Crowdworker Filtering with Support Vector Machine

Hohyon Ryu
School of Information
University of Texas at Austin
hohyon@utexas.edu

Matthew Lease
School of Information
University of Texas at Austin
ml@ischool.utexas.edu

ABSTRACT
Crowdsourcing has been recognized as a possible technique to complement costly user studies, usability studies, relevance judgment for information retrieval studies, and training set build-up for automatic document classification. However, the quality of crowdworkers varies by diverse factors and we often cannot tell whether their answers are right or wrong immediately due to the lack of gold standard answers. In this paper, we present a machine-learning based crowdworker filtering technique that can be used to assess workers immediately after they finish their assigned tasks. A Support Vector Machine (SVM)-based crowdworker filter, called a Smart Crowd Filter (SCFilter), was used to predict the probability that each label is correct and identifies those crowdworkers that consistently provide answers that are unlikely to be correct. To verify the performance of the SCFilter, a bad worker detection simulation test and an experiment in an actual crowdsourcing environment at the Amazon Mechanical Turk (AMT) website were performed. In the simulation test, bad worker detection performance was assessed in terms of precision and recall. In the experiment at the AMT website, a statistically significant improvement was observed for automatic document classification.

Keywords
Crowdsourcing, Worker Qualification, Support Vector Machine, Automatic Classification.

INTRODUCTION

Recently, crowdsourcing has attracted the attention of the information science community as a possible technique to complement costly user studies (Kittur, Chi, & Suh, 2008), usability studies, relevance judgment for information retrieval studies (Alonso, 2009), and training set build-up for automatic document classification (Brew, Greene, & Cunningham, 2010; Kazai, Kamps, Koolen, & Milic-Frayling, 2011). However, the biggest trade-off for fast and cheap answers is quality (Hsueh, Melville, & Sindhwani, 2009; Snow, O'Connor, Jurafsky, & Ng, 2008). The quality of crowdworkers' answers varies since these kinds of workers do not operate under a controlled environment. Also, we do not know the workers' socio-economical backgrounds, and we often cannot tell whether their answers are right or wrong due to the lack of pre-existing gold standard answers.

To minimize the impact of possible erroneous or abusive answers from crowdworkers, answers from "bad" workers need to be excluded from the training data. In this paper, we present a machine-learning based crowdworker filter, called a Smart Crowd Filter (SCFilter), which can predict the quality of each of the workers by examining each of their answers. The SCFilter is based on a Support Vector Machine (SVM) which predicts the probabilities of each answer for every example, and finds the crowdworkers whose average probability of answers is below a given threshold.

To verify the performance of the SCFilter, we first analyzed the performance of bad worker detection in a simulation test. This test used 3068 hand-labeled examples of Infochimps[1] documents that included title, description, source, and user-assigned tags. Good and bad workers were generated according to a preset error rate, and the SCFilter was used to identify bad workers based on automatically generated answers. After the simulation test, an experiment was performed on the on Amazon Mechanical Turk (AMT)[2] website to verify the performance of the SCFilter in an actual crowdsourcing environment. This AMT experiment was performed to obtain labels for training examples in order to predict the labels of 10447 unlabeled examples of Infochimps documents. The SCFilter was applied to the crowdworkers in the AMT website, and the performance of SVM-based automatic categorization trained on training labels filtered with the SCFilter was compared to the unfiltered baseline.

RELATED WORK

Most traditional supervised machine learning systems have relied on expensive human-labeled training sets produced

ASIST 2011, October 9-13, 2011, New Orleans, LA, USA.

[1] http://www.infochimps.com

[2] https://www.mturk.com

by a small number of experts (Sebastiani, 2002). To avoid expensive human labor to build training examples, crowdsourcing has been utilized in multiple machine learning studies. For example, Snow et al. (2008) assessed the usability of AMT for natural language processing problems, mostly for classification. Brew et al. (2010) used crowdsourcing along with active learning for sentiment classification, and Ambati, Vogel, & Carbonell (2010) applied crowdsourcing and machine learning to machine translation. Individual crowdworkers are usually not experts, but the aggregation of the wisdom of the crowd can often provide labels of fair quality that can partially or entirely replace expensive expert labeling.

However, unlike expert annotators, the quality of crowdworkers' feedback is hard to control. The most widely used way of measuring the quality of crowdworkers' annotation is inter-annotator agreement. Nowak & Rüger (2010) analyzed the degree of agreement between workers for the same example and filtered out noisy annotations. Another approach was used to balance the load between machine learning and crowdsourcing to optimize the machine learning performance. In addition, Quinn, Bederson, Yeh, & Lin (2010) presented CrowdFlow that tunes the balance between crowdworkers and machine learning for optimal performance. Other approaches include detecting extremely short task durations, verbal answer analysis (Kittur et al., 2008), and using hidden gold standard questions (Eckert et al., 2010). The task design also affects the quality of crowdsourcing (Kazai et al., 2011).

SMART CROWD FILTER

A Smart Crowd Filter (SCFilter) is a machine learning based model that can be used to detect a "bad" crowdworker, who randomly guesses labels or constantly assigns wrong labels. Given a collection of N examples $X_{1:N}$, $X_{1:N}$ can be partitioned into a seed set $X_{1:M}$ and a test set $X_{M+1:N}$. Trained on examples from the seed set $X_{1:M}$ with $Y_{1:M}$ corresponding labels chosen from categories $C_{1:|C|}$, the SCFilter can be used to predict the normalized probability of $P(c|x)$ for every possible label for every example in the test set $X_{M+1:N}$.

W denotes workers, and every worker $w \in W$ is assigned λ tasks. We defined a task to be a single labeling event in which the worker is presented with the title, description, and tags of a document with category options and asked to categorize the document with one of the labels. For example, a given worker w generates a label $\hat{y}$ for a given example $x \in X_{1:\lambda}$. The SCFilter predicts whether the given worker w is good or not as shown in Figure 1. SCF(w,t) returns 1 (w is a good worker) when the average probability of w's label $\hat{y}$ for $x \in X$ is above a threshold t, otherwise it returns 0 (w is a bad worker).

function SCF(w, t):

$$\text{return} \begin{cases} 1, & \text{if } \frac{1}{\lambda}\sum_{j=1:\lambda} P(\hat{y}_j|x_j) > t\,(x \in X_{1:\lambda} \text{assigned to } w) \\ 0, & \text{otherwise} \end{cases}$$

Figure 1. The function that was used to detect a bad worker.

SIMULATION TEST

Test Collection

The test collection consists of a selection of documents of obtained from the Infochimps website. As of April 26, 2011, there were 13515 document examples containing the title, tags, description, and source of documents. An example of an Infochimps document is as follows:

> **Title**: Average Hours Worked Per Day by Employed Persons: 2005
>
> **Tags**: America, persons, per, average, day, worked, employed, hours, demographics, government, census, population
>
> **Description**: The Statistical Abstract files are distributed by the US Census Department as Microsoft Excel files. These files have data mixed with notes and references, multiple tables per sheet, and, worst of all, the table headers are not easily matched to their rows and columns.
>
> **Source**: Census Bureau

The documents fell into thirteen categories including Science, Computers, Engineering, Medicine, Economics, Social Sciences, Geography, History, Linguistics, Poliltics and Law, Art and Culture, Encyclopedic, and Sports.

Among these document examples, 3068 (22.7%) were previously hand-labeled. 2068 examples were used as seeds for initial SVM training, and the other 1000 document examples were used for testing.

Experimental Design

The SVM was implemented with LIBSVM[3] for Ruby[4]. After testing different models by cross-validation, the linear model was chosen. Let $Z_{1:|W|} \in \{1, 0\}$ denote the quality of workers such that $Z_i = 1$ if and only if w_i is a good worker. In the simulation test, a good worker Z_i was generated by a Bernoulli distribution (B) with probability 0.8, and labels for a good worker were also generated by a Bernoulli distribution with probability 0.8. A bad worker was assumed to follow a simple random model that chooses a label $c \in C$ uniformly at random. All other notations followed the definitions in the section "Smart Crowd Filter." We simulated crowd labeling for the 1000 test

[3]http://www.csie.ntu.edu.tw/~cjlin/libsvm/

[4]https://github.com/tomz/libsvm-ruby-swig

1: **for each** $w_i \in W$ **do**
2: **sample** $Z_i \sim B(.8)$
3: **for each** $x_j \in X_{(i-1)\lambda+1:i\lambda}$ **do**
4: **Generate** $\gamma \sim B(.8)$
5: **if** $Z_i = 1$ and $\gamma = 1$ **then**
6: **generate** $\hat{y}_j = y_j$
7: **else**
8: **generate** $\hat{y}_j = c \in C \sim P(c)$
9: **end if**
10: **end for**
11: **end for**

Figure 2. Algorithm for generating simulated crowdworkers and labels.

examples $X_{M+1:N}$ by generating labels $\hat{y}_{M+1:N}$ as shown in Figure 2. We set $\lambda = 5$, and thus $|W| = 200$.

Results

Figure 3 shows the simulated bad worker detection performance of the SCFilter in terms of precision, recall and F-score. Precision, recall, and F-score in this experiment were defined as follows:

$$Precision = \frac{Number\ of\ Correctly\ Classified\ Bad\ Workers}{Workers\ Classified\ as\ Bad\ Workers}$$

$$Recall = \frac{Number\ of\ Correctly\ Classified\ Bad\ Workers}{Number\ of\ All\ Bad\ Workers}$$

$$F-score = \frac{2PR}{P+R}$$

When the precision was higher, the SCFilter filtered out less bad workers and sacrificed fewer good workers. When the recall was high, SCFilter detected more bad workers and sacrificed more good workers. The F-score refers to the harmonic mean of precision and recall. The result is the average of five trials, and the SCFilter performed the best at the threshold of 0.1 in terms of the F-score, which was 1.25 of the random selection probability ($\frac{1}{13} = 0.08$). To filter out all the bad workers (100% Precision = 37% Recall), we had to sacrifice an average of 73% of the good workers at the threshold of 0.28. However, a high F-score may not guarantee the best performance improvement in terms of document classification. The next experiment with the AMT tests was done to determine the best threshold for using the SCFilter for actual document classification.

PERFORMANCE TEST ON AMT

Test Collection

Some of the 10447 unlabeled Infochimps documents were used in the AMT performance test. 250 examples were used for the seed set to train the SCFilter, and 500 were used for the training set. In sum, the SVM classifier was trained on 750 crowdsourced examples.

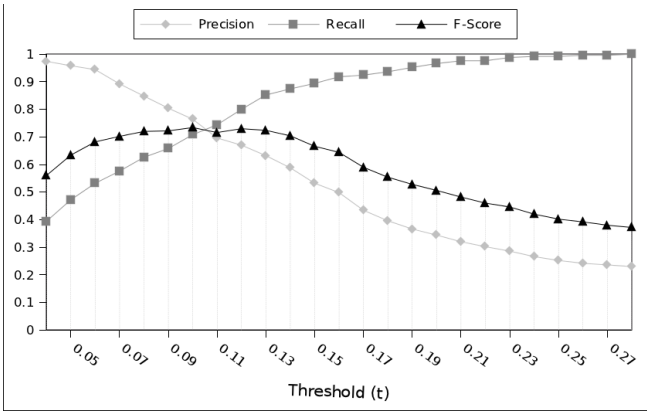

Figure 3. Simulated bad worker classification performance of the SCFilter .

To evaluate the performance of the SVM classifier, 300 gold standard examples were randomly sampled from the remaining examples and each example was labeled by ten or more AMT crowdworkers. Among the 300 gold standard examples, 82 examples were excluded from the gold standard documents, since fewer than 30% of the crowdworkers agreed on a single majority category.

Experimental Design

To verify the performance of the SCFilter in an actual crowdsourcing environment, the crowdsourcing test was performed at the AMT website. To initialize the SCFilter, 250 examples were crowdsourced by 50 workers, and the performance of the SVM classifier with the filter was compared to the gold standard examples. An AMT worker labeled $\lambda = 5$ randomly chosen examples. If the SCFilter with a threshold predicted that a worker was a bad worker, the worker's answers were ignored. This process was repeated until the number of valid workers reached 100. Trained on the answers of 100 workers, the SVM classifier was then used to predict the categories of the test set examples. The predictions of the SVM classifier were compared to the gold standard examples in order to measure the performance of the classifier. This process was done for six different thresholds including the baseline. The algorithm for this process is presented in Figure 4. In Figure

1: $W = \emptyset$
2: **while** $|W| < 100$ **do**
3: **for each** $x_j \in X_{(i-1)\lambda+1:i\lambda}$ **do**
4: Let $w \in W_{AMT}$ label x_j with $\hat{y}_j \in C$
5: **end for**
6: **if** $SCF(w, t) = 1$ **then**
7: $W = \{W, w\}$
8: **end if**
9: **end while**

Figure 4.Algorithm for a performance test on the AMT website.

4, W is a set of accepted workers, and W_{AMT} is the pool of the AMT workers.

HIT (Human Intelligence Tasks) refer to a batch of tasks presented to crowd workers at the AMT website. One HIT includes 5 tasks. In each task, the title, tags, description, and thirteen categories to choose from are shown to a crowd worker. Every worker is paid $0.05 per HIT and the time is limited to 8 minutes. With this HIT design, it takes 23 seconds in average to finish one given task. When the time limit was higher, fewer workers were attracted and it took more time to get the desired number of labels. It takes 1 minute and 55 seconds on average to finish 1 HIT.

Results

In the AMT performance test, the filtered training set significantly outperformed the unfiltered baseline at the expense of hiring more crowdworkers. Crowdsourcing answers filtered with a threshold greater than 0.11 showed statistically significant performance improvements at $p<0.01$. The best trials at the thresholds 0.13 and 0.14 showed a 19.6% improvement from 0.51 to 0.61 in terms of the F-score.

Table 1 shows the number of filtered out crowdworkers and the classification performance per threshold. To achieve the best performance at the threshold of 0.14, 306 workers were filtered out. This presents a cost of about four times more money than when using the unfiltered baseline. The threshold of 0.13 achieved the same performance by filtering out 207 workers, and the threshold of 0.11 yielded a slightly lower performance (0.6 compared to 0.61) but it filtered out only 116 workers which is 38% of that of the threshold of 0.14. However, in practice, the requester may sometimes refuse to pay the filtered out workers as they would not be considered qualified workers.

Threshold	0	0.1	0.11	0.12	0.13	0.14
Additional workers	0	69	116	130	207	306
Precision	0.52	0.54	**0.61	**0.61	**0.62	**0.62
Recall	0.5	0.52	**0.59	**0.59	**0.6	**0.6
F-Score	0.51	0.53	**0.6	**0.6	**0.61	**0.61

Table 1. The performance test of the SCFilter when tested in on the AMT website.

CONCLUSION

This paper presents a SCFilter that sorts out bad crowdsourcing workers. In a simulation test on a hand-labeled document collection, the SCFilter filtered out 70.8% of the bad workers while sacrificing an average of about 9% of the good workers. In an actual crowdsourcing environment, the SCFilter significantly improved the automatic classification performance by 19.6% (with a 0.51 to 0.61 F-score improvement) at the threshold of 0.14, which required about 306% more crowdworkers. In optimal settings at the threshold of 0.12, which maximized the performance improvement per cost, 17.6% of improvement was achieved from 0.51 to 0.6, which decreased the number of additional required workers from 306% to 116%.

The SCFilter can be utilized for live filtering that determines a worker's quality immediately upon finishing a HIT. In practice, a requester may force a worker to redo their HIT or reject the worker's HIT by a preset policy. Thus, without additional monetary costs, a requester can improve the quality of the labels. It may also be helpful for crowdworkers as they do not need to wait until their hit gets accepted or rejected. The qualification can be presented immediately to workers right after finishing the job, and they may get a chance to revise their answers to be accepted.

REFERENCES

Alonso, O. (2009). Can we get rid of TREC assessors? Using Mechanical Turk for relevance assessment. *Proceedings of the SIGIR 2009 Workshop*.

Ambati, V., Vogel, S., & Carbonell, J. (2010). Active learning and crowd-sourcing for machine translation. *LREC*, *11*(1), 2169-2174.

Brew, A., Greene, D., & Cunningham, P. (2010). Using crowdsourcing and active learning to track sentiment in online media. *ECAI 2010.* (p. 145–150). IOS Press.

Eckert, K., Niepert, M., Niemann, C., Buckner, C., Allen, C., & Stuckenschmidt, H. (2010). Crowdsourcing the assembly of concept hierarchies. *JCDL '10*, 139.

Hsueh, P. Y., Melville, P., & Sindhwani, V. (2009). Data quality from crowdsourcing: a study of annotation selection criteria. *NAACL HLT 2009* (p. 27–35). ACL.

Kazai, G., Kamps, J., Koolen, M., & Milic-Frayling, N. (2011). Crowdsourcing for Book Search Evaluation: Impact of HIT Design on Comparative System Ranking. *SIGIR 2011, July 24-28, 2011, Beijing, China*. ACM.

Kittur, A., Chi, E., & Suh, B. (n.d.). Crowdsourcing for Usability: Using Micro-Task Markets for Rapid, Remote, and Low-Cost User Measurements. *Proc. CHI 2008*.

Kittur, A., Chi, E. H., & Suh, B. (2008). Crowdsourcing user studies with Mechanical Turk. *CHI '08*, 453. New York, New York, USA: ACM Press.

Nowak, S., & Rüger, S. (2010). How reliable are annotations via crowdsourcing. *MIR '10*, 557. New York, New York, USA: ACM Press.

Quinn, A. J., Bederson, B. B., Yeh, T., & Lin, J. (2010). *CrowdFlow: Integrating machine learning with mechanical turk for speed-cost-quality flexibility*. Technical Report HCIL-2010-09, University of Maryland (2010).

Sebastiani, F. (2002). Machine learning in automated text categorization. *ACM Computing Surveys*, *34*(1), 1-47.

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

This work was partially supported by a grant from the Science and Technology Foundation of Portugal (FCT) and a John P. Commons fellowship.