Mentor: A Visualization and Quality Assurance Framework for Crowd-Sourced Data Generation

Siamak Faridani, Georg Buscher, Johnny Ferguson
Microsoft
One Microsoft Way,
Redmond, WA 98052

Abstract

Crowdsourcing is a feasible method for collecting labeled datasets for training and evaluating machine learning models. Compared to the expensive process of generating labeled datasets using dedicated trained judges, the low cost of data generation in crowdsourcing environments enables researchers and practitioners to collect significantly larger amounts of data for the same cost. However, crowdsourcing is prone to noise and, without proper quality assurance processes in place, may generate low quality data that is of limited value. In this paper we propose a human-in-the-loop approach to deal with quality assurance (QA) in crowdsourcing environments. We contribute various visualization methods and statistical tools that can be used to identify defective or fraudulent data and unreliable judges. Based on these tools and principles we have built a system called *Mentor* for conducting QA for datasets used in a large commercial search engine. We describe various tools from *Mentor* and demonstrate their effectiveness through real cases for generating training and test data for search engine caption generation. Our conclusions and the tools described are generalizable and applicable to processes that collect categorical and ordinal discrete datasets for machine learning.

Introduction

Human judgments are frequently used in academia and industry to train and evaluate a great variety of machine learning models. For example, a key evaluation criterion for information retrieval systems comprises manually assessing the relevance of query-document pairs (Hofmann, Behr, and Radlinski 2012) on top of which metrics like NDCG are computed (Järvelin and Kekäläinen 2002). Human judgments are usually collected at scale using dedicated trained judges or crowdsourcing (Alonso, Rose, and Stewart 2008). To collect data using crowdsourcing is significantly cheaper than using trained judges but comes at the cost of poorer data quality which makes it more difficult to use for training and evaluation of machine learned models. To reasonably deal with this cost, proper quality control mechanisms are of high importance. While data quality can be influenced by the task design (Callison-Burch and Dredze 2010a), e.g., using concepts of collaborative games (von Ahn 2006), in this work we assume that the task design is static and focus on quality control of crowd-sourced labeled datasets.

A crowd-sourced task commonly known as a *Human Intelligence Task* or *HIT*, presents input information to the judges (also known as workers or Turkers) which they have to label, usually by answering simple questions. For more complex tasks, judges are often provided with guidelines specifying what aspects they should pay attention to when assigning a label.

There are generally two preeminent reasons for poor data quality using crowd judges in practice, i.e., *spammer* judges who produce very noisy to random labels, often in automated ways, and *judges with low data quality output*. A low quality judge might be providing low quality labels because he is not trained well or has not studied, followed, or understood the task guidelines completely. In contrast to spammers, low quality judges do not provide random data as much, but often make systematic mistakes.

Detecting spammers and identifying and potentially correcting low quality judges in practice is a time-consuming process for large-scale data gathering applications. Several statistical models as well as fully automated algorithms have been studied in various settings targeting very specific scenarios. e.g., spammer detection (Smucker and Jethani 2012), bias correction (Ipeirotis, Provost, and Wang 2010), etc. Such techniques can be combined and can feed in to an overall comprehensive system for judge quality control.

In this paper, we report on best practices for active judge quality control. Instead of using purely automated algorithms, we use a human-in-the-loop approach and apply algorithms to produce easy to understand visualizations and quantitative data summaries for a crowd admin to quickly identify quality issues and potentially correct problems in judges, judgments and the final dataset. We explain what types of problems the different visualizations and data summaries can point out, and and when they are most useful.

The contributions of this paper are threefold:

- We propose a human-in-the-loop approach to monitor the quality of data and judges in a crowdsourcing environment.
- We introduce and explain the value of various visualiza-

tion techniques and statistical elements to diagnose judge and data quality problems.

- We share a number of best practice principles based on our experience managing crowd judges.

We have built an effective large-scale data generation pipeline and dashboarding system, internally called ***Mentor*** based on these tools and principles. *Mentor* is used for training and evaluating machine learning models that are in use in a large commercial search engine.

Overview of *Mentor*, A quality assurance framework for training data generation

The purpose of *Mentor* is twofold: Firstly, from a data quality standpoint, it allows the crowd admin[1] to find out which judgments are noisy and should be removed from the dataset, or if the dataset needs to be regenerated again completely. Secondly, it helps the crowd admin produce training plans for the judges and identify low quality judges that should not be qualified for the next batch of work.

A high level overview of the *Mentor* framework is presented in Fig. 1. A domain expert prepares the task including appropriate guidelines and the raw data that needs to be presented to the judges. Crowd judges go through the guidelines before they can start judging and then produce data labels. Once all task units have been judged, the labeled dataset is uploaded into the *Mentor* dashboarding system where it is automatically aggregated, analyzed, and visualized. Next, in order to keep the judge pool as healthy as possible for the next round of labeled data generation, the crowd admin studies the judge quality indicators provided by the dashboard and may disqualify spammer judges or translate *Mentor*'s feedback into training plans for low quality judges. At the same time, the crowd admin inspects the data quality summaries and drilldowns for the labeled dataset in the dashboard and decides what data has to be discarded and whether systematic biases in the data should be corrected before using it for machine learning training or evaluation.

In this paper, we use the judgment task of indicating the relevance of summary snippets of web documents as they are displayed on search engine results pages as an example scenario. However, the presented principles can be generalized to various tasks especially in cases where the judgment labels are categorical or ordinal.

In our example scenario, we use various types of judgment tasks. Most HITs ask for only one labeled data while super HITs ask for labels on many items at the same time. Judges provide two types of judgments. In the first stage a judge will flag a data point if it is a defect. Defects are removed from the set before being used for any machine learning training or evaluation. In our case, defects are often computer-generated document summaries that are not readable, may contain adult content, or might be in another language not familiar to the judge. If the input data is not a defect then the judge will move to the next step and provide a label for the data point. After the judgment of a dataset is complete, *Mentor* produces visualization metrics that can be easily looked at by the crowd admin. Performance metrics for a judge are measured against a ground truth for which we often use gold standard data sets (Callison-Burch and Dredze 2010b). In cases that gold standard data is not at hand, *Mentor* uses a ground truth inference model to generate ground truth from current judgments. There are various models available to the crowd admin to choose from, i.e., Majority, GLAD (Whitehill et al. 2009), or Minimax Entropy (Zhou et al. 2012) for categorical labels, and for cases where the data is ordinal we can also use Mean, Median and or a modified EM model to infer the ground truth. The crowdsourcing platform may have dynamic stopping rules for how many judgments per item it needs. As a result the number of judgments per item may not be equal for all items.

In the next step the crowd admin looks at the various human-readable visualizations that are produced by *Mentor*. These metrics allow her to perform two tasks, i.e., 1) produce a performance report and build a training plan for a single judge, and 2) analyze the quality of the produced data set and decide whether or not to use the set for machine learning training or evaluation. The admin is also able to modify the dataset, e.g., by discarding judgments from bad judges.

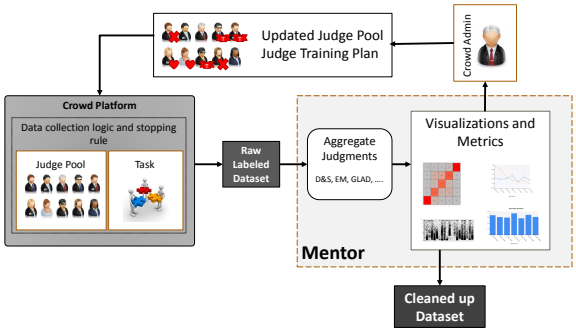


Figure 1: The data generation and quality assurance pipeline (*Mentor*)

Visual and Numerical Judge Quality Metrics

We found it valuable to look at five levels of metrics and scorecards for quality assurance, i.e., metrics for

1. Each individual judged item
2. Each individual judge
3. The judge pool (group of judges)
4. The dataset (collection of judged items)
5. The collection of datasets with various judge pools

There are various statistics, metrics and visual diagnostic tools that can be used for each one of the levels. Detailed descriptions are following in the next sections.

There is a hierarchical relationship between these levels of metrics. The crowd admin can look at the highest level of metrics to see how the metrics change over various datasets

[1]Crowd admin, also known as *data admin* or *crowd operator* who is the person who oversees the quality of data

that are collected for the same type of task, usually at different points in time. She can then drill down and look at the metrics for each dataset and examine the judge pool that worked on that dataset. She can further drill down to each individual judge and see the judge's performance on various datasets. Finally the crowd admin can look at individual judgments for an item in order to find interesting examples that can be used, e.g., in training plans.

Simple Metrics for Judge Quality

Binary Agreement/Disagreement: The binary agreement metric for a judge is the probability that the judge will insert the correct label for an item. It is calculated by taking the empirical probability in Eq. 1 in which C_j is the number of times that judge j's label for an item agrees with the ground truth, and G_j is the total number of items judged by j for which the ground truth is available.

The binary agreement is useful when multiple discrete classes of labels are present without any ordinal relationship. Fig. 2 shows agreement scores for judges in a pool sorted in descending order.

$$A_j = \frac{C_j}{G_j} \tag{1}$$

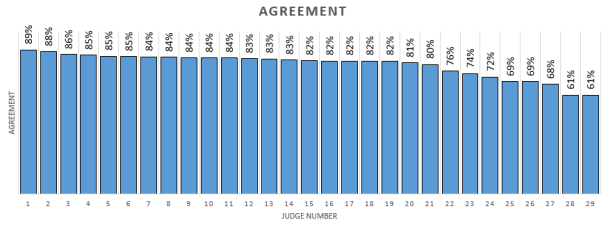


Figure 2: Agreement values for the pool of judges. An agreement diagram allows a crowd admin to quickly identify underperforming judges.

Judge Pool Accuracy (JPA)

It is beneficial to have an estimate for the expected number of correct labels based on the quality of the judges. We contribute the Judge Pool Accuracy metric (JPA) as depicted in Fig. 3. Based on the individual agreement values for the judges in the pool, the aggregation function (i.e., majority vote) and the work routing policy JPA is the expected number of correctly labeled items. In simple settings it is easy to derive a closed form expression for JPA. However as the form of the market becomes more complicated, closed form solutions become significantly harder to derive. For various work routing policies in the market we use simulations to calculate the value of JPA based on the agreement values of the judges with the ground truth.

Absolute Errors and Squared Errors

Both JPA and Agreement are used when there exists no ordinal relationship between the alternatives for the input label. However when labels are ordinal, one can use various error

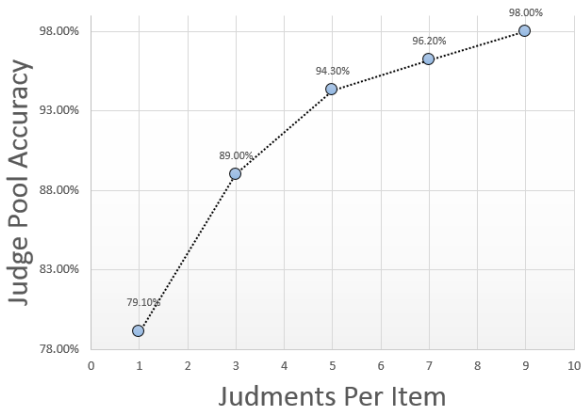


Figure 3: Judge Pool Accuracy (JPA) is the approximate percentage of items that are expected to be correctly labeled by the pool of judges. In *Mentor* these values are calculated by simulating a judging process based on the distribution of binary agreement values for the judges.

metrics instead or in addition. The simplest of these measures is the *absolute error* which is defined as the Euclidean distance from the input label x_i to the ground truth x. It is computed as Absolute Error $= |x_i - x|$. Mean Squared Errors (MSE) and Root Mean Squared Errors (RMSE) are two other measures of error that represent the variance of the error estimator and the standard deviation of the estimator.

Custom Error Measures (CEM)

We also use custom error measures for which we assign penalties based on the input label and the ground truth. CEM can be represented as matrices. For example, Table 1 demonstrates the CEM penalty table for a crowd-based query-document relevance measurement. One of the advantages of these tables is that they can be used in the absence of ordinality in labels. For example, while they both have the same error, classification of a *not relevant* pair of query-document as *very relevant* might be more problematic than classifying a *very relevant* pair as *not relevant*. As a result, we assign a higher penalty in the former case.

		Ground Truth		
		Very relevant	Somewhat relevant	Not Relevant
judgments	Very relevant	0	-3	-9
	Somewhat relevant	-1	0	-2
	Not Relevant	-5	-3	0

Table 1: CEM tables can be used for categorical and ordinal data. It can model errors and agreements and is flexible.

CEM tables can be considered as a methodical way of extending error and agreement metrics to a more flexible error measure. When the labels are finite, CEM tables provide a convenient way of quantifying how critical the error in the judgment is. The asymmetric nature of these tabels provides an additional level of flexibility to quantifying judge errors.

Error	Categorical	Ordinal
Binary Agreement	✓	×
MSE, Error, RMSE	×	✓
Custom Errors Measures (CEM)	✓	✓

Table 2: Applicability of error measures.

How to select the proper error measure

Table 2 summarizes the applicability of the error measures. Selecting the proper error measure is task-dependent. For complex tasks, selecting the correct metric can be challenging and is usually performed by metrics specialists. For the cases where the error measure is not straight-forward to select, we follow two guidelines:

1. By consulting the team that will be using the dataset (i.e., the *feature team*), we make sure that the proper error measure as well as appropriate error weights are used.
2. If the proper error measure is hard to select, we ask expert members of the feature team to judge a small set of tasks along with some of the judges. We examine the errors of both the feature team experts and the judges and sort judges and experts based on various error measures. The ideal error measure will place our own internal judges as the best judge before the crowd sourcing judges.

Judge Improvement Graph

As part of *Mentor*, we introduced a Judge Improvement Graph (JIG). The Judge Improvement Graph is designed based on the *parallel coordinate visualization* (Inselberg and Dimsdale 1991) to highlight the improvement of a judge over time compared to others. Each vertical bar consists of the name of the judge or an ID. The Y axis represents the desired quality metric. For example Fig. 4 presents the binary agreement of each judge[2]. For each dataset, the judges are sorted on the y axis from the lowest agreement to the highest. The improvement of each judge over time can be examined by connecting the position of the judge over all the datasets. For a large number of judges in the pool, the diagram can be represented by using a dynamic query as in the parallel coordinate elements in Protovis[3].

Color Coded Confusion Matrices

Confusion Matrices can be powerful visualization tools for diagnostics and training when the labels are ordinal or categorical. Fig 5 shows a confusion matrix for a judge with normal behavior. Each element of a confusion matrix highlights the performance of the judge against the ground truth. In the ideal case, ground truth should come from a gold set for which the data is carefully labeled by experts. As previously mentioned, when a gold set is not at hand, the ground truth may be generated by employing a ground truth inference model such as D&S, GLAD or Minimax Entropy

[2]To protect the identity of our judges we do not use real data and each judge is represented by an ID

[3]http://mbostock.github.io/protovis/ex/cars.html

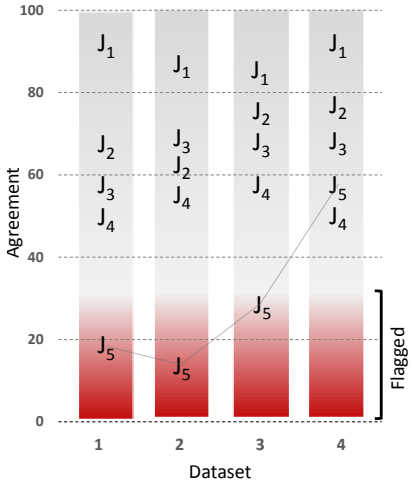


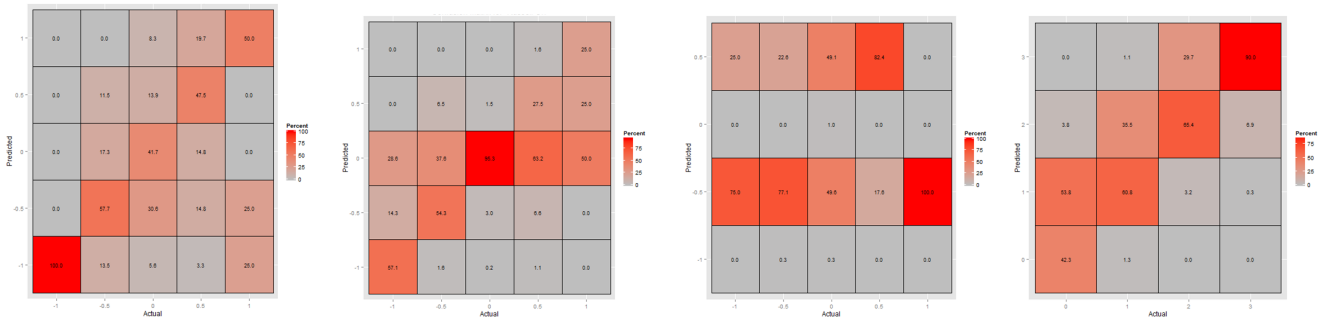
Figure 4: An example JIG graph for the agreement of 5 judges over 4 different datasets. In parallel coordinate visualizations the improvement of a judge can be highlighted by hovering over his ID.

(Dawid and Skene 1979), (Whitehill et al. 2009), (Zhou et al. 2012).

Confusion Matrix with Percentages

When presented in percentages, each element of the confusion matrix is equal to one minus the error of the judge for that particular label. In Fig. 5 the labels are from a number line [-1, -0.5, 0, 0.5, 1]. We showed two short summaries of the same web document to the judge that were the output of two document summarization models A and B. The judge indicated the relative relevance of the short summaries to the document as follows: "1" if for a web document, model A has clearly produced a better summary, "0.5" if the summary by model A is slightly better, "0" if the two summaries are equal, "-0.5" if the summary by model B is slightly better, and "-1" if the summary by model B is clearly better. The judge was provided with a detailed guideline and examples to be able to accurately judge the pair. In the confusion matrix, columns demonstrate the ground truth and rows demonstrate the input from the judge. For example, the value in the cell [row = -0.5, col = 0] = 10.3 indicates that the probability that the judge inserts label "-0.5" for a "0" pair is 10.3%. Similarly the probability that the judge correctly labels a "-0.5" value is 68%. Here, the intensity of the color red is determined by the value in each cell. And since we have ordinal data, large diagonal cell values are desired meaning that the judge labeled most items correctly. In the ideal case, the confusion matrix should have red diagonal elements (100%) with gray cell values (0%) elsewhere.

Confusion matrix for a careless judge Fig. 6a demonstrates an example of a careless and therefore noisy judge. The judge has been able to identify label "-1" correctly but the performance on other labels are noisy. On the other hand, the judge is biased toward label "-1" as shown by his or her performance on label "1".



(a) (b) (c) (d)

Figure 6: Visualization of the confusion matrices for different types of judges. (a) Confusion Matrix for an uneducated judge. While the error rate is low for "-1", two diagonal elements have a value below 50%. (b) The confusion matrix for a center-leaning judge. The middle row is more red compared to the confusion matrix of a good judge. (c) Confusion Matrix for a spammer. The spammer has never provided label "1" and has rarely inserted a "-1". (d) Confusion Matrix for a systematically biased judge. Overall this judge avoids providing labels on the extreme ends of the label spectrum. He is frequently off by one label.

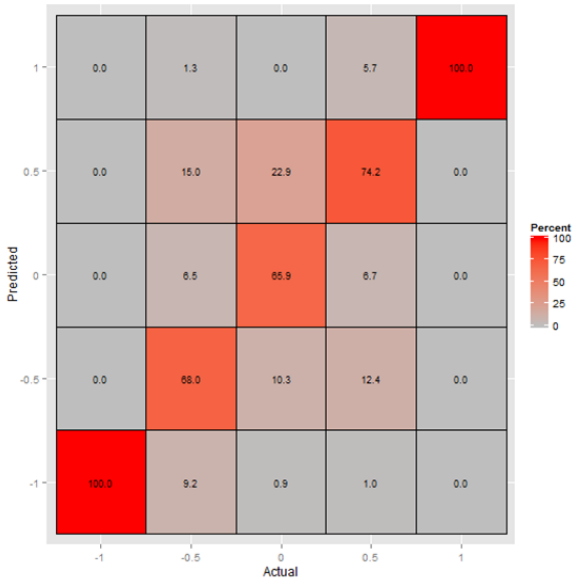


Figure 5: Confusion Matrix for a good judge.

Confusion matrix for a center-leaning judge One of the problems in producing labeled datasets for machine learning models is the fact that a judge may not be willing to spend enough time to correctly label the document but still takes the route with smaller penalty and smaller probability of being detected as a spammer.

Fig 6b is the confusion matrix for a judge that is center-leaning. This means regardless of the summarization model he often labels the two summaries as being equal. Unfortunately, this behavior does not show on visualizations that work based on error measures. In fact, among 20 judges that worked on one dataset this judge had the smallest Root Mean Squared Error (RMSE). Also the agreement metrics cannot expose this type of behavior and 7 other judges had a worse agreement value than this judge. The confusion matrix allowed us to immediately pinpoint this behavior by visually inspecting it.

Confusion matrix for a spammer

Fig 6c demonstrates the confusion matrix for a smart spammer. As we see, the spammer uses two labels only.

Confusion matrix for a systematically biased judge

We call a judge systematically biased if he shows a consistent bias in a specific direction. Fig 6d is an example of the judge whose labels are often one label off from the ground truth. Here the red elements above the diagonal of the matrix suggest a systematic bias in judge's input. For example, the judge has constantly provided label "1" for 53.4% of the cases for which the ground truth was "0". He also provided label "-2" in 35.5% of the cases for which the ground truth was "1". And in 29.7% of the cases that the true label was "2" he inserted "3" as his label.

Correcting Systematic Bias for Judges When the systematic bias of a judge is characterized, then it is possible to correct these biases through a probabilistic model. Similar to (Smyth et al. 1995) and (Dawid and Skene 1979), these models consider a confusion matrix of the error rates for each judge. Based on David and Skene's (Dawid and Skene 1979) model we have also developed a probabilistic Hard-Assignment Expectation Maximization (EM) for correcting these biases, however, the details of such a model are out of the scope of this paper.

Color coded confusion matrix using counts

While confusion matrices based on error rates are informative, they have an inherent problem of not showing the number of items for each cell. For example, a 25% error rate in

Fig. 6a can be due to the fact that the judge only had 1 wrong judgment for that cell. We solved this problem using color-coded confusion matrices based on counts (see Fig. 7). In these confusion matrices, the value of each cell is a count value while the color comes from the error rates from the range between 0% to 100%.

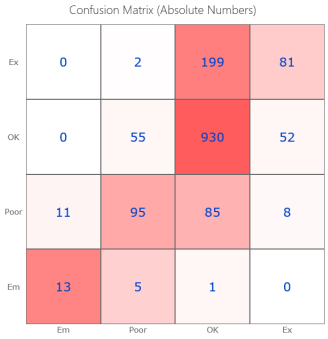


Figure 7: Confusion matrix using counts. Background colors for each cell are calculated similar to the ones for the confusion matrices with percentages but the values of the cells contain raw counts.

Side Bias and Position Bias

In simple HITs one label per HIT is collected. This prevents a position bias from appearing in the data. Sometimes, especially in cases where relative labels are needed, it is desirable to include multiple items on a HIT page and produce a super HIT. In our experiments, a HIT takes one of the forms that are presented in Fig 8. The items for which the labels are needed are either placed side by side (often when relative data on two items are needed) or in the cases that labels on more than two items are needed, the items are placed one after another from top to bottom. This arrangement may produce an undesirable position bias in the output of the task (Craswell et al. 2008), i.e., judges might unintentionally examine the first items more carefully than others. This can have an effect on the quality of the labels for items presented at different positions. Even in cases where only one item is presented in the HIT, the item that is presented earlier in the judge session might be examined more (or less) than the items presented in subsequent HITs, e.g., due to fatigue.

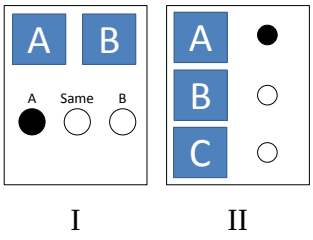


Figure 8: Item arrangement in HITs and super HITs

The effect of position bias is demonstrated in Fig. 9. For this HIT, we asked the judges to select the best item among 2 items that were presented side by side similar to HIT I in Fig. 8. We randomly placed the outcome of different web document summaries in positions one and two and looked at the number of times that each side was selected as the best summary. Because of the random placement we expect each side to be selected as the best summary in about 50% of the times, but the graph in Fig 9 shows that, on average, judges have a significant bias toward the summary on the right.

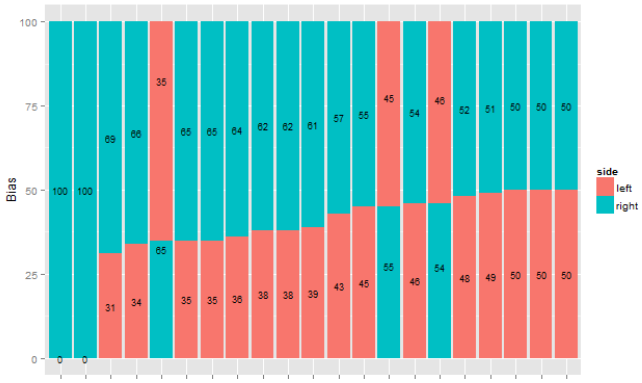


Figure 9: Side Bias. Each column shows how many times a judge has selected the left or the right side. Performing a proportion hypothesis test can highlight the judges that have a significant bias.

If the number of items in the HIT configuration II in Fig. 8 varies in one data set, then we need to examine the HITs that had only two items per HIT separately from the ones that had three items per HIT. Fig. 10 demonstrates the position bias for hits with 2 to 9 items per HIT. We looked at how many times each position was selected as the best web summary by one of the judges in the pool. Because we randomly placed the outcome of various summarization models in position one we expected to see equal numbers for each position. However, the outcome of the HIT showed a statistically significant bias toward higher positions. In other words, the results of proportion test for n=[2,3,4] all suggested that the number of times that the web summary in position 1 was selected as the best summary was significantly higher than 50%, 33.3% and 25% respectively ($p < 0.05$).

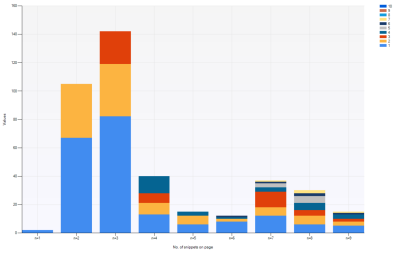
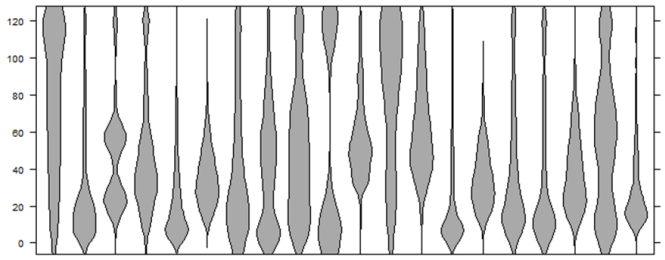
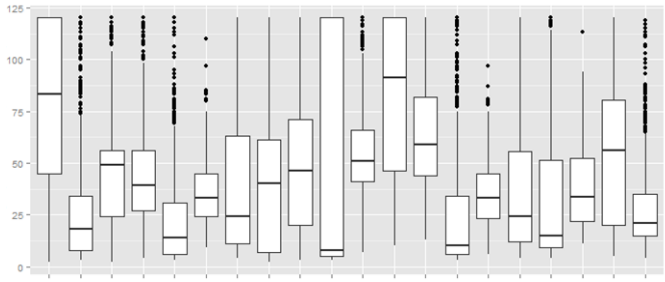


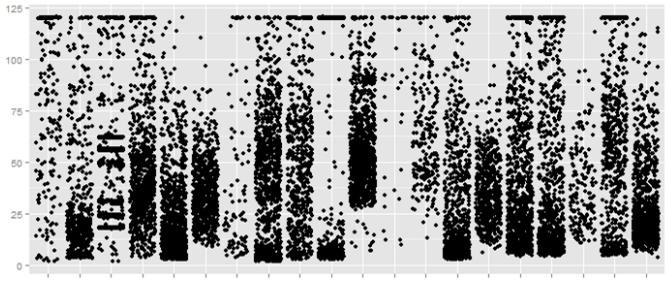
Figure 10: Position bias for type II HITs that have varying number of items



(a) Distribution of time spent on each HIT by each judge



(b) Box-and-whisker plot for time spent on each HIT by each judge



(c) Jitter plot of every submission time by each judge

Figure 11: Three different visualization methods for examining the task submission time for the judges.

Time Based Metrics for Quality of Each Judge

Smucker and Jethani show the relationship between "time to judge" for each judgment and the error for the judgment (Smucker and Jethani 2012). In this work, we demonstrate example visualizations that can highlight various task submissions. In Fig. 11 (a) we plot the submission time distribution for each judge. Columns are sorted based on RMSE values for the judges so the judge on the left has the highest error. We paid the judges by the hour, so an adversarial judge may try to open the HIT and then work on other HITs that pay per task, and later come back to our HIT to submit a label. The heavy tail in the distribution of the judge on the left for both Figures 11 a and b suggest such a behavior. As we see, not only the judge had the highest error, he also showed the adversarial behavior. Additionally, the bimodal shape of the submission time in the third column from the left in Fig. a is suspicious. Fig (c) allowed us to examine this behavior further. We realized that the judge was probably using an automated task submission script to submit the task after some time thresholds.

Flagging System

As the number of metrics and visualizations grows, it is desirable to have a flagging system that automatically flags users when their metrics values pass preset alarming thresholds. JIP graphs in Fig. 4 enable the crowd admin to quickly set a threshold for flagging judges for failing under various metrics. In *Mentor* we sort the judges in the pool based on how many flags they have raised. This allows us to be able to quickly identify judges that are flagged the most and look at their selected labels. It is also important to consider the difficulty of an item when the metrics are calculated. There are various definitions for the difficulty of a task. Whitehill et al. define the difficulty of item i as β_i where $\beta_i = 0$ means i is so ambiguous that a proficient judge has only 50% chance of labeling the item correctly. In their model $\beta_i = \infty$ means the item is easy and a noisy judge will always label it correctly (Whitehill et al. 2009). In another work Abraham et al. define the difficulty of a task as the gap between the empirical likelihood of the most frequent label and the second most frequent label (Abraham et al. 2013). In this work if items a and b have the same number of judgments, and the labels of a have a smaller dispersion than b, then a is considered to be an easier task. Table 3 presents a subset of the numerical and visual metrics in *Mentor*.

1	Binary Agreement
2	Average Absolute Error $\pm$ std
3	MSE & RMSE
4	Average CEM $\pm$ std
5	Cohen's Kappa against the ground truth
6	Average CEM $\pm$ std
7	JOP Graph
8	List of textual comments by judges
9	Number of items that are indicated as defects, inappropriate, irrelevant
10	Fleiss's Kappa for the dataset
11	Histogram for labels
12	Aggregated score of judgments (Using mean, median, majority, EM, GLAD, D& S, Minimax Entropy, Ensemble Models)
13	Spread / Variance around the inferred label
14	Names of the judges who have worked on each item
15	Inferred difficulty of item

Table 3: Visual and numerical metrics used in *Mentor*.

Final Advice

We have been developing and improving *Mentor* for about 7 months. Through our work on *Mentor* we have established a number of best practice principles that we would like to share with the community.

Principle 1: Communicate with your judges. Allow them to critique your HIT design, listen to their pain-points, and always address them. Judges always appreciate it when they realize that you are listening to their concerns.

Principle 2: Mandate independence for labels and judges. For collecting labeled data, it is important to make sure each label for an item is independent from the other labels for that item. While dependence of labels can be tested with statistical techniques we also make sure that the judges do not physically sit close to each other and do not communicate.

Principle 3: Look for significance in your metrics. Allow statistical hypothesis testing to guide the actions that you take based on the visualization tools. For example we do not take any actions based on position bias visualization unless a statistical proportion test validates that the judge has shown position bias.

Principle 4: Keep a human in the loop. Whether it is an expert engineer or a super judge selected from the crowd, a human in the loop decreases the likelihood of a major bug appearing in the system.

Principle 5: Review outliers by inspecting individual data points. Sometimes outliers can help you identify corner cases.

Principle 6: Do not over-correct. Do not make the judges aware of their biases as it will produce a bias in the opposite direction. For example when we realize a judge is providing incorrect labels for one specific label, we ask them to review the guidelines for that label and send them examples that can help them correct their behavior without explicitly telling them that their labels are inaccurate.

Principle 7: Make performance visible. Share your results with the crowd admin so that they can monitor judge behavior across different tasks. Typically, judges show certain behavior for specific tasks, but poor quality in one task may not preclude them from working on another task.

Future Work

We are working on building an automated system that can generate training plans for the judges based on their confusion matrices and error rates. We are also working an ensemble model for judgement aggregation to combine various algorithms for ground trough inference.

Conclusion

In this paper we presented best practices and principles for judge quality control in crowdsourcing environments. We demonstrated how a collection of different error measures, statistical tests, and visualizations can provide the crowd admin with a comprehensive and detailed picture on the quality of judges, the quality of labeled data sets, and the development of the judge pool quality over time.

Over several months, we worked extensively on managing the health of judge pools and controlling the quality of resulting labeled datasets using the techniques described in this paper. Based on our experience, we also created and shared a list of best practice principles for managing crowd judges in order to optimize the judge pool's health, which in turn results in higher quality data.

Overall we found that this collection of techniques works well in practice as *Mentor* let us identify spammers, led to the successful correction of low quality judges, and finally to a drastic improvement of data quality. We hope that the shared techniques and principles can be adopted in similar systems.

References

Abraham, I.; Alonso, O.; Kandylas, V.; and Slivkins, A. 2013. Adaptive crowdsourcing algorithms for the bandit survey problem. *arXiv preprint arXiv:1302.3268*.

Alonso, O.; Rose, D. E.; and Stewart, B. 2008. Crowdsourcing for relevance evaluation. In *ACM SigIR Forum*, volume 42, 9–15. ACM.

Callison-Burch, C., and Dredze, M. 2010a. Creating speech and language data with amazon's mechanical turk. In *Proceedings of the NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk*, 1–12. Association for Computational Linguistics.

Callison-Burch, C., and Dredze, M. 2010b. Creating speech and language data with amazon's mechanical turk. In *Proceedings of the NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk*, CSLDAMT '10, 1–12. Stroudsburg, PA, USA: Association for Computational Linguistics.

Craswell, N.; Zoeter, O.; Taylor, M.; and Ramsey, B. 2008. An experimental comparison of click position-bias models. In *Proceedings of the 2008 International Conference on Web Search and Data Mining*, WSDM '08, 87–94. New York, NY, USA: ACM.

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

Hofmann, K.; Behr, F.; and Radlinski, F. 2012. On caption bias in interleaving experiments. In *Proceedings of the 21st ACM international conference on Information and knowledge management*, CIKM '12, 115–124. New York, NY, USA: ACM.

Inselberg, A., and Dimsdale, B. 1991. Parallel coordinates. In Klinger, A., ed., *Human-Machine Interactive Systems*, Languages and Information Systems. Springer US. 199–233.

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

Järvelin, K., and Kekäläinen, J. 2002. Cumulated gain-based evaluation of ir techniques. *ACM Trans. Inf. Syst.* 20(4):422–446.

Smucker, M. D., and Jethani, C. P. 2012. Time to judge relevance as an indicator of assessor error. In *Proceedings of the 35th international ACM SIGIR conference on Research and development in information retrieval*, 1153–1154. ACM.

Smyth, P.; Fayyad, U.; Burl, M.; Perona, P.; and Baldi, P. 1995. Inferring ground truth from subjective labelling of venus images. *Advances in neural information processing systems* 1085–1092.

von Ahn, L. 2006. Games with a purpose. *Computer* 39(6):92–94.

Whitehill, J.; Ruvolo, P.; Wu, T.; Bergsma, J.; and Movellan, J. 2009. Whose vote should count more: Optimal integration of labels from labelers of unknown expertise. *Advances in Neural Information Processing Systems* 22(2035-2043):7–13.

Zhou, D.; Platt, J.; Basu, S.; and Mao, Y. 2012. Learning from the wisdom of crowds by minimax entropy. In *Advances in Neural Information Processing Systems 25*, 2204–2212.