quality of crowdsourcing results. In [20], the problem of crowdsourcing labeling is modeled as a multi-armed bandit (MAB) problem where the goal is to learn the most effective combination of labelers for a given labeling task to execute. However, we stress that this solution is specifically conceived for the labeling problem under majority-voting employment. Other approaches based on (multi-armed) bandits algorithms are presented in [15,28] where the focus is on optimizing task assignment in relation with predefined budget constraints.

Original Contribution. With respect to the related work discussed above, the Argo+ framework proposed in this paper is characterized by the following original contribution. First, the features used for representing the worker expertise are not predefined in type and number and the degree of worker expertise on a certain feature dynamically changes through learning functions based on the successful rate of executed tasks. Moreover, the specification of an initial worker profile is not mandatory in Argo+, thus allowing the system to start the crowdsourcing activities without requiring the execution of an extra bulk of profiling tasks/questions devoted to recognize the worker abilities in advance with respect to the participation to real crowdsourcing campaigns. The evolving nature of the Argo+ worker profile captures the worker capabilities that have been concretely employed and shown in task execution, thus allowing to choose the work-force of a task by selecting workers that are really expected to provide a successful contribution/answer.

3 The Argo+ Framework

Consider a *requester*, namely a *campaign manager*, submitting a crowdsourcing campaign C based on a set of tasks $T = \{t_1, \ldots, t_n\}$ for execution on a crowdsourcing system CS by a crowd of workers $W = \{w_1, \ldots, w_k\}$.

A *task* $t \in T$ is defined as $t = \langle id_t, a_t, m_t, d_t, F_t \rangle$, where id_t is the unique *task identifier*, a_t is the *task action*, m_t is the *task modality*, d_t is the *task description*, and F_t is the set of *task-features*. A task action a_t denotes the task target, namely the goal that needs to be satisfied through crowd execution (e.g., picture labeling, movie recognition, sentiment evaluation). A modality m_t represents the kind of worker answer required in task execution. Conventional modalities are (i) *creation* for denoting that the worker is called to generate a free task answer, and (ii) *decision* for denoting that the task answer is chosen by workers within a set of possible alternative options. Further task modalities are possible, such as for example *rating* and *ranking*, and details are provided in [6]. A description d_t represents the task request given to each worker for illustrating what is demanded to her/him in the task execution. For instance, in a task t with action picture labeling and decision modality, the description d_t can be recognize the historical period of the following artwork among the given options. A set of task-features F_t provides a description of task requirements, namely a specification of the capabilities expected from a worker for being involved in the execution of the task t. For each feature $f \in F_t$, a *task-feature weight* $\omega(f)$ is associated to denote the relevance of f within the task-features F_t. A task-feature $f \in F_t$ is a label that denotes

an expected worker expertise in a specific domain knowledge. For instance, the task-features of the considered task t can be art history and renaissance to denote required domain knowledge.

A *worker* $w \in W$ is defined as $w = \langle id_w, F_w \rangle$, where id_w is the unique *worker identifier* and F_w is the **worker profile** expressed as a set of worker-features. A worker-feature $f \in F_w$ denotes a worker capability, namely knowledge expertise, and it is associated with a *worker-feature weight* $\omega(f)$ denoting the "degree" of expertise/ability associated with the worker. In Argo+, a worker profile is *adaptive* and *evolving*, meaning that (i) the worker-features F_w of a worker $w \in W$ are dynamically determined based on the executed tasks and (ii) the associated worker-feature weights are progressively adjusted (i.e., learned) based on the quality of executed tasks. For instance, given a set of tasks T with task-features F_t, the more a worker w successfully executes the tasks T, the more the worker-features in F_w and the corresponding worker-feature weights become similar to the task-features of F_t.

The Argo+ framework and the conceptual schema of the underlying worker/task management repository are shown in Fig. 1. The framework is articulated in the following components:

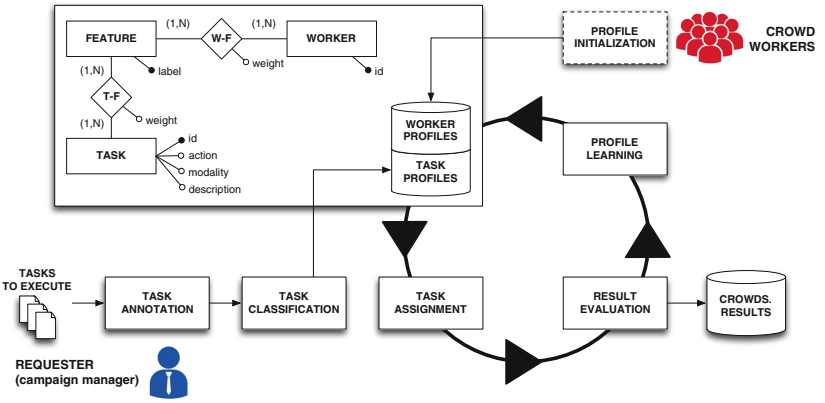


Fig. 1. The Argo+ framework

Task Annotation. This component has the goal to associate a task $t \in T$ with the corresponding set of task-features F_t and related task-feature weights. In a basic scenario, we expect that task annotation is manually performed by the requester, who has the role to choose the set of task-features F_t and to setup the corresponding weights in force of her/his understanding of the crowdsourcing campaign. The requester determines the composition of the set F_t by specifying requirements on the worker domain knowledge. Usually, tasks characterized by a common action, modality, and description are associated with the same set of features F_t. Since a campaign is typically characterized by few different types of task requests, manual task annotation can be considered a viable and effective solution.

Profile Initialization. This component has the goal to associate a worker $w \in W$ with the worker-features F_w and corresponding weights $\omega(f)$. In a crowdsourcing campaign, for each worker, the set of worker-features F_w coincides with the set of task-features F_t associated with the tasks T to execute in the crowdsourcing campaign. A weight $\omega(f)$ is then assigned to each worker-feature $f \in F_w$. Two alternative solutions are envisaged in Argo+ for profile initialization. Similarly to most of the existing crowdsourcing systems, profile initialization can be enforced through an initial *positioning questionnaire* based on ad-hoc questions in which workers self-evaluate their knowledge/abilities before their involvement in task execution (*custom profile initialization*). The questionnaire is usually provided by the campaign manager and it is defined according to the features of tasks T constituting the campaign. In this solution, the result of profile initialization is that task assignment is initially based on worker preferences as determined through the questionnaire answers. As an alternative, the worker profile can be initialized according to a default configuration which is common to any worker, thus avoiding both the preparation and the execution of a positioning questionnaire (*flat profile initialization*). In this solution, each worker profile gradually evolves from the common initialization towards the personal profile as learned from executed tasks. The flat initialization grounds in the idea that the self-evaluation of worker knowledge is subjective and untrustworthy due to possible over-/under-estimation of real worker expertise, thus a flat profile initialization is more effective than a custom one.

Task Classification. This component has the goal to aggregate the tasks T into K classes, so that tasks with similar features F_t are associated with a same class. A number of categorization approaches can be exploited for task aggregation, ranging from distance-based algorithms to probabilistic models based on latent features mined from data [23]. As a general remark, we recommend that the adopted solution for task classification is characterized by *overlap support*, meaning that a task can be associated with more than one class due to different similarities causes with other tasks. This way, it is possible that a task with a plurality of features has multiple associated classes and it can be exploited by workers with different expertise, each one focused on a different class.

Task Assignment. This component has the goal to choose the work-force of each task $t \in T$ and to schedule the related execution. In Argo+, task assignment is expected to follow a *worker-centric* criterion. This means that assignment is triggered by a worker w asking for a task to execute, which is determined "on-the-fly" by the system according to the w profile. Argo+ exploits the results of task classification and it detects the most appropriate task class k for the worker w, then the task t with the highest probability of being associated with k is assigned to w. The most appropriate task class for a worker w is the class k that best fits the w profile, thus maximizing the probability of w to successfully execute a task t in k.

Result Evaluation. For each task $t \in T$, this component has the goal to determine the final task result $\alpha(t)$ on the basis of the answers provided by workers

involved in the t execution. Different solutions can be employed by a crowdsourcing system CS for determining $\alpha(t)$. Popular solutions are based on *majority voting* mechanisms where the final task result corresponds to the answer that obtained the majority of preferences by the involved workers [5]. Alternative solutions are characterized by *statistics-based* techniques where the final task result is determined by considering the distribution of collected worker answers through the combination of one or more statistical indicators, like for example arithmetic mean, variance, or deviation [6]. As a result, the final task result $\alpha(t)$ is determined by relying on the adopted solution. Given a task t, a further goal of result evaluation is to assess the quality of each collected worker answer in light of the determined $\alpha(t)$. We say that a worker w provided a *successful contribution* to the task t when the worker answer coincides with (or is equivalent to) $\alpha(t)$ according to the employed techniques for task result evaluation. Otherwise, we say that a worker w provided an *unsuccessful contribution* to the task t. According to this, we define the *worker-task result* $\rho(w,t)$ as follows:

$$\rho(w,t) = \begin{cases} 1 \text{ if } w \textit{ provided succ. contrib.} \\ 0 \text{ otherwise} \end{cases}$$

Profile Learning. This component has the goal to update/evolve the worker profiles according to the results of executed tasks. The idea of Argo+ is to progressively learn the knowledge of a worker w by considering the features of tasks and related weights on which w provided successful contributions. For effective profile learning, update/evolution operations on the profile of the worker w should be triggered each time w executes a task t, on the basis of the quality of provided task answer. This way, learning results can affect all the subsequent task assignments for the worker w. However, we need to consider that real crowdsourcing systems are mostly characterized by *group-based task assignment* in which a task t is assigned to a set of independent workers, each one called to autonomously execute t. As a result, the final task result $\alpha(t)$ of a task t can be determined only when a certain number of worker answers are collected. In most situations, this means that, the quality of answer provided by a worker w to a task t cannot be determined immediately after the execution of t by w. A basic solution to this issue is the use of *delayed learning actions*, in which update operations to worker profiles are enforced when the final results of executed tasks become available. A more effective solution is the use of *gang-based learning actions*, in which the quality of worker answers is immediately determined after a task execution according to probability-based predictions determined by observing the behavior of similar workers (that can be considered as a "gang") [7].

4 Techniques for Argo+

In the following, we present a possible implementation of Argo+ based on the use of probabilistic topic modeling for realizing the task classification component. In particular, the proposed solution is characterized by the use of Latent Dirichlet Allocation (LDA) [4] over the task-features, which has the goal to produce a soft

task aggregation based on two discrete probability distributions, namely ϕ and θ. ϕ describes the probability distribution of task-features on classes. In particular, ϕ^k denotes the probability of each task-feature f of being associated with the kth class on the K possible classes. θ describes the probability distribution of classes on tasks. In particular, θ_t denotes the probability of the task t of belonging to each class k among the K possible classes. Finally, we denote θ_t^k as the probability of the task t to be associated with the class k. The choice of K, namely the number of classes on which LDA works for task classification, is a configuration parameter and it is discussed in Sect. 5.

The results of task classification has an impact on (i) how to select the tasks for assignment to a requesting worker (task assignment component), (ii) how to update the worker profile according to the task results (profile learning component).

4.1 Assigning Tasks to Workers

Consider a worker w and associated worker profile F_w. When w asks for a task t to execute, the probability distributions (ϕ, θ) created by task classification are exploited. Through ϕ, Argo+ calculates the maximum a posteriori estimation θ_w given the worker features F_w. This is done by using collapsed Gibbs sampling [26] to learn the latent assignment of features to classes given the observed features F_w. In particular, we repeatedly estimate the probability $p(f \mid \phi^k)$ of a feature f to be assigned to a class k and we exploit this to estimate the probability $p(k \mid w)$ of the class k to be the correct assignment for the worker w. This sampling process is repeated until convergence, so that for each class $k \in K$ we finally estimate:

$$\theta_w^k \propto \frac{\sum\limits_{f \in F_w} \omega(f)_k}{\sum\limits_{f \in F_w} \sum\limits_{j \in K} \omega(f)_j}, \tag{1}$$

where $\omega(f)_i$ denotes the weight of features of type f that have been assigned to class i. Then, from the distribution θ_w, we select the class z such that:

$$z = \arg\max_{z \in K} \theta_w^z. \tag{2}$$

Given z as the most relevant task class for the worker w, we assign to w the task t such that:

$$t = \arg\max_{t \in T} \theta_t^z. \tag{3}$$

We stress that a task t is available for assignment until the number of task executions expected by the system is reached, then it is marked as finished and it is excluded from the assignment mechanism.

Example 1. Consider to enforce Argo+ on a system with task classification based on $K = 10$. A worker w asks for a task to execute and the profile F_w is defined by the following features:

$F_w = \langle$ (web search, 0.85), (classification, 0.85), (smartphone, 0.51), (text, 0.34), ... $\rangle$

Starting from F_w, we exploit Eq. 1 in order to classify the worker w with respect to the classes K. The resulting distribution θ_w is:

K	1	2	3	4	5	6	7	8	9	10
θ_w	0.07	0.57	0.02	0.08	0.06	0.02	0.02	0.04	0.02	0.07

From θ_w, we exploit Eq. 2 to select the most relevant class for the worker profile, that is $k = 2$. The top-3 features associated with $k = 2$ in ϕ^2 are: classification, tweets, and web search, which motivates the relevance of the class with respect to the worker profile F_w. Given the class, it is now possible to exploit Eq. 3 in order to select a task t for worker execution. The features F_t of the task selected for assignment to w are:

$$F_w = \langle \text{(web search, 1.0), (classification, 1.0), (smartphone, 1.0)} \rangle$$

4.2 Learning Worker Profiles

For updating a worker profile, Argo+ relies on learning techniques inspired to the *Rocchio relevance feedback* [21]. When a worker w executes a task t, we associate the worker w with a new set of features $F'_w = F_w \cup F_t$. We denote $\omega(f)^w$ as the weight of the feature f in F_w (possibly being 0 if f was not in F_w) and $\omega(f)^t$ the weight of feature f in F_t (possibly being 0 if f was not in F_t). Then, the new weight $\omega(f)'$ for each feature in F'_w is updated as follows:

$$\omega(f)' = \delta \cdot \omega(f)^w + (1 - \delta) \cdot \theta_t^z \cdot \rho(w, t) \cdot \omega(f)^t, \tag{4}$$

where δ is a dumping factor in [0, 1] that determines how much of the original weight of the profile features contributes to the new weight, and z is the class chosen for the task assignment. The idea behind profile update is that when a worker profile feature is not included in the task features, its weight is reduced by a factor δ. In the other case, the new profile feature weight $\omega(f)'$ is computed as the weighted sum between the previous profile feature weight $\omega(f)^w$ and the task feature weight $\omega(f)^t$, which contribution is proportional to the relevance θ_t^z of the task t in the class z. The task feature weight $\omega(f)^t$ is forced to be equal to 0 when the worker does not provide a successful contribution on the task (resulting in a reduction of the corresponding profile feature weight).

Example 2. Consider the task assignment of Example 1. The worker w executed t and $\rho(w, t) = 1$. We update F_w by applying Eq. 4. The updated worker profile F'_w is the following (the class-task relevance $\theta_t^2 = 0.77$):

$F_w = \langle$ (web search, 0.78), (classification, 0.78), (smartphone, 0.74), (text, 0.03), ... $\rangle$

We note that the three features of t affects the worker profile by changing the relative feature weights. Features web search and classification remain the most relevant, but the weight of smartphone that is a feature of t is increased. On the opposite, the feature text of F_w becomes less relevant in the new worker profile, due to the fact that it is not part of the task feature set F_t. After the profile update, Argo+ will exploit the new worker profile for the subsequent task assignments to w.

5 Evaluation of Learning-Based Task Assignment in Argo+

For evaluation purposes, we rely on the Argo+ implementation presented in Sect. 4 and we consider two different experiments characterized by different task datasets called Amt-dts and Argo-dts, respectively.

Amt-dts contains the results of a third-party crowdsourcing campaign run on Amazon Mechanical Turk and presented in [25]. In Amt-dts, tasks are assigned to workers according to a motivation-based criterion by relying on the intuition that the quality of worker contributions can be improved when tasks are assigned to workers according to their motivation to execute tasks. Through Amt-dts, we consider a third-party dataset with the aim to compare Argo+ against a well-known crowdsourcing system equipped with a task assignment mechanism characterized by totally-different design principles.

Argo-dts, contains the results of a crowdsourcing campaign run through our Argo crowdsourcing system [5]. In Argo-dts, tasks are assigned to workers according to a trustworthiness-based mechanism, in which workers that demonstrate high reliability are rewarded and involved in tasks where commitment is hard to obtain (i.e., complex tasks). Through Argo-dts, the goal is to have a baseline comparison for observing the behavior of Argo+ against a basic version of expertise-based task assignment mechanism.

Our experimental evaluation is organized in two different experiments based on Amt-dts and Argo-dts, respectively. In both the experiments, the task answers contained in the datasets are considered as a baseline for comparison against Argo+ and the goal is to assess whether Argo+ succeeds in improving the task assignments with respect to the considered baseline.

In the following, we first present the experimental setup and associated evaluation metrics, then we discuss the obtained results in the two experiments with Amt-dts and Argo-dts, respectively.

5.1 Experimental Setup and Evaluation Metrics

Both Amt-dts and Argo-dts consist in a set of task answers collected from crowd workers on a given crowdsourcing campaign. Multiple answers are provided for each task by different workers and each worker is involved in the execution of a variable number of tasks.

Given a crowdsourcing execution over a given set of tasks T, we call **stream of answers** $\mathcal{S} = \{a_1, \ldots, a_n\}$ a sequence of worker task answers where $a = \langle w, t, \rho(w,t), r \rangle$ denotes that the worker w executed the task $t \in T$ by providing the worker-task result $\rho(w,t)$. The r value is the *request timestamp* denoting when the worker submitted the task answer and completed the task execution. According to Sect. 3, the worker-task result $\rho(w,t)$ is specified to distinguish the worker answers that represent a successful contribution from the others. In a stream of answers, given a worker w, it is possible to retrace the time-sequence of tasks executed by w which coincides with the task assignments received by w during time.

In both the experiments, the considered dataset, namely Amt-dts or Argo-dts, represents a stream of answers considered as a baseline for comparison against Argo+. Since the set of tasks concretely executed by each worker cannot be changed, our experiments are based on the idea to post-analyze the stream of answers in the baseline and to change the assignment sequence of tasks to workers according to the assignment schedule determined by Argo+. We aim at verifying whether the task assignment of Argo+ succeeds in capturing the worker profile and in improving the time-sequence of executed tasks, so that the tasks successfully executed by a worker w are assigned to w before than other tasks.

Given a stream of answers $\mathcal{S}$, we call $\sigma(r, \mathcal{S})$ the *success rate* of $\mathcal{S}$ at the request timestamp r and it is defined as follows:

$$\sigma(r, \mathcal{S}) = \frac{1}{r} \sum_{i=1}^{r} \rho(w, t)^i$$

where $\rho(w, t)^i$ is the worker-task result of the ith task execution at the request timestamp r in the stream of answers $\mathcal{S}$. In the experiments, we compare two different streams of answers. One is the baseline stream (i.e., Amt-dts or Argo-dts) and one is the stream of answers of Argo+.

To this end, given two different streams of answers $\mathcal{S}_1$ and $\mathcal{S}_2$, we call *increment value* $I_r(\mathcal{S}_1, \mathcal{S}_2)$ the ratio between the success rate of $\mathcal{S}_1$ and the success rate of $\mathcal{S}_2$ at the timestamp r. The increment value $I_r(\mathcal{S}_1, \mathcal{S}_2)$ is defined as follows:

$$I_r(\mathcal{S}_1, \mathcal{S}_2) = \frac{\sigma(r, \mathcal{S}_1)}{\sigma(r, \mathcal{S}_2)}$$

Considering a stream of answers $\mathcal{S}$, we call *assignment performance* $\sigma_R^{\mathcal{S}}$ the comprehensive success rate of $\mathcal{S}$ defined as follows:

$$\sigma_R^{\mathcal{S}} = \int_1^R \sigma(r, \mathcal{S}) dr$$

where R the overall number of successfully executed tasks in the stream of answers $\mathcal{S}$, namely R is the sum of all the $\rho(w, t) = 1$ in $\mathcal{S}$.

5.2 Results on the Amt-dts Dataset

The Amt-dts dataset contains 707 crowd answers about 22 different kinds of tasks, each one associated with a specific set of tags/keywords taken from a set of 39 thematic tags. The task keywords have been exploited for setting up the task features F_t expected by Argo+. In particular, for each keyword associated with a task t in Amt-dts, a corresponding task feature f is created in Argo+ for the task t with a corresponding weight $\omega(f) = 1$. Moreover, 23 workers are involved in the execution of dataset tasks, each one associated with a static set of featuring keywords. The same set of thematic tags is exploited in Amt-dts for describing both tasks and workers. In the experiment, two different Argo+ configurations are

considered. One configuration with a flat worker profile (called Argo+noprofile) where $\omega(f) = 0$ is initially defined for each thematic tag (i.e., worker-feature), and one configuration with a custom worker profile (called Argo+profile) where $\omega(f) = 1$ if the tag/worker-feature is associated with the worker in Amt-dts.

In the experiment, Argo+ has been configured with $K = 10$ classes for task classification and a dumping factor $\delta = 0.3$ for worker profile learning. We note that choosing a high value of K produces classes with few and (usually) precise associated tasks, but the presence of few tasks per class negatively affects the learning mechanism. For instance, consider to choose K so that it corresponds to the number of available tasks. This way, each class is associated with very few tasks (probably just one per class). As a result, once that the task of a class k is assigned to a worker w, learning that w is capable to successfully execute the tasks of k is not useful since the class k is empty for w and subsequent tasks need to be picked up from other classes, thus enforcing a task assignment mechanism that is equivalent to a random choice. On the opposite, choosing a low value of K produces classes with many associated tasks. Given a class k, there are tasks t that are associated with k through a high probability value θ_t^k, but there are also a (usually long) tail of tasks with low probability values. This means that a worker w with a profile fitting the class k is satisfied of the initial tasks taken from k, but subsequent assignments risk to be inappropriate due to the fact that tasks are not so relevant for the class k (i.e., low probability value θ_t^k). In our experiments, the choice of $K = 10$ has been determined experimentally by exploiting *perplexity* which measures the ability of a model to generalize to unseen data [3]. In the experiment, we run the LDA algorithm considering a variable number of K and we calculated the corresponding perplexity value for each execution. Perplexity decreases as K increases and we decided to set K to the minimum value for which perplexity reaches a stable value (i.e., $K = 10$).

The comparison of the baseline Amt-dts ($\mathcal{S}_1$) against Argo+noprofile ($\mathcal{S}_2$) and Argo+profile ($\mathcal{S}_3$) is performed by comparing (i) the success rate $\sigma(r, \mathcal{S}_1)$, $\sigma(r, \mathcal{S}_2)$, and $\sigma(r, \mathcal{S}_3)$ (Fig. 2(a)), and (ii) the increment value $I(\mathcal{S}_2, \mathcal{S}_1)$ and $I(\mathcal{S}_3, \mathcal{S}_1)$ (Fig. 2(b)). We observe that both Argo+noProfile and Argo+Profile succeed in improving the success rate of Amt-dts, since successfully executed tasks are assigned to workers before than others in most cases. For the first 150 requests, the success rate of Argo+noProfile is around 20% better than Amt-dts. It is also interesting to note that at the very beginning of the system execution ($r < 10$) the behavior of Argo+noProfile and Argo+Profile is characterized by an unstable trend. We believe that this behavior is due to the fact that learning has insufficient information for recognizing the appropriate task class for each worker. However, Argo+ quickly learns the worker profile ($r \geq 10$) and this has a positive impact on the assignment of subsequent tasks. The performance of Argo+noProfile becomes similar to Amt-dts after the 300th worker request. This is due to the fact that Argo+ first selects tasks that are highly relevant for the worker profile, but subsequent assignments are about residual tasks of the K classes on which the relevance for the worker profile is weaker.

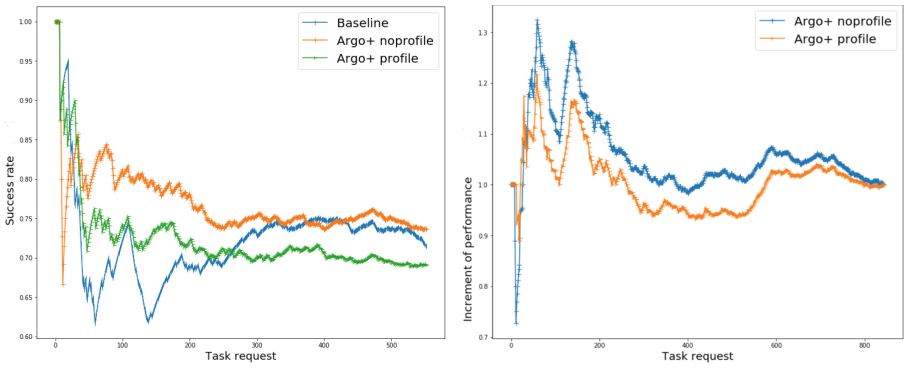


Fig. 2. (a) the success rate on executed tasks, (b) the increment value of Argo+ with respect to Amt-dts

Finally, we compare Amt-dts and Argo+ through the assignment performance measure σ_R and we obtain that $\sigma_R^{S_1} = 399.59$, $\sigma_R^{S_2} = 424.66$, and $\sigma_R^{S_3} = 399.61$. As a result, we observe that the use of a flat initialization of worker profile provides the best performance on the three considered streams of answers (see also Fig. 2(b) on the increment value). This confirms the intuition behind the use of flat profiles which argues that the auto-evaluation of worker knowledge is usually misplaced with respect to the real worker expertise, and thus unreliable and sometimes damaging for the performance of the crowdsourcing system.

5.3 Results on the Argo-dts Dataset

The Argo-dts dataset contains 14,016 crowd answers about 1,507 tasks on paintings. The Argo-dts dataset has been collected during a crowdsourcing campaign between November and December 2017 which involved 367 students from the Faculty of Arts and Literature at the University of Milan. The painting tasks are about paintings from 56 different authors spanning from the 13th century to the 20th century. For each task, the worker is asked to examine a painting and to choose the correct author among a set of six possible painters. For painting, both task and worker features are taken from Wikidata[1] and they include the name of the author, the year, and the Wikidata thematic categories available for a painting. An example of task and worker answer is given in Fig. 3. Two different configurations of Argo+ are considered. One called Argo+noprofile is characterized by a flat worker profile where $\omega(f) = 0$ is initially defined for each feature (i.e., worker-feature) taken from Wikidata. One called Argo+profile is characterized by a custom worker profile where $\omega(f) = 1$ for each feature on which the worker has declared an expertise. A self-evaluation questionnaire has been submitted to workers about knowledge of painters and different periods in the art history

[1] https://www.wikidata.org.

for collecting the perceived worker expertise before starting the crowdsourcing activities. In the experiment, Argo+ has been configured with a number of classes $K = 30$ for task classification and a dumping factor $\delta = 0.3$ for worker profile learning. The number K has been determined through perplexity by following the same approach discussed in the Amt-dts experiment.

WORKER OPTIONS

- ◯ Raffaello Sanzio
- ◯ Gustav Klimt
- ◯ Piero della Francesca
- ◯ Francisco Goya
- ◯ Giotto
- ◯ Michelangelo Buonarroti

TASK FEATURES

Raffaello Sanzio
1516
High Renaissance
Portrait paintings of cardinals

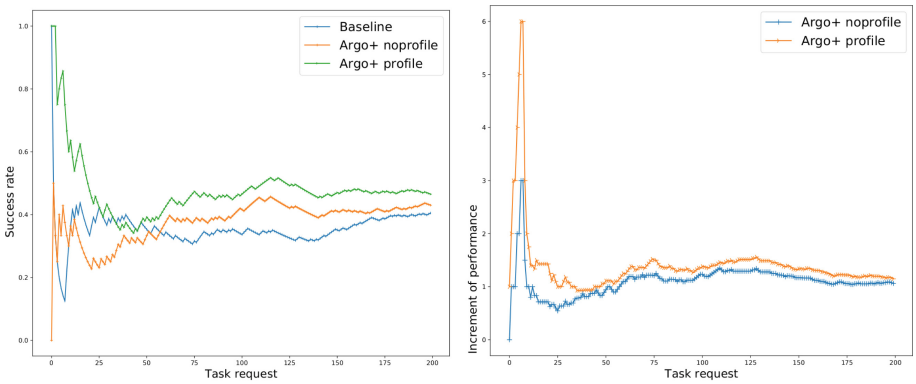
EXAMPLE OF WORKER ANSWER

{ "gold_answer" : "Q5597",
"argo_answer" : "Raffaello Sanzio",
"worker_answer_id" : "Q5432",
"worker_answer" : "Francisco Goya",
"task_id" : 1102,
"answer_timestamp" : 2017-11-13T14:42:19,
"worker_id" : 527,
"task_refused" : false }

Fig. 3. Example of painting task in Argo-dts with associated task features and an example of (wrong) worker answer

The comparison of the baseline Argo-dts ($\mathcal{S}_1$) against Argo+noprofile ($\mathcal{S}_2$) and Argo+profile ($\mathcal{S}_3$) is shown for the first 200 tasks requests in Fig. 4(a) and (b), respectively. The Argo-dts experiment confirms that both Argo+noProfile and Argo+Profile succeed in significantly improving the success rate of the baseline. In comparison with the previous Amt-dts, this experiment with Argo-dts show that Argo+noProfile and Argo+Profile follow a very similar trend. This means that, in Argo-dts, the initialization of a custom worker profile does not provide a relevant impact, neither positive or negative, on the success rate of executed tasks. Similarly to the experiment with Amt-dts, it is interesting to note that an unstable trend on the success rate occurs at the beginning of task assignment. In this case, the instability behavior of the two Argo+ streams can be observed within an initial window-frame of around 30–50 tasks. This value is higher than the one observed in Amt-dts and (more or less) coincides with the number K of classes used for task classification (i.e., $K = 30$ in Argo-dts). We argue that this result highly depends on the use of a learning mechanism inspired to the Rocchio relevance feedback, in which the learned worker expertise is immediately exploited for subsequent task assignments. As a result, apart from the perplexity considerations, we argue that the use of a high number of classes for task classification is recommended in large crowdsourcing campaigns. This way, the initial window-frame in which tasks of different classes are assigned to each worker is enlarged and the learning mechanism can discover more worker expertnesses before starting focused assignments based on learned knowledge.

According to the increment values in Fig. 4(b), we note that Argo+Profile provides a slightly better performance than Argo+noProfile. However, the behavior of the two Argo+ configurations is very similar in comparison with the baseline. We argue that this result is due to the training information received by the involved workers before starting the crowdsourcing activities. In particular, workers were recommended to carefully fill out the self-evaluation questionnaire by providing fair degrees about their expertise in Arts and paintings. As a result, we have



(a) (b)

Fig. 4. (a) the success rate on executed tasks, (b) the increment value of Argo+ with respect to Argo-dts

that the use of a trustworthy crowd has a positive impact on the performance of the task assignment mechanism. However, the degree of trustworthiness of a crowd cannot be determined a-priori, before the crowdsourcing activities. Thus, we confirm that the adoption of a flat profile initialization is a more reliable solution that provides the better performance in the general case (see the results of the Amt-dts experiment).

6 Concluding Remarks

In this paper, we presented the Argo+ framework and a related techniques for crowdsourcing task assignment with online learning of worker profiles. An implementation based on techniques inspired to topic modeling and Rocchio relevance feedback is illustrated. Experimental results on a third-party dataset and a real crowdsourcing campaign show promising results by increasing the number of successfully executed tasks in a considered time frame.

Future research activities are devoted to extend the Argo+ framework to consider/support human skills, such as *originality*, *perceptual speed*, and *deductive reasoning*, in addition to knowledge expertise when modeling task features and worker profiles. The possible use of alternative techniques with respect to LDA and Rocchio-inspired learning mechanism will be also considered in future work. In addition, ongoing work are focused on worker profile management over different crowdsourcing campaigns. The idea is that the profile of a worker is saved when a crowdsourcing campaign is terminated, so that it can be eventually exploited in subsequent campaigns. This way, the learned worker experience contributes to positively affect task assignments on startup of crowdsourcing campaigns subsequently joined by a worker. This intuition goes in the direction to consider the crowd as a *permanent* component of a crowdsourcing system where the workforce to involve in a campaign can be dynamically composed

according to the expertise degree of each worker on the tasks to execute. As a result, like in a real job-marketplace, it becomes possible that workers can negotiate their participation to crowdsourcing activities based on the contribution they can provide in terms of (profile-certified) expertise.

References

1. Alsayasneh, M., et al.: Personalized and diverse task composition in crowdsourcing. IEEE Trans. Knowl. Data Eng. **30**(1), 128–141 (2018)
2. Amer-Yahia, S., Roy, S.B.: Human factors in crowdsourcing. PVLDB **9**(13), 1615–1618 (2016)
3. Arun, R., Suresh, V., Veni Madhavan, C.E., Narasimha Murthy, M.N.: On finding the natural number of topics with latent Dirichlet allocation: some observations. In: Zaki, M.J., Yu, J.X., Ravindran, B., Pudi, V. (eds.) PAKDD 2010. LNCS (LNAI), vol. 6118, pp. 391–402. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-13657-3_43
4. Blei, D.M., Ng, A.Y., Jordan, M.I.: Latent Dirichlet allocation. J. Mach. Learn. Res. **3**(Jan), 993–1022 (2003)
5. Castano, S., Ferrara, A., Genta, L., Montanelli, S.: Combining crowd consensus and user trustworthiness for managing collective tasks. Futur. Gener. Comput. Syst. **54**, 378–388 (2016)
6. Castano, S., Ferrara, A., Montanelli, S.: A multi-dimensional approach to crowd-consensus modeling and evaluation. In: Johannesson, P., Lee, M.L., Liddle, S.W., Opdahl, A.L., López, Ó.P. (eds.) ER 2015. LNCS, vol. 9381, pp. 424–431. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-25264-3_31
7. Cesa-Bianchi, N., Gentile, C., Zappella, G.: A gang of bandits. In: Proceedings of the 27th Internatioanl Conference on Neural Information Processing Systems, Lake Tahoe, Nevada, USA, pp. 737–745 (2013)
8. Difallah, D.E., Demartini, G., Cudré-Mauroux, P.: Pick-a-crowd: tell me what you like, and I'll tell you what to do. In: Proceedings of the 22nd WWW International Conference, Rio de Janeiro, Brazil (2013)
9. Fan, J., Li, G., Ooi, B.C., Tan, K.l., Feng, J.: iCrowd: an adaptive crowdsourcing framework. In: Proceedings of the ACM SIGMOD International Conference on Management of Data, pp. 1015–1030. ACM, Melbourne (2015)
10. Gadiraju, U., Fetahu, B., Kawase, R.: Training workers for improving performance in crowdsourcing microtasks. In: Conole, G., Klobučar, T., Rensing, C., Konert, J., Lavoué, É. (eds.) EC-TEL 2015. LNCS, vol. 9307, pp. 100–114. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-24258-3_8
11. Goel, G., Nikzad, A., Singla, A.: Allocating tasks to workers with matching constraints: truthful mechanisms for crowdsourcing markets. In: Proceedings of the 23rd WWW International Conference, Seoul, Korea (2014)
12. Goncalves, J., Feldman, M., Hu, S., Kostakos, V., Bernstein, A.: Task routing and assignment in crowdsourcing based on cognitive abilities. In: Proceedings of the 26th WWW International Conference, Perth, Australia (2017)
13. Hassan, U., Curry, E.: A capability requirements approach for predicting worker performance in crowdsourcing. In: Proceedings of the 9th International Conference on Collaborate Computing, Austin, Texas, USA (2013)
14. Jain, A., Sarma, A.D., Parameswaran, A., Widom, J.: Understanding workers, developing effective tasks, and enhancing marketplace dynamics: a study of a large crowdsourcing marketplace. Proc. VLDB Endow. **10**(7), 829–840 (2017)

15. Jain, S., Narayanaswamy, B., Narahari, Y.: A multiarmed bandit incentive mechanism for crowdsourcing demand response in smart grids. In: Proceedings of the 28th AAAI Conference on Artificial Intelligence, Qulébec, Canada, pp. 721–727 (2014)
16. Karger, D.R., Oh, S., Shah, D.: Budget-optimal task allocation for reliable crowdsourcing systems. Oper. Res. **62**(1), 1–24 (2014)
17. Kazai, G., Kamps, J., Milic-Frayling, N.: Worker types and personality traits in crowdsourcing relevance labels. In: Proceedings of the 20th CIKM, Glasgow, Scotland, UK (2011)
18. Khazankin, R., Psaier, H., Schall, D., Dustdar, S.: QoS-based task scheduling in crowdsourcing environments. In: Kappel, G., Maamar, Z., Motahari-Nezhad, H.R. (eds.) ICSOC 2011. LNCS, vol. 7084, pp. 297–311. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-25535-9_20
19. Li, G., Wang, J., Zheng, Y., Franklin, M.J.: Crowdsourced data management: a survey. IEEE Trans. Knowl. Data Eng. **28**(9), 2296–2319 (2016)
20. Liu, Y., Liu, M.: An online learning approach to improving the quality of crowdsourcing. IEEE Trans. Netw. **25**(4), 2166–2179 (2017)
21. Manning, C.D., Raghavan, P., Schütze, H.: Introduction to Information Retrieval, vol. 1. Cambridge University Press, Cambridge (2008)
22. Mo, K., Zhong, E., Yang, Q.: Cross-task crowdsourcing. In: Proceedings of the 19th ACM SIGKDD International Conference, Chicago, Illinois, USA (2013)
23. Müller, E., Günnemann, S., Färber, I., Seidl, T.: Discovering multiple clustering solutions: grouping objects in different views of the data. In: Proceedings of the 28th IEEE ICDE International Conference, Washington, DC, USA, pp. 1207–1210 (2012)
24. Organisciak, P., Teevan, J., Dumais, S.T., Miller, R., Kalai, A.T.: A crowd of your own: crowdsourcing for on-demand personalization. In: Proceedings of the 2nd AAAI HCOMP, Pittsburgh, USA (2014)
25. Pilourdault, J., Amer-Yahia, S., Lee, D., Roy, S.B.: Motivation-aware task assignment in crowdsourcing. In: Proceedings of the 20th EDBT International Conference, Venice, Italy (2017)
26. Porteous, I., Newman, D., Ihler, A., Asuncion, A., Smyth, P., Welling, M.: Fast collapsed gibbs sampling for latent Dirichlet allocation. In: Proceedings of the 14th ACM SIGKDD International Conference, pp. 569–577 (2008)
27. Simpson, E., Roberts, S.: Bayesian methods for intelligent task assignment in crowdsourcing systems. In: Guy, T., Kárný, M., Wolpert, D. (eds.) Decision Making: Uncertainty, Imperfection, Deliberation and Scalability. SCI, vol. 538, pp. 1–32. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-15144-1_1
28. Tran-Thanh, L., Stein, S., Rogers, A., Jennings, N.R.: Efficient crowdsourcing of unknown experts using bounded multi-armed bandits. Artif. Intell. **214**, 89–111 (2014)
29. Tranquillini, S., Daniel, F., Kucherbaev, P., Casati, F.: Modeling, enacting, and integrating custom crowdsourcing processes. ACM Trans. Web **9**(2), 7:1–7:43 (2015)
30. Zheng, Y., Li, G., Li, Y., Shan, C., Cheng, R.: Truth inference in crowdsourcing: is the problem solved? Proc. VLDB Endow. **10**(5), 541–552 (2017)