A Transcription Task for Crowdsourcing with Automatic Quality Control

Chia-ying Lee and James Glass

MIT Computer Science and Artificial Intelligence Laboratory
Cambridge, Massachusetts 02139 USA
{chiaying, glass}@csail.mit.edu

Abstract

In this paper, we propose a two-stage transcription task design for crowdsourcing with an automatic quality control mechanism embedded in each stage. For the first stage, a support vector machine (SVM) classifier is utilized to quickly filter poor quality transcripts based on acoustic cues and language patterns in the transcript. In the second stage, word level confidence scores are used to estimate a transcription quality and provide instantaneous feedback to the transcriber. The proposed design was evaluated using Amazon Mechanical Turk (MTurk) and tested on seven hours of academic lecture speech, which is typically conversational in nature and contains technical material. Compared to baseline transcripts which were also collected from MTurk using a ROVER-based method, we observed that the new method resulted in higher quality transcripts while requiring less transcriber effort.

Index Terms: Transcription, crowdsourcing, quality control

1. Introduction

Transcribing speech data has historically been an expensive and time-consuming task requiring well-trained experts. Recently however, crowdsourcing platforms such as Amazon Mechanical Turk (Mturk), have emerged to supply a relatively inexpensive labor pool capable of transcribing data. In experimental studies, researchers have investigated crowdsourcing methods to produce reasonable transcripts for spoken language data. Many have reported producing good or near expert-level transcripts depending on the task [1, 2, 3, 4, 5, 6].

Given the fact that most workers from a crowd-based workforce are not well-trained experts, task design can have a large impact on the resulting transcription quality. Previous studies have examined a variety of crowd-based transcription methods. For example, in [2], multiple independently collected transcripts were combined using ROVER [9]. In [3], an iterative corrective approach was proposed to improve the transcription word error rate (WER). In [6, 7], it was pointed out that by inserting utterances with a known "gold standard" reference into the transcription task, better quality control on the collected data could be achieved. An unsupervised method was suggested in [5] to create a "gold standard" by finding a worker who had the highest agreement with transcripts produced by other workers. This "gold standard" worker's transcripts could then be used as a reference to assess future incoming transcripts.

The goal of this work is to find an efficient approach to generating high-quality transcripts for long audio recordings. In this paper, we propose a two-stage transcription task with an automatic quality control embedded in each stage. In the first stage, an SVM classifier is used to automatically distinguish between poor quality transcripts and good quality transcripts. The SVM classifier prevents workers from submitting poor quality transcripts and allows poor quality transcripts to be screened out quickly during post-processing. In the second stage, word level confidence scores are utilized to provide instantaneous feedback to workers regarding their performance on the transcription task. This two-stage transcription task was tested on academic lecture speech via Mturk and compared to transcripts collected by using ROVER with three workers [2, 9].

2. Two-stage Transcription Task

The transcription task we set up has two phases: first, a *Short Transcription* stage, and second, a *Transcript Refinement* stage. Each stage has an ASR-enabled quality controller that measures the input quality and makes it easy to filter out poor quality transcripts. We describe each phase in detail in the following.

2.1. Short Transcription

For the first stage of transcription, we wanted to create a set of small tasks that would require a consistent amount of effort from workers. Since the audio data we were transcribing were on the order of an hour or more, we decided to automatically chop the data into more reasonable segment lengths. A phonetic recognizer was used to decide on the presence of speech/non-speech. The audio, was then automatically partitioned at silence intervals into short clips that were, on average, about five to six seconds long [12].

2.1.1. HIT Design

Based on our previous crowdsourcing experience [10], we decided that requiring workers to transcribe five short utterances was a reasonable unit of labor. We therefore created a human intelligence task (HIT) based on this concept.[1] In this HIT, workers were required to listen to the short audio clips and then transcribe them. The reward was $0.01 for each short utterance. Workers were able to listen to the clips as many times as they desired, but they were not allowed to submit their HITs before listening to all the audio clips in order. In other words, they had to wait until all the audio clips finished playing before they could submit their HIT. Workers were also asked to indicate background noise, laughter, or any non-speech sounds; therefore, no empty transcripts could be submitted. Despite these requirements however, we observed many poor quality transcripts, which we defined as having a word error rate (WER) larger than 65%.

Based on these observations we decided to design an automatic quality detector to try to reduce the number of poor transcripts that were submitted. We modeled the problem as

[1] An online demo for design of the first stage can be found at https://ssls.csail.mit.edu/turks/FirstStage/

a classification problem and designed a classifier that could distinguish between good and poor quality transcripts.

2.1.2. Poor Quality Transcript Detection

By examining poor quality transcripts we observed consistent patterns of transcription behavior that could act as features for a transcription quality classifier. For example, poor quality transcripts tended to have a limited set of words for each utterance in a HIT. For example, a poor transcript of the HIT described in Section 2.1.1 could contain word *um* for all five utterances. This observation inspired us to include 1) a similarity measure among transcripts for utterances in a HIT, $Sim(h)$, and 2) the vocabulary size used by a worker in a HIT, $|Voc(h)|$, into the feature set for classifier training. The two metrics, $Sim(h)$ and $|Voc(h)|$, are defined as follows,

$$Sim(h) = \frac{\sum_{i<j} S(i,j)}{C_2^N} \tag{1}$$

$$Voc(h) = \frac{|\cup_1^N \{V(i)\}|}{N} \tag{2}$$

where $V(i)$ stands for the word vocabulary used in transcript for the i^{th} utterance and N stands for the number of utterances in a HIT, which is five for our HIT design. $S(i,j)$ means the transcript similarity score between the transcript for utterance i and the transcript for utterance j,

$$S(i,j) = \frac{|V(i) \cap V(j)|}{|V(i) \cup V(j)|} \tag{3}$$

Even though these two features capture the poor quality transcript pattern described before, there are still cases these features are unable to catch. For example, workers could just type the first two words of each utterance or type in completely random words and still get relatively high Sim and Voc scores. To prevent these type of issues, we utilized an automatic speech recognizer (ASR) and added two ASR auxiliary features.

The first ASR-enabled feature, $WER(h)$, is defined as follows: worker HIT transcripts are compared to the n-best hypotheses generated by an ASR (e.g., $n = 20$). For each utterance, the ASR hypothesis that produces the lowest WER when compared with the transcript is selected to be the reference for that utterance. The N utterances transcripts are then concatenated to form a single transcript, and the corresponding selected n-best references are also concatenated to form a single reference. $WER(h)$ is then the WER produced when aligning the transcript with the reference.

The second ASR-enabled feature, $PER(h)$, has a similar definition to $WER(h)$, except that instead of comparing worker transcripts to ASR word hypotheses, worker transcripts are converted to phoneme sequences and compared to ASR n-best phoneme hypotheses for each utterance in a similar way.

There are two benefits to embedding a poor transcript quality detector in a HIT. First, it prevents workers from submitting poor quality work at the point of origin. Whenever a worker attempts to submit a poor quality transcript, the worker will be warned and asked to improve the transcript. If the worker insists on submitting the HIT, a check-box can be used to automatically identify problematic transcripts that can subsequently be more easily screened during post-processing.

2.2. Transcript Refinement

After each utterance was transcribed by one worker, we collected transcripts from the *Short Transcription* stage. We compiled the transcripts and created a larger *Transcript Refinement* task for the second stage. Our experiments with the first stage indicated that we could expect to achieve an approximately 15% WER after the first stage of short utterance transcription. Many of the errors that we observed were difficult for workers to correct because not enough audio context was provided in the initial HIT. The goal of the second transcription task was to bundle the first round HITs together to provide a larger context. Since we knew the approximate WER, and since we provided workers with the initial transcripts, we believed the longer audio sequence would not be too onerous for workers to process. In addition, we designed the HIT to provide an easy interface for the users to edit the transcript. The following sections describe this stage in more detail.

2.2.1. HIT Design

In this stage, the short audio clips used in Section 2.1 were merged to form longer audio segments of about 75 seconds. Each of these audio segments was published along with the corresponding transcripts collected from the *Short Transcription* stage, and workers were asked to listen to the audio and correct transcript errors. The reward was $0.10 for each HIT in this stage. To help workers follow the audio while examining the transcripts, we enabled a pointer in our interface, which is synchronized with the audio and underlines the word that is being said in the audio. Workers can follow the pointer and click on words (or click and drag over word sequences) that appear to be different from what is being said in the audio and make corrections.[2]

2.2.2. Confidence Scores for Performance Estimation

Just as in the first stage HIT, workers were able to get live feedback on their transcription performance during a HIT, and were able to improve their work if necessary. For the second stage, the performance feedback was generated by using ASR word level confidence scores [8]. Word confidence scores were used to estimate the number of corrections needed for a HIT, and by comparing the number of corrections made by a worker to the estimated number, we were able to inform the worker with an estimate of the quality of their transcription editing.

In order to generate the confidence scores, we first generated ASR hypotheses for an independent lecture training set of about three hours. We computed confidence scores for the hypothesized words, and divided the range of the scores into M intervals. We then estimated the probability, p_i, that a word with a confidence score within interval i is incorrect, as defined as follows,

$$p_i = \frac{W(i)}{T(i)} \tag{4}$$

where $W(i)$ stands for the number of words with confidence score within interval i that were incorrect, while $T(i)$ stands for the total number of hypothesis words with confidence scores in interval i.

After collecting short transcripts from the first stage, we performed forced alignment to align the transcripts with the audio and computed confidence scores for all words in the transcripts. The word confidence scores were then used to predict the total number of errors in a HIT using p_i defined in Eq. 4.

[2] An online demo for design of the second stage can be found at https://ssls.csail.mit.edu/turks/contextTrans/

$$E(h) = \sum_{i=1}^{M} T(i) \times p_i \quad (5)$$

$E(h)$ is the estimated number of errors for HIT h. With this estimation, we are able to give workers feedback, $F(h)$, on their performance for a HIT, which is defined as the total number changes made by a worker, $Cor(h)$, divided by the estimated number of changes needed for the HIT, see Eq. 6. The correlation between a worker's performance and the payment rule for the HIT is also provided to workers, which serves as a guide for workers to improve their work.

$$F(h) = \frac{Cor(h)}{E(h)} \quad (6)$$

A submitted HIT from this stage will be accepted if the number of changes made is more than 80% of the estimated number for the HIT. If fewer changes are made, heuristic checks are used to flag low quality transcripts such as metrics based on worker edit counts over multiple HITs. However, this is clearly an area for future investigation.

3. Experiment Setup and Results

In this section, we examine the performance of the proposed two-stage transcription task by 1) investigating how effective the classifier is for screening out poor quality transcripts, and 2) analyzing the quality of the transcripts collected using the proposed design by comparing to baseline transcripts collected using a ROVER-based method with three workers. [2, 9]. In all cases, performance was measured by word error rate (WER) between a transcript and a reference transcript that had been previously generated. A set of 5,870 utterances that cover seven lectures in biology, computer science, and mathematics were used for these experiments.

3.1. SVM Classifier

3.1.1. Training and Offline Test

We collected 400 positive and 400 negative examples for training a support vector machine (SVM) classifier, where positive examples were defined as HITs with a WER below 40%, while negative examples were defined as HITs with a WER above 65%. We also explored training the classifier with fewer examples, as well as training without using ASR-derived features. Note that the WER of the ASR used to generate features was approximately 30% for the lecture task. In our experiments, we used LIBSVM [11] to train the SVM classifiers and tested the classifiers on 163 good quality and 163 poor quality HITs.

The results are shown in Table 1. From the table, we see that the performance of the SVM trained without ASR-derived features degraded about 4% absolute, when just 50 positive and 50 negative examples were used, compared to the best performing classifier. Even though we did not publish a task embedding this classifier trained with fewer data and features, according to the offline test results, we believe the SVM classifier should still be suitable for new domains, where initial ASRs are not available, as long as enough samples are collected.

3.1.2. Online Test

We integrated the SVM classifier trained with 400 good quality and 400 poor quality examples and the full set of features into our first stage HIT design, and published a transcription task for the 5,870 utterances. Before publishing the task with the SVM classifier, we also published another task for these utterances that did not use the transcript classification for the purpose of comparison. Note that to reduce the effect of rejected/accepted work in the first published task interfering with the other, the tasks were published three months apart. Among the 60 unique workers who submitted to the task without the classifier and the 78 unique workers who worked on the task with the classifier, only 2 workers did both tasks.

# training samples (good/bad)	# features	Accuracy (%)
400 / 400	4	98.2
400 /400	2	94.5
50 / 50	4	97.6
50 / 50	2	93.9
20 / 20	4	97.5
20 / 20	2	90.8

Table 1: *Performance of using SVM classifiers to detect poor quality transcripts for offline tasks.*

	w/ SVM	w/o SVM
Poor quality transcript (%)	3.5	23.9
Classification accuracy (%)	96.6	-

Table 2: *Performance of using an SVM classifier to detect bad transcripts for an online task.*

Table 2 shows that the classifier was able to judge 96.6% of the submitted transcripts correctly. It also shows the percentage of poor quality transcripts in the collected data from the first stage with and without using the classifier. We can see that by using a classifier, the number of submitted poor quality transcripts was reduced by over 85% relative, which shows that a classifier can prevent workers from submitting poor quality transcripts.

3.2. Transcription Quality Analysis for the Two-stage Method

Transcripts for the 5,870 utterances collected from the first stage were merged into 311 HITs for the second stage as described in Section 2.2. We published the 311 HITs on MTurk with the embedded confidence score based performance estimation. Note that with the estimation mechanism, poor quality transcripts can be detected by checking whether workers make enough changes in the HITs they submit. Poor quality transcripts can then be filtered and republished. In this work, we published one HIT at most three times. For comparison, we also published a task for these 311 HITs without the automatic performance estimation. Transcripts of the 5,870 utterances were also collected by using ROVER [2, 9] with three workers transcribing each utterance. Table 3 shows the WER of the published 5,870 utterances after the first stage and after the second stage with different HIT setups as well as the WER of baseline ROVER task.

Table 3 shows that the two-stage transcription task with performance feedback to workers and filtering produced the largest reduction in WER among all the methods being discussed. The WER improvement was statistically significant at the 0.001 level. We believe the gain came from providing workers with their performance estimation and informing workers the correlation between their performance and the payment. Filtering and republishing also allowed us to exclude poor quality transcripts. From Table 3, we can tell that there is still a 10% WER difference between the reference and the transcripts collected using the two-stage transcription task design. In Table 4, we show the average difference between the *actual* number of changes that need to be made, $E^*(h)$, for each HIT and

	WER (%)
Stage1	16.2
ROVER(3)	12.2
Stage2_$noconf(h)$	13.2
Stage2_$conf(h)$	11.7
Stage2_$conf_filter(h)$	10.2

Table 3: *Word error rate of the 5,870 utterances after the first stage, Stage1, and after the second stage with different HIT setups as well as that of ROVER with 3 independent transcripts, ROVER(3). Stage2_*$noconf(h)$ *means the second stage design without performance estimation. Stage2_*$conf(h)$ *stands for the second stage design with performance estimation. Stage2_*$conf\_filter(h)$ *is the second stage design with performance estimation and filtering as well as republishing.*

	Avg	Std
$E^*(h) - E(h)$	1	13
$E^*(h) - E_{noconf}(h)$	21	15
$E^*(h) - E_{conf}(h)$	12	14
$E^*(h) - E_{filter\_conf}(h)$	2	15

Table 4: *Average difference between the actual number of changes needed,* $E^*(h)$ *for the 311 HITs and the number of changes estimated* $E(h)$ *by the confidence score based mechanism. Also, average difference between the actual number of changes needed and number of changes workers made for different second stage setups is also shown in this Table.* $E_{noconf}(h)$ *is the number of changes workers made without performance feedback.* $E_{conf}(h)$ *represents the number of changes workers made with performance feedback.* $E_{filter\_conf}(h)$ *corresponds to the number of changes workers made with both performance estimation and filtering.*

the number of changes estimated by the confidence score based mechanism $E(h)$ described in Section 2.2.2. The average difference between the actual number of changes needed and the number of changes made by workers for different second stage set ups are also presented in Table 4.

From the table, we see that even though the average number of changes workers made at the second stage with feedback and filtering is close to the actual number of changes needed, the variance of the difference between the two is not as small as desired. For future work, we plan to improve the estimation method, and we expect the WER will be reduced further.

4. Discussion

It is interesting to observe that the two stage approach produced a higher-quality transcript in terms of WER, it also required approximately half the worker resources. The three worker ROVER-based approach required over 3K HITs of 5 utterances each, while the two stage approach required approximately 1K HITs for stage one, and 311 HITs for stage two. Even when factoring in that we paid workers twice as much for second stage HITs, the amount of human resources required for the two stage approach is considerably less than the ROVER-based approach.

Even though the two-stage approach generated the best result, we believe there is still much room for improvement. For example, in the second stage, there are no mechanisms embedded to detect *what* changes workers make. Workers may just type random changes and successfully game the system. Fortunately, we did not observe this type pattern of behavior from the data we collected. We suspect it was because workers did not realize how their performance was being estimated. To prevent this type of behavior in the future, a quality control mechanism similar to the SVM classifier for the first stage can be built.

5. Conclusion

A two-stage transcription task designed for crowdsourcing with automatic quality control has been proposed and tested on lecture speech. The experimental results showed that an SVM classifier trained with both ASR-enabled features and language pattern of workers' input allows poor quality transcripts to be detected with high accuracy. A confidence score based method was also shown to provide effective performance feedback to workers. Compared to a baseline transcription task using a ROVER-based method with three workers, the two stage task with embedded ASR-based features ultimately produced higher quality transcripts with significantly fewer worker resources.

In this paper, the two-stage approach was tested in a domain where an initial ASR existed. For future work, we are interested in adapting this design to new domains.

6. Acknowledgment

The authors would like to thank Cathy Wu for her help in setting up the baseline comparison. This work is funded by the T-Party Project, a joint research program between MIT and Quanta Computer Inc., Taiwan.

7. References

[1] A. Gruenstein, I. Mcgraw, and A. Sutherland, "A self-transcribing speech corpus: collecting continuous speech with an online educational game," in *Proc. SLaTE Workshop*, 2009.

[2] M. Marge, S. Banerjee, and A. Rudnicky, "Using the amazon mechanical turk for transcription of spoken language," in *Proc. ICASSP*, 2010, pp. 5270 –5273.

[3] M. Marge, S. Banerjee, and A. I. Rudnicky, "Using the amazon mechanical turk to transcribe and annotate meeting speech for extractive summarization," in *Proc. NAACL HLT Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk*, 2010, pp. 99–107.

[4] K. Evanini, D. Higgins, and K. Zechner, "Using amazon mechanical turk for transcription of non-native speech," in *Proc. NAACL HLT Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk*, 2010, pp. 53–56.

[5] S. Novotney and C. Callison-Burch, "Cheap, fast and good enough: automatic speech recognition with non-expert transcription," in *Proc. HLT-NAACL*, 2010, pp. 207–215.

[6] M. E. Gabriel Parent, "Toward better crowdsourced transcription: Transcription of a year of the let's go bus information system data," in *Proc. SLT*, 2010.

[7] C. Callison-Burch and M. Dredze, "Creating speech and language data with amazon's mechanical turk," in *Proc. NAACL HLT Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk*, 2010, pp. 1–12.

[8] S. O. Kamppari and T. J. Hazen, "Word and phone level acoustic confidence scoring," in *Proc. ICASSP*, 2000, pp. 1799–1802.

[9] J. G. Fiscus, "A post-processing system to yield reduced word error rates: Recognizer output voting error reduction (rover)," in *Proc. ASRU*, 1997, pp. 347–352.

[10] I. McGraw, C. ying Lee, L. Hetherington, S. Seneff, and J. Glass, "Collecting voices from the cloud," in *Proc. LREC*, 2010, pp. 19–21.

[11] C.-C. Chang and C.-J. Lin, *LIBSVM: a library for support vector machines*, 2001, software available at http://www.csie.ntu.edu.tw/ cjlin/libsvm.

[12] J. Glass, T. Hazen, S. Cyphers, I. Malioutov, D. Huynh, and R. Barzilay, "Recent progress in the MIT spoken lecture processing project," in *Proc. Interspeech*, 2007, pp. 2553–2556.