Crowdsourcing preference tests, and how to detect cheating

Sabine Buchholz, Javier Latorre

Toshiba Research Europe Ltd., Cambridge Research Lab, Cambridge, UK
sabine.buchholz@crl.toshiba.co.uk, javier.latorre@crl.toshiba.co.uk

Abstract

We describe an approach to crowdsource the evaluation of TTS systems by preference tests and report on lessons learnt from running 127 real-life crowdsourced tests. We show that at least one type of cheating becomes more prevalent over time if left unchecked and develop metrics to exclude cheaters. We demonstrate that their exclusion improves test outcomes.

Index Terms: TTS, speech synthesis, listening test, preference test, crowdsourcing, cheating

1. Introduction

Listening tests are important for TTS research and development as no overall objective method exists that correlates well enough with human jugdments. Traditionally, listening tests have been done under very controlled conditions in the lab by external subjects. However, this is time-consuming and costly. Tests are now increasingly done over the internet, however finding subjects is still a problem (e.g. [1]). This is where crowdsourcing provides a solution. A special case of crowdsourcing are so-called microtask platforms where work providers can post tasks, specify a remuneration and transfer funds to the platform, and "workers" do tasks of their choosing and get paid by the platform. The most well-known microtask platform is Amazon's Mechanical Turk (MTurk)[1] and it is increasingly used for speech and language tasks [2].

However due to the anonymous nature of the interaction, there is a risk of workers cheating. In the broadest sense, cheaters[2] can be defined as workers who do not meet the requirements or do not follow the instructions outlined for the task. Given the instructions we presented to workers on all crowdsourced tests discussed in this paper, we therefore define as cheaters those workers who do our preference tests but do not listen at all or without proper attention to both samples before making a choice, do not use headphones, do not work in a quiet environment, are not native speakers, have a hearing impairment, or do not choose the sample they think sounds better.

There are clear incentives for people to cheat: Not listening to samples before answering the preference question means workers can earn money faster. Listening without proper attention requires less concentration. As our tasks pay reasonably well, non-native or hearing-impaired workers might be tempted to do ours, rather than other lower-paying ones. Also, our tasks are relatively short and so disappear from offer quickly, which is an incentive for workers not to waste time finding that headphone or reaching a quiet place. Finally, there seems less of an incentive to answer questions untruthfully but some people might find the idea amusing.

While we can never hope to detect all cheaters and all types of cheating, we will investigate what we can detect.

[1] https://www.mturk.com/mturk/welcome

[2] called "cheats" in the UK

2. Related Work

There is literature on crowdsourcing other types of tasks; particularly related is work on speech and language data collection, annotation and evaluation, see [2] for a recent overview.

With regard to listening tests, [3] describe crowdsourcing intelligibility tests via MTurk and [4] a framework for crowdsourcing mean opinion score (MOS) tests via MTurk. [5] describe a crowdsourced framework for evaluating algorithms that process multimedia files, such as audio or video codecs. For evaluating n algorithms, the framework runs a task consisting of $\binom{n}{2}$ paired comparisons (i.e. preference tests) on e.g. MTurk. In each comparison, users randomly see/hear the output of one or the other algorithm depending on whether they press or release the space bar. They then make a forced choice between the two states (pressed/released). The presentation method described in [5] is not suitable for TTS listening tests (except potentially for vocoding research) as the outputs of two different (versions of) TTS systems will typically have different timing structure and cannot simply be switched mid-sample. As such, the present work is the first to describe crowdsourced preference tests suitable for evaluating TTS systems.

There are at least three different approaches for excluding cheaters in crowdsourcing: using a gold standard, intrinsic metrics or additional information. *Gold standard* data is provided by a group of trusted annotators/listeners or comes from an objective ground truth, such as the true text for an intelligibility test. For example, [3] excluded workers with a Word Error Rate above 0.9 from the results. The Blizzard Challenges, who, in the broad sense, crowdsource part of their listeners, in effect use the human samples as gold standard when they exclude listeners who rate these samples very low in the MOS test [6]. By *intrinsic metrics* we mean metrics that use only the raw test results, i.e. the information about which worker answered what to which test item, and nothing else (see [7] for a general example of such a metric). [5] and [4] both use intrinsic metrics. [5] checks the transitivity of results: if A is preferred over B and B over C then A should also be preferred over C. Workers are rejected (i.e. not paid) if 80% or less of triples satisfy this transitivity rule.[3] [4] rejects workers if the sample correlation coefficient between their MOS values and those of all workers for each system is less than 0.25. [4] also uses *additional information* – in the form of timestamps – by rejecting tasks "which were submitted too quickly to have been listened to".[4] While all work discussed uses some metrics to exclude workers deemed

[3] Note that this metric assumes that there are at least three systems in the comparison, which is typically not the case for a TTS preference test.

[4] Note that a simplistic implementation of this might miss some cases of cheating: By looking at the overlap of start and end timestamps of tasks we have seen workers who must have worked in 10 browser windows in rotation, giving the impression that they had spent 10 times more time on an individual task than they had in reality.

unreliable, none of them discuss what effect, if any, this has on results. To the best of our knowledge, the present work is the first to quantify the effect of cheater exclusion metrics for listening tests.

3. Crowdsourcing preference tests

3.1. Set-up

Officially only people or institutions located in the US can post work on MTurk.[5] We therefore use CrowdFlower[6] as an interface to MTurk workers. In addition to allowing access from outside the US, CrowdFower recently implemented IP address geolocation, which helps to restrict workers to a country of interest.[7] For the tests reported in this paper, worker location was always restricted to the US, as the TTS voices evaluated were always US English voices.

We break a preference test into microtasks consisting of just one paired comparison. Workers see instructions and two buttons which they need to click to play the two audio samples, and indicate their preference for the first sample, the second sample, or none. To ensure the samples are played in sequence and in the intended order, the second button is only enabled once the first one has been clicked and the duration of the first audio file has elapsed. Every time a worker clicks a button to play an audio sample, a hidden counter for that button is incremented. For each microtaks, the final values of the two counters are stored together with the worker's preference. To avoid presentation order bias, we post separate microtasks for each order (AB as well as BA) of each sample pair. CrowdFlower/MTurk ensures that workers get served microtasks in random order.

Sometimes researchers wish to compare not just two but three or more versions of a TTS system. To facilitate this, preference tests can be grouped together, i.e. all the microtasks for all the tests in the group will show up in the same HIT group on MTurk. This will become relevant once we look at excluding all work done by a cheater for a specific test, or in this case, test group.

For each preference test, we compute the test outcome in terms of percentages of preference expresses for each system, A and B, and "No preference". To check whether any observed difference in preference for the systems is statistically significant, we apply a two-tailed t-test to the results after splitting the "No preference" votes equally over the two systems, simulating a forced choice situation in which people who really have no preference can be expected to vote for A as often as for B.

3.2. Test results

The above method to crowdsource preference tests has been used within our TTS research group extensively. Since June 2010, 127 preference tests have been run, combined into 75 test groups. Research evaluated comprised everything from prosody to acoustic modelling to vocoding. Typically, all but the largest test groups finish overnight, which greatly increases researcher efficiency. On average, a test consists of 70 sample pairs, and each pair is evaluated by 7.3 workers. Workers so far came from 48 different US states and therefore should be more representative than any group one could hope to assemble in a traditional lab experiment.

To verify whether results would be similar if tests were not crowdsourced, we repeated five of the early tests internally, i.e. with speech researchers as test subjects. Note that the majority of our speech researchers are fluent but non-native English speakers. Figure 1 shows the test outcomes for the two sets of five tests. As one can see, the crowdsourcing workers actually express a preference more often than the speech experts.[8] To make it easier to compare relative preferences for systems, we split the "No preference" vote in two halves, as described above. By comparing the boundaries between these two halves, we can see that relative preferences are very similar between the two subject groups, except for Test 4. However, while the preference differences observed in Tests 1 and 5 are highly significant, the ones in Tests 2 to 4 are not significant, so the slight reversal of system preferences in Test 4 is probably just noise. We conclude that crowdsourced preference tests allow us to draw the same conclusions as listening tests with trusted subjects.

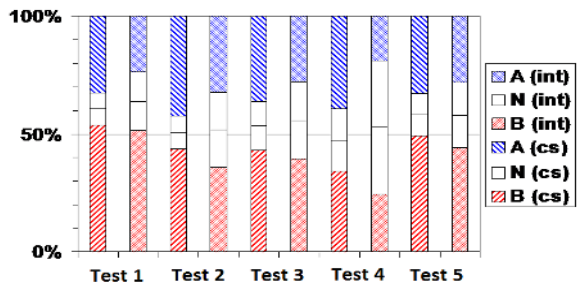

Figure 1: Preference for one system or the other (A or B) or none (N) in 5 tests done by crowdsourcing (cs) or internally (int). To facilitate comparison, the "N" vote is split in halves.

4. Analysis of cheating

As explained earlier, there is an incentive for workers to cheat. In this section, we will discuss methods to uncover cheating, and estimate its prevalence and effect on test outcomes.

4.1. Using additional information

Given that we know for each sample how often it was played by each worker, the simplest way to detect cheating is by checking these values: any worker who does not play both samples before answering is clearly cheating *on that comparison*. For each test group, we can compute the *not-played percentage*, i.e. the percentage of submitted answers for which one or both samples were not played by the worker. Figure 2 shows the not-played percentages for all 75 test groups, ordered chronologically. One can see that there is a wide variation of values (which we cannot explain) but that for many of the earlier tests, not-played percentages were very low, sometimes below 1%. More recent tests never show such low percentages, which further analysis revealed is due to *returning* cheaters, i.e. workers who cheat on more than one test group. It seems that some workers have figured out that they can get away with cheating (as CrowdFlower does not allow to directly block or reject workers) and are now on the lookout for our jobs.

[5] https://www.mturk.com/mturk/help?helpPage=requester#do_support_outside_us

[6] http://crowdflower.com/

[7] This is another example of using additional information (the IP address) for cheater exclusion. Note that MTurk only restricts workers based on the country *they* fill in when signing up.

[8] Either because they can hear more of a difference because they are native speakers, or because we expressly instruct them to not use the no-preference option too often.

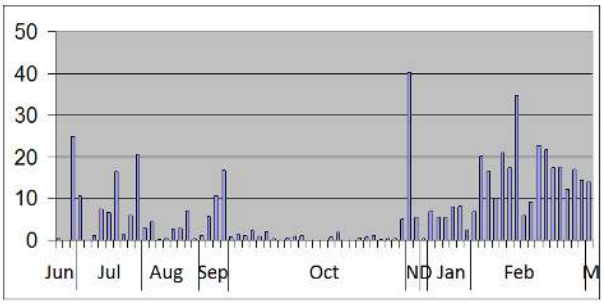

Figure 2: Percentage of submitted answers for which one or both samples were not played by the worker, for each of 75 test groups ordered chronologically, with months indicated.

4.2. Gold standard data

In our case, no ground truth or trusted annotations are available for the two systems under test, as typically one or both of these systems are the result of recent research. However, we can construct crude gold questions by pairing a sample from the human voice used to train the TTS with a sample from a TTS system (typically an older version of the TTS versions in the main comparison). We would expect people to prefer the human over the TTS and will henceforth say that they "fail" the gold question if they do not. To confirm whether our gold samples work as expected, we conducted a preference test in which all 103 questions were gold questions. 12 of the participating 20 workers always preferred the human. A further 2 always preferred the human except once (Note how excluding these two would also exclude many correct answers). 4 of the 20 workers are clear cheaters: they preferred the human and the TTS about equally often. The remaining 2 workers fall somewhere in between, preferring the human 92% resp. 78% of the time.

It was left to researchers to decide whether they wanted to use gold samples in their tests. Therefore, only 46 out of the 75 test groups include gold samples. An obvious reason for failing a gold question is if the samples were not even played; however those are the uninteresting cases. We therefore only investigated gold questions where both samples *were* played. We plotted the percentage of (played) gold questions that were failed per test group over time (not shown) and did not find the same increase (yet...). The average percentage of failed gold was 7.8.

4.3. Intrinsic metrics

The previous section introduced two ways to detect cheating in preference tests. Using that information, we can now define the following two subgroups of workers:

Known cheaters Never play both samples of a microtask.

Trusted workers Always play both samples, got asked at least 4 or 5 gold questions and do not fail a single gold question in the whole test group.[9]

By comparing these two groups, we can identify intrinsic metrics that differentiate well between the two groups. In effect, we have "recruited" a group of "genuine" cheaters and can now study their behaviour, in order to better identify and exclude similar cheating in the future. Note that while these known cheaters did not even play the samples, their answer pattern should be identical to workers who play, but do not listen to, samples.[10] To avoid sparse data issues we will concentrate on those known cheaters and trusted workers that have done at least 25 relevant microtasks per test group.

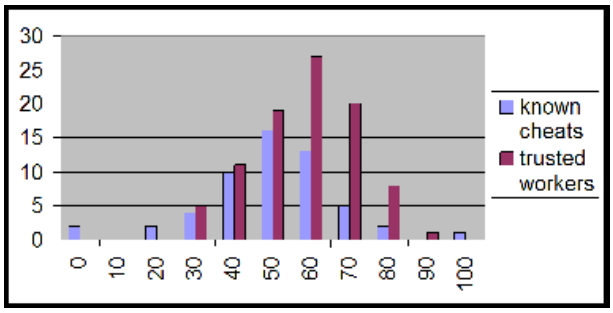

Figure 3: Counts (y-axis) of workers who had a **preference for the second sample** $x\%$ (rounded to nearest multiple of 10) of the time that they had a preference at all, for workers who had a preference in at least 25 microtasks and a) never played samples (cheaters; 55 workers) or b) played everything, did at least 4 gold and passed all gold (trusted; 91 workers; raising the threshold to 5 gold resulted in too few data points.)

Presentation order bias describes the tendency of people to prefer the second of two samples if there is no clear difference between them. Cheaters however can be expected to have either no preference for a specific position, as they choose their answer basically at random, or a very clear preference, due to always choosing the same answer. This is confirmed by the data shown in Figure 3: The peak of the cheaters' distribution is at 50% while that of the trusted workers is at 60%. Also, all the data points at the extremes are from the known cheaters. By looking at the values prior to rounding, we determined that a suitable definition of "extreme" here is $x < 30$ or $x > 90$.

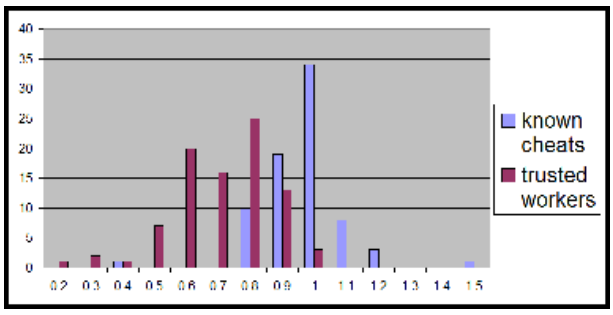

Figure 4: Counts (y-axis) of workers who had a mean distance of x (rounded to nearest multiple of 0.1) between the worker's answer and everybody else's answer for those sample pairs, for workers who did at least 25 microtasks and a) never played samples (cheaters; 76 workers) or b) played everything, did at least 5 gold and passed all gold (trusted; 88 workers).

[9]Note that for these definitions, we consider a person that took part in e.g. two test groups as two different workers. So the same person could be a trusted worker for one test group but a known cheater for another, as they might indeed have changed their behaviour.

[10]In fact, we are going to switch our interface to one that prevents workers from submitting answers without having played both samples. However, this still cannot fully prevent cheating as workers could e.g. play samples while sound is muted.

A cheater exclusion method proposed by [4] is to look for workers whose answers deviate widely from the average of all workers. This approach is based on the tacit assumption that cheaters display outlier behaviour, and conversely that outliers are probably cheaters. [4] treats MOS values as numeric and computes the sample correlation coefficient. For a preference test, the answers are not numeric. We decided to compute the **deviation in system preference** for a worker as the mean distance between the worker's answers and the other answers for the same sample pair. The distance between two answers is 0 if the answers are identical, 1 if one of the answers is "No preference" and 2 if different systems are preferred. Figure 4 shows that, indeed, the two worker groups have different distributions for this value. By looking at the values prior to rounding, we saw that no trusted worker in this subset had a mean distance of more than 1.03.

4.4. Cheater exclusion metrics and their effects

The previous section presented several cheater detection mechanisms and metrics. We will now investigate how test outcomes are affected when these are used to exclude answers/workers. Note that this constitutes a best-case scenario as exclusion thresholds were determined on a subset of the data itself. As cheaters introduce noise and therefore "dilute" test outcomes, we would expect that on average their exclusion results in bigger preference differences between systems, higher confidence ($(1-p)*100$) in the outcome, and more tests with a highly significant ($p < 0.01$) or significant ($p < 0.05$) difference.

Exclude	% ans. excl.	mean pref. diff.	mean conf.	hs	s	ns
None	0.0	11.6	86.3	45	12	70
not played answers	6.4	12.4	86.9	45	12	70
+ fail gold	14.6	12.8	86.7	45	11	71
+ pref 2nd <30 or >90	14.7	13.1	87.4	46	12	69
+ dev.syst.pref. >1.03	10.6	13.1	87.6	47	11	69
+ both above	18.5	13.6	87.8	47	13	67
+ random	14.5	12.5	86.6	43	13	71

Table 1: Effect of cheater exclusion metrics: Percent of answers excluded, mean preference difference, mean confidence, number of tests with highly significant (hs), significant (s) and non-significant (ns) differences. "+" means named metric plus exclusion of non-played answers.

The first row of Table 1 shows the simplest case, in which no answers or workers are excluded. The second row shows the effect of excluding answers for which the worker did not play both samples. This excludes 6.4% of answers, and increases the mean preference difference and the mean confidence, as we would expect. The number of tests in each significancy category stays the same.[11] In the remainder of this table, reported effects are for the combination of non-played answer exclusion and additional metrics.

If we exclude all workers who fail at least one gold question, mean preference difference increases but mean confidence decreases, and one more test is non-significant. This seems to indicate that this metric excludes not only cheaters but also some genuine workers who maybe by mistake failed one gold question. The next two rows show results for the two intrinsic metrics developed in the previous section (preference for the second sample, deviation in system preference). Both yield the highest values seen so far for mean preference difference and mean confidence and both result in more tests with (higher) significancy. Applying both metrics together results in a further boost, showing that they complement each other. Finally, as a sanity check, we show that simply excluding a percentage of workers at random does result in worse test outcomes (except a very slight increase in the mean preference difference).

[11] Although some tests changed their significancy category, these changes cancelled each other out in terms of numbers.

5. Discussion and conclusions

We have presented a method for crowdsourcing preference tests for TTS, and shown that it gives similar results to tests conducted internally. We have shown how monitoring the number of times samples were played and mixing in gold questions enabled us to quantify the prevalence of cheating over time, and to study the behaviour of cheaters. This in turn lead to two intrinsic cheater exclusion metrics, which were shown to improve test outcomes, in particular if combined. Of the two metrics, the one based on deviation in system preference gives slightly better results while excluding far fewer answers. However, it also risks biasing the results by inadvertently excluding genuine workers who happen to have a preference different from the majority. The metric based on the preference for the second sample introduces no such bias. It does however exclude some workers simply because they have done few samples and therefore the percentage is skewed (in the extreme case, 0% or 100% if they have done only one sample). This could be remedied by a more sophisticated metric incorporating that additional information.

From the comparison of known cheaters and trusted workers it should be clear that these metrics can only hope to exclude some cheaters, and that there is a remaining risk of excluding genuine workers. We therefore do not recommend these metrics for rejecting workers, i.e. withholding payment, but only for post-hoc exclusion of workers from results. In addition, the metrics can be used for monitoring change in cheating prevalence.

6. References

[1] A. W. Black and K. Tokuda, "The blizzard challenge – 2005: Evaluating corpus-based speech synthesis on common datasets," in *Proc. of Interspeech – Eurospeech*, 2005.

[2] C. Callison-Burch and M. Dredze, "Creating speech and language data with Amazon's Mechanical Turk," in *Proc. of NAACL HLT Workshop on Creating Speech and Language Data With Amazon's Mechanical Turk*. ACL, 2010.

[3] M. K. Wolters, K. B. Isaac, and S. Renals, "Evaluating speech synthesis intelligibility using Amazon Mechanical Turk," in *Proc. 7th Speech Synthesis Workshop (SSW7)*, 2010.

[4] F. Ribeiro, D. Florêncio, C. Zhang, and M. Seltzer, "CrowdMOS: An approach for crowdsourcing mean opinion score studies," in *Proc. of ICASSP*. IEEE, 2011.

[5] K.-T. Chen, C.-C. Wu, Y.-C. Chang, and C.-L. Lei, "A crowdsourceable QoE evaluation framework for multimedia content," in *Proc. of the 17th ACM international conference on Multimedia (MM '09)*. ACM, 2009.

[6] C. L. Bennett, "Large scale evaluation of corpus-based synthesizers: Results and lessons from the blizzard challenge 2005," in *Proc. of Interspeech – Eurospeech*, 2005.

[7] P. G. Ipeirotis, F. Provost, and J. Wang, "Quality management on Amazon Mechanical Turk," in *Proc. of KDD-HCOMP*. ACM, 2010.