Classifying and Clustering Survey Respondents in a World With Bad Actors

Cibele Freire
cibelemf@cs.umass.edu

Emma Tosch
etosch@cs.umass.edu

The traditional statistical treatment of inference over surveys concentrates on reducing sampling bias and nonresponse [7, 8, 6]. The rise in popularity of web surveys introduces a new threat to validity: respondents are much more likely to provide low-integrity responses. With the scale and ease of crowd-sourced survey responses, threats to validity from the population of respondents expand to include bots and automated survey answering assistants [4].

Crowdsourcing platforms have responded to these threats with *some* internal controls: Amazon maintains worker statistics and permits job requesters to require qualification tests for jobs. SurveyMonkey manages worker recruitment through email. While curation on the part of the crowdsourcing platform helps prevent bot responses, it also restricts the pool of workers to those who actively participate in the forum. As Ipierotis and others have shown [11], active workers on Mechanical Turk come from specific subpopulations. Restricting the survey sample to the population of workers who possess stellar Mechanical Turk work histories limits the pool of respondents in the same way as conducting a psychological experiment on college freshman.

Many external quality control mechanisms have arisen in response to this need for greater quality control. Frameworks such as AUTOMAN [3] ensure quality automatically for tasks the user expects will have high agreement on correct answers. Other frameworks such as MACE [9] model annotator reliability and task difficulty explicitly.

When it comes to survey responses, where ground truth is much more difficult to obtain, there are far fewer options available. One approach is to seed the data with "attention-check questions" [1] that have known responses. However, these questions are easy to spot for malicious respondents, and easy to miss for fatigued honest respondents [10, 12]. Approaches such as Welinder et al. [14] model individual variation more forgivingly than in Hovy et al. [9], but still rely upon the existence of task-specific ground truth.

Extant methods for quality control in survey responses are very *ad hoc*. They frequently rely on domain-specific knowledge and labor-intensive curation on the part of the survey writer. We believe that this does not need to be the case, and that a researcher can spend less time curating data and more time doing experiments when they write survey instruments that better lend themselves to catching bad actors. The SURVEYMAN system comes equipped with a simulator that can be used to generate data. Our hope is that a researcher can use this system to explore the survey's robustness to various bad actors, and adjust the survey accordingly.

We view data collection as an iterative process; conducting a pilot study and modifying the survey instrument can be represented as a debug-loop.

Framework

A survey can be thought of as a collection of random variables. The joint probability for a survey having n questions where each question is denoted as $Q_i, 1 < i < n$ is $P(Q_1, \ldots, Q_n)$. The joint distribution is exchangeable, but not necessarily independent.

For this project, we treat the survey as completely flat – there is no branching, and no grouping of questions. Therefore, the set of questions the respondents see may appear in any order. The possible answers to questions are all unordered and exclusive – they would appear to the respondent

as radio buttons in a web survey, and may appear in any order. Therefore, all questions are nominal measurements, modeled as categorical variables. We let all questions have equal number of response options, which we denote m. The parameter θ_{ij} represents the population probability of seeing answer j for question i, where $\sum_j \theta_{ij} = 1$. However, the population is composed of some unknown number of K subpopulations, each of which we denote by π_k. The parameters Θ_i for each Q_i may differ between subpopulations.

We define the following respondent profiles:

Lexicographic Chooses the response option with the lexicographically first surface string. Although this respondent is typically seen as an adversary[1], in the context of this project, we use the Lexicographic respondent as a proxy for a subpopulation that has 0 variance in its response set.

NoisyLexicographic Same as Lexicographic respondent, except that this respondent chooses an option randomly for a small, tunable ϵ percentage of the time. This respondent would typically be seen as our strongest adversary – unlike the Lexicographic respondent, the small variation between responses makes detection more difficult.

Profiled This respondent most closely models the problems we see in real data. Each response is drawn by a "profile," represented as a table of preferences. SURVEYMAN generates the profile according to the following procedure:

Result: The response profile
Data: n-dimensional preference vector (P), n-dimensional answer vector (A)

for $i \leftarrow 1$ **to** n **do**
  $A[i] \leftarrow$ `SelectRandom(`Q_i`)`
  $P[i] \leftarrow \frac{1}{|Q_i|} + \left(\texttt{Random}(0, 1) \times \frac{|Q_i|-1}{|Q_i|}\right)$
end

This ensures that the respondents have some preference for a particular answer between uniform and 1. When we generate a response for a survey, we chose the preferred answer for Q_i with probability $P[i]$ and the remaining answers with probability $\frac{1-P[i]}{|Q_i|-1}$. It may be the case that some respondents have preferences not appreciably stronger than random. We believe that this models many real-world surveys.

Random This respondent chooses an answer for question Q_i uniformly at random.

We also have implementations for choosing with strong positional preference (e.g., always the first option, always the last option, "Christmas Tree" style), but under the restrictions of this project (i.e., all unordered, exclusive questions), these are equivalent to the random respondent.

1 Experiments

We describe a series of experiments to assess different approaches to classifying bad actors. The experiments increase in difficulty and decrease in expected accuracy.

Baseline data To start with, we have treated each question as if it were independent. The purpose of this project is to improve upon this approach.

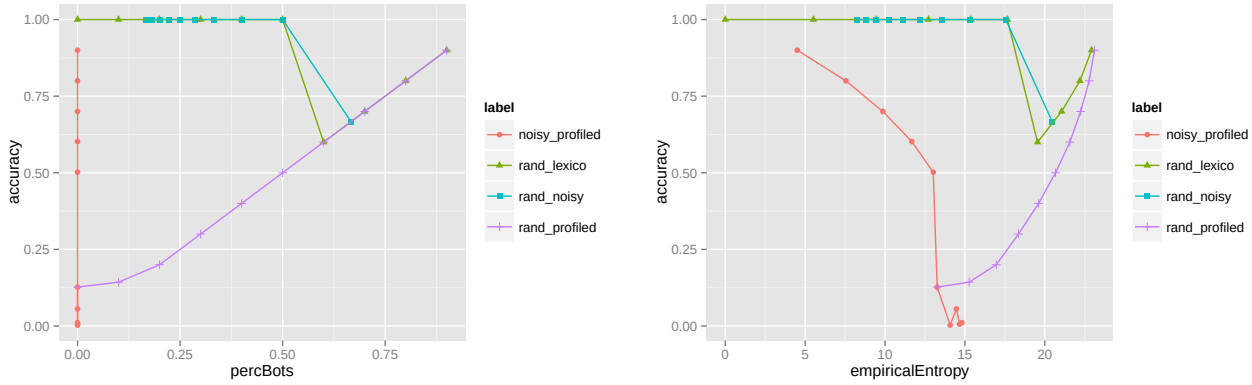
Under our independence assumption, we compute $\hat{P}(Q_1, \ldots, Q_n) = \prod_{i=1}^{n} \hat{P}(Q_i)$, where $P(Q_i)$ is estimated by the bootstrap method [5]. We classify responses that have likelihood scores that are too low as bad actors. This is, of course, an imperfect approach – it only works when we expect honest respondents of interest to cluster together and bad actors to behave randomly. It is slightly better than a naive baseline, which would guess the label with the highest frequency.

[1] We generally assume no collusion between bad actors. Since we are looking for agreement, our strongest adversaries under this model are those that identify a policy *a priori* that can lead to agreement. Lexicographic choice is one such policy that not only leads to agreement, but can be automated. Another policy it is to choose the most controversial responses. See [13] for an example of this policy as a case study in threats to validity.

As can be seen in Figure 1, accuracy is bounded by the percentage of bad actors, as we might expect; if we have a prior on the percentage of bad actors and a weak classifier, in the worst case we should always guess the most common class. Once more than 50% of the respondents are bad actors, the classification reduces to bias coin flips. Not evident from the graphs presented is that this classifier is very conservative – it never returns any false positives for this data.

It's clear that this classifier is not robust to an absence of bad actors. In the plot of percent bad actors vs. accuracy for the response mixture of NoisyLexicographic and Profiled, the accuracy increases once more than 50% of the respondents are NoisyLexicographic. For this respondent, accuracy decreases as entropy increases. For all remaining respondents, accuracy dips when the respondent composition is split 50/50 between honest respondents and bad actors.

Figure 1: Baseline accuracy data.



Supervised Approaches Since we have synthesized the data and know the labels, we should be able to predict classes using standard supervised methods.

We are comparing results using 4 different methods learned in class: Naive Bayes, Average Perceptron, Kernelized Average Perceptron, and Decision trees. The data was drawn from a pool of Lexicographic, Profiled and Random respondents, each described in Section . Random respondents are always the bad actors, whereas we can combine Lexicographic and Profiled in the same training set, or not, as the good actors. We consider that each question of the survey as a features, and we have 2 sets of data: surveys with 10 questions and 5 response options each; and surveys with 36 questions and 10 response options each. For the decision tree, we expanded the features in order to have boolean variables, giving a total of 50 features in the first set of data, and 360 on the second set of data.

Table 1 displays the accuracy on test and validation data, showing the constant proportion of good and bad actors, 80% and 20% respectively. In most of the case, the accuracy is higher than 80%, in particular, naive Bayes performs very well with Lexicographic vs. Random, with 100% accuracy.

At 20% random respondents, averaged perceptron fails to do as well as the baseline for the Lexicographic respondent, whereas the naive Bayes classifier performs as well a the baseline approach (and indeed could not do better!). The supervised approaches perform much better for the Profiled vs. Random respondents – the baseline always returns that all results are invalid, whereas both naive Bayes and averaged perceptron do much better. In fact, they ought to be compared against a less conservative baseline – one that simply returns the most common label. In that case, we would expect both Lexicographic and Profiled respondents vs. Random to return 80% accuracy, since they should always predict that the response is valid. Since the accuracy is better than this, they must be detecting bad actors.

Overall we can see that the classifiers are more successful when comparing Lexicographic and Random respondents, being followed by Profiled and Random respondents. Most of the methods had better performance when given more features. For decision trees, we can see that the accuracy decreases when we increase the maxdepth for Lexicographic vs. Random and Profiled vs. Random, what suggests that the model does not need to be complicated on those case. However, when using

the mixture, it could perform slightly better by increasing the maxdepth, indicating the model we need to learn is more complex, as expected.

Classifier	Validation Accuracy	Test Accuracy
Lexicographic (80%) vs. Random Respondents (20%)		
Naive Bayes (10 features)	100.00%	100.00%
Naive Bayes (36 features)	100.00%	100.00%
Averaged Perceptron(10 features)	97.20%	95.20%
Averaged Perceptron (36 features)	91.60%	90.40%
Kernelized Averaged Perceptron (10 features)	99.80%	99.80%
Kernelized Averaged Perceptron (36 features)	100.00%	100.00%
Decision Tree (10 feat, 5 maxdepth)	99.80%	99.60%
Decision Tree (36 feat, 5 maxdepth)	99.80%	99.60%
Decision Tree (36 feat, 10 maxdepth)	98.60%	97.60%
Profiled (80%) vs. Random Respondents (20%)		
Naive Bayes (10 features)	93.60%	93.20%
Naive Bayes (36 features)	95.40%	95.20%
Averaged Perceptron(10 features)	87.80%	87.80%
Averaged Perceptron (36 features)	95.00%	94.40%
Kernelized Averaged Perceptron (10 features)	90.00%	87.60%
Kernelized Averaged Perceptron (36 features)	97.40%	95.60%
Decision Tree (10 feat, 5 maxdepth)	98.60%	99.00%
Decision Tree (36 feat, 5 maxdepth)	98.60%	98.40%
Decision Tree (36 feat, 10 maxdepth)	90.60%	92.60%
Profiled (40%) vs. Lexicographic Respondents (40%) vs. Random Respondents (20%)		
Naive Bayes (10 features)	89.00%	90.00%
Naive Bayes (36 features)	87.60%	86.40%
Averaged Perceptron(10 features)	77.20%	73.60%
Averaged Perceptron (36 features)	95.40%	94.20%
Kernelized Averaged Perceptron (10 features)	92.00%	92.00%
Kernelized Averaged Perceptron (36 features)	96.60%	95.80%
Decision Tree (10 feat, 5 maxdepth)	91.00%	92.80%
Decision Tree (36 feat, 5 maxdepth)	81.60%	84.60%
Decision Tree (36 feat, 10 maxdepth)	81.80%	85.40%

Table 1: Classification accuracy using supervised techniques.

Unsupervised Approaches In practice, we do not have access to the underlying distribution of the data. Typically the most supervision we have available is to inject known bad actors and then separate the data.

Social scientists handle this problem by augmenting their surveys with "attention check questions" and "catch trials." Both serve as kinds of control questions – attention check questions are meant to be very basic and simply verify that the respondent is paying attention. An example might be :

> Please read the following paragraph. In the next section you will be reading about the rise of pottery in Ancient Greece. Please pay careful attention to the reading selections. For example, to answer this question, do not click next, but instead click on the image in the far right. Instructions such as these allow us to determine the best course of action when evaluation a response.

Conversely, catch trials are meant to be covert; they allow the researcher to determine if the respondent is actually part of the population of interest.

Although control questions are a tempting solution, at least one study has shown that they are too easily recognizable by bad actors and can be easily overlooked by fatigued honest respondents [].

n	k_j	m_j	μ	δ	α
90	1	3	$n/3$	0.5	0.05
70	1	3	$n/3$	0.5	0.10
34	9	10	$(9n)/10$	0.5	0.05
26	9	10	$(9n)/10$	0.5	0.10
m/k	k	m	1	1	0.72
$(2m)/k$	k	m	2	1	0.51
$(7m)/k$	k	m	7	1	0.10

Table 2: Some sample values for n, μ, δ, and α. We assume for simplicity that the k_j and m_j are the same for every question.

For our baseline in the unsupervised framework, we propose using low-frequency answers as a means to reject bad actors. In order for low-frequency answers to "stand out" sufficiently, the following must be true: *Catch trial, attention check, and related control questions have non-uniform distributions in the underlying population.*

We expect bots and random respondents to have a different distribution of responses from honest respondents. If a bot's choice of response is on the basis of position, we consider the bot a random respondent *even if the bot is choosing position on the basis of a pattern*. Since the position of unordered options are randomized, choosing a pattern of positions (e.g. always the first option, alternating options, the "Christmas Tree" pattern), is indistinguishable from choosing random positions.

If we expect an equal distribution of responses in the target population, there will be no way to tell the difference between bad actors and honest respondents. If however there exists a least popular answer to some subset of questions, we can catch bad actors.

Suppose we have n unordered questions, each having k least popular elements. We say that k_j/m_j is the probability that a random respondent will choose the least popular element of question i. Let X_j denote the event that a respondent chooses a least popular option for question j and let X be the total number of questions that are answered with a least popular option. Then the expected number of questions answered with the least popular option for bots will be

$$\mathbb{E}(X) = \mu = \sum_{j=1}^{n} \frac{k_j}{m_j}. \tag{1}$$

We want to be able to say that a respondent who differs from μ by more than a certain amount is, with high probability, an honest respondent. Since least popular answers are, by design, bad-actor-heavy, we only care about the case where a respondent answers significantly fewer of the control questions with the least popular answer. The marginalizing effect of randomization allows us to treat each X_j as a Bernoulli trial with probability k_j/m_j. Since each question is answered independently and responding with a least popular answer is a Bernoulli trial, we can use Chernoff bounds to describe the relationship between the expected number of least-popular responses to catch trials, deviation from that expectation, and our confidence that a response that deviates is indeed a human and not a bot.

Let δ be the tolerance for deviation from μ and let α be our significance level. By the Chernoff bound, we have

$$\mathbb{P}(X < (1-\delta)\mu) \leq e^{\frac{-\delta^2\mu}{2+\delta}}. \tag{2}$$

If a respondent answers fewer than $(1-\delta)\mu$ control questions, we would like to classify them as a bad actor. The probability of this event is at most $e^{\frac{-\delta^2\mu}{2+\delta}}$. If we would like our rejection region α to be less than 0.05, we should pick an appropriate δ and design the survey to have a μ such that

$$e^{\frac{-\delta^2\mu}{2+\delta}} < 0.05. \tag{3}$$

Table 2 gives some examples of possible values for each of our parameters. While the parameters are listed for a survey of all unordered questions, the settings will detect respondents that are behaving uniformly randomly.

An important detail to note is that μ is tunable, since any question can have response options added to it to turn it into a control question. Different applications will require different types of control questions. For example, if the application is an annotation task for a large number of examples, it would make the most sense to interleave known annotations throughout the survey. It may also happen that most annotations are unambiguous and so there may be a large number of catch trials already in the survey instrument. If the application is a short psychological experiment, it may be necessary to introduce catch trials. This could be done by introducing a handful of questions with many options, where honest human respondents are sure to answer in a particular way.

Table 3: Meaning of variables used in establishing bounds.

n :	the number of control questions
μ :	the expected number of control questions a Random Respondent will answer with the least popular element
δ :	the acceptable deviation from μ
α :	area under the curve corresponding to the rejection region

Baseline Unsupervised In order to implement the least popular option classifier, we need to define what it means for a response to be "least popular." In simulation, we tested by hand a number of parameters and picked an increase of 0.5 as the cutoff for whether an option was in our discriminating set.

Figure **??** shows the baseline's performance against various populations. As with the baseline for the supervised classifiers, LPO is conservative and rarely selects false positives. However, unlike the supervised baseline, it does incorrectly classify a bad actor as an honest respondent from time to time.

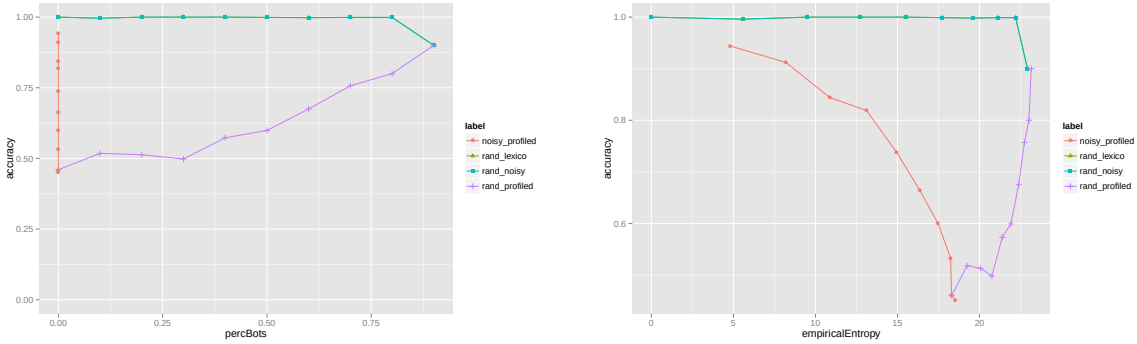
kmeans++ Unsupervised Clustering As an alternative, we consider the kmeans++ algorithm for clustering [2]. Since the surveys contain nominal data and thus do not sit in Euclidean space, we need to consider another way of measuring the distance between two surveys. We encode question responses with numbers corresponding to differentiate individual response options and then use Hamming distance to compute the distance between two vectors.

We use the most common label for the cluster as its label. Table 1 holds plots of the clustering algorithm's accuracy for varying mixtures of respondents. We ran the simulation over two surveys, to get a sense of how survey size impacted the quality of the classification. We ran the classifier for 2- and 3-clusters.

We also implemented a version that added a small amount of noise to the n-dimensional bitvectors of differences, but were sufficiently satisfied with our results that we did not use this version.

Discussion: Additional Features and Classifiers This project only used survey questions as part of its feature set. However, one could also use features particular to the web environment, such time to answer questions and mouse clicks.

If we had the time, we would like to have completed a GMM classifier for the data, to see if representing mixtures helped with classification. Since any given respondent could belong to multiple mixtures, LDA may provide some insight into the population of interest.



References

[1] Surveymonkey best practices: How many people do i need to take my survey?, 2011.

[2] David Arthur and Sergei Vassilvitskii. k-means++: The advantages of careful seeding. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 1027–1035. Society for Industrial and Applied Mathematics, 2007.

[3] Daniel W. Barowy, Charlie Curtsinger, Emery D. Berger, and Andrew McGregor. AUTOMAN: A platform for integrating human-based and digital computation. In *Proceedings of the ACM International Conference on Object Oriented Programming Systems Languages and Applications*, OOPSLA '12, pages 639–654, New York, NY, USA, 2012. ACM.

[4] Mick P Couper. Review: Web surveys: A review of issues and approaches. *Public opinion quarterly*, pages 464–494, 2000.

[5] B. Efron. Bootstrap Methods: Another Look at the Jackknife. *The Annals of Statistics*, 7(1):pp. 1–26, 1979.

[6] Andrew Gelman and John B Carlin. Poststratification and weighting adjustments. In *In*. Citeseer, 2000.

[7] Morris H Hansen and William N Hurwitz. The problem of non-response in sample surveys. *Journal of the American Statistical Association*, 41(236):517–529, 1946.

[8] D Holt and TMF Smith. Post stratification. *Journal of the Royal Statistical Society. Series A (General)*, pages 33–46, 1979.

[9] Dirk Hovy, Taylor Berg-Kirkpatrick, Ashish Vaswani, and Eduard H Hovy. Learning whom to trust with mace. In *HLT-NAACL*, pages 1120–1130, 2013.

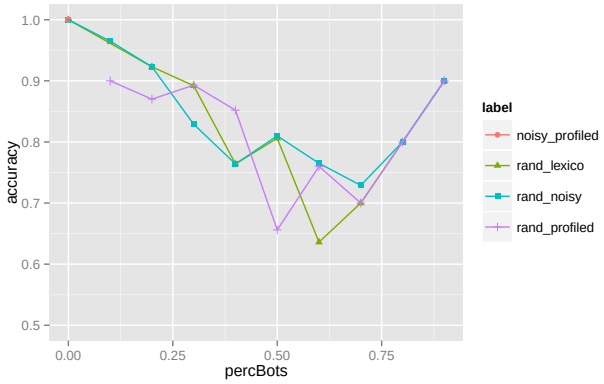
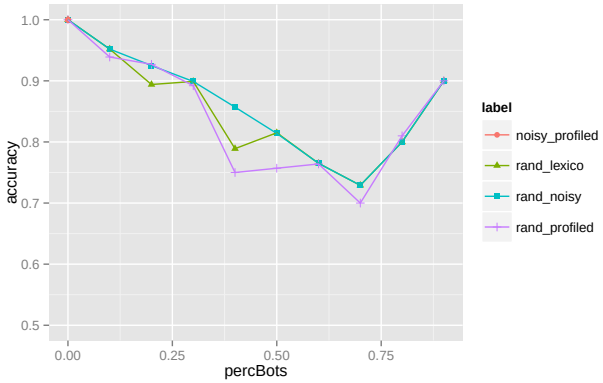
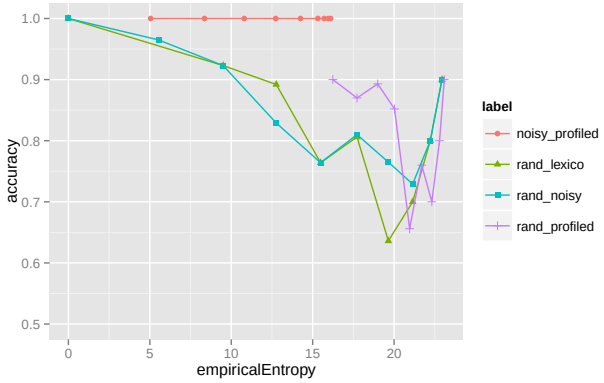
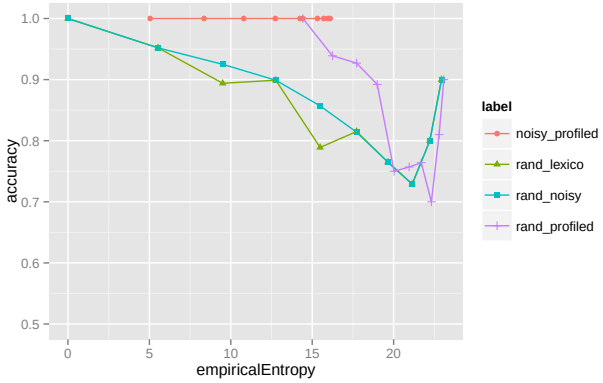
[10] Jason L Huang, Paul G Curran, Jessica Keeney, Elizabeth M Poposki, and Richard P DeShon. Detecting and deterring insufficient effort responding to surveys. *Journal of Business and Psychology*, 27(1):99–114, 2012.

[11] Panagiotis Ipeirotis. Demographics of Mechanical Turk. Technical Report NYU working paper no. CEDER-10-01, 2010.

[12] Adam W Meade and S Bartholomew Craig. Identifying careless responses in survey data. *Psychological methods*, 17(3):437, 2012.

[13] Joseph P Robinson-Cimpian. Inaccurate estimation of disparities due to mischievous responders several suggestions to assess conclusions. *Educational Researcher*, 43(4):171–185, 2014.

[14] Peter Welinder, Steve Branson, Pietro Perona, and Serge J Belongie. The multidimensional wisdom of crowds. In *Advances in neural information processing systems*, pages 2424–2432, 2010.

2 Clusters	3 Clusters
10-question surveys, each having 5 options.	
	
	
36-question surveys, each having 10 options