Detecting Non-Adversarial Collusion in Crowdsourcing

Ashiqur R. KhudaBukhsh
Department of Computer Science
Carnegie Mellon University
akhudabu@cs.cmu.edu

Jaime G. Carbonell
Language Technologies Institute
Carnegie Mellon University
jgc@cs.cmu.edu

Peter J. Jansen
Language Technologies Institute
Carnegie Mellon University
pjj@cs.cmu.edu

Abstract

A group of agents are said to collude if they share information or make joint decisions in a manner contrary to explicit or implicit social rules that results in an unfair advantage over non-colluding agents or other interested parties. For instance, collusion manifests as sharing answers in exams, as colluding bidders in auctions, or as colluding participants (e.g., Turkers) in crowd sourcing. This paper studies the latter, where the goal of the colluding participants is to "earn" money without doing the actual work, for instance by copying product ratings of another colluding participant, adding limited noise as attempted obfuscation. Such collusion not only yields fewer independent ratings, but may also introduce strong biases in aggregate results if undetected. Our proposed unsupervised collusion detection algorithm identifies colluding groups in crowd sourcing with fairly high accuracy both in synthetic and real data, and results in significant bias reduction, such as minimizing shifts from the true mean in rating tasks and recovering the true variance among raters.

1 Introduction

Over the last decade, crowdsourcing systems have emerged as a significant resource for sharing knowledge, accomplishing tedious tasks, collecting opinions, and gathering information. Several widely-used online systems have facilitated access to a varied range of people for collecting opinions in a short amount of time. Amazon's Mechanical Turk (AMT) is one such system that has gained tremendous popularity. However, crowdsourcing for opinions is often vulnerable to certain threats such as collusion among participants, either to intentionally bias a rating, or simply to avoid work where a set of colluding individuals designate one to do the actual work, the others merely copy, and all are paid, negating much of the benefit to whoever posts and pays for the tasks. Worse yet, copying may skew the statistics in crowd ratings, misleading the entity posting the tasks.

In order to avoid getting caught, Turkers, instead of exact copying, often make minor changes to their responses such as adding noise, which makes collusion detection more challenging. We identify this type of colluding behavior as non-adversarial, as the goal of the participants is to accumulate payments without doing the actual work, rather than having any malicious intent to boost up certain product ratings or to lower ratings of competing products or sites. Although at first glance, it may seem that the damage caused by this type of collusion is mostly financial in nature, our analysis reveals that several statistical metrics such as the mean, median, and variance experience significant shifts as an unfortunate side-effect of collusion. In this paper, we propose algorithms to detect this type of non-adversarial collusion and counter its side-effects on real collected data and synthetic data with similar properties.

Social networks, crowdsourcing, e-auctions, and e-education open several new possibilities in exchanging information. However, such interactions are beneficial to all the parties involved if participants follow certain rules of engagement, whether explicit or tacit. One common type of rule is one-persona per person, another is non-collusion among participants. We see our work as a part of a growing literature on detecting violations of these rules and countering the damage caused by such violations (Lasecki, Teevan, and Kamar). Past work in collusion detection has focused on player-player collusion in competitive games (Vallve-Guionnet 2005; Smed, Knuutila, and Hakonen 2007; Mazrooei, Archibald, and Bowling 2013), on cheating by colluding on multiple-choice exams (Ercole et al. 2002; Wollack 2003), on plagiarism as a form of collusion (Irving 2004), and most recently on on-line reputation and rating systems (Liu, Yang, and Sun 2008; Allahbakhsh et al. 2013). In contrast, we address collusion in crowd-sourcing such as in AMT, which seeks to minimize work, maximize profit, but causes harm as a side effect unlike falsely inflating or trashing on-line reputations, or theft by collusion in on-line gambling. Our research addresses the case of "real" colluding teams of users and the multi-doppelganger case of one person creating phantom colluding partners, and any combination of k persons controlling $n \geq k$ colluding e-personas, since the methods detect patterns of potential collusion, regardless of the bi-partite mapping between e-personas and real people, or between people who do the real work and those who merely copy and obfuscate.

The first contribution of this paper is to identify non-adversarial collusion as a serious threat in introducing certain biases to the obtained data and to make a real-world

data set with admitted collusion of this type publicly available for the first time [1]. To the best of our knowledge of the literature, this is the first work that realizes this type of non-adversarial collusion as a potential threat and explores the extent of damages caused by such behavior. Through a series of experiments on both real and synthetic data, we show that in the presence of non-adversarial collusion, there is a significant shift in the mean and the KL divergence between the true distribution and the observed distribution. As a second contribution we propose methods to detect such collusion and show that after taking corrective measures based on the findings of our detection algorithm, we can eliminate most of the bias caused by it. Our results show that our method is fairly robust to the percentage of colluding participants even when the number of colluders is high as 50% of the total participants. Finally, we offer insights into how to design rating tasks to facilitate early detection by a thorough analysis on several datasets varying the parameters of our collusion model.

Throughout the entire paper, we used the following terms raters, Turkers and participants interchangeably. The rest of the paper is organized as follows. After a discussion of related work (Section 2), we illustrate the damage of non-adversarial collusion detection with a simple example. (Section 3). This is followed by a description of our data (Section 4), detection methods (Section 5), evaluation metrics (Section 6) and a presentation and discussion of the results from our experiments (Section 7). We end with some general conclusions and an outlook on future work (Section 8).

2 Related Work

In view of the sizeable literature on detection of malicious collusion in Collaborative Filtering systems (Lam and Riedl 2004; Allahbakhsh et al. 2013; Mehta, Hofmann, and Nejdl 2007), the scarcity of work on non-adversarial collusion in crowdsourcing situations is rather surprising. While many aspects of both situations are similar, the differences imply the need to explore a different approach and new detection algorithms.

A first distinction lies in the colluder's *intent*. In shilling attacks or profile injection attacks in Collaborative Systems, malicious shillers aim to influence the outcome of the recommendation system. For a similar reason, our work is different from (Lasecki, Teevan, and Kamar) which also identifies collusion as a serious threat to crowdsourcing where colluders can extract sensitive data by exchanging information. In non-adversarial collusion situations, colluders are not concerned with the outcome of the recommender system or extracting sensitive data per se, just with avoiding the effort required for producing the ratings without reduction in remuneration. That is, their primary aim is laziness.

(Marcus et al. 2012) raised the possibility of a sophisticated group of Turkers coordinating a Sybil attack on a truncated mean-based crowd-powered algorithm and proposed an effective and scalable solution of randomly distributing gold standard items throughout the tasks. Although this work raises a pertinent potential threat in crowd-powered systems, our work is different in the following ways. First, our techniques are different: our unsupervised algorithm detects collusion by computing the pairwise similarities and does not require gold standard labels. In fact, for product or service ratings, there is no gold standard whatsoever. Secondly, the intention of Turkers setting up a Sybil attack involves skewing the means for their own benefit whereas in a non-adversarial collusion scenario, the skew is merely a side-effect.

Additionally, the situations to which the different styles of collusion apply may be different in the *information* the raters are given. Large-scale online rating systems like Amazon shopping or IMDB provide a detailed summary of how other people have rated a specific product with mean, median, histogram and often demographic information. In a paid online survey, such information is not usually provided to raters.

Potential collusion *strategies* will be different between colluders who strive to promote or sink a product, service or task (shillers), and those who seek to collect payment without work (non-adversarial colluders). In order to hide from the detection mechanism, a typical malicious shiller will rate many products close to their respective means, and, depending on intent, give extreme low or extreme high ratings for a small subset of the target products he or she is interested in trashing or boosting. A typical non-adversarial colluder, on the other hand, will provide ratings derived from the lead clique member, with small modifications to avoid detection. Hence, *detection methods* used to detect malicious attacks, usually based on coordinated deviations from the mean for a subset of tasks, may not be of much use in the case of non-adversarial collusion. On the other hand, certain plausible malicious attack types (such as automated profile injection attacks or impersonation) are much less likely in our scenario because of stricter user validation policies for paid crowd workers.

Another context in which collusion has been studied extensively is that of test taking, e.g. on multiple choice exams (Ercole et al. 2002; Argenal, Cruz, and Patungan 2004; Wollack 2003). The nature of this scenario is more akin to our own, and several of the indices proposed to detect collusion (generally based on (in)dependence), could be adapted. Most of these metrics, however, cannot take advantage of the ordinal nature of rating values (there is generally only one correct answer in a test).

Yet another context in which collusion has received considerable attention is the socio-economic sense of *Game Theory* and *Auctions* (Mazrooei, Archibald, and Bowling 2013). Although the same term is used, "collusion" in this case arguably describes a rather different phenonenon than in the cases considered above, and the majority of the literature on cartel detection or collusion detection in voting or sequential games does not address non-adversarial collusion. In most cases, this form of collusion must be deemed malicious, as it is almost by definition intended to harm other parties.

[1] The data set can be downloaded from *http://www.cs.cmu.edu/ akhudabu/projects.html*.

3 The Damage of Non-adversarial Collusion

In this section, we illustrate the problem of non-adversarial collusion by using a simple example. Table 1 shows a small 5×5 window of our real dataset with five tasks denoted as T_1, T_2,$\cdots$, T_5, and five participants denoted as P_1, P_2,$\cdots$, P_5 of which P_1 and P_2 are non-colluding, whereas P_3, P_4 and P_5 constitute one colluding clique (as per the obtained ground-truth labels).

	T_1	T_2	T_3	T_4	T_5
P_1	8	6	5	9	2
P_2	2	9	5	5	3
P_3	4	9	3	3	5
P_4	6	8	3	2	5
P_5	4	8	6	3	5

Table 1: A 5×5 window of a real dataset with colluding participants

We observe that the colluders are not submitting identical copies of their ratings, presumably to avoid detection. A careful examination for possible dependencies reveals however that the differences in ratings between colluders are small for each of the tasks (in this example, the maximum disagreement of 3 is observed, for instance, between colluders P_3 and P_5 on task T_3). The difference between non-colluding raters tends to be larger (for example, the difference is 6 between P_1 and P_2 on T_1).

We can also note that the average ratings submitted by colluders and non-colluders are generally close, indicating that plausibly the colluders did not intend to influence the rating statistics in any way (non-adversarial). However, the collusion may nevertheless introduce an unintentional bias in these statistics can be observed from the table. Without collusion (i.e., of the colluders, only the one genuine rater is considered), we have that $\mu = 3.3$ ($\sigma = 1.25$). Wich collusion present but undetected, on the other hand, $\mu = 4.0$ ($\sigma = 1.27$). That is, collusion introduces a *mean shift* of 20%. While the results on a small example are not statistically significant (see the Results section for a more detailed analysis), the example illustrates that collusion may have a large impact on the statistial reliability of the results, and it is in the interest of the crowdsourcer to eliminate its influence.

4 Data

Obtaining reliable real-world data for testing the effectiveness of our collusion-detection algorithm is difficult for the following reasons. First, much of the crowdsourced data for product ratings is closely held by marketing companies or the product producers. It is gathered because its competitive business value, and sharing it may negate such value. Secondly, labeling data for colluding cliques requires a significant amount of time and effort, and in many cases domain expertise. This, points to the need for unsupervised methods, and was in fact one of the motivating factors behind our automatic detection endeavors. Finally, since collusion is not generally acceptable behavior and sometimes subject to penalties (e.g., cancellation of payment or future barring), colluders may not admit to their actions, making the collection of gold-standard ground truth even more challenging.

For these reasons, in addition to a real dataset with ground-truth ratings, consisting of anonymized product ratings obtained from a large e-commerce organization, we have generated a large number of synthetic rating scenarios roughly modelled to have the same properties as the real data set. The real data set consists of (incomplete) rating data (values on an integer rating scale from $[1..10]$), from 123 participants, over 20 densely populated tasks. The identification of collusions is by obtaining an admission from the colluders after the fact. Hence it is very unlikely to contain false positives. The data set is smaller than we would like, and is somewhat biased towards higher collusion probability, even after later addition of data from (presumed) non-colluding participants. 36 users (approx 30%) admitted to colluding, in approximately 6 cliques ranging from $2-11$ clique members per clique. In the part of the data from the same surveys that was not made available, there were no known collusions. While the condition of releasing the data is to remain anonymous (the e-commerce organization do not want to publicize to their customers that crowd-surveys can and do have colluders), if we respect confidentiality, there are no restrictions in sharing this data publicly. Quite the opposite, the company would benefit from more work in this area. Also, we encourage others to collect similar data in greater quantities and make it publicly available.

Figure 1 presents the rating distributions of colluders vs. non-colluders, and shows that there are no major disagreements in the average rating for any of the tasks. However, note that colluders sometimes may shift the mean considerably, and thus present a false picture of the aggregate opinion.If the colluding group just happens to express the average opinion, then this will not occur (only a variance shift leading the false belief of greater concurrence, but that is a secondary effect). But if the colluding group represents a more extreme opinion, the mean shift can cause a major misperception of the true aggregate opinion. Hence detecting and removing colluding groups corrects for the false mean shift.

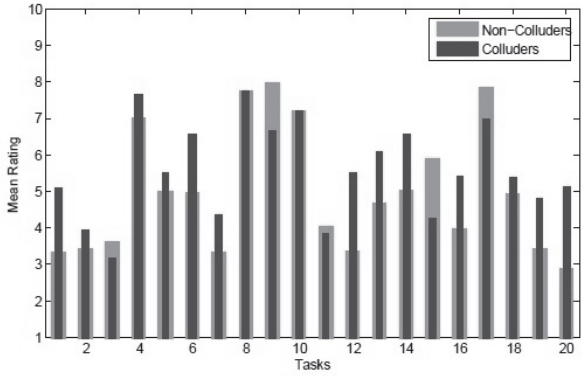


Figure 1: Rating distribution of collluders vs non-colluders.

We generated our synthetic dataset to be comparable to the real-world data under the following initial assumptions.

- **Task type**: Numerical rating task using the same fixed scale.
- **Task difficulty**: equal difficulty and no colluder has any additional incentive to collude on some specific subset of tasks.
- **Task interdependence**: none. That is, for any given rater R_i, her ratings for any task pair T_j and T_k are independent.
- **Distributional assumption**: rating distributions for each task are mixture distributions of two truncated Gaussians (many real-world rating distributions are approximately bimodal).
- **Agent participation**: dense. Since crowdsourcers need statistically sufficient data to compute rating statistics, we assume that all participants take part in a significant block of tasks.
- **Collusion behavior**: colluding participants form cliques (of minimum size 2). Within each clique one participant (the *clique leader*) actually performs the rating task. The others (*clique followers*) copy the leader's ratings, and submit these after adding limited noise to obfuscate. Our real dataset is too small to reliably identify the optimal distribution that characterizes the noise. Given that the obfuscation attempts appear to be symmetric about the mean, and mostly not too far from the mean, an initial assumption of Gaussian noise was made – and it is not unreasonable given that conditional on said assumption we can recover the known collusion cliques. A larger data set would be required to characterize the obfuscation-injected noise more precisely.

For our initial experiments, we have kept fixed the number of tasks (20) and participants (60), and generated 5100 instances of rating scenarios for varying collusion prior $p_{coll} \in [0.00, 0.01..0.50]$ (100 instances for each collusion prior value). Later we increased the number of tasks.

We were informed by the provider of the collusion data that the number of expected colluding raters are a minority compared to the total parties genuinely providing ratings. Hence we chose to run the collusion ratios between [0.0 and 0.5] to cover the range of expected colluding fraction of raters. However, there is nothing in our method that sets a hard limit of 0.5; we expect our algorithm to degrade gracefully as the number of colluders increases. In fact, we ran one experiment with the ratio at 2/3 colluders (0.67). On 100 such instances, we obtained an overall accuracy of 87.71% with no false positives (precision of 84.34%).

5 Detection Methods

Our basic algorithm FINDCOLLUDERS is presented in Algorithm 1.

In general, we try to detect collusion by finding strong inter-rater dependence across tasks, especially as these diverge from the mean ratings. Of the various schemes we have explored, the following fairly direct algorithm performed the most accurately and robustly on both our real-world data and the synthetic data we modeled on the basis thereof.

Input: A set of m participants $P_1, P_2, \cdots, P_m$. A set of n tasks $T_1, T_2, \cdots, T_n$. An $m \times n$ matrix R. $R_{\langle i,j \rangle}$ denotes the response of the i-th participant to j-th task.
Output: An $m \times m$ matrix C such that $C_{\langle i,j \rangle} = 1$ if P_i has colluded with P_j, or 0 otherwise.
Initialize
$C_{\langle i,j \rangle} = 0,\ \forall 1 \leq i \leq m, 1 \leq j \leq m$
Subtract the column mean from individual columns
$R'_{\langle i,j \rangle} \leftarrow R_{\langle i,j \rangle} - \frac{1}{n} \sum_j R_{\langle i,j \rangle}$
$R \leftarrow R'$
Compute the $m \times m$ similarity matrix S based on a **similarity measure**
for $pOne \leftarrow 1$ ***to*** m **do**
  for $pTwo \leftarrow 1$ ***to*** m **do**
    if $S_{\langle pOne,pTwo \rangle} >$ ***threshold*** **then**
      $C_{\langle pOne,pTwo \rangle} \leftarrow 1$
    end
  end
end
return C

Algorithm 1: FINDCOLLUDERS Basic collusion detection algorithm

In our implementation, we use *cosine similarity* as our similarity measure. Cosine similarity between two vectors **A** and **B** is defined as

$$\text{sim}(\mathbf{A},\mathbf{B}) = \frac{\sum_{i=1}^{n} A_i \times B_i}{\sqrt{\sum_{i=1}^{n} (A_i)^2} \times \sqrt{\sum_{i=1}^{n} (B_i)^2}}$$

For the sake of correctness of our algorithm, we define S in the following way:

$$S_{\langle i,j \rangle} = \begin{cases} \text{sim}(R_{\langle i,. \rangle}, R_{\langle j,. \rangle}) & \text{if } i \neq j \\ 0 & \text{if } i = j \end{cases}$$

where $R_{\langle i,. \rangle}$ denotes the response of P_i on all tasks.

Note that since collusion is symmetric, derivation of the cliques from the pairwise collusion relationships is straightforward. Also, if we only care about if a rater has colluded with any other rater or not, the binary classification problem is trivially solvable from the pairwise collusion relationships. For a given pair of raters, the complexity of computing the similarity measure is $O(n)$ where n is the number of tasks, and we need to compute this for every pair of raters. So the overall complexity of our algorithm is $O(m^2 n)$.

Our method has the following limitations. Since we identify pairwise-similarity beyond a certain threshold as indicative of collusion, our approach is limited to opinion-based rating problems. Our approach may perform less well on more objectively grounded tasks (e.g., multiple choice labeling or testing) because it may identify raters choosing correct answers as potential colluders. Also, we acknowledge that it is possible that people with extremely similar preferences may lead to false positives.

Our main classification criterion (and only continuous parameter) is the `threshold`, which we set empirically to a fixed value of 0.85. As we show below, this setting proved robust regarding the performance of FINDCOLLUDERS across a variety of parameters (such as users, tasks, and distributions), with some bias towards precision rather than recall (i.e., benefit of the doubt).

6 Evaluation Metrics

Evaluating the performance of our detection algorithm can be done in different ways depending on our objective. If we only care about barring the colluders from future participation, identifying the colluders is sufficient. In that case, this problem can be treated as a standard binary classification problem and a high recall and precision value will achieve the objective. However, if we want to make necessary adjustments in order to do meaningful analysis on the data (e.g., computing the mean, estimating the maximum-likelihood estimation (MLE) of the multinomial for a given task), we further need to correctly identify the cliques to which the colluders belong. For example, identifying a big clique as many smaller cliques will not affect the standard measures of accuracy, precision and recall, but we may not be able to recover the true mean as well as we could if we identified the big clique accurately.

Consider a simple scenario of just one clique of size 5 which the detection algorithm identifies as a clique of size 3 and a clique of size 2. Consider there is a task which the two-membered clique has rated 1,2 (effective mean 1.5) and the three-membered clique has rated 3,4,5 (effective mean 4). If we consider these as two separate cliques, our effective mean will be $\frac{1.5+4}{2} = 2.75$. However, since all these ratings are coming from the same clique, the actual effective mean should be $\frac{1+2+3+4+5}{5} = 3$. Clearly, for this task, our incorrect clique-detection results in only partial recovery of the mean. We evaluate the performance of FINDCOLLUDERS both in terms of detection-accuracy and its ability to eliminate side-effects (e.g., mean-shifts) caused by non-adversarial collusion.

7 Results

7.1 Accuracy, precision and recall

On our synthetic data, we found that FINDCOLLUDERS performed consistently well. Figure 2 summarizes the performance of FINDCOLLUDERS on our synthetic data (in terms of precision, recall, and accuracy) as a function of collusion prior. We found that even when the collusion prior was as high as 0.50, we achieved a very high precision (99%) and recall (93%).

On the real-world dataset, FINDCOLLUDERS achieved an accuracy of 93.50% (recall: 94.74%, precision: 85.71%). These results are somewhat lower than in the simulated data, perhaps because the assumptions do not fully hold in the wild. Alternatively, some false positives may be the result of actual colluders failing to admit to their copying behavior in establishing ground truth.

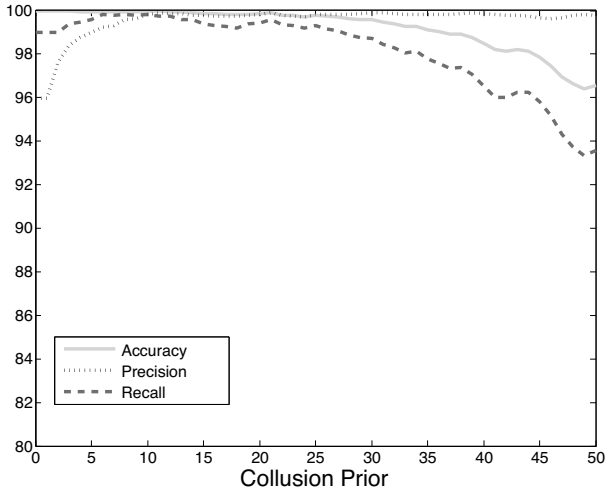


Figure 2: Performance of FINDCOLLUDERS on synthetic dataset comprising 5100 instances. Each instance is a rating scenario with 60 participants and 20 tasks. Collusion prior is varied from 0 to 0.50 at a step-size of 0.01.

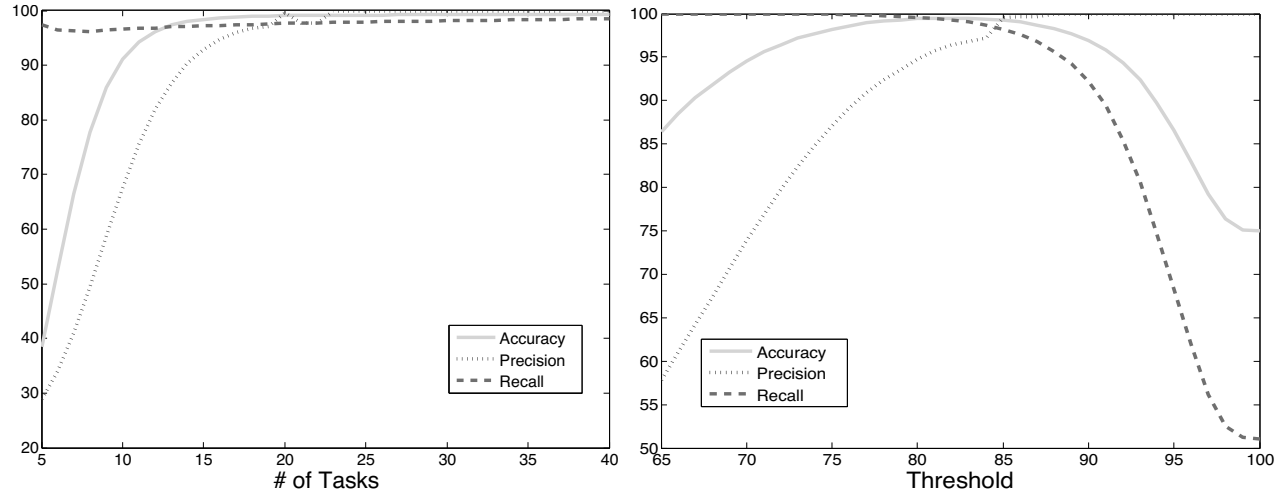
7.2 Robustness with respect to system parameters

We acknowledge that it is possible that people with similar preferences may lead to false positives. However, the colluders agree on ratings for a considerable number of vastly different products, far more than having similar opinions about certain products or services would imply. The more ratings for different uncorrelated products obtained per rater, the smaller the probability of agreement via similar opinions. Figure 3(a) shows the expected result that performance increases with the number of tasks, and 99% or higher accuracy and precision is achieved when we have 25 tasks or more per worker.

Although realistically, we don't expect to have a very high number of tasks per worker, in order to examine the scalability of our algorithm, we ran it on 51 instances where each instance consists of 1000 raters and 1000 tasks, collusion prior is varied from 0 to 0.5 at a step size of .01 (one instance for each collusion prior value). On this dataset, FINDCOLLUDERS achieved an accuracy of 100% without a single false positive. On an intel core i5-2400 machine running MATLAB 7.12.0 (R2011 a) in a Windows 7 environment, our detection algorithm took less than a minute per instance which confirms that this algorithm is computationally tractable and can be suitably modified to perform detection online.

To gain insight into the sensitivity of the `threshold` parameter to small changes in its value, we plot its influence on accuracy, precision, and recall (Figure 3(b)). The algorithm appears robust with respect to this parameter, and our choice at the high end of this range ensures the number of false positives will be limited.

Also on the real-world data, we varied the number of tasks (i.e. selecting a subset of the 20 given). Our results are qualitatively similar to the results obtained on the synthetic dataset.



(a) # of Tasks Versus Accuracy, Precision and Recall

(b) Threshold Versus Accuracy, Precision and Recall

Figure 3: Performance of FINDCOLLUDERS with varying number of tasks and threshold on our synthetic dataset.

7.3 Countering the side-effects of non-adversarial collusion

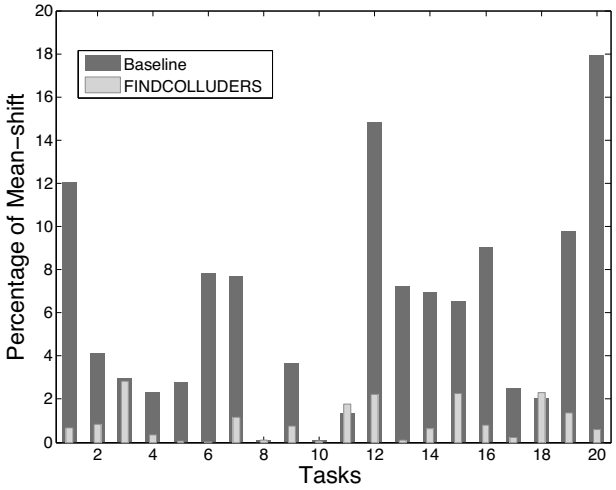


Figure 4: Comparison of mean-shifts on our real dataset in a per-task basis.

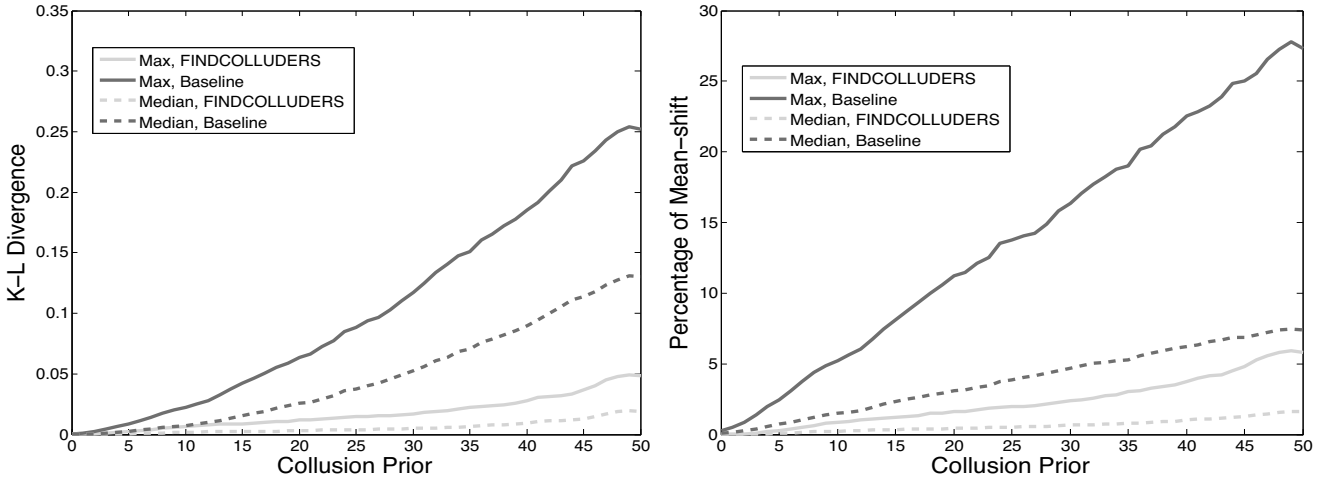
We have already illustrated in the previous sections how mean-shifts can result from non-adversarial collusion and how inaccurate idenfication of cliques affects recovery. On our synthetic and real-word dataset, we observe that indeed, substantial mean-shifts can happen if we use no collusion-detection mechanism and naively assume that every participant is non-colluding. In order to underscore the importance of detecting non-adversarial collusion and to explore the extent of its potential damage, we compare the performance of FINDCOLLUDERS against a baseline that assumes no raters colluded.

For a given task T_l, a baseline that assumes no raters colluded estimates the mean as $\hat{\mu}_{ND} = \frac{1}{n}\sum_i R_{\langle i,l\rangle}$. A detection algorithm that has identified k cliques- $c_1, c_2, \ldots, c_k$, estimates the mean as $\hat{\mu}_{DA} = \frac{1}{(n+k-\sum_k |c_k|)}(A+B)$ where $\mid c_r \mid$ denotes the size of the r-th clique, $I(L)$ is just an indicator function that returns 1 if L is true, otherwise returns 0, $A = \sum_i R_{\langle i,l\rangle} I(P_i \notin \cup_k c_k)$, and $B = \sum_k \frac{1}{|c_k|}\sum_i R_{\langle i,l\rangle} I(P_i \in c_k)$. Basically, a clique is treated as just one individual rater with an effective rating as the mean of the ratings of all members belonging to the particular clique. The mean-shift for a given task is computed as $\frac{abs(\hat{\mu}-\hat{\mu}_O)}{\hat{\mu}_O}$ where $\hat{\mu}_O$ is the estimate by an oracle that has perfect information about the cliques.

Figure 4 presents the mean-shifts on our real-world dataset for each task. The green bar denotes $\frac{abs(\hat{\mu}_{DA}-\hat{\mu}_O)}{\hat{\mu}_O}$ and the red bar denotes $\frac{abs(\hat{\mu}_{ND}-\hat{\mu}_O)}{\hat{\mu}_O}$ for each task. Figure 4 shows that different tasks experienced different amount of shifts in their respective means, which implies a constant compensation will not be particularly useful. At least half of the tasks experienced a mean-shift of 5.33% or higher (up to 17.94%). With adjustments based on our detection algorithm, this improves by roughly a factor of 6 (median and maximum reduced to 0.70% and 2.85% respectively) and capped under 5% across all tasks.

We summarize the experimental results on our synthetic data in 5(b). In the baseline (no detection) the mean-shift steadily increases as we increase the collusion prior. At 0.5, the median of the mean-shifts is 7.5% with the maximum being 27%. With adjustments based on FINDCOLLUDERS, we obtain a substantial improvement – the maximum mean shift remains below 6% even for a collusion prior as high as 50%.

Another way to measure damage is to observe the changes in the distributions extracted from the data. Let us consider the maximum likelihood estimates of the multinomial distributions on a per-task basis. For a given task T, let an oracle



(a) KL Divergence (b) Mean-shift

Figure 5: Performance comparison on eleminating strong biases incurred due to collusion on our synthetic dataset.

estimate the multinomial distribution as $\hat{P}_O$, where the baseline estimate (no detection) is $\hat{P}_{ND}$, and our detection algorithm combined with necessary adjustments obtains $\hat{P}_{DA}$.

As a measure of corruption of the distributions, we use the *Kullback Leibler Divergence*, denoted as $D_{KL}(P \parallel Q)$, defined as $\Sigma_i ln(\frac{P(i)}{Q(i)})P(i)$ (P and Q both are discrete probability distributions), which, simply put, is a measure of the information loss if Q is used to approximate P.

For each task, we compute $D_{KL}(\hat{P}_O \parallel \hat{P}_{ND})$ and $D_{KL}(\hat{P}_O \parallel \hat{P}_{DA})$. Figure 5(a) shows a similar qualitative trend as 5(b). We find that $D_{KL}(\hat{P}_O \parallel \hat{P}_{ND})$ steadily increases as collusion prior increases. When collusion prior is 0.5, the median of $D_{KL}(\hat{P}_O \parallel \hat{P}_{ND})$ across tasks is 0.13 with the maximum being 0.25. However, throughout the entire dataset, the maximum of $D_{KL}(\hat{P}_O \parallel \hat{P}_{DA})$ is restricted below 0.05 which clearly shows that our detection algorithm brings in a considerable improvement in eliminating the biases introduced by collusion. On our real-world dataset, we obtain a similar performance (roughly, a factor of 7 improvement). The maximum and the median of $D_{KL}(\hat{P}_O \parallel \hat{P}_{NA})$ across tasks were 0.079 and 0.0394, respectively, whereas the maximum and the median of $D_{KL}(\hat{P}_O \parallel \hat{P}_{DA})$ were 0.0112 and 0.0053, respectively.

Finally, all our experiments on synthetic datasets are repeated with 40 tasks instead of 20. We find all the results qualitatively similar to what we present here. Moreover, we achieved higher accuracy as precision improves with additional tasks per rater (See, Figure 3(a)).

8 Discussion

8.1 Conclusion

In this paper we have identified and investigated non-adversarial collusion as a potentially serious threat in crowdsourcing. We have shown that its unintentional results may lead to significant shifts in the mean and and KL divergence between distributions. We have proposed a method to detect non-adversarial collusion, and demonstrated that in our current setting, on both real-world and synthetic data, a simple algorithm suffices to not only detect most cases of collusion, but also eliminate most of its side-effects. The `FINDCOLLUDERS` algorithm proved fairly robust with respect to the parameters of the scenario as well as the algorithm itself.

8.2 Future work

This paper opens the gates to research in non-adversarial collusion detection and counter measures; there is still much to be done, including:

- *Occasional colluders and other variants:* Initially, we tried to adhere to the simplest possible assumptions while generating our synthetic dataset. This was mainly because a) since this type of collusion has not been studied in the literature, we wanted a reasonable starting point to begin our investigation b) we wanted to illustrate that even under the simplest assumptions, non-adversarial collusion can have substantial side-effects on estimating statistics from the data. However, we have already started relaxing some of these assumptions as a potential future direction.

  It is possible that some raters collude on some occasions and not others. Perhaps only if they can round up their friends, or they find the rating tasks onerous. This makes the collusion detection more challenging, since it can become task-group or temporally dependent. We have in fact already set out to investigate this case, and our preliminary results indicate that our algorithm maintains good results as long as the collusion behavior is systematic enough. One rater colluding with multiple cliques is another direction we are looking into.

- *Validation* on multiple real data sets. These are very difficult to obtain. We will make our anonymized data set public, as well as our synthetic data.

- *Outliers:* In the forseen arms race between colluders and detectors, the former could attempt to thwart our detection methods by including a few ratings that are very far from the ones copied. This would require a method to detect and disregard sparse outliers.
- *Matrix completion:* In cases where there is a large matrix of raters and tasks, but it is only semi-sparsely populated, it is possible that matrix completion could yield a more meaningul dense, or partially-dense matrix on which to do the analysis.
- Incorporate *Machine Learning techniques:* We can learn which types of tasks (based on task features) or types of raters (based on rater features) are more likely to collude, and use these as priors for the detection model. This would require supervised learning (Mitchell 1997; Hastie, Tibshirani, and Friedman 2009), and would be enhanced by active learning (Roy and Mccallum 2001; Donmez, Carbonell, and Bennett 2007).

References

Allahbakhsh, M.; Ignjatovic, A.; Benatallah, B.; Bertino, E.; Foo, N.; et al. 2013. Collusion detection in online rating systems. In *Web Technologies and Applications*. Springer. 196–207.

Argenal, R. N.; Cruz, E.; and Patungan, W. R. 2004. A new index for detecting collusion and its statistical properties.

Donmez, P.; Carbonell, J. G.; and Bennett, P. N. 2007. Dual strategy active learning. In *Machine Learning: ECML 2007*. Springer. 116–127.

Ercole, A.; Whittlestone, K.; Melvin, D.; and Rashbass, J. 2002. Collusion detection in multiple choice examinations. *Medical education* 36(2):166–172.

Hastie, T.; Tibshirani, R.; and Friedman, J. 2009. The elements of statistical learning: Data mining, inference, and prediction, (springer series in statistics).

Irving, R. W. 2004. Plagiarism and collusion detection using the smith-waterman algorithm. *University of Glasgow*.

Lam, S. K., and Riedl, J. 2004. Shilling recommender systems for fun and profit. In *Proceedings of the 13th international conference on World Wide Web (WWW)*, 393–402. ACM.

Lasecki, W. S.; Teevan, J.; and Kamar, E. Information extraction and manipulation threats in crowd-powered systems. In *Proceedings of the 17th ACM conference on Computer supported cooperative work & social computing (CSCW)*, 248–256.

Liu, Y.; Yang, Y.; and Sun, Y. L. 2008. Detection of collusion behaviors in online reputation systems. In *Signals, Systems and Computers, 2008 42nd Asilomar Conference on*, 1368–1372. IEEE.

Marcus, A.; Karger, D.; Madden, S.; Miller, R.; and Oh, S. 2012. Counting with the crowd. *Proceedings of the VLDB Endowment* 6(2):109–120.

Mazrooei, P.; Archibald, C.; and Bowling, M. 2013. Automating collusion detection in sequential games. In *Proceedings of the Twenty-Seventh Conference on Artificial Intelligence (AAAI)*, 675–682.

Mehta, B.; Hofmann, T.; and Nejdl, W. 2007. Robust collaborative filtering. In *Proceedings of the 2007 ACM conference on Recommender systems*, 49–56. ACM.

Mitchell, T. M. 1997. *Machine learning*. McGraw-Hill Boston, MA:.

Roy, N., and Mccallum, A. 2001. Toward optimal active learning through monte carlo estimation of error reduction. In *Proceedings of the International Conference on Machine Learning (ICML)*, 441–448.

Smed, J.; Knuutila, T.; and Hakonen, H. 2007. Towards swift and accurate collusion detection. In *GAMEON*, 103–107.

Vallve-Guionnet, C. 2005. Finding colluders in card games. In *Information Technology: Coding and Computing, 2005. ITCC 2005. International Conference on*, volume 2, 774–775. IEEE.

Wollack, J. A. 2003. Comparison of answer copying indices with real data. *Journal of Educational Measurement* 40(3):189–205.