Luna: An Adaptive Context-Aware Crowdsourcing Framework

Christopher Jonathan
University of Minnesota
Minneapolis, MN, USA
cjonathan@cs.umn.edu

1. PROBLEM AND MOTIVATION

In recent years, crowdsourcing has been gaining a lot of popularity due to its capability of solving various computer-hard tasks including, image recognition [16], entity resolution [15], and sentiment analysis [18]. This triggers the existence of several crowdsourcing platforms, e.g., Amazon Mechanical Turk (MTurk) [11] and Upwork [14]. On such platforms, requesters submit a task with a budget to pay to workers (i.e., the crowd) to solve the task.

While many tasks can be solved by crowdsourcing efficiently and accurately, e.g., a survey; many other tasks are not fit to be solved by the general crowd as it requires some expertise of the workers in order to provide the accurate result. We call these tasks *context-aware* tasks as they require the workers to be familiar with the context of the tasks in order to provide high-quality answers. An example of a context-aware task is a task to find a location of a given object, often referred as geotagging, which has a context of *location*. Asking workers who are located close to the location of the object will result in a much better accuracy as people know their nearby area better. To confirm this, we run an MTurk experiment by asking the workers to find the location of images that we already know their locations. By ignoring the *location* context of the task, we recruit workers randomly, thus, results in a low accuracy as shown by Random in Table 1. By considering the *location* context, we recruit workers located at the same state of each image and are able to receive a much higher accuracy as shown by Expert.

The biggest challenge faced by existing crowdsourcing frameworks is that the context of a task is often unavailable. Had we know the context of the task in advance, it would have been easy to get the corresponding workers from their profiles. In the geolocation example, it is easy to find workers who live nearby the location of the images if we know the location of the images in advance. Furthermore, each context has different resolution which by asking workers who are familiar with a finer-grained context of the

	Country	Region	State	City
Random	0.54	0.39	0.28	0.22
Expert	1.0	1.0	1.0	0.95

Table 1: Percentage of Workers Accuracy

task will result in a better accuracy. For example, in the geolocation example, asking workers who are familiar with the state where the images are located is preferred to asking workers who are familiar with the country where the images are located.

These challenges motivate us to provide a better crowdsourcing framework. In this paper, we introduce a novel crowdsourcing technique called *adaptive crowdsourcing* to extract the context of a given task efficiently. The main idea of adaptive crowdsourcing is that we do not use the whole budget that is given by the requester at once. Instead, we split the crowdsourcing task into multiple iterations where the goal of each iteration is to find a finer-grained context for the task and to recruit workers within this refined context to do the task. Using the geolocation example, the first iteration of the geolocation task is used to find the country of the images by sending the tasks to workers from around the world. Then, we recruit workers all within the resulting country to find the state of the images, and so on. To realize this, we develop a novel crowdsourcing system, called Luna, that is able to adaptively predict the context of a given task and choose expert workers to solve the task, thus, results in a more accurate result.

2. BACKGROUND AND RELATED WORK

Crowdsourcing Tasks and Frameworks. In recent years, crowdsourcing has been extensively studied within the Database community and mainly used to solve computer-hard tasks including, sorts and joins in [9], skyline in [7], labeling in [4], data mining in [1], search in [16], macrotask processing in [3], as well as integrating crowdsourcing into the query plan of relational database systems in [2, 10, 12]. In this work, we do not provide a new type of computer-hard task that can be solved by using crowdsourcing. Instead, we provide a novel context-aware adaptive crowdsourcing framework where existing crowdsourcing tasks can be executed efficiently to output more accurate results without additional budget.

Quality-Control in Crowdsourcing. There are numerous research efforts done to increase the quality of a crowdsourcing result including, providing a worker model to match the worker to any given tasks in [5, 17], providing a worker elimination method to eliminate low-quality workers or spammers in [6, 8, 13], and aggregating multiple workers' answers in [6, 18]. Worker modeling techniques assume that the context of the task is known in advance which is not the case in our case. We see that low-quality workers elimination as an orthogonal work to our work and we can use these methods to eliminate low-quality workers or spammers.

*This work is partially supported by the National Science Foundation, USA, under Grants IIS-1525953, CNS-1512877, IIS-1218168, and IIS-0952977.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).

SIGMOD '17 Student Research Competition, May 14-19 2017, Chicago, IL, USA

© 2017 Copyright held by the owner/author(s).

ACM ISBN 978-1-4503-4199-8/17/05.

DOI: <http://dx.doi.org/10.1145/3055167.3055171>

Figure 1: Luna System Architecture

3. APPROACH AND UNIQUENESS

Figure 1 provides the system architecture of Luna. In Luna, there are three different users, namely the requester, the admin who manages the system, and the workers; and six different modules, described briefly below:

Index Creation Module. This module allows admin user to create a new context. Once executed, Luna creates an index for the context, which can be any space partitioning multi-resolution index structures, and index all workers from the workers pool into the newly defined context. Each index cell contains a pointer to the workers that satisfy the context value defined by the cell. An example of a context is *location*. The top level cell of the index covers the whole world and each cell of the second level of the index covers each country in the world, and so on.

New Worker Management Module. This module registers new workers into the system by storing their information in the workers pool and update existing indexes with the new workers information.

Task Processor Module. This module receives the task and the budget B from the requester. Then, it has two main tasks. First, it uses existing techniques, e.g., topic extraction technique, computer vision technique, and knowledge based software, to extract a hint about the context of the task. Second, it calculates the number of iterations, say H , as well as the number of workers that Luna can use in each iteration, say $N = B/H$. Then, it sends this information to the worker selection module. The value of H may differ depending on the setting of the system which can be tuned by the admin. The module also maintain the status of each task and returns the completed task back to the requester.

Worker Selection Module. The goal of this module is to find the N workers who are suitable for the task. As Luna is an adaptive crowdsourcing framework which runs a single task in multiple iterations, this module will be executed $H \times$ for each task. If this is the first iteration of a task, the module will uniformly assign N workers across different context based on the hint given by the task processor module. Otherwise, the module takes the advantage of the multi-resolution index to find the N workers from the children context of the refined context to send the task. In the Manchester United example, the module selects N workers from different context, e.g., sports, music, and theater, in the first iteration. In the next iteration, given a context of sports which inferred from the crowd answers, the module selects N workers from different context within sports, e.g., soccer, tennis, and baseball.

Crowdsourcing API Module. This module receives the workers information from the worker selection module and send the task to

Figure 2: Luna Accuracies

them. Then, it waits for the result from the workers and send the results to the result calculation module once it is available.

Result Calculation Module. The goal of this module is to infer the results from the crowdsourcing job and decides on the decision for the task. If it is not the last iteration, the module calculates the most promising context to be checked next based on the result and sends this information to the worker selection module to continue on the next iteration. In the Manchester United example, the module might see that most workers within sports context have more agreement on their results. Thus, it chooses sports as the resulting context of the iteration. If it is the last iteration of the task, the module will calculate the answer for the task and send it back to the task processor module to be returned to the requester.

In order to increase the quality of the result, Luna has different optimizations that are generally geared towards ensuring that the recruited workers are more expert than the ones recruited without the optimizations, which will end up in increasing the overall accuracy of the final result. Unfortunately, due to space limitation, we omit the details of these optimizations.

4. RESULTS AND CONTRIBUTIONS

We use our running example of the object geolocation task to show the effectiveness of our approach. We compare the performance of Luna without any optimizations as well as Luna equipped with its optimizations with the two basic crowdsourcing techniques, namely the random and the uniform workers assignment approach. Random approach chooses the workers randomly regardless of their locations while Uniform approach chooses workers uniformly based on their locations.

We run our experiment by running geolocation tasks for four different images using Amazon Mechanical Turk (MTurk) with 360 workers. Two of the images depict a popular object, such as the Statue of Liberty, while the other two images depict a not popular object, such as the University of Minnesota campus. We ask the workers to provide the exact location of the images and we calculate the accuracy of the workers by calculating the distance between the resulting location to the exact location of the image. We use distance above 42 miles to be treated as inaccurate and the accuracy is calculated linearly based on the resulting distance.

Figure 2 shows the average accuracy of each technique in geolocating the given images. As we can see from the figure, even when the object is popular, Luna is able to achieve 95% of accuracy compared to 80% of accuracy that are achieved by the two existing approaches. The reason is that Luna is able adaptively find the workers who live in the area of the image who know the exact location of the image better compared to other workers. While Luna is able to maintain its accuracy in asking not popular object, both random and uniform approaches achieve 0% accuracy in for this task as most workers do not know the exact location of the image. This experiment shows that Luna is able to adaptively recruit expert workers and by being able to retrieve expert workers to solve the task, we will get more accurate results.

5. REFERENCES

- [1] Y. Amsterdamer, Y. Grossman, T. Milo, and P. Senellart. Crowd mining. In *SIGMOD*, pages 241–252, 2013.
- [2] M. J. Franklin, D. Kossmann, T. Kraska, S. Ramesh, and R. Xin. Crowddb: Answering queries with crowdsourcing. In *SIGMOD*, pages 61–72, 2011.
- [3] D. Haas, J. Ansel, L. Gu, and A. Marcus. Argonaut: macrotask crowdsourcing for complex data processing. In *PVLDB*, pages 1642–1653, 2015.
- [4] D. Haas, J. Wang, E. Wu, and M. J. Franklin. Clamshell: speeding up crowds for low-latency data labeling. In *PVLDB*, pages 372–383, 2015.
- [5] C.-J. Ho and J. W. Vaughan. Online task assignment in crowdsourcing markets. In *AAAI*, pages 45–51, 2012.
- [6] P. G. Ipeirotis, F. Provost, and J. Wang. Quality management on amazon mechanical turk. In *SIGKDD*, pages 64–67, 2010.
- [7] C. Lofi, K. El Maarry, and W.-T. Balke. Skyline queries in crowd-enabled databases. In *EDBT*, pages 465–476, 2013.
- [8] A. Marcus, D. Karger, S. Madden, R. Miller, and S. Oh. Counting with the crowd. In *PVLDB*, pages 109–120, 2012.
- [9] A. Marcus, E. Wu, D. Karger, S. Madden, and R. Miller. Human-powered sorts and joins. *PVLDB*, pages 13–24, 2011.
- [10] A. Marcus, E. Wu, D. R. Karger, S. Madden, and R. C. Miller. Crowdsourced databases: Query processing with people. In *CIDR*, pages 211–214, 2011.
- [11] Amazon mechanical turk. <https://www.mturk.com/>.
- [12] A. G. Parameswaran, H. Park, H. Garcia-Molina, N. Polyzotis, and J. Widom. Deco: declarative crowdsourcing. In *CIKM*, pages 1203–1212, 2012.
- [13] V. C. Raykar and S. Yu. Eliminating spammers and ranking annotators for crowdsourced labeling tasks. *Journal of Machine Learning Research*, pages 491–518, 2012.
- [14] Upwork. <https://www.upwork.com/>.
- [15] N. Vesdapunt, K. Bellare, and N. Dalvi. Crowdsourcing algorithms for entity resolution. In *PVLDB*, pages 1071–1082, 2014.
- [16] T. Yan, V. Kumar, and D. Ganesan. Crowdsearch: exploiting crowds for accurate real-time image search on mobile phones. In *MobiSys*, pages 77–90, 2010.
- [17] C. J. Zhang, L. Chen, H. Jagadish, and C. C. Cao. Reducing uncertainty of schema matching via crowdsourcing. In *PVLDB*, pages 757–768, 2013.
- [18] Y. Zheng, J. Wang, G. Li, R. Cheng, and J. Feng. Qasca: a quality-aware task assignment system for crowdsourcing applications. In *SIGMOD*, pages 1031–1046, 2015.