Programmatic Gold: Targeted and Scalable Quality Assurance in Crowdsourcing

David Oleson, Alexander Sorokin,
Greg Laughlin, Vaughn Hester, John Le, Lukas Biewald
CrowdFlower
2111 Mission Street, Suite 302,
San Francisco, CA, USA, 94110
{dave,alex,greg,vaughn,john,lukas}@crowdflower.com

Abstract

Crowdsourcing is an effective tool for scalable data annotation in both research and enterprise contexts. Due to crowdsourcing's open participation model, quality assurance is critical to the success of any project. Present methods rely on EM-style post-processing or manual annotation of large gold standard sets. In this paper we present an automated quality assurance process that is inexpensive and scalable. Our novel process relies on programmatic gold creation to provide targeted training feedback to workers and to prevent common scamming scenarios. We find that it decreases the amount of manual work required to manage crowdsourced labor while improving the overall quality of the results.

1 Introduction

Human-annotated data is the cornerstone of scientific method in machine learning. Many business applications require human judgment to supplement AI-based systems. Traditionally, this manual annotation has been carried out by small teams of human annotators over long periods of time. Although businesses often can afford to hire and manage annotator workforces, traditional hiring practices become inefficient or insufficient in the face of the demands of modern applications. These needs are characterized by unpredictable scaling, spiky traffic and short deployment cycles. Crowdsourcing - the use of large, distributed groups of people to complete tasks on demand (Howe 2008) - is one solution to these problems. Crowdsourcing has changed data annotation practices by distributing large amounts of work efficiently.

Crowdsourcing is a growing industry, and many online platforms exist for completion of microtasks. **(Amazon Mechanical Turk 2005)** was the first large scale labor-on-demand platform. It provides a powerful API for application development, international payment infrastructure and basic quality assurance mechanisms. For the purposes of this paper we also consider two more recent general-purpose platforms. **(SamaSource 2011)** is a crowdsourcing non-profit with a mission to lift people out of poverty through training and provision of digital work. **(CrowdFlower 2011)** is a general-purpose crowdsourcing platform with built-in quality assurance. Unlike other platforms, CrowdFlower does not have its own workers. Instead it pushes the crowdsourcing tasks to partners (like AMT or Samasource), who engage their worker pools to complete the available tasks. Partnering with a multitude of labor pools significantly increases scalability of the platform and diversifies worker base.

Copyright © 2011, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

Crowdsourcing has become a global phenomenon with a low barrier to entry. Millions of people around the world now participate in a digital and virtual workplace. The crowdsourcing ecosystem is one in which work is being redefined as an online meritocracy in which skilled work is rewarded in real time and job training is imparted immediately via feedback loops (Lehdonvirta and Ernkvist 2011). The breadth and diversity among crowdsourcing participants, workers and researchers is immense. This means that quality control mechanisms should account for a wide range of worker behavior: scammers, lazy workers, ethical workers, slow learners, etc.

With these services, researchers can manage a huge pool of workers on a crowdsourcing platform to perform various tasks. To name a few, (Snow et al. 2008) has shown applicability of crowdsourcing to NLP, (Sorokin and Forsyth 2008) demonstrated how to efficiently crowdsource annotation for computer vision, (Ipeirotis, Provost, and Wang 2010) applied crowdsourcing to content moderation, (Holmes, Kapelner, and Lee 2009) applied crowdsourcing to identify bacteria and (Alonso, Rose, and Stewart 2008) used crowdsourcing for assessments of search relevance.

All authors agree that crowdsourcing requires explicit quality assurance mechanism to deal with scammers, insufficient attention and incompetent workers. (Snow et al. 2008) relies on averaging and bias correction, (Ipeirotis, Provost, and Wang 2010) develops bias correction into an EM-based procedure to detect scammers. (Kittur, Chi, and Suh 2008) argues that verifiable questions must be included into otherwise subjective tasks and answering the verifiable part correctly should take as much effort as doing the whole task. (Dow et al. 2011) designs interactive and peer-to-peer feedback system to encourage quality work. (Le et al. 2010) develop a gold standard-based quality assurance framework that provides direct feedback to the workers and targets specific worker errors. The approach of (Le et al. 2010) requires extensive manually-generated collection of gold data. In this paper we present an algorithm for generating gold data automatically while maintaining important gold proper-

ties: transparency to workers, mistake targeting and in-task training. Our approach still requires manual intervention to detect errors and codify the specific gold generator, but it provides significant time savings and ensures overall scalability of crowdsourced data processing.

2 Quality in crowdsourcing

In the paper, we refer to any piece of data that needs to be processed as **unit** and denote it with u. We call any worker output **judgment** j. We apply this term more generally than the original meaning of judging relevance. We apply it to any work provided for a single unit by a single worker. We assume that there are units that have a single unambiguous answer j^*, which we call the correct answer. This is not always the case, but we must leave these special cases outside of the scope of the paper.

2.1 Worker screening

Pre-screening workers is a simple strategy: set up multiple-choice questions and ban people who do not pass the test. If the workers pass the test, assume they work well. This strategy is simple to implement and can improve accuracy by preventing unskilled or unethical workers from entering a task. It is readily available on the (Amazon Mechanical Turk 2005; CrowdFlower 2011) platforms. Other platforms (SamaSource 2011; Microtask 2011) pre-screen the entirety of their workforces.

Worker screening has limitations. The introduction of a barrier to entry affects good and bad workers alike. Unethical workers ("scammers") looking for easy tasks are less likely to put in effort to pass the screening test, but diligent workers may also opt not to enter the task. Once the worker passes the screening, there is no continued incentive to submit accurate responses. The screening module itself may be vulnerable to scammers who may share the answers to the questions. Finally, the screening test must be manually designed.

2.2 Inferring worker trust

The second strategy (Dawid and Skene 1979), introduced by (Ipeirotis, Provost, and Wang 2010) in the context of crowdsourcing, is to simultaneously infer worker accuracy and the correct answer. We can formulate an EM optimization that accounts for worker biases $p_w(j|j^*)$, estimates these biases and the correct response for each unit $p(j^*|u)$ (see (Dawid and Skene 1979) and (Ipeirotis, Provost, and Wang 2010) for details and a sample implementation). The estimation is necessarily iterative, because the resulting EM formulation is NP-hard, and needs all data to be present.

This strategy requires batch post-processing. It is necessary to obtain multiple judgments for many units of data, then to estimate worker accuracy, true answers and worker biases. One then iterates this process several times to provide true responses. Once all data is collected, it is possible to identify trusted workers and untrusted workers, the latter which should be rejected. If some units of data are missing the required number of independent trusted judgments, those should be collected again and the estimation process repeated. Alternatively, one might collect an excessive number of judgments for EM purposes (P. Welinder 2010), thereby making the process much more expensive. While this is a perfect tool to study the behavior of workers, collecting excessive number of judgments is impractical for any large-scale application.

Offline estimation introduces large delays and allows workers to submit large batches of work that might be later rejected. Most workers are highly sensitive to rejections and will contact the requester if they feel rejections were unfair. The requester would then have to respond to a very large number of e-mails. As a result, rejecting large number of judgments becomes expensive, it can damage requester reputation and deter workers. These factors make EM-style QA difficult for large scale, ongoing projects.

Gold-based quality assurance Gold-based quality assurance provides a direct estimate of worker accuracy and allows for in-task worker training. A gold standard dataset is a set of gold units, each with a known correct answer j^* and a feedback message explaining why the answer is chosen. We can explicitly estimate worker accuracy by randomly injecting gold units into the workflow. If the worker gives an incorrect answer for a gold unit ("misses the gold unit"), we show the feedback message to train the worker. The worker has an option to contest the gold message if they believe the gold answer j^* is incorrect. If their complaint is accepted (e.g. j^* is corrected), the worker answer is considered correct.

Gold-based worker accuracy is then the $a_{gold} = 1 - \frac{N_{missed}}{N_{shown}}$, where the N_{missed} is the number of gold units they missed and N_{shown} is the number of gold units they have seen. The accuracy a_{gold} is computed only after the worker has seen at least $N_0 = 4$ gold units. As the workers progress through the task, gold sampling frequency is reduced if the workers maintain high accuracy levels. If at any point in time the worker falls below task-specific accuracy threshold t_{reject}, they are rejected from the task. To increase transparency, we warn workers when their accuracy falls below a higher threshold t_{warn}.

To ensure that workers are adequately prepared to work on the task, we start with a training session. In training, workers see only gold units and receive feedback for every mistake made. Training continues until the worker has completed $N_{training} = 4$ gold units correctly. After that the workers are switched into regular work mode. The mistakes made in training mode do not affect the worker's accuracy estimate for subsequent questions answered. The workers are paid for training responses only upon successful completion of training mode.

Gold-based QA has a number of advantages. First, it explicitly measures worker accuracy. This accuracy can then be used to make decisions regarding the worker: can we use their work? Should we block them? Do they deserve a bonus? Second, the process is transparent to the workers. They see the current accuracy estimate and how each missed gold affects it. Last, workers receive in-task training. By presenting special cases as gold units, we can ensure that workers understand the nuanced or challenging details

of the task requirements. By balancing the answer distribution within the gold set like in (Le et al. 2010), we can ensure that the workers are trained to choose the correct answers in edge cases that may only comprise a small fraction of the data.

3 Challenges of gold-based quality assurance

Gold questions are used to improve the accuracy of results collected in crowdsourced tasks. CrowdFlower relies extensively on gold-based quality assurance efforts in all projects. These questions aim to: a) remove unethical workers from a task and b) educate untrained or incompetent workers to improve the accuracy of their responses. Key considerations for successful implementation of gold units include the time, cost and efficacy of gold units.

3.1 Size of gold dataset

Repeat exposure to specific gold units increases the likelihood that a worker will recognize a particular unit (particularly one which he or she has answered incorrectly previously) as belonging to the quality control system. This can result in a subsequent minimal effort by a worker on non-gold units. In jobs with a high number of units, if a particular gold unit is viewed repeatedly by a single worker, it could invalidate the accuracy estimate (Figure 1). Limiting the maximum of amount of work per worker may minimize repeat views of gold units, but can also result in decreased throughput. Use of extra (or excessive) questions (or very large gold datasets) to minimize repeat views of gold units involves a considerable cost. Most tasks with automated quality control will compensate workers for all judgments submitted during a task up until the point at which the worker fails to meet a minimum accuracy level. In addition, there are costs of internal resources and time for creation, oversight and responses to worker feedback when quality control strategies raise questions or confusion.

3.2 Composition

The selection of gold units should focus on those with objective, irrefutable true answers. True answers should encompass as wide/even of a distribution as possible; no single response category should dominate the gold unit distribution. Gold units can provide an active learning environment if accompanied by a clear, specific reason as to why a particular response would be the correct answer. This process flows under the assumption is that the creators of these gold units (know as "gold-diggers") are ethical and strategic: they must plan ahead to anticipate likely types of worker errors on a particular task. The creation of gold units manually is a time-consuming and expensive process.

3.3 Disguising gold units

A key best practice for optimizing crowdsourcing tasks is iteration. However, this iteration is frequently accompanied by minor adjustments to the language or images in each question which can expose that unit as belonging to a previous version of the job. Gold units must be undetectable from the non-gold units in a job, and iteration becomes difficult

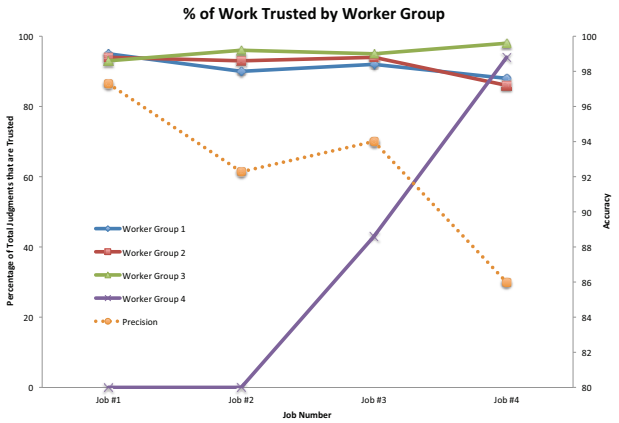

Figure 1: Observed example of accuracy degradation over time. In Job 1, there is new gold, workers from group 1 cannot pass the gold accuracy targets, none of their work is counted, and accuracy is at 97%. By Job 4, group 1 workers have learned the gold, allowing them to submit inaccurate judgments on non-gold units, and overall accuracy is down to 85%.

when gold units must be manually recreated due to iteration in the job. All data fields and visual clues in one job must be consistent, otherwise workers will have a clear opportunity to scam the task and extract compensation correctly answering (only) the gold units. The accuracy on non-gold units is severely impacted when analyzed post-processing.

3.4 Detecting new error types

Creation of gold units must be as iterative as the task design, layout, and UI. Processing multiple batches of data will, however, expose the most frequently occurring patterns of worker behavior and errors. Identifying these errors makes it possible to then target them via specific types of gold units: those which ask the workers to make judgments on challenging edge cases, to complete a specific type of research, or to complete a particular field within the task interface.

3.5 Scammer behavior

Strict quality control measures give rise to a variety of types of scamming, generally by workers who are removed from a particular task (for failing to maintain the 70% accuracy required). Creation of multiple worker accounts by the same individual is common if a worker has too low of a trust rating to continue working on a particular job of interest. Workers may collude regarding gold unit answers in order to inflate their accuracy ratings, especially if they are located in the same physical location.

A final consideration is that it can be difficult to generate gold data that targets specific worker behavior or errors on a particular task. Traditionally, gold units have been manually created using source data units from the batch of source data from the present job; this is a time-consuming, laborious process. Additionally, it is a process that is difficult to

a) rapidly scale and b) adapt to precisely target certain behavior. The experiments in this paper attempt to reduce the effects of many of these challenges. Our work extends basic, traditional strategies of quality control in crowdsourcing by attempting to mitigate the impact of the scenarios listed here.

4 Programmatic gold

Programmatic gold is a process of generating gold units with known answers. New units are generated from previously collected correct data. By injecting known types of errors into the data, we gain control over the training experience of the workers and force all workers to consider most common error cases. When the workers make mistakes, they see automatically-generated feedback. We refer to this manipulated data as **pyrite**. The simplest application of programmatic gold is use of high confidence answers to benchmark future workers. This approach is similar to EM-based quality assurance, however it does not provide as clear of an impact with respect to targeting specific errors as **pyrite** does.

The first step in creating **pyrite** is identifying worker errors through manual audits. Then, mutating the conditions for correct worker responses so that pyrite directly targets the most common errors by workers. For example, in a URL verification task, workers might have difficulty with businesses which are similarly named but are located in the wrong city. For instance, if a local search engine wanted the "Bob's Restaurant" located in Seattle, WA, but the URL provided was for a "Bob's Restaurant" in Portland, OR, workers often have difficulty distinguishing between the two and incorrectly state the URL is for the right business in the given city and state. As described below, it is possible to generate pyrite that directly targets this type of worker error ("city-state" gold). Worker errors usually fall into discrete sets, once these are identified through manual spot checks, programmatic gold can be generated for the universe of worker errors, insuring consistently high accuracy of results.

The second step is to define a set of data mutations $\mu(u, j^*, U)$. Each mutation applies to a single unit u with correct judgment j^*, but generally requires all available data as input. The mutations alter individual attributes of the data to produce a new unit that (a) differs the original unit enough to violate task requirements and (b) looks similar to original data. For example, to create city-state gold, we assigned variables to each units address, city, state, postal code, and phone number ("location information"). We randomize each unit's location information with location information from another entity. For instance, Gramercy Tavern in *New York , NY, 10003 212-477-0777* would now have the location information from an entity outside of New York. Workers would be presented with the information Gramercy Tavern, *San Francisco, CA 94103 415-555-1234*. The link (http://www.gramercytavern.com) takes workers to the NY-based Gramercy Tavern website. The workers should answer "no" because the entity is not in the correct location or with the correct phone number.

Another type of pyrite is name gold. Name gold targets behavior where workers would say that the website for Enterprise Lawyers located in Durango, CO, was the correct website for Enterprise Group, also located in Durango, CO. To address this problem, we create name gold by taking answers where workers had answered yes with 1 confidence, then parsing out the words into their component pieces (i.e. first word, second word) and rearranging both the component pieces of the names, as well as the last 7 digits of the phone number.

For instance, for the unit Nulook Party in Greenville, NC with phone number *252-367-7444*, we scrambled the name and the phone number so that the gold unit would be Automated Party in Greenville, NC with phone number *252-8985-9733*. Worker have to identify that this is not the correct website.

Each mutation function produces the gold unit that corresponds to a particular error ϵ identified earlier, so we use the description of the error to automatically generate the feedback message. If the workers answered city-state gold incorrectly, they would see a message of "The URL is for Gramercy Tavern located in New York, NY. We asked for the Gramercy Tavern in San Francisco, CA. Be sure to check that the restaurant is located in the correct city and state!" If workers answered name_gold incorrectly, they would see a message of "The URL is for Nulook Party in Greenvile, NC, but we asked for Automated Party. Be sure that the name is the same!"

The third step is to collect a set of units with known correct answers. When such a set is not available, we start with a small data run that requires little gold to complete (as described below in Experiment 2). The results with highest agreement are then used as source data for programmatic gold. If there is another set of accurate data, that can be used instead (as described below in Experiment 1). Occasionally all workers agree on an incorrect response, and an incorrect response is turned into a gold unit. Since each gold unit is seen by many workers in the course of normal data processing, workers contest the wrong gold and we automatically disable the gold unit if the gold contention rate is high. This makes programmatic gold tolerant to inevitable errors in the source data set.

The last step is to define the target distribution of gold. As shown by (Le et al. 2010), sampling from underlying distribution of examples is inferior to sampling uniformly from possible worker errors. We follow that idea and generate gold with uniform distribution of target errors identified by manual spotchecks. For each sampled error ϵ we select a source unit u. We then apply the transformation $\mu_\epsilon(u, j^*, U)$ corresponding to this particular error ϵ to generate gold units. We automatically attach the respective error description as feedback for the generated gold unit. Note that the errors ϵ are sampled with replacement, while the source units are sampled without replacement to prevent detectability.

5 Experiments

We evaluate programmatic gold in two experiments: Experiment 1 compares the effect of manual gold with programmatic gold, while Experiment 2 tests the scaling up of the gold set from 10 intial gold units to a total of 22,000 units.

For Experiment 1 we used a dataset of 9,659 business listings which had web addresses. The goal was to first verify the correct name of the business, and, if the business name was different on the website vs. the information in the business listing, change the business name to the name listed on the website. Workers were instructed to find the correct business name on the website and, if the name provided by the local index was incorrect, copy and paste the correct name from the URL onto the page. Some businesses have different names for specific locations or divisions of the business. In that situation, workers were asked to use the name that best matches the location or division.

We used previous data runs to look for units where workers agreed with each other with 1 confidence. Then, we turned those units into gold units (without any additional transformations), and processed the data again. To measure the effect of programmatic gold, we performed 3 separate data runs: with no gold, with manually-dug gold and with programmatic gold. To measure accuracy we performed internal audits (spot checks) of 100 units on each set. For each unit, we measure if the aggregate result is correct or not. The accuracy is then the percentage of correct aggregate results in the spotcheck. Different kinds of errors are not reflected in the accuracy measure - any incorrect result is treated the same.

For Experiment 2 we used a 22,000-unit subset of a larger set of 500,000 local business listings with business web address (candidate URL), business name, physical address, and phone number data. The goal was to verify whether or not the candidate URL was correct.

Workers are given the business data and the URL candidate and are instructed to answer "yes, this is the correct url" (a "yes unit") if it meets at least one of the following conditions:

1. A location of the business on the website is located in the provided city or has the provided phone number and the business name on the website is similar to the provided name
2. A location of the business on the website is located in the provided city and the business name on the website is nearly identical to the provided name

To process such a large dataset, we would need, at minimum, 200 gold units for every 20,000 units of data processed. When processing this data at the rate of 20,000 units/day, gold units must be refreshed on a daily basis. These units would have to be created manually. As explained in Section 3, gold-diggers would have to identify instances in which workers are likely to make mistakes, turn them into gold units and provide the correct answer and feedback message. This process is time-consuming and it is challenging to find good gold candidates that target specific worker errors.

5.1 Scaling up with programmatic gold

In the aforementioned set of 22,000 units, we first manually created 10 gold standard test questions. We used those gold standard test questions to maintain high worker quality on the processing of 300 listings. Those 300 listings yielded 60 listings on which multiple workers had agreed that the website in question was correct. We then turned those 60 units into programmatic gold units: 30 units using the city-state gold algorithm described above, and 30 "yes units" with confidence values of 1.0 (indicating perfect agreement) were turned into "yes" gold units. We then used these 60 newly created gold units to process 3,000 units with a limit of 1,000 judgments per worker.

After processing those 3,000 units, we audited the worker results and found that workers were missing similarly named businesses located in the correct city. To educate workers, we created name gold as described in Section 4.

We then used 200 programmatically created gold test units (100 "yes" gold units, 50 "no" city-state gold units, and 50 "no" name gold units) to process the next 3,000 units with a limit of 1,500 judgments per worker. On the next iteration we created 233 gold for 15,000 units and the same 1,500 judgments per worker limit, refreshing the gold units so that workers would not see the same gold unit multiple times. We performed a 100-unit spot check on each set as described above with the exception of 15,000-unit set, where we evaluated 253 units.

	Gold		
	No	Programmatic	Manual
Experiment #1	88%	99%	99%
Experiment #2	83%	92%	N/A

Table 1: Results accuracy: No gold, Manual and Programmatic gold

6 Experimental results

The results of Experiment 1 are shown in Table 1. Programmatic gold yielded 99% accuracy: which is comparable to the results using manual gold. Comparing this to the no-gold scenario, in which only 88% of responses were accurate, shows a substantial difference in the quality of the results.

Experiment 2 shows that the programmatic gold allows us to scale up the task and reduces the amount of manual gold digging. As shown in Table 2, the use of programmatic gold resulted in overall accuracy of 92.2%. The baseline accuracy for this job is 85%. The most common scenario in this sample of data is that the URL is not correct and no actual work can be performed for that unit. However, it is the other cases that we are really interested in. As mentioned in Section 3, gold balancing is extremely important in this case. By using uniform distribution of **non-trivial** examples, we prevent scammers and lazy workers from getting a free pass on our task. While maintaining the accuracy level of 92% at scale, programmatic gold also reduced the amount of gold digging by a factor of **50** from 511 to just 10.

In Table 1, accuracy dropped from 95% in the 1st iteration with manual gold to 89% in the 2nd iteration with programmatic gold before increasing to 94% in the 3rd iteration and dropping to 92.5% in the 4th iteration. We note that the ratio

Gold	Units	Accuracy	MAX work/worker	Gold Ratio	Time (hr)	Accuracy at confidence 1.0
10 manual	213	95.0%	100	1:10	4	100.0%
60 programmatic	3,004	89.0%	1,000	1:17	5	96.0%
208 programmatic	3,103	94.0%	1,500	1:8	9	97.0%
233 programmatic	15,509	92.5%	1,500	1:7	27	97.0%

Table 2: Scaling up with programmatic gold

of gold units to the amount of work a single worker can do can affect the accuracy of the results and should be a very important consideration in task design. The gold:unit ratio was 1:10 in the first job, 1:17 in the second, 1:8 in the third and 1:7 in the fourth. As workers in the second job have higher recollection of gold, they could put less effort into non-gold units. Programmatic gold allows project managers to scale the ratio of gold to units without inflated costs and to minimize repeat viewing of a gold unit by a worker. Accuracy is thus increased and scammer activity is reduced.

7 Discussion

In this paper we present a novel method to achieve scalable quality control for crowdsourcing. Previous methods relied on batch post-processing or manual gold digging, an expensive and time-consuming process that is difficult to scale. Programmatic gold is an efficient way to generate gold. It is as easy to create 100 gold units are it is to create 10,000 gold units. This allows us to lift the limits on the maximum number of units per workers, allowing the best workers to work as much as they want, increasing both accuracy and throughput.

As presented here, programmatic gold relies on manual spot checks and detection of worker errors. In the future, we plan to develop methods to automatically detect common errors and identify confusing cases for manual analysis. The applicability of programmatic gold is limited to tasks with definitive answer, which comprise a large share of crowdsourcing tasks that we run. As presented here, programmatic gold can't be applied to highly subjective tasks.

Programmatic gold mitigates the challenges associated with manual gold creation (as described in Section 3). We reduce the chance that two workers see the same gold unit, provide extensive feedback to train workers and target specific worker errors. Programmatic gold creation improves the quality and scalability of crowdsourced data collection. We currently apply this strategy to most tasks that we run.

8 Acknowledgments

The authors of this paper are indebted to the whole team at CrowdFlower. We specifically thank Adam Abeles for observing the effects of programmatic gold on scammer detection and prevention, Aaron Spector for running some of the initial experiments with programmatic gold, and Lukas Bergstrom for discussions and help in editing the paper.

References

Alonso, O.; Rose, D. E.; and Stewart, B. 2008. Crowdsourcing for relevance evaluation. *SIGIR Forum* 42(2):9–15.

2005. Amazon mechanical turk. http://www.mturk.com/.

2011. Crowdflower. http://www.crowdflower.com/.

Dawid, A. P., and Skene, A. M. 1979. Maximum likelihood estimation of observer error-rates using the em algorithm. *Applied Statistics* 28(1):20–28.

Dow, S.; Kulkarni, A.; Bunge, B.; Nguyen, T.; Klemmer, S.; and Hartmann, B. 2011. Shepherding the crowd: managing and providing feedback to crowd workers. In *Proceedings of the CHI Extended Abstracts on Human factors in computing systems*, 1669–1674. New York, NY, USA: ACM.

Holmes, S.; Kapelner, A.; and Lee, P. P. 2009. An interactive java statistical image segmentation system: Gemident. *Journal of Statistical Software* 30(10):1–20.

Howe, J. 2008. *Crowdsourcing: Why the Power of the Crowd Is Driving the Future of Business*. New York, NY, USA: Crown Publishing Group, 1 edition.

Ipeirotis, P.; Provost, F.; and Wang, J. 2010. Quality management on amazon mechanical turk. In *KDD-HCOMP '10*.

Kittur, A.; Chi, E. H.; and Suh, B. 2008. Crowdsourcing user studies with mechanical turk. In *CHI*, 453–456.

Le, J.; Edmonds, A.; Hester, V.; and Biewald, L. 2010. Ensuring quality in crowdsourced search relevance evaluation: The effects of training question distribution. In *Proceedings of the ACM SIGIR 2010 Workshop on Crowdsourcing for Search Evaluation (CSE 2010)*.

Lehdonvirta, V., and Ernkvist, M. 2011. Knowledge map of the virtual economy. http://www.infodev.org/en/Publication.1056.html.

2011. Microtask. http://www.microtask.com/.

P. Welinder, P. P. 2010. Online crowdsourcing: rating annotators and obtaining cost-effective labels. In *CVPR*.

2011. Samasource. http://www.samasource.org/.

Snow, R.; O'Connor, B.; Jurafsky, D.; and Ng, A. Y. 2008. Cheap and fast—but is it good?: evaluating non-expert annotations for natural language tasks. In *EMNLP '08: Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 254–263. Association for Computational Linguistics.

Sorokin, A., and Forsyth, D. 2008. Utility data annotation with amazon mechanical turk. In *First International Workshop on Internet Vision, CVPR 08*.