# Introducing Online Profile Learning in Crowdsourcing Task Routing
## Discussion Paper

Silvana Castano[1], Alfio Ferrara[1], and Stefano Montanelli[1]

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

**Abstract.** In this paper, we present an initial implementation with experimental results of online profile learning in Argo+, a framework for crowdsourcing task routing characterized by i) feature-based representation of both tasks and workers, and ii) learning techniques inspired to *Rocchio relevance feedback* for prediction of the most appropriate task to execute by a given worker.

## 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. 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 topics and skill requirements not compliant with her/his attitudes. As a result, the capability to effectively discover and represent the profile of engaged crowd workers is becoming a strategic asset of emerging crowdsourcing marketplaces. The goal is to selectively choose a qualified and motivated crowd to recruit/involve in a given campaign according to the required knowledge/abilities based on the features of the tasks to execute. In this direction, the use of machine-learning techniques in crowdsourcing applications is being appearing in the recent literature for mining emerging worker skills from the analysis of executed tasks [6, 7, 12]. Online learning techniques based on (multi-armed) bandits algorithms have been also proposed for improving the quality of crowdsourcing results [8, 10, 14]. Online learning of worker skills is also concerned with the so-called *task routing* issue, that is the capability of a crowdsourcing system to assign a task to a worker

SEBD 2018, June 24-27, 2018, Castellaneta Marina, Italy. 

based on the expectations to obtain a successful contribution from her/his answer. In the literature, popular solutions are characterized by the idea to rely on human factors for addressing task routing (e.g., [1, 9]).

In this paper, we present an initial implementation of online profile learning in Argo+, a framework for crowdsourcing task routing characterized by i) feature-based representation of both tasks and workers, and ii) learning techniques inspired to *Rocchio relevance feedback* for prediction of the most appropriate task to execute by a given worker. A detailed description of the Argo+ framework is provided in [5]. In the following, we focus on discussing the preliminary results obtained on a real crowdsourcing campaign, by comparing the performance of Argo+ against a baseline with conventional task routing techniques.

The paper il organized as follows. Section 2 presents the basic elements of the proposed Argo+ implementation. Experimental results are then discussed in Section 3. Concluding remarks are finally provided in Section 4.

## 2 Learning-based task routing

A crowdsourcing campaign is characterized by a crowd of workers $W = \{w_1, \ldots, w_k\}$ involved in the execution of a set of tasks $T = \{t_1, \ldots, t_n\}$. 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 (e.g., creation, decision). A description $d_t$ represents the task request given to each worker for illustrating what is demanded to her/him in the task execution. A set of task-features $F_t$ manually associated with the task for providing 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 **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, either knowledge expertise or skill, and it is associated with a *worker-feature weight* $\omega(f)$ denoting the "degree" of expertise/ability associated with the worker.

### 2.1 Assigning tasks to workers

For enforcing task routing, Argo+ relies on a task classification procedure for aggregating the tasks $T$ to execute into $K$ classes, so that tasks with similar features $F_t$ are associated with a same class. In the proposed implementation of Argo+, probabilistic topic modeling are exploited for task classification. The choice is motivated by the need to enforce a *soft aggregation* mechanism, where a task with a plurality of features can have multiple associated classes and it can

be exploited by workers with different expertise, each one focused on a different class. In particular, the proposed solution is characterized by the use of Latent Dirichlet Allocation (LDA) [3] over the task-features, characterized by two discrete probability distributions, namely $\phi$ and $\theta$. $\phi$ describes the probability distribution of task-features on classes. In particular, $\phi^k$ denotes the probability of each task-feature $f$ of being associated with the $k$th class on the $K$ possible classes. $\theta$ describes the probability distribution of classes on tasks. In particular, $\theta_t$ denotes the probability of the task $t$ of belonging to each class $k$ among the $K$ possible classes. Finally, we denote $\theta_t^k$ 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 Section 3.

Consider a worker $w$ and associated worker profile $F_w$. When $w$ asks for a task $t$ to execute, the probability distributions $(\phi, \theta)$ created by task classification are exploited. Through $\phi$, Argo+ calculates the maximum a posteriori estimation $\theta_w$ given the worker features $F_w$. This is done by using collapsed Gibbs sampling [13] 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:

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

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

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

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

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 $K = 10$ and a set of thematic tags used as features both for tasks and workers. 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 Equation 1 in order to classify the worker $w$ with respect to the classes $K$. The resulting distribution $\theta_w$ is:

| $K$ | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| $\theta_w$ | 0.07 | 0.57 | 0.02 | 0.08 | 0.06 | 0.02 | 0.02 | 0.04 | 0.02 | 0.07 |

From $\theta_w$, we exploit Equation 2 to select the most relevant class for the worker profile, that is $k = 2$. The top-3 features associated with $k = 2$ in $\phi^2$ 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 Equation 3 in order to select a task $t$ for worker execution. The features $F_t$ of the task selected for assignment to $w$ are $F_w = \langle$ (web search, 1.0), (classification, 1.0), (smartphone, 1.0)$\rangle$

### 2.2 Learning worker profiles

Given a task $t$ executed by a worker $w$, we need to assess the quality of the provided worker answer for deciding how to update the worker profile, and thus how to enforce learning. We call $\alpha(t)$ the *final task result* determined by the crowdsourcing system. We note that different solutions can be employed 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. Alternative solutions are also possible, such as for example *statistics-based* techniques [4]. We say that a worker $w$ provided a *successful contribution* to the task $t$ when the worker answer coincides with (or is equivalent to) $\alpha(t)$. 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:

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

For updating a worker profile, Argo+ relies on learning techniques inspired to the *Rocchio relevance feedback* [11]. 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:

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

where $\gamma$ 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 $\delta$. 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 $\theta_t^z$ 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 Equation 4. The updated worker profile $F'_w$ is the following (the class-task relevance $\theta^2_t = 0.77$):

$F_w = \langle$ (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 remarkably 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 $w$.

## 3 Experimental results

For evaluation, we present some preliminary experimental results based on the comparison of the proposed Argo+ implementation against a basic task routing mechanism called from now on baseline.

### 3.1 Experiment setup

Our experiment relies on a crowdsourcing campaign (named paintings) run through the Argo crowdsourcing system (i.e., the system version implementing baseline routing) between November and December 2018. The experiment involved 367 students from the Faculty of Arts and Literature at the University of Milan, who have been asked to examine a dataset of paintings in order to choose for each painting the correct author among a choice of six possible painters. The paintings dataset is composed by 948 paintings from 56 different authors spanning from the 13th century to the 20th century. Each task has been executed by more than one worker, for a total of 8,573 executions. The fact that the paintings dataset is featured by a correct answer for each task (i.e., the correct name of the painting author) makes it possible to easily evaluate the effectiveness of the work done by each worker (i.e., successful contribution) in terms of number of correct answers given to the tasks question.

To setup the experiment for evaluating Argo+, we compare the success rate of the baseline execution of paintings against the success rate obtained through two different executions of paintings in Argo+: i) one execution with a flat worker profile (called Argo+noprofile) where $\omega(f) = 0$ is initially defined for each feature (i.e., worker-feature), and ii) one execution with a custom worker profile (called Argo+profile) where $\omega(f) = 1$ for each feature on which the worker has declared a competence. Competences declared by workers have been collected through a self evaluation questionnaire about knowledge of painters and different periods in the art history. Task and worker features have been taken from Wikidata (`https://www.wikidata.org`) and they include the name of the author, the

**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. 1.** Example of task with task feature and an example of (wrong) worker answer

year, and the Wikidata thematic categories available for a painting. An example of task and worker answer is given in Figure 1.

The goal of our experimental evaluation is to assess whether Argo+ improves the success rate with respect to baseline. In order to simulate the execution of Argo+noprofile and Argo+profile on exactly the same set of workers and answers used in baseline, our experiments are based on the idea to change the time-sequence of tasks executed by each worker in baseline according to the assignment schedule determined by Argo+. We aim at verifying whether the tasks successfully executed by a worker $w$ are assigned to $w$ before than others. Being the task answers the same for all the experiments, the overall success rate will be the same as well. However, if Argo+ performs better than baseline, we expect to execute correct tasks before. In other terms, we aim at verifying if Argo+ reaches a success rate better than baseline by taking into account the first $r$ tasks assignments. In particular, we call $r$ (*request timestamp*), the timestamp at which the crowdsourcing system receives the request for a task to execute by a worker, and we call $\sigma(r, \tau)$ the *success rate* of the system execution $\tau$ at the request timestamp $r$. A *system execution* is a stream of task answers, each one collected from a worker at a certain timestamp. In our evaluation, baseline is the reference system execution, while Argo+noprofile and Argo+profile represent alternative system executions of baseline where the time-sequence of task answers is changed according to the Argo+ routing mechanism. The success rate $\sigma(r, \tau)$ is defined as follows:

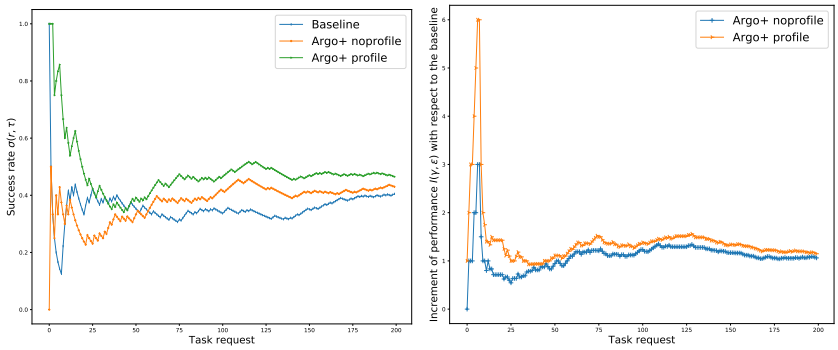
$$\text{(a) } \sigma(r,\tau) = \frac{1}{r}\sum_{i=1}^{r}\rho(w,t)^i \;\; ; \;\; \text{(b) } \sigma_R^\tau = \int_1^R \sigma(r,\tau)dr$$

where $\rho(w,t)^i$ in (a) is the worker-task result received by the system at the $i$th request timestamp in the execution $\tau$ and (b) measures the overall system performance of task routing with $R$ representing the overall number of successfully executed tasks in a system execution $\tau$ (i.e., $R$ is the sum of all the $\rho(w,t) = 1$ in $\tau$). Given two different system executions $\epsilon$ and $\gamma$, the delta value $\delta_r(\gamma, \epsilon)$ represents how much the success rate of $\gamma$ changes with respect to $\epsilon$ at time $r$. The delta value $\delta_r(\gamma, \epsilon)$ is defined as $\delta_r(\gamma, \epsilon) = \frac{\sigma(r,\gamma)}{\sigma(r,\epsilon)}$.

### 3.2 Considerations

Experiments have been performed with a number of classes $K = 30$ for task classification and a dumping factor $\gamma = 0.3$ for worker profile learning. The com-

parison of baseline against Argo+noprofile and Argo+profile on success rate $\sigma(r, \tau)$ and delta value $\delta(\gamma, \epsilon)$ are shown for the first 200 tasks requests in Figures 2(a) and 2(b), respectively. We observe that both Argo+noProfile and Argo+Profile suc-




**(a)** **(b)**

**Fig. 2.** (a) success rate on executed tasks, (b) increment of performances with respect to baseline

ceed in improving the success rate of baseline, since successfully executed tasks are assigned to workers before other tasks in most cases. For the first 200 requests, the success rate of Argo+Profile is around 20% better than baseline. It is also interesting to note that, at the very beginning of the system execution ($r < 50$), the behavior of Argo+noProfile and Argo+Profile is very unstable since learning has insufficient information for recognizing the appropriate task class for each worker. However, Argo+ quickly learns the worker profile ($r \geq 50$) and this has a positive impact on the assignment of subsequent tasks. The performance of Argo+noProfile becomes similar to baseline after the 300th worker request. This is due to the fact that Argo+ first selects tasks that are highly relevant for the worker profile, while subsequent assignments are about residual tasks of the $K$ classes for which the relevance for the worker profile is weaker.

Finally, we compare baseline and Argo+ through $\sigma_R^\tau$ and we obtain that $\sigma_R^\tau = 399.59$ for baseline, $\sigma_R^\tau = 424.66$ for Argo+Profile, and $\sigma_R^\tau = 399.61$ for Argo+noProfile. As a result, we observe that the use of a questionnaire for initializing the worker profile provides the best performance on the three considered system executions (see also Figure 2(b) on the increment value). However, after a small number of executions, the performance of the learning system without the initial set-up of the worker profile becomes similar to the one of the system execution initialized with the questionnaire. This confirms the intuition behind the use of flat profiles which argues that the auto-evaluation of worker skill/abilities could be misplaced with respect to the real worker expertise, and thus sometimes damaging the performance of the crowdsourcing system.

## 4 Concluding remarks

In this paper, we presented an implementation of profile learning techniques in Argo+ crowdsourcing framework [5] with some experimental results. Ongoing research activities are aimed i) to extend the experimentation for considering multiple kinds of tasks with different action, modality, and description, and ii) to improve the Argo+ framework to support worker profile management over different crowdsourcing campaigns with different task/worker features.

## References

1. Amer-Yahia, S., Roy, S.B.: Human Factors in Crowdsourcing. PVLDB **9**(13), 1615–1618 (2016)
2. Arun, R., Suresh, V., Madhavan, C.V., Murthy, M.N.: On Finding the Natural Number of Topics with Latent Dirichlet Allocation: Some Observations. In: Proc. of the 14th PAKDD Conference. Hyderabad, India (2010)
3. Blei, D.M., Ng, A.Y., Jordan, M.I.: Latent dirichlet allocation. Journal of machine Learning research **3**(Jan), 993–1022 (2003)
4. Castano, S., Ferrara, A., Montanelli, S.: A Multi-Dimensional Approach to Crowd-Consensus Modeling and Evaluation. In: Proc. of the 34th ER Int. Conference. Stockholm, Sweden (2015)
5. Castano, S., Ferrara, A., Montanelli, S.: A Conceptual Framework for Crowdsourcing Task Assignment with Online Profile Learning. In: Submitted to the 37th ER Int. Conference. Xi'an, China (2018)
6. Gadiraju, U., Fetahu, B., Kawase, R.: Training Workers for Improving Performance in Crowdsourcing Microtasks. In: Proc. of the 10th EC-TEL. Toledo, Spain (2015)
7. Goncalves, J., Feldman, M., Hu, S., Kostakos, V., Bernstein, A.: Task Routing and Assignment in Crowdsourcing Based on Cognitive Abilities. In: Proc. of the 26th WWW Int. Conference. Perth, Australia (2017)
8. Jain, S., Narayanaswamy, B., Narahari, Y.: A Multiarmed Bandit Incentive Mechanism for Crowdsourcing Demand Response in Smart Grids. In: Proc. of the 28th AAAI Conference on Artificial Intelligence. pp. 721–727. Qulébec, Canada (2014)
9. Karger, D.R., Oh, S., Shah, D.: Budget-Optimal Task Allocation for Reliable Crowdsourcing Systems. Oper. Res. **62**(1), 1–24 (2014)
10. Liu, Y., Liu, M.: An Online Learning Approach to Improving the Quality of Crowd-Sourcing. IEEE Transactions on Networking **25**(4), 2166–2179 (2017)
11. Manning, C.D., Raghavan, P., Schütze, H.: Introduction to Information Retrieval, vol. 1. Cambridge university press Cambridge (2008)
12. Organisciak, P., Teevan, J., Dumais, S.T., Miller, R., Kalai, A.T.: A Crowd of Your Own: Crowdsourcing for On-Demand Personalization. In: Proc. of the 2nd AAAI HCOMP. Pittsburgh, USA (2014)
13. Porteous, I., Newman, D., Ihler, A., Asuncion, A., Smyth, P., Welling, M.: Fast Collapsed Gibbs Sampling for Latent Dirichlet Allocation. In: Proc. of the 14th ACM SIGKDD Int. Conference. pp. 569–577 (2008)
14. Tran-Thanh, L., Stein, S., Rogers, A., Jennings, N.R.: Efficient Crowdsourcing of Unknown Experts using Bounded Multi-armed Bandits. Artificial Intelligence **214**, 89–111 (2014)