Adaptive Performance Optimization over Crowd Labor Channels

Saraschandra Karanam, Deepthi Chander
Koustuv Dasgupta, Vaibhav Rajan
Xerox Research Centre-India (XRCI)
Bangalore, India

L. Elisa Celis
École Polytechnique Fédérale de Lausanne (EPFL)
Lausanne, Switzerland

Abstract

We describe a system which monitors the performance of labor channels within a crowdsourcing platform in an online manner. This allows us to automatically determine if and when to switch between labor channels in order to improve overall performance of crowd tasks.

Introduction

Most crowdsourcing platforms, either implicitly or explicitly, hire different pools of workers with varying geographies, demographics, and skill sets. We call each such (potentially overlapping) pool a labor channel. The quality of work for a given task can vary widely across different channels (Dasgupta et al. 2013), yet the requestor is rarely privy to this information. Some crowdsourcing platforms provide the flexibility for the requestor to choose a specific labour channel either explicitly, or by selecting different settings (e.g., geography or worker accuracy rating), however platforms do not

- provide a characterization of the performance of these labor channels,
- automatically route tasks to appropriate labour channels based on the requirements of the requester, or
- adapt to performance changes or switch tasks between channels.

Therefore, the requestor is left to guess by trial and error the best labour channel that suits her requirements. This leads to loss of time, revenue, and quality.

In this paper, we fill this gap by introducing a system that provides functionality for the three points identified above. Such a system would be useful in at least two important scenarios:

Enterprise Crowdsourcing: Large volumes of tasks are processed on a regular basis, so there is both scope and incentive to learn and optimize over labor channels.

Platform Improvement: Offering such optimization as a feature for requesters would be a novel and useful feature for a crowdsourcing platform.

Note that it is important, in both scenarios, that the solution be adaptive. Since the crowd within a labor channel is changing (due to people joining, leaving, or acquiring skills), learning once and optimizing is not enough.

Related Work

Task assignment strategies to improve performance of crowdsourcing platforms has been studied before. Most of them require knowledge of platform internals such as worker skills and preferences (Difallah, Demartini, and Cudré-Mauroux 2013), historical worker performance (Liu et al. 2012), or availability of best workers (Khazankin, Schall, and Dustdar 2012). Even those works that do not assume knowledge of platform internals such as (Dasgupta et al. 2013; Rajan et al. 2012) overlook the fact that the performance of these platforms is an aggregate of the performances of multiple labour channels.

Our Approach

We consider an explore-exploit framework when thinking about this problem. While we wish to improve performance as much as possible by "exploiting" the best labour pool, we also wish to "explore" to make sure we detect potential better pools as and when they arise. We conduct a very simple experiment where we post tasks several times a day over several days. We compare an adaptive posting algorithm to the naïve (yet commonly used) strategy of simply posting to a single labor pool.

The Experiment

We considered a digitisation task where workers were asked to type in fields of handwritten text from a form. We used four different labor channels (DiamondTask, Neodev, Prodege & RewardingWays) on the CrowdFlower platform which allows the requester to explicitly select labor channels. We posted batches of 20 tasks on each of the four channels at six different hours of the day (02, 06, 10, 14, 18 & 22) for seven days in a week, which generated our data.

We considered optimizing three different metrics: 1) only accuracy (Acc) 2) only response time (RT), and both accuracy and response time (AccRT). Accuracy is given by $1-\frac{L}{N}$ where, L is the Levenshtein Distance between the original text field in the form and the response (i.e. minimum number

of string manipulations required to transform one string to other) and N is the length of the string. RT is the time elapsed between posting the task and retrieving the results. To normalize for outliers, we set any value below 1st quartile to 0, any value above 3rd quartile to 1 and normalized the remaining values. Our optimization function was $Acc + (1 - RT)$. We used regret against each labor channel (the difference between the performance of our switching algorithm with the algorithm that does not switch) as a measure of our algorithms efficacy. Since our algorithm is randomized, we ran it counterfactually on the above data 100 times, and report the mean regret along with the standard deviation over the trials.

We used a simple exploration/exploitation algorithm with one simple observation: not every labor pool is worth exploring. We only explore under two scenarios: 1) it was *close* to our current best, or 2) it has been a long time since we tried it. In either case, the performance of the pool may have improved and surpassed our current best. We exploit the best observed pool with probability $1 - \epsilon$ and explore a pool which satisfies the above conditions with probability ϵ. The algorithm has a short memory, and simply uses the last observed measurement as an estimate for the performance of a labor pool. The pseudocode is given in Algorithm 1. Many more sophisticated algorithms could be used, however we see below that even this simple version gives significant improvement.

Algorithm 1 : ϵ-Smart Algorithm.
Input: number of pools k, and learning parameters ϵ, γ.

$\tau = 0$
for $i \leftarrow 1$ to k **do**
  $\tau_i = i$, $\hat{\mu}_i = r \sim F_i^i$
end for
for $i \leftarrow k, k+1$ to ... **do**
  $\mu^* = max_i\{\hat{\mu}_i\}, i^* = argmax_i\{\hat{\mu}_i\}$
  for $i \leftarrow 1$ to k **do**

$$a_i = \begin{cases} 1 & \text{if } \mu_* - \hat{\mu_i} \leq \gamma\sqrt{t - \tau_i}; \\ 0 & \text{otherwise.} \end{cases}$$

  end for

$$i_t = \begin{cases} i_* & \text{with probability } 1 - \epsilon; \\ i \text{ s.t. } a_i = 1 & \text{with probability } \frac{\epsilon}{\sum_i s_i}. \end{cases}$$

end for

Results

When running our algorithm, we saw that it indeed choses to switch between the different labor channels. Moreover, despite being rather simple, it still outperforms any single labor channel. This difference is significant when we optimize for both accuracy and time, and comparable when optimizing only one of the two (see Figure 1).

We expect a more sophisticated algorithm, with better memory or more sophisticated switching to improve the results further. Developing such algorithms remains a key component of future work. Further, we wish to test larger data sets with more labor channels over longer periods of time.

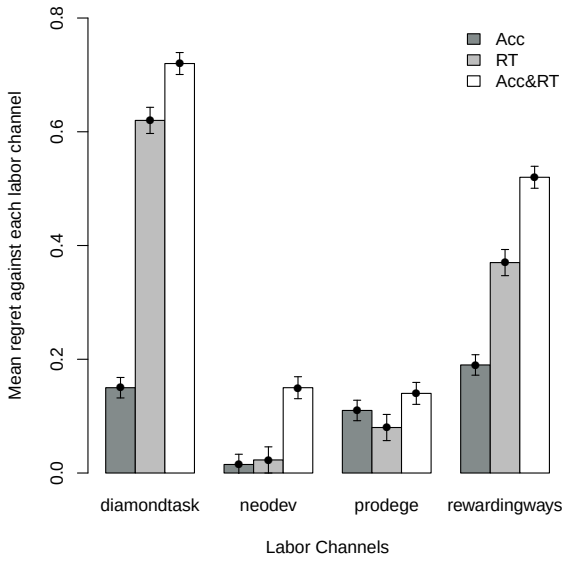

Figure 1: Performance of our algorithm compared to an algorithm that has a fixed choice of labor channel.

Conclusions

In this work, we proposed a system which opportunistically and automatically selects labour channels in such a way that the final performance can be improved. Enterprise requesters and crowdsourcing platforms could benefit significantly by leveraging their labor pools in an efficient manner. Even a simple first attempt has shown improvement, and we expect our results to only improve with more sophisticated algorithmic techniques.

References

Dasgupta, K.; Rajan, V.; Karanam, S.; Ponnavaikko, K.; Balamurugan, C.; and Piratla, N. M. 2013. Crowdutility: know the crowd that works for you. In *CHI'13 Extended Abstracts on Human Factors in Computing Systems*, 145–150. ACM.

Difallah, D. E.; Demartini, G.; and Cudré-Mauroux, P. 2013. Pick-a-crowd: tell me what you like, and i'll tell you what to do. In *Proceedings of the 22nd international conference on World Wide Web*, 367–374. International World Wide Web Conferences Steering Committee.

Khazankin, R.; Schall, D.; and Dustdar, S. 2012. Predicting qos in scheduled crowdsourcing. In *Advanced Information Systems Engineering*, 460–472. Springer.

Liu, X.; Lu, M.; Ooi, B. C.; Shen, Y.; Wu, S.; and Zhang, M. 2012. Cdas: a crowdsourcing data analytics system. *Proceedings of the VLDB Endowment* 5(10):1040–1051.

Rajan, V.; Bhattacharya, S.; Celis, L. E.; Chander, D.; Dasgupta, K.; and Karanam, S. 2012. Crowdcontrol: An online learning approach for optimal task scheduling in a dynamic crowd platform.