A Review on Manually Detecting Errors for Data Cleaning Using Adaptive Crowdsourcing Strategies (VChecker)

Valerios Stais
valerios1910@gmail.com
University of Athens
Athens, Greece

Figure 1. Ranking Accuracy against time randomness

Abstract

VChecker[4] is a tool made to optimize the crowdsourcing process when one has sets of questions that involve different data values. In this review I reproduce the authors' code and repeat their experiments to validate their results. Additionally, I introduce a simple method of simulating crowdsourcing in-silico for research purposes, and use VChecker's results to pinpoint the hyperparameter values that most closely simulate a real-world environment.

Keywords: reproducibility, crowdsourcing, simulation, data regions

1 Introduction

This is a review of the paper 'Manually Detecting Errors for Data Cleaning Using Adaptive Crowdsourcing Strategies' [4] by Zhang et al., hence called the paper or simply **VChecker**. Besides validating the authors' results, I have created the beginnings of an in-silico crowdsourcing simulation process useful for experimental purposes.

In the paper, the authors deal with an artificial e-shop storage dataset, using a crawler to scrape several items off the internet along with their details. Some of these details are wrong, and the objective of crowdsourcing is to determine these wrong values. The authors propose that a question's response time is related to the question's difficulty. They verify this natural hypothesis and make use of it to optimize the crowdsourcing process.

Taking response time into consideration alongside the crowdsourcing workers' average accuracy, they are able to achieve a large decrease in cost while maintaining similar accuracy, or significantly increase the accuracy without increasing the cost, among other results.

In the process of reviewing the paper, the first and largest problem I encountered was the lack of data. The paper authors utilized web crawling and crowdsourcing to form their dataset and get crowdsourcing answers at various steps of their research, but I lack these means. However, I decided to turn this problem into an opportunity by introducing a new direction to this review.

I have created a process for artificially creating a question dataset, as well as one for 'crowdsourcing' questions, where several hyperparameters are used to tweak the applicable real-world circumstances. I use this artificial data to approximate the paper's results, looking for inconsistencies that

would not be adequately explained by the difference between datasets.

I also test different combinations of hyperparameters to create different sets of artificial data, and then use this data with the reproduced code of the paper to find the one that best approaches the results reported by the authors. In this way, I find the combination of hyperparameters that best approximates the real-world conditions of crowdsourcing. I hope that this research can serve as a reference point for later researchers with limited resources, who are looking to create in-silico crowdsourcing data.

This review's main findings, in order of perceived importance, are:

1. Proposing a method to in-silico simulate crowdsourcing (though more research is needed to make this method serviceable).
2. An experimental reinforcement of the natural idea that a question's difficulty is strongly correlated with the time taken to answer it. This is a core idea of *VChecker*.
3. A reproduction of the paper's code.
4. A good reproduction of the paper's results using artificial data.

The following sections are organized as follows; the Methods section describes the data creation process and the parts of the original paper that I reproduced, as well as the several implementation choices I made. It also details the limitations of this research, due to time and resource constraints. The Experiments section describes the experiments I made, in contrast with the paper's results. Finally, the Conclusions and Future Work section summarizes the results of my research and details avenues that I did not have the time to explore.

2 Methods

2.1 Data Creation

The original dataset used by the authors is not public, and I was unable to find a dataset which fit all the necessary criteria; we required a crowdsourced dataset with information on individual answers as well as the time taken by each crowdsourcing worker. Moreover, for *VChecker's* algorithms to be applicable, the dataset needed to contain several sets of items organized in data regions.

Therefore, unable to procure a suitable dataset, I created my own.

I assume a single data field for each item in our dataset, which I conveniently name 'color'. For this data field, I introduce eleven values (blue, green,...) and assign a difficulty to each These data values each represent what is called a data region, which is the data division explored in the paper. A data region's difficulty is considered to be constant among all questions of this region, and it is only used during the in-silico crowdsourcing to represent the probability of an erroneous answer by a crowdsourcing worker. Ranking the regions by difficulty is one of the main functions of *VChecker*.

For each color value, its difficulty belongs to the range *[0.025,0.325]* and represents the probability of a crowdsourcing worker getting the answer wrong when the question involves this value. No two data values have the same difficulty. Among the n items created, the data values are assigned randomly, based on a wide normal distribution. Therefore, for *n=690* items and *M=11* regions, the population of each data region ranges between *8* and *121*.

2.2 In-Silico Crowdsourcing

The crowdsourcing workers (henceforth, we will refer to them as workers) are assumed to be *i.i.d.* (independently identically distributed) like in the paper. Spammers and unskilled workers are present to a certain ratio, but the rest of the workers are assumed to have the same level of ability to answer questions.

When t answers are requested through crowdsourcing, for an item with the data value *v*, t random workers are chosen. Spammers have a *50%* chance of getting it right, unskilled workers face double the difficulty of a regular worker, and each normal worker's answer is random with a probability *d* of making a mistake, where d is the difficulty of *v*. These answers are concatenated and returned to the requester, hence simulating a simplified version of the black-box operation of crowdsourcing.

2.3 Hyperparameters

Below, I present the hyperparameters employed during data creation. This can be tuned to attempt to represent real-world conditions.

1. *time randomness*: The randomness inherent in the time it takes for an answer to be produced. It exists in the range *[0,1]*, where *0* means that the time taken is directly correlated with the question's difficulty and *1* means that the time taken to answer is completely random.
2. spammer ratio: The ratio of spammers present in the worker pool, where spammers are assumed to reply very quickly and randomly to each question.
3. unskilled ratio: The ratio of workers who are, for some reason, ill-suited for the particular questions they are facing. When calculating their answer to a question, the chance of a mistake is twice as high as that of a regular worker.

The spammer ratio and unskilled ratio hyperparameters are set to *0*, meaning that no spammers or unskilled workers are assumed to be present in the worker pool. The reason for this is that VChecker's experiments only include workers with an excellent track record, so adding spammers or unskilled workers in my experiments would only muddy the results.

These hyperparameters do exist in my code, however, so they can easily be tuned for future research on other subjects.

The following hyperparameters are connected to the paper's algorithms, and we tune them to get results similar to those reported by the paper authors. These two hyperparameters are not emphasized in this review, and their values are set through empirical testing.

1. *G-size*: After acquiring the metrics of each data region (accuracy, time taken, and disagreement between workers) we use the SVMRank[3] algorithm to rank them by difficulty. We use a subset G of the region metrics to train the ranking algorithm before having it rank all regions. The size of this subset is determined by *G-size*, in the range of *(0,1]*. This parameter is used to regulate the time required of our expert, who is asked to sort G before using it as golden data.
2. threshold: To determine the difficulties of each region, the paper's algorithm originally crowdsources a small part of the dataset and gets some metrics for each data region. x items are queried for each data region, and y answers are requested for each item, where the optimal x and y are found by iterative expansion. We compare the current iteration's metrics with the metrics of the previous two iterations and, if their Spearman correlation[1] is over the threshold, we consider the iterative algorithm to have converged.

For this review, the *G-size* hyperparameter is set to *1*, so the entire set of data regions are used to train the SVMRank[3] algorithm. This is because we only have *11* data regions, so an expert could very quickly sort them. The *G-size* hyperparameter is only useful when the data regions number in the dozens or hundreds. The threshold is set at *0.7*, as I experimentally found that this value produces similar convergence to what the paper reports.

2.4 Implementation Choices

Below, I enumerate the implementation choices I have made that are different from what the paper suggests:

- In this review, the golden ranking of the region difficulties is known (golden data), as they are explicitly set before data creation. The authors of the paper use extensive crowdsourcing to get a good approximation of these difficulties (silver data).
- The ranking algorithm is trained on pairs of data values sorted by an expert. Specifically, the expert is given the metrics of some data values and is asked to split them into two groups, with all values of one group being more difficult than the other group's. As I lack an expert, I use a K-means[2] algorithm to separate these region metrics into two groups. Compared against the golden ranking, these groupings are found to be accurate *99%* of the time.

2.5 Limitations

Due to time and resource constraints, some parts of the paper have not been explored in this review. Specifically, the part about generating explanations about the difficulty ranking produced is omitted, as it involves the pursuit of a rule-based approach to determining the reason for a region's ranking (spammers, unskilled workers, inherent difficulty, etc.) As I am aware of all of the artificial dataset's and crowdsourcing's perplexities, looking for the rules that I myself made would hold little academic interest.

Additionally, I only investigate one crowdsourcing problem, that of minimizing the cost while keeping accuracy over a limit. The paper mentions four other problems, but they focus on this one. For this review, this problem is enough to produce reliable and insightful results.

3 Experiments

Each experiment detailed below is repeated *100* times for reliability, and I report the average results.

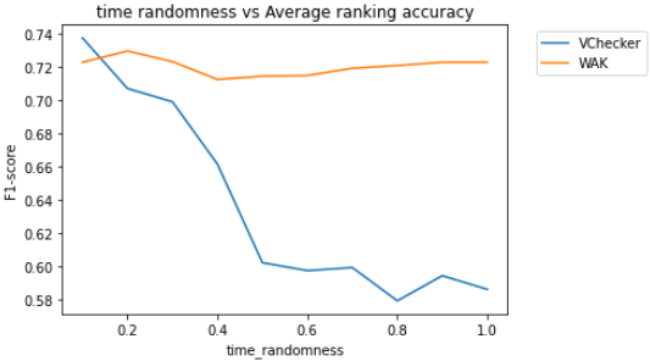
Using the artificial data creation and crowdsourcing process described above, we simulated the dataset 'Apparel' of the original paper. This is the smallest one, as it contains *690* items split between *11* data regions.

3.1 Learning to Rank

I first simulate the ability of VChecker to learn a good region difficulty ranking based on the provided data. The ranking is split into sets of two regions *(v,v)*, with the difficulty of v greater than that of v, and then the set of sets produced by this ranking is compared against the set of sets produced by the golden ranking.

Figure 1 shows the efficiency of the paper's ranking method against the value of time- randomness, as described in the Methods section. For comparison purposes, I also present the results of a baseline ranking method *WAK* (worker accuracy-based ranking), which simply orders the regions by average worker accuracy.

Figure 2. Ranking accuracy against time randomness.



We can observe that, for all values of time-randomness above *0.1*, the baseline offers better performance than the method proposed in the paper. However, for *time-randomness=0.1*, the improvement of VChecker over the baseline is similar to what the authors report.

Therefore, it seems that the value of time-randomness is approximately *0.1* in real-world situations, and that is the value we will use in the rest of these experiments. Note that this result could be affected by other parameters, like the distribution of data over regions or the distribution of difficulty over regions, but I am unable to check those in the context of this review. However, even if that is the case, this experiment indicates that the real randomness of answer time is low, meaning that the authors are correct to assume a strong correlation between a question's difficulty and the time taken to answer.

For *time-randomness=0.1*, Table 1 summarizes the results found by my experiments (artificial data) and those reported by the paper authors (real data). All results are reported in %.

Table 1. 'Learning to Rank' Results Comparison.

	WAK			VChecker		
	Precision	Recall	F1-score	Precision	Recall	F1-score
Art. Data	72.27	72.27	72.27	73.72	73.72	73.72
Real Data	76.67	63.89	69.70	72.22	72.22	72.22

For the two datasets, the results are similar. We can observe, however, that WAK shows a significant gap between precision and recall when tested on real data, while no such difference can be found when tested on our artificial data. I originally hypothesized that this might be caused by a worker bias towards positive or negative answers, but further experimenting showed that adding bias does not produce such an effect. Therefore, I assume that this gap is caused by the inherent qualities of the paper's dataset, though further research is needed.

But regardless of whether or not that assumption is true, the paper results show that VChecker can neutralize whatever causes this gap. In light of my research, this is an important strength of VChecker that was not suitably emphasized in the paper.

3.2 Solving crowdsourcing problems

Table 2 shows the performance of VChecker on reducing the crowdsourcing cost while maintaining the same accuracy as a baseline method. UCS (uniform crowdsourcing) was used as a baseline, which requests the same number of answers t for each data region.

The differences in cost when using artificial data are larger than the respective differences in real data. This could be due to the real data's inherent randomness, to a degree that the artificial data have problems emulating. The accuracies concerning artificial data are significantly lower than in the real data, possibly due to an increased population of difficult data regions, and this increased difficulty could be related to the increase in VChecker's improvement over the baseline. Nevertheless, these results reinforce the paper's findings, showing that the numbers they report are indeed feasible.

Table 2. Reducing the cost while maintaining a steady accuracy. Reduction is given in %.

Dataset	t-a	Cost (num of questions)			Accuracy (%)	
		UCS	VChecker	Reduction	UCS	VChecker
Artificial Data	3	2070	1454	29.7	81.31	81.54
	5	3450	1819	47.2	81.49	81.61
	7	4830	2014	58.2	81.25	81.36
	9	62.10	2274	63.3	81.29	81.37
Real Data	3	2070	2016	2.6	97.60	97.62
	5	3450	2384	30.9	98.03	97.82
	7	4830	2866	40.7	98.36	97.90
	9	6210	3202	48.4	98.84	98.02

4 Conclusions and Future Work

In the 'Learning to Rank' area of the paper, the artificial data seem to behave similarly to the real ones, barring a gap between precision and recall when the baseline method WAK is used on the real data. When it comes to the core findings of the paper, 'Finding Good Plans', the paper's results have been similarly reproduced, and even to a greater degree than reported.

This review's main findings are, in order of perceived importance:

1. Proposing a method to in-silico simulate crowdsourcing.
2. An experimental reinforcement of the natural idea that a question's difficulty is strongly correlated with the time taken to answer it.
3. A reproduction of the paper's code.
4. A good reproduction of the paper's results using artificial data.

This research could be significantly expanded by adding more hyperparameters to the data creation process and observing whether this brings the results closer to what was expected. We could also model the second part of the paper, 'Generating Explanations', that we omitted. This could also be helpful in better exploring the data creation process. Different distributions of difficulties could also be explored, as well as the effect of a larger dataset with more items and data regions. Finally, we could explore more variations of crowdsourcing plans, as the authors of the paper do.

References

[1] 2008. *Spearman Rank Correlation Coefficient.* Springer New York, New York, NY, 502–505. https://doi.org/10.1007/978-0-387-32833-1_379

[2] Xin Jin and Jiawei Han. 2010. *K-Means Clustering.* Springer US, Boston, MA, 563–564. https://doi.org/10.1007/978-0-387-30164-8_425

[3] Thorsten Joachims. 2002. Optimizing Search Engines Using Clickthrough Data. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (Edmonton, Alberta, Canada) *(KDD '02)*. Association for Computing Machinery, New York, NY, USA, 133–142. https://doi.org/10.1145/775047.775067

[4] Haojun Zhang, Chengliang Chai, A. Doan, Paris Koutris, and Esteban Arcaute. 2020. Manually Detecting Errors for Data Cleaning Using Adaptive Crowdsourcing Strategies. In *EDBT*.