Mechanical Cheat: Spamming Schemes and Adversarial Techniques on Crowdsourcing Platforms

Djellel Eddine Difallah, Gianluca Demartini, and Philippe Cudré-Mauroux
eXascale Infolab
U. of Fribourg—Switzerland
{firstname.lastname}@unifr.ch

ABSTRACT

Crowdsourcing is becoming a valuable method for companies and researchers to complete scores of micro-tasks by means of open calls on dedicated online platforms. Crowdsourcing results remains unreliable, however, as those platforms neither convey much information about the workers' identity nor do they ensure the quality of the work done. Instead, it is the responsibility of the requester to filter out bad workers, poorly accomplished tasks, and to aggregate worker results in order to obtain a final outcome. In this paper, we first review techniques currently used to detect spammers and malicious workers, whether they are bots or humans randomly or semi-randomly completing tasks; then, we describe the limitations of existing techniques by proposing approaches that individuals, or groups of individuals, could use to attack a task on existing crowdsourcing platforms. We focus on crowdsourcing relevance judgements for search results as a concrete application of our techniques.

Categories and Subject Descriptors

H.3.3 [**Information Storage And Retrieval**]: Information Search and Retrieval; H.3.4 [**Information Storage And Retrieval**]: Systems and Software

General Terms

Algorithms, Experimentation, Performance

Keywords

Crowdsourcing, Spam, Adversarial IR, Malicious workers

1. INTRODUCTION

Crowdsourcing is the process of indirectly employing anonymous people over the internet, often for a nominative amount of money, to complete concise tasks (called *micro-tasks*) that are typically too complex for today's computers but relatively simple for humans. Examples of such micro-tasks include image annotation, relevance judgement, sentiment analysis, language translation, etc. Currently, crowdsoucing platforms like Amazon Mechanical Turk[1] (AMT) allow requesters to create tasks in the form of web pages, decide on how much to pay per task, and restrict participants by declaring filters on acceptance rate, country, etc. Once the tasks are completed, the requester gets back results in the form of raw files from which they are supposed to filter out bad answers, and decide whether or not to pay for each given answer. One particular appeal of crowdsourcing is to complete large collections of tasks that a requester cannot do by himself in a reasonable amount of time; hence, going through all the response manually is also not practical.

Crowdsourcing calls attracts different categories of potential workers; a study of the demographics showed that workers are spread over country clusters (mostly in India and USA), age and occupation. The incentives for completing a task varies per task type and per individuals. For more than 50% of the Indian workers, for instance, crowdsourcing is a primary or secondary source of income[2]. This obviously negatively influences the efforts and time taken to complete tasks on the crowdsourcing platform, since many requesters do not use tight quality control schemes (previous research has shown that it is better to systematically pay for all completed tasks rather than to risk not paying honest workers [2]).

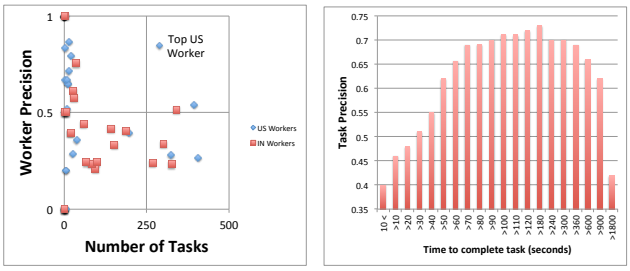
The focus of this paper is the increasing adoption of crowdsoucing in a context where the workers' incentive is solely monetary. Skimming through the tasks and hastily or randomly filling out web forms is the simplest form of crowdsourcing treachery. Knowledgeable individuals can go much further and create automated programs that employ advanced methods to complete tasks. We can for example imagine organized groups sharing information to complete collections of tasks faster. In the following, we show that in the absence of strict control and monitoring mechanisms on the crowdsourcing platforms, the requesters are reduced to rely on manual labor, artificial intelligence or statistical methods to filter out potentially erroneous responses.

Our final claim is that

> Current crowdsourcing quality control techniques are *insufficient* to counter organized groups of workers who maliciously aim at gaining money disregarding the quality of their completed tasks.

CrowdSearch 2012 workshop at WWW 2012, Lyon, France

[1] https://www.mturk.com/mturk/welcome
[2] http://hdl.handle.net/2451/29585



(a) (b)

Figure 1: (a) Worker precision distribution (from [4]). (b) Task quality over task completion time.

The remaining of this paper is organized as follows: In Section 2, we present the motivation of this work. We give an overview of anti-adversarial techniques in Section 3. In Section 4, we describe a series of possible attacks that could undermine today's anti-adversarial techniques. We draw our conclusions and highlight future directions in Section 5.

2. MOTIVATION

Despite many recent research efforts dedicated to result filtering and cheaters detection, quality control methods remain largely bound by the following factors: 1) Task type 2) Time 3) Cost 4) Participants. First, we believe that the current techniques can hardly be generalized to any type of tasks, especially for subjective types of tasks like review writing and text translation. Second, in the absence of a reputation system[3] participants remain anonymous and little information is communicated to the requesters, it is therefore difficult to conclude that an experiment is repeatable and to assess how many dishonest workers were involved. Cost is also a sensitive variable: it has been shown that paying a higher reward for a task does not lead to higher quality but only to lower completion time (see, for example, [1]). In an experiment we previously conducted [4] where we manually judged all worker answers, we observed that more than 75% of Indian workers achieved a precision of 50% or less. Figure 1a shows the precision of the workers against the number of tasks they performed. Out of the 4088 submitted tasks, only 1 worker achieved a precision of 0.85 while answering 288 tasks. Do the available quality control schemes respond to the behavior of the current worker population? In the following we will describe adversarial techniques providing examples based on the relevance judgement scenario: documents are shown to the user together with a keyword query; the worker has to judge the relevance of the document (possibly on multiple levels) with respect to the query.

3. ANTI-ADVERSARIAL TECHNIQUES

We can distinguish two categories of crowdsourcing anti-adversarial schemes: a priori cheater dissuasion and a posteriori quality control. In the following, we summarize briefly some of the common techniques from both perspectives, and discuss some of their advantages and drawbacks.

[3]It has to be mentioned that recently AMT introduced the concept of Masters, that is, workers which have proved to perform well on certain task types. Anyhow, it is out of the requester's control to decide who is a quality-worker at this stage.

3.1 Task Design

Task design is the sole responsibility of the requester. At a minimum, it consists of choosing the right way to formulate the task and the right incentives. A study on the impact of incentives was recently conducted [14] and partly concluded that crowdsourcing platforms favor monetary incentives instead of social ones. The study also hypothesized that explicit worker conditioning (e.g., inform the worker that disagreement with other workers on the same task will be punished) on top of quality control can lead to better quality results. In [12], Kitture *et al.* stressed the importance of task formulation and of having verifiable results with two variants of a given task formulated differently; along the same line, [7] observed that cheaters are less attracted to "Novel tasks that involve creativity and abstract thinking". Incentives and sophisticated task formulation form a good barrier for cheaters but constitute a burden for the requester who only needs the task to be done.

3.2 Task Repetition and Aggregation

For many tasks, result of the aggregation of multiple answers from non-expert workers (the so-called *wisdom of the crowds*) can be compared to the results of more expensive expert workers as shown in several evaluations of crowdsourced relevant judgment or other labeling experiments [16, 15, 3]. In general, and in the presence of noisy answers, the same task is offered multiple times to different workers; once all the tasks are completed, the requester decides what answers to pick up and how to aggregate them. The aggregation of the final results is a well studied topic; the most straightforward approach in this context is to proceed with a majority decision (e.g., [13]). The authors of [9] formalized the majority decision approach and proposed the use of a control group that double-checks the answers of a prior run.

The primary goal of task repetition is to diversify the output by asking different workers, which is desired and even required for many task types; it comes however with the price of multiplying the cost by the number of repetitions.

3.3 Test Questions

As part of the recommendations in [12], Kittur *et al.* also suggest to formulate verifiable tasks. Cheaters and non-serious workers will likely start tasks without reading the requester directions, and then randomly click generating most likely a wrong answer. In this case, a simple screening process could be used [6]. The requester chooses K first tasks as qualification test that the worker has to pass first. Moreover, in our recent work ZenCrowd [4] we applied a posteriori continuous testing, where for every 10 tasks we tested worker answers against a gold standard set. Other test methodologies can be used throughout the experiments with classic anti-spamming techniques like CAPTCHAs to filter out automatic answers, for which the requester will not have to worry about creating a test set of questions.

Test questions are powerful traps for cheaters and spammers, especially when they cannot be differentiated from regular tasks. This comes at a cost: for large amounts of tasks however, a bigger gold standard set is needed to avoid workers spotting recurrent questions. Moreover, test questions should be selected carefully so that a) they do not trick real workers and b) they are not easy for robots to answer.

3.4 Machine Learning Filtering

The distribution of workers and the number of tasks they perform is usually characterized by a power law distribution [1] where many workers do few tasks and few workers do many tasks. The quality of aggregating the results in such a context (e.g, with a majority decision scheme) is self-contained in the judgment of the task. Using machine learning algorithms [17, 5, 10, 4] allows one to carry over some knowledge about the workers across tasks. In our ZenCrowd entity-linking system [4], we started with a learning phase to label workers with a confidence score to decide how to weight the worker's answer, then used a probabilistic network to propagate and update scores across workers and tasks.

3.5 Current AMT Techniques

Requesters on AMT can already benefit from some basic aggregation and anti-spamming features provided by the Amazon platform. A requester on AMT is not able to directly assign a task to a specific worker; he can only publish the task on the "market", and typically the first worker who agrees to pick the task will do it. In addition, the platform allows requesters to add a few constraints when the task is published, in order to help in avoiding low-quality and malicious workers. First, requesters can filter workers based on their previous acceptance rate: if on previously submitted tasks a worker had an acceptance rate lower than, for example, 95%, then he is not allowed to accept such task. Additionally, requesters can add filters on worker location (at the granularity of the country), number of tasks performed so far, and add a qualification test before the task is assigned. After a task has been completed, requesters can always decide not to pay for poorly performed tasks and can even report bad workers to AMT, which may lead to the worker account being suspended. Naturally, this can only happen after poor results and/or malicious workers are detected. Such simple features can be sufficient for simple adversarial techniques, but not for organized group attacks as explained in the next section.

4. ADVERSARIAL TECHNIQUES

This section presents an overview of possible attacks to crowdsourcing platforms that we envision. We define a *dishonest answer* in a crowdsourcing context as a task that has been either: i) randomly posted ii) artificially generated or iii) duplicated from another source. We differentiate spamming attacks by the level of collaboration used to generate dishonest answers. Individual attackers try to proceed as fast as possible to earn additional money but are more likely to run into easy test traps; group attacks are more organized and exploit the repeatability of a task to build knowledge, and hence become more difficult to detect. In the following, and without loss of generality, we use the scenario where workers are asked to judge the relevance of search results given a keyword query.

4.1 Individual Attack

4.1.1 Random Answers

When a worker has spent some time trying to solve the task and realizes that he is not able to provide a good answer, it is more likely that a random answer will be given rather than the task will be returned. Moreover, malicious workers will quickly and randomly answer in order to obtain the reward fast. In our previous experiment [4] we have observed that 10% of $0.01 tasks have been completed in less than 5 seconds. As we can see in Figure 1b quickly completed tasks provide lower quality work. Therefore, completion time is a strong indication of a malicious random answer.

Random answers jeopardize tasks designed with monetary incentives and no test questions. The workers prefer to provide a random answer to collect the money rather than skipping the task. However, depending on the number of random answers in the collected results, they could be filtered out by task repetition.

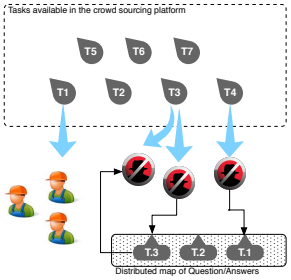


Figure 2: Group attack mechanism using a distributed map of question answers

4.1.2 Automated Answers

Spammers create generic malicious programs –or bots– capable of registering to do tasks and submitting answers either randomly or with minimal artificial reasoning.

Test questions constitute a good trap for this attack. However bots can massively attack a task and thus increase their chances to pass qualification test questions.

4.1.3 Semi-Automated Answers

We identify semi-automated answers as bots specifically designed for a given task (e.g., relevance judgement). Spammers can use pre-existing packages and tailor their attacks to a given context. In our use-case, the spammers can create a bot that opens all the links, parses the corresponding HTML content and attempt to complete the relevance judgment task whenever possible. Or, it can run the query in a search engine and identify which of the proposed links has been ranked first. If the bot is not sure about its answer, it can even ask a human, for a given ratio of questions, to increase its answer accuracy or return low confidence HITs to preserve its approval rate.

Such semi-automatic approaches can considerably improve the time/reward ratio of dishonest workers and they target task collections with easy-to-answer test questions.

4.2 Group Attack

By a group attack we mean a group of individuals or bots focusing on the same batch of tasks. This group can use a distributed dictionary of questions and answers. Depending on the attack, answers (e.g., query-result pairs) are recorded in this dictionary called Shared Question Answer Dictionary (SQAD) which is shared among the group (see Figure 2).

4.2.1 Agree on Answers

Typically, the requester would expect some agreement on

the answers received for a given task; it is typically recommended to shuffle the order of the questions and their answers to prevent an all-agree-on-first strategy. In our use case, the attackers could use the following strategy: the first worker who sees a task will select a link randomly, and then an automated system will create an entry in a SQAD with the query string and the chosen link. If the same task is encountered again, the automated system will highlight or automatically submit the stored answer.

This attack makes majority vote filtering ineffective as it may discard valid answers.

4.2.2 Answer Sharing

From Section 3.3, we can categorize test questions and their respective attacks into:

- *Gold standard*: The requesters can only input a limited number of these questions and often will require redundancy. Spammers will exploit this weakness by having all the workers agree on honestly answering some questions and submitting their answers to a SQAD. The answers to test questions get shared as well.
- *Turing test questions*: Such questions (e.g., Captcha) are widely used to stop bots, they can also be generated indefinitely, which makes it impossible to track with a SQAD. Since only humans can pass these tests, it is sufficient that the task is recognized as a test to require full human attention. The remaining of the task can be completed automatically.

4.2.3 Artificial Clones

This attack is similar to Answer Sharing, with the difference that the malicious worker acts alone by answering honestly to questions and storing them, then spawns automated programs that duplicate the spammer's behavior by reading his answers. If the program encounters an unseen question, it can either skip the question (if allowed by the platform), answers randomly, or ask a human.

This attack presents a challenge for all the anti-adversarial schemes we are aware of, as the bots merely replicate a "honest" answer which is an increase in gain for the spammer and a loss for the requester.

5. CONCLUSIONS AND FUTURE WORK

The crowdsoucing market is flourishing and it is strongly based on financial incentives. Because of this, it may attract more and more cheaters and thus give rise to novel cheating-schemes. We could not find any hard evidence about the amount of spam in crowdsourcing platforms. Anyhow, we expect that adversarial approaches will become more advanced as the popularity of crowdsourcing raises.

In this paper we overviewed adversarial crowdsourcing mechanisms and showed that many of current quality control mechanisms can fail naively in detecting well-organized spammers. Based on the presented overview, we claim that in the process of evaluating spam-filtering schemes, the usual methodology applied—often based on self designed experiments—is not adequate to real crowdsourcing environments where organized groups of malicious workers are present.

Such reasons motivate the need for further studies in the area of spam detection and quality control in crowdsourcing platforms which will be the focus of our future work. Specifically, there is the need for new benchmarks on which to evaluate and compare existing and novel spam detection techniques for crowdsourcing platforms. Moreover, a study of how much this problem affects the quality of crowdsourced tasks in a real-world large-scale setting is necessary. Existing research started to understand which tasks attract more cheaters and which task features have to be controlled (e.g., high reward as well as simple task design attract more malicious workers [7]). Therefore, the conclusion is that it is better to discourage cheaters rather than invest resources in a posteriori filtering. With respect to post-filtering, the requester can use information like assignment time, submission time, feedback etc. to classify an answer as spam. Systems for worker analytics that help gather and share data in real time about the tasks in progress and about workers may help in identifying malicious behaviors [8]. Reward mechanisms different than the financial one should be also taken into account [11].

6. ACKNOWLEDGMENTS

This work was supported by the Swiss National Science Foundation under grant number PP00P2_128459.

7. REFERENCES

[1] O. Alonso and R. A. Baeza-Yates. Design and implementation of relevance assessments using crowdsourcing. In *ECIR*, pages 153–164, 2011.

[2] O. Alonso and M. Lease. Crowdsourcing 101: putting the wsdm of crowds to work for you. In *WSDM*, pages 1–2, 2011.

[3] O. Alonso and S. Mizzaro. Can we get rid of TREC assessors? Using Mechanical Turk for relevance assessment. In *SIGIR 2009 Workshop on The Future of IR Evaluation*, 2009.

[4] G. Demartini, D. E. Difallah, and P. Cudre-Mauroux. ZenCrowd: Leveraging Probabilistic Reasoning and Crowdsourcing Techniques for Large-Scale Entity Linking. In *WWW 2012*, Lyon, France, 2012.

[5] P. Donmez, J. G. Carbonell, and J. Schneider. A probabilistic framework to learn from multiple annotators with time-varying accuracy. In *SDM'10*, pages 826–837, 2010.

[6] J. S. Downs, M. B. Holbrook, S. Sheng, and L. F. Cranor. Are your participants gaming the system?: screening mechanical turk workers. In *CHI*, pages 2399–2402, New York, NY, USA, 2010. ACM.

[7] C. Eickhoff and A. P. de Vries. Increasing Cheat Robustness Of Crowdsourcing Tasks. *Information Retrieval*, 2012.

[8] P. Heymann and H. Garcia-Molina. Turkalytics: analytics for human computation. In *WWW*, pages 477–486, New York, NY, USA, 2011. ACM.

[9] M. Hirth, T. Hoßfeld, and P. Tran-Gia. Cost-Optimal Validation Mechanisms and Cheat-Detection for Crowdsourcing Platforms. In *Workshop on Future Internet and Next Generation Networks (FINGNet)*, Seoul, Korea, June 2011.

[10] P. G. Ipeirotis, F. Provost, and J. Wang. Quality management on amazon mechanical turk. In *Proceedings of the ACM SIGKDD Workshop on Human Computation*, HCOMP '10, pages 64–67, New York, NY, USA, 2010. ACM.

[11] R. Jurca and B. Faltings. Mechanisms for making crowds truthful. *J. Artif. Int. Res.*, 34:209–253, March 2009.

[12] A. Kittur, H. Chi, and B. Suh. Crowdsourcing user studies with mechanical turk. In *Proc. CHI 2008, ACM Pres*, pages 453–456, 2008.

[13] J. Le, A. Edmonds, V. Hester, and L. Biewald. Ensuring quality in crowdsourced search relevance evaluation: The effects of training question distribution. In *SIGIR Workshop on Crowdsourcing for Search Evaluation*, pages 21–26, 2010.

[14] A. D. Shaw, J. J. Horton, and D. L. Chen. Designing incentives for inexpert human raters. In *CSCW*, pages 275–284, New York, NY, USA, 2011. ACM.

[15] V. S. Sheng, F. Provost, and P. G. Ipeirotis. Get another label? improving data quality and data mining using multiple, noisy labelers. In *KDD*, pages 614–622, New York, NY, USA, 2008. ACM.

[16] R. Snow, B. O'Connor, D. Jurafsky, and A. Y. Ng. Cheap and fast—but is it good?: evaluating non-expert annotations for natural language tasks. In *EMNL*, pages 254–263, Stroudsburg, PA, USA, 2008. Association for Computational Linguistics.

[17] J. Whitehill, P. Ruvolo, T. fan Wu, J. Bergsma, and J. Movellan. Whose Vote Should Count More: Optimal Integration of Labels from Labelers of Unknown Expertise. In *NIPS*, pages 2035–2043, 2009.