



Continual Subjective Evaluation Method of Speech by Merging Sort-based Preference Tests Towards Ever-Expanding Corpus of Human Ratings

Yusuke Yasuda^{2,1}, Junichi Yamagishi², Tomoki Toda¹

¹Information Technology Center, Nagoya University, Japan

²Digital Content and Media Sciences Research Division, National Institute of Informatics, Japan

yasuda@nii.ac.jp, jyamagis@nii.ac.jp, tomoki@icts.nagoya-u.ac.jp

Abstract

Mean Opinion Scores (MOS) are widely used method for subjective evaluation of speech, and automatic quality assessment can predict MOS from speech by training models with MOS as labels. However, MOS are not suitable training labels for the automatic quality assessment models because they are context-dependent and their data size is limited and fixed, which limits reliability of the automatic quality assessment. To overcome the limitation, this study defines a continual subjective evaluation of speech to keep expanding scores and systems in a subjective evaluation corpus by merging. The objective of continual subjective evaluation is to derive a ranking of systems in a situation where the number of systems increases over time. The continual subjective evaluation consists of a loop of two subproblems: sorting subsets of systems in the quality order and merging the subsets of sorted systems into a single ranking. We propose a preference test method integrated with sort-and merge-based online learning algorithms to solve the continual subjective evaluation efficiently. Our experiments show that our method can realize the continual subjective evaluation by deriving a ranking of 60 systems from 216 pairs with 65,460 preference scores.

Index Terms: subjective evaluation, speech evaluation, speech quality assessment, preference test, online learning

1. Introduction

Subjective evaluations are widely used to assess the quality of speech based on human ratings. Recently, a large-scale corpus of subjective evaluations of speech has been developed that contains hundreds of synthetic speech systems and tens of thousands of scores [1, 2, 3, 4, 5]. The main application of these corpora is training of automatic quality assessment models [6, 7, 8]. Because subjective evaluations are costly due to the involvement of human evaluators, the automatic quality assessment is expected to reduce the cost of speech quality evaluations.

However, the reliability of automatic quality assessment models trained with the subjective evaluation corpus is limited. There are two reasons that limit the reliability of the automatic quality assessment models. The first reason is that the size of the training data is very limited due to the high cost of subjective evaluation to construct a corpus. The automatic quality assessment models are deep learning models, so tens of thousands of scores as training labels are too small to train generalizable models. The second reason is that the subjective scores used as training labels are context-dependent scores evaluated in mean opinion scores (MOS), which are known to suffer from a variety of biases depending on the context of experiments [9, 10]. One of the most detrimental effects of the biases in MOS is range equalizing bias [11]. Because MOS uses scales to rate

scores from 1 (Bad) to 5 (Excellent), the range equalizing bias causes scores to span the entire scale regardless of the actual quality. It causes the meaning of 4.0 in MOS in a certain experiment to differ from 4.0 in MOS in other experiments. This results in treatments that different MOS corpora are out of domain with each other, as the annual VoiceMOS challenge offers out-of-domain MOS prediction tasks as a new MOS corpus is introduced [4, 5]. Moreover, this bias forces the construction of a MOS corpus in a single shot of the experiment, which limits the number of systems that can be evaluated and requires an unreasonably large budget in one experiment. These two reasons prevent us from constructing a large-scale subjective evaluation corpus beyond the scale of the existing corpora by merging corpora consistently, limiting the reliability of the upper bound of the automatic quality estimation models.

This study defines a new subjective evaluation scheme called continual subjective evaluation to overcome the limitations of the existing subjective evaluation corpora. The continual subjective evaluation derives a ranking of systems in a quality order from many subsets of subjective evaluations. Instead of conducting a large-scale subjective evaluation involving many systems in a single shot, a continual subjective evaluation divides the systems into many sets and evaluates them to derive rankings in sub-experiments. After rankings of the system subsets are derived from the sub-experiments, a total ranking is derived by merging the sub-rankings. The continual subjective evaluation conducts these evaluation processes iteratively to keep expanding the subjective evaluation corpus consistently. This concept of the continual subjective evaluation enables us to add newly proposed speech synthesis methods to the existing corpus evaluated with the continual subjective evaluation.

We propose our first method to realize the continual subjective evaluation. Our method is based on preference tests that incorporate sort-based online learning. Because our method adopts preference tests instead of MOS, our method can divide subjective evaluation into several experiments and merge them afterwards. Unlike MOS, the preference tests use relative scores independent of quality scales, so scores are consistent and less biased even if experiments are divided to evaluate only a subset of systems. In addition, sort-based online learning can efficiently derive a ranking of systems by evaluating only part of the system pairs. These features of our proposed method realize a first step towards the continual subjective evaluation for expanding a subjective evaluation corpus continually as new speech synthesis methods emerge over time.

Our contributions are summarized as follows:

- we define the continual subjective evaluation as a new subjective evaluation task that can expand systems to evaluate over time;

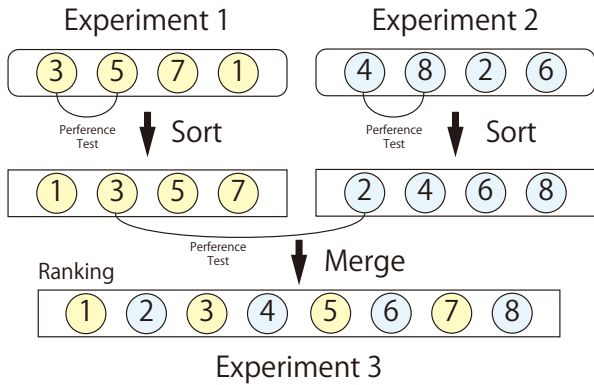


Figure 1: Concept of Continual Subjective Evaluation

- we propose a method to realize the continual subjective evaluation based on preference tests and merge- and sort-based online learning;
- we conduct an iteration of the continual subjective evaluation in three experiments to derive a ranking of 60 systems;
- our experiments show that our method can realize the continual subjective evaluation by deriving a ranking of 60 systems efficiently from preference tests evaluating 216 pairs.

2. Proposed Method

Figure 1 shows our concept of the continual subjective evaluation. The objective of continual subjective evaluation is to derive a ranking of systems in a situation where the number of systems increases over time. Therefore, the continual subjective evaluation consists of a loop of two subproblems: (1) sorting subsets of systems in the quality order and (2) merging the subsets of sorted systems into a single ranking. The increase of systems may occur when, for example, new text-to-speech (TTS) methods are proposed, and we want to evaluate the ranking of the TTS methods about naturalness, including both the new and existing TTS methods. Another situation may be the case where we want to construct a corpus of human ratings for synthetic speech, and we want to keep updating the corpus to increase the size of the score data and coverage of TTS methods. Our definition of continual subjective evaluation is expected to be a solution to these applications.

In these situations to which continual subjective evaluation is applied, the following challenges arise: (1) evaluations must be divided into several experiments to add systems at different time points; (2) the derived ranking from multiple evaluations should be consistent; and (3) cost efficiency is required for evaluations to be continual up to a large-scale system set.

To address the challenges to realize the continual subjective evaluation, we adopt preference tests as a subjective evaluation method. The preference test is a subjective evaluation method that rates a sample pair by selecting the better one. Regarding the challenges (1) and (2), preference tests can derive a consistent ranking even if evaluation is divided into several experiments. This is because the preference score is a relative score between two systems, so the score is not affected by a context in which systems are evaluated in an experiment if other conditions, such as the question to ask, are the same. On the other hand, MOS test is not suitable evaluation method for the continual subjective evaluation. MOS is not consistent across different experiments evaluating different system sets: a system

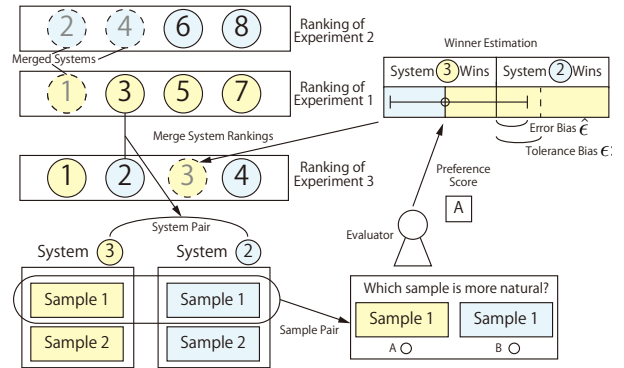


Figure 2: Overview of Proposed Method

with high MOS in an experiment may get low MOS in another experiment targeting only high-quality systems.

To address the challenge (3) of the continual subjective evaluation, we adopt sort-based online learning as an optimization method for preference tests [12]. Normal preference tests are not scalable to a large number of systems due to the huge number of pair combinations¹. Because our objective is ranking evaluation, we do not have to evaluate all combinations of pairs. Instead of evaluating $O(|S|^2)$ pairs where $|S|$ is the number of systems in set S , we only evaluate $O(|S| \log |S|)$ pairs to sort $|S|$ systems. Similarly, we only evaluate $O(|S_1| + |S_2|)$ pairs to merge the sorted $|S_1|$ systems and $|S_2|$ systems. The sort-based online learning performs a sort or merge algorithm to select the minimum pairs to derive a ranking during preference tests.

Figure 2 shows an overview of our proposed method to realize continual subjective evaluation by using preference tests that incorporate sort-based online learning. Our method is based on an existing listening test server that supports preference tests optimized with sort-based online learning [13]. With this existing method, we can obtain two sorted sets of systems efficiently by conducting two preference tests in experiments 1 and 2. Our proposed method extends it to perform the merging of the two sorted sets of systems in experiment 3. In both sorting and merging experiments, system pairs are selected by the online learning algorithm. Then, a sample pair is selected from the system pair to present to evaluators. Evaluators rate the sample pair by selecting a better sample and submitting a preference score to the server. The server updates the mean and confidence intervals of the preference score. The evaluation of a system pair continues until a winner can be estimated with low error. The error of the winner estimation is defined as an overlap of confidence intervals. If an overlap of confidence intervals becomes smaller than a desired threshold, the online learning algorithm determines a winner of the system pair. Based on the winner estimation, a system pair is sorted, and the next system pair is newly selected by the online learning algorithm. The server iterates these processes until the online learning converges to output a ranking of all systems.

Algorithm 1 shows the MERGE online learning algorithm for merging two sorted sets of systems into a single ranking in the experiment 3. The MERGE algorithm is based on the merge algorithm [14]. The only difference between the MERGE and merge algorithms is that the comparisons are based on the eval-

¹Usually, normal preference tests evaluate up to 10 systems. This study evaluates 60 systems, which is beyond the realm of normal preference tests.

Algorithm 1 MERGE

Input: Sorted sets S_1, S_2 , bias ϵ , confidence δ .
Initialize: $i = 1, j = 1$ and $O = \emptyset$.
while $i \leq |S_1|$ and $j \leq |S_2|$ **do**
 if $S_1(i) = \text{COMPARE}(S_1(i), S_2(j), \epsilon, \delta)$ **then**
 append $S_2(j)$ at the end of O and $j = j + 1$.
 else
 append $S_1(i)$ at the end of O and $i = i + 1$.
 if $i \leq |S_1|$ **then**
 append $S_1(i : |S_1|)$ at the end of O .
 if $j \leq |S_2|$ **then**
 append $S_2(j : |S_2|)$ at the end of O .
Output: Sorted set O

Algorithm 2 COMPARE

Input: element pair i, j , bias ϵ , confidence δ .
Initialize: $\hat{p}_{ij} = \frac{1}{2}, m = \frac{1}{2\epsilon^2} \log \frac{2}{\delta}, r_{ij} = 0, w_{ij} = 0$.
Define: $\hat{c}(r) = \sqrt{\frac{1}{2r} \log \frac{4r^2}{\delta}}$ if $r > 0$ else $\frac{1}{2}$.
Define: $\hat{\epsilon}(r, \hat{p}) = \hat{c}(r) - |\hat{p} - \frac{1}{2}|$.
while $\epsilon \leq \hat{\epsilon}(r_{ij}, \hat{p}_{ij})$ and $r_{ij} \leq m$ **do**
 Compare i and j . **if** i wins, $v_{ij} = 1$ **else** $v_{ij} = 0$.
 $w_{ij} = w_{ij} + v_{ij}, r_{ij} = r_{ij} + 1, \hat{p}_{ij} = \frac{w_{ij}}{r_{ij}}$.
if $\hat{p}_{ij} \leq \frac{1}{2}$ **Output:** winner j **else Output:** winner i

uator’s stochastic preference scores instead of a deterministic comparison function. To reliably estimate a winner with the evaluator’s stochastic preference scores, the MERGE algorithm relies on the COMPARE algorithm.

Algorithm 2 shows the COMPARE algorithm. COMPARE determines the winner of a pair by checking if the mean of preference scores is smaller or greater than 0.5. Let i and j be a comparing pair, r_{ij} be the number of comparisons, w_{ij} be the win counts of i over j , and \hat{p}_{ij} be the win rate of i . The winner can be determined by comparing the win rate $\hat{p}_{ij} = \frac{w_{ij}}{r_{ij}}$ with 0.5: i is the winner if $\hat{p}_{ij} > \frac{1}{2}$. To reliably estimate a winner, COMPARE guarantees that the overlap of the confidence intervals called error bias $\hat{\epsilon}$ is at most tolerance bias ϵ . Researchers can configure how accurate and costly the winner estimation is by setting the tolerance bias ϵ which is an acceptable overlap of confidence intervals, and confidence δ which is a tail probability to calculate confidence intervals. To guarantee the configured tolerance bias, COMPARE evaluates at most the maximum number of comparisons m . If there are distinct preferences, the winner can be estimated before reaching m comparisons. In that case, COMPARE performs early stopping by checking if the error bias $\hat{\epsilon}$ becomes below the tolerance bias ϵ . This feature of the COMPARE algorithm can reduce the cost of preference tests while ensuring that winners are estimated below the desired error.

We investigate two sort-based online learning algorithms to obtain the sorted sets of systems in experiments 1 and 2. The first algorithm is the MERGE-RANK algorithm [15] shown in Algorithm 3. The MERGE-RANK algorithm is based on the merge sort algorithm [14]. Thus, the MERGE-RANK algorithm is also a divide-and-conquer approach with $O(|S| \log |S|)$ complexity in the best, average, and worst cases. Thus, the MERGE-RANK algorithm is expected to sort systems efficiently with a small number of pairs, even if the systems are

Algorithm 3 MERGE-RANK

Input: Set S , bias ϵ , confidence δ .
 $S_1 = \text{MERGE-RANK}(S(1 : \lfloor |S|/2 \rfloor), \epsilon, \delta)$
 $S_2 = \text{MERGE-RANK}(S(\lfloor |S|/2 \rfloor + 1 : |S|), \epsilon, \delta)$
Output: MERGE(S_1, S_2)

Algorithm 4 INSERT-RANK

Input: Set S , bias ϵ , confidence δ .
Initialize: $i = 1, j = 2$.
for $j = 2, \dots, |S|$ **do**
 $i = j - 1$
 while $i > 0$ AND $\text{COMPARE}(S(i), S(j), \epsilon, \delta) = S(i)$
 do
 Insert in place $S(i + 1) \leftarrow S(i)$
 $i = i - 1$
 Insert in place $S(i + 1) \leftarrow S(j)$
Output: Sorted set S

already sorted or in a completely random order. The MERGE-RANK algorithm relies on the MERGE algorithm, so it also uses the COMPARE algorithm to reliably estimate a winner from the evaluator’s preference scores. Because the MERGE-RANK algorithm uses the MERGE algorithm internally, it can handle both experiments 2 and 3 seamlessly by configuring the first set S_1 in Algorithm 3 as a sorted set in experiment 1.

The second algorithm is the INSERT-RANK algorithm shown in Algorithm 4. The INSERT-RANK algorithm is based on the insert sort algorithm. The only difference between the INSERT-RANK and insert sort algorithms is that it uses the COMPARE algorithm to estimate winners reliably from the evaluator’s stochastic preference scores. Because of inheriting properties from the insert sort, the INSERT-RANK algorithm is an incremental approach with $O(|S|)$ complexity in the best and $O(|S|^2)$ complexity in average and the worst cases. Therefore, the INSERT-RANK algorithm is expected to be efficient to sort systems by evaluating only a small number of pairs if the systems are already sorted, but inefficient if the systems are ordered reversely in the initial set. In addition, the INSERT-RANK algorithm is expected to be more efficient than the MERGE-RANK algorithm for small sets of systems because the insert sort is efficient for small sets than the merge sort. These properties of the INSERT-RANK algorithm are promising for subjective evaluations of speech for two reasons. First, typical subjective evaluations of speech target the number of systems between a few and hundreds, which is where the INSERT-RANK algorithm works efficiently. Second, the initial order of systems can be roughly sorted before the experiment, for example, based on automatic quality assessment such as the MOS predictor [6]. If the initial order is roughly sorted, the INSERT-RANK algorithm might sort in the best case with $O(|S|)$ complexity.

3. Related Works

Our method is not the only method that can solve the continual subjective evaluation. Another candidate method to solve the continual subjective evaluation is the Elo rating system [16]. The Elo system derives a ranking of players from pairwise matches by estimating player’s skill as rating. In this study’s setting, players in the Elo system corresponds to TTS systems, matches correspond to pairwise comparisons, player’s skill corresponds to quality of TTS system. A preferable property of the

Experiment No.	1	2	3
Sort Algorithm	Insert Rank	Merge Rank	-
Merge Algorithm	-	Merge	Merge
#Sort Systems	30	30	-
#Merge Systems	-	10	50
#Scores in Budget	24,960	24,960	15,540
#Convergence Cost	14,977	19,658	9,761
#Evaluated Pairs	70	98	48
#Significant Pairs	28	49	21
#Max Cost per Pair	528	413	465
#Min Cost per Pair	219	60	127

Table 1: *Settings and results of three experiments.*

Elo rating is the preservation of total rating: the sum of ratings from all players is always constant in the past and future. These features of the Elo system realizes requirements of the continual subjective evaluation: divisibility of experiments, ranking consistency, and cost efficiency.

The Elo system and its extension, such as TrueSkill [17], have been applied to preference-based subjective evaluation with crowd-sourcing. The Elo system is an online algorithm that updates ratings based on each preference observation, which is desirable for the optimal pair selection, similar to our online learning-based approach. Speaker ranking evaluation [18] and machine translation evaluation [19] are successful application of rating-based approach to large-scale preference-based evaluations.

The advantages of our online learning-based approach versus Elo rating-based approaches are guaranteed evaluation error, optimal budget allocation by maximum error reduction pair selection, reduced cost by explicit stopping condition, and consistent pair selection by sort algorithm. At convergence, the Elo system does not guarantee the upper bound of evaluation errors of each pair, which is not desirable for evaluation purposes. The Elo system does not consider the uncertainty of the ratings. Therefore, additional pair selection criteria such as the number of evaluations must be considered in addition to the similarity of the ratings, as stated in [18]. There are no stopping criterion in the Elo system, so it can not be used for cost reduction. In the early phase, the Elo ratings are unstable [17], so random evaluation targets are paired that are only evaluated few times. This is in contrast to our sort-based online learning, which selects consistent pairs and evaluates them until achieving the desired error bias.

4. Experimental Evaluation

4.1. Experimental Settings

To investigate the feasibility of the continual subjective evaluation with our method, we applied our method to derive a ranking of 60 systems. Table 1 summarizes the settings of our experiments. We divided the 60 systems into two sets containing 30 systems. For the two sets, we designed three experiments. The first experiment was to sort the first set using the INSERT-RANK algorithm. The second experiment was to sort the second set using the MERGE-RANK algorithm. The third experiment was to merge the two sorted sets into a single ranking using the MERGE algorithm. To conduct the experiment efficiently, we configured the second experiment so that merging two sorted sets could start by utilizing the remaining budget after finishing sorting by the MERGE-RANK. As a result, the first experiment sorted 30 systems, the second experiment sorted 30 systems, followed by merging 10 systems with the first sorted

set, and the third experiment merged 50 systems. Note that how many systems could be merged by the second experiment was not known in advance because the remaining budget after sorting was unknown. Therefore, we fed all 30 systems, sorted by the first experiment, to the MERGE algorithm in the second experiment.

We used VoiceMOS challenge 2022 (VMC2022) as a speech corpus to apply our method [1]. VMC2022 contained 175 systems, including natural and synthetic samples. We selected the top 60 systems regarding MOS from the 175 systems to focus on ranking high-quality regions. The top 60 systems had MOS ranging from 3.29 to 4.48 and consisted of 9 natural samples and 51 synthetic speech methods from Blizzard Challenge [20, 21, 22, 23, 24, 25], Voice Conversion Challenge [26, 27, 27, 28], and ESPNet [29]. The total pair combination of the 60 systems was 1,770 pairs, so it was hard to evaluate them all by brute force, which motivated us to utilize our optimization algorithm approach. We divided the top 60 systems into two sets based on whether their rank was odd or even. We considered the situations where we wanted to evaluate 60 systems, but we could not afford to evaluate them in a single shot, or where we evaluated 30 systems in the past, and another 30 systems emerged later. Our experiments investigated how we could rank all 60 systems in these situations.

To conduct the three experiments, we recruited evaluators by crowdsourcing. The evaluators were presented with two speech samples and asked to select a sample that sounded more natural than the other. We modified the question sentence used in the VMC2022 to ask for a comparison. We recruited 255, 292, and 236 Japanese evaluators and obtained 24,960, 24,960, and 15,540 preference scores for the three experiments, respectively. Each pair was rated by as many evaluators as possible by balancing assignments so that the evaluator bias could be offset by averaging.

We set the initial order of system sets in the MOS order for the INSERT-RANK and MERGE-RANK algorithms. It meant that the number of pairs to sort would be the minimum if the final ordering by the preference tests was same as that of MOS. We configured the confidence δ and tolerance bias ϵ of INSERT-RANK, MERGE-RANK, and the MERGE algorithm with $\delta = 0.05$ and $\epsilon = 0.0877$. This configuration meant that each pair was evaluated with 240 scores at maximum to determine its winner. Some pairs did not need 240 scores to determine a winner, so some budgets remained. In that case, the remaining budget was consumed to reduce error bias below the tolerance bias ϵ . Thus, we measured the actual convergence cost, which was the number of scores evaluated at the convergence of the algorithms.

Speech samples presented to evaluators were selected from the systems so that parallel utterances could be paired. If parallel utterances were not available for the system pair, random utterances were selected. The unparallel utterance pairs happened in the case where systems from different subsets were paired, for example, the Blizzard Challenge 2008 and Voice Conversion Challenge 2020.

As a post-hoc analysis, we calculated the number of significantly different pairs using the Binomial test.

4.2. Experimental Results

The ranking of 60 systems was obtained successfully by the three experiments, which proved the feasibility of the continual subjective evaluation with our method. Figure 3 shows the ranking obtained from our preference tests with a ranking cor-

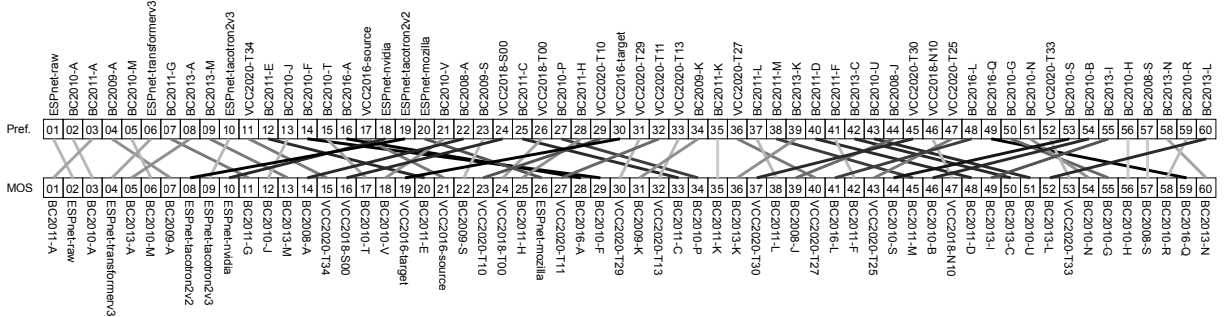


Figure 3: Ranking comparison between preference and MOS tests.

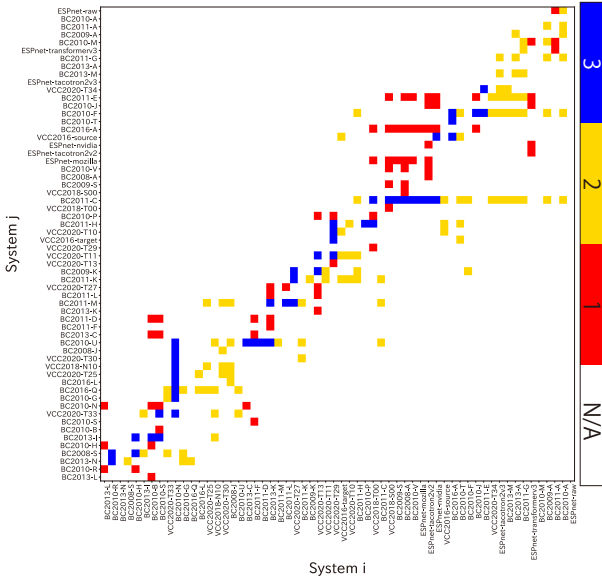


Figure 4: Distribution of pairs classified by experiments.

relation from the MOS test recorded in VMC 2022. The ranking correlation coefficients were 0.798 in Kendall’s tau, and 0.943 in Spearman correlation coefficient, which indicated relatively high agreement between the ranking from our evaluations and the MOS evaluation. The top ranked systems were natural samples and the bottom ranked systems were speech synthesis methods with low MOS, as expected. During sorting systems from the initial ranking based on MOS, many reordering occurred in the ranking ranges between 10 and 30 and between 40 and 60. Some reorderings were expected because many MOS in the VMC 2022 were known to lack statistically significant differences, especially in the middle MOS region between 2.5 and 3.8 [30]. Many systems were concentrated in this region, so the MOS might suffer from contraction bias, which was the compression of scores within a fixed quality scale regardless of actual perceptual quality differences. On the other hand, our method did not suffer from contraction bias because we adopted preference tests. Thus, our method could be used for fine-grained analysis for the quality region where many systems were concentrated. We considered this was promising as an evaluation method to record training labels for more reliable automatic quality assessment models.

Table 1 shows the statistics of the experimental results of the three experiments. The first experiment compared 70 pairs to sort 30 systems, the second experiment compared 98 pairs to

sort 30 systems and merge 10 systems, and the third experiment compared 48 pairs to merge the two sorted system sets. In the second experiment, 89 pairs were compared by the MERGE-RANK algorithm, and 9 pairs were compared by the MERGE algorithm.

Regarding the sorting efficiency, the INSERT-RANK algorithm was more efficient than the MERGE-RANK algorithm: only 70 pairs were evaluated by the INSERT-RANK, whereas 89 pairs were evaluated by the MERGE-RANK to sort 30 systems. However, we observed slow execution performance of the INSERT-RANK in practice when many evaluators participated simultaneously because the algorithm was not parallelizable: only one pair can be evaluated simultaneously in the INSERT-RANK, whereas up to 15 pairs can be evaluated simultaneously in the MERGE-RANK.

Regarding the statistical significance of preference scores from each experiment, the first experiment got 28 significant pairs (40%) from the 70 pairs, the second experiment got 49 significant pairs (50%) from 98 pairs, and the third experiment got 21 significant pairs (44%) from 48 pairs. The fewer significant pairs of the INSERT-RANK and MERGE algorithms indicated that they focused on pairs with similar quality. On the other hand, it indicated that the MERGE-RANK algorithm evaluated not only pairs with similar quality but also pairs with distinct quality differences.

Figure 4 shows the distribution of evaluated pairs classified by the three experiments. The vertical and horizontal axes were systems sorted in the final order. The first experiment using the INSERT-RANK algorithm and the third experiment using the MERGE algorithm mainly evaluated systems in the diagonal region. It also supported that they focused on pairs with similar quality, resulting in a small number of pairs to sort. The second experiment using the MERGE-RANK and MERGE algorithm showed a relatively broader distribution, indicating it also evaluated pairs with distinct quality differences, resulting in a larger number of pairs to sort. It also showed that merging the top 10 systems was handled by the second experiment by using the MERGE sub-algorithm seamlessly in MERGE-RANK, so the third experiment using the MERGE algorithm merged systems from 11 to 60 ranks.

Figure 5 shows the centered preference score $\hat{p}_{ij} - \frac{1}{2}$, evaluation cost r_{ij} , and error bias $\hat{\epsilon}$ of the evaluated pairs (i, j) . The centered preference score is the shifted average preference score from $[0, 1]$ to $[-0.5, 0.5]$ to show that zero indicates equal naturalness. The preference score in Figure 5a showed that pairs in the diagonal region had similar naturalness, whereas pairs in the off-diagonal region had distinct preferences. It indicated that comparing pairs with distant ranks could be skipped or reduced because their winners were obvious, supporting our sort-

- yond speech quality prediction,” in *SLT*. IEEE, 2024, pp. 803–810.
- [6] C. Lo, S. Fu, W. Huang, X. Wang, J. Yamagishi, Y. Tsao, and H. Wang, “Mosnet: Deep learning-based objective assessment for voice conversion,” in *INTERSPEECH*. ISCA, 2019, pp. 1541–1545.
 - [7] Y. Leng, X. Tan, S. Zhao, F. K. Soong, X. Li, and T. Qin, “MB-NET: MOS prediction for synthesized speech with mean-bias network,” in *ICASSP*. IEEE, 2021, pp. 391–395.
 - [8] T. Saeki, D. Xin, W. Nakata, T. Koriyama, S. Takamichi, and H. Saruwatari, “UTMOS: UTokyo-SaruLab System for Voice-MOS Challenge 2022,” in *Proc. Interspeech 2022*, 2022, pp. 4521–4525.
 - [9] S. Zielinski, F. Rumsey, and S. Bech, “On some biases encountered in modern audio quality listening tests—a review,” *J. Audio Eng. Soc.*, vol. 56, no. 6, pp. 427–451, 2008. [Online]. Available: <http://www.aes.org/e-lib/browse.cfm?elib=14393>
 - [10] S. Zielinski, “On some biases encountered in modern audio quality listening tests (part 2): Selected graphical examples and discussion,” *J. Audio Eng. Soc.*, vol. 64, no. 1/2, pp. 55–74, 2016. [Online]. Available: <http://www.aes.org/e-lib/browse.cfm?elib=18105>
 - [11] E. Cooper and J. Yamagishi, “Investigating range-equalizing bias in mean opinion score ratings of synthesized speech,” 2023.
 - [12] V. Bengs, R. Busa-Fekete, A. E. Mesaoudi-Paul, and E. Hüllermeier, “Preference-based online learning with dueling bandits: A survey,” *J. Mach. Learn. Res.*, vol. 22, pp. 7:1–7:108, 2021.
 - [13] Y. Yasuda and T. Toda, “Automatic design optimization of preference-based subjective evaluation with online learning in crowdsourcing environment,” *CoRR*, vol. abs/2403.06100, 2024.
 - [14] H. H. Goldstine and J. von Neumann, “Planning and coding of problems for an electronic computing instrument, part II, volume 2, reprinted in John von Neumann Collected Works, volume V: Design of computers, theory of automata and numerical analysis,” pp. 152—214, 1963.
 - [15] M. Falahatgar, A. Orlitsky, V. Pichapati, and A. T. Suresh, “Maximum selection and ranking under noisy comparisons,” in *ICML*, ser. Proceedings of Machine Learning Research, vol. 70. PMLR, 2017, pp. 1088–1096.
 - [16] A. Elo, *The Rating of Chess Players, Past and Present*. Arco Publishing, 1978.
 - [17] R. Herbrich, T. Minka, and T. Graepel, “Trueskilltm: A bayesian skill rating system,” in *NIPS*. MIT Press, 2006, pp. 569–576.
 - [18] T. Baumann, “Large-scale speaker ranking from crowdsourced pairwise listener ratings,” in *INTERSPEECH*. ISCA, 2017, pp. 2262–2266.
 - [19] K. Sakaguchi, M. Post, and B. V. Durme, “Efficient elicitation of annotations for human evaluation of machine translation,” in *WMT@ACL*. The Association for Computer Linguistics, 2014, pp. 1–11.
 - [20] V. Karaiskos, S. King, R. A. J. Clark, and C. Mayo, “The blizzard challenge 2008,” in *The Blizzard Challenge 2008*, 2008, pp. 1–18.
 - [21] S. King and V. Karaiskos, “The blizzard challenge 2009,” in *The Blizzard Challenge 2009*, 2009, pp. 1–24.
 - [22] —, “The blizzard challenge 2010,” in *The Blizzard Challenge 2010*, 2010, pp. 1–32.
 - [23] —, “The blizzard challenge 2011,” in *The Blizzard Challenge 2011*, 2011, pp. 1–10.
 - [24] —, “The blizzard challenge 2013,” in *The Blizzard Challenge 2013*, 2013, pp. 1–12.
 - [25] —, “The blizzard challenge 2016,” in *The Blizzard Challenge 2016*, 2016, pp. 1–16.
 - [26] T. Toda, L.-H. Chen, D. Saito, F. Villavicencio, M. Wester, Z. Wu, and J. Yamagishi, “The voice conversion challenge 2016,” in *Interspeech 2016*, 2016, pp. 1632–1636.
 - [27] J. Lorenzo-Trueba, J. Yamagishi, T. Toda, D. Saito, F. Villavicencio, T. Kinnunen, and Z. Ling, “The voice conversion challenge 2018: Promoting development of parallel and nonparallel methods,” in *The Speaker and Language Recognition Workshop (Odyssey 2018)*, 2018, pp. 195–202.
 - [28] Z. Yi, W.-C. Huang, X. Tian, J. Yamagishi, R. K. Das, T. Kinnunen, Z.-H. Ling, and T. Toda, “Voice conversion challenge 2020 — intra-lingual semi-parallel and cross-lingual voice conversion —,” in *Joint Workshop for the Blizzard Challenge and Voice Conversion Challenge 2020*, 2020, pp. 80–98.
 - [29] T. Hayashi, R. Yamamoto, K. Inoue, T. Yoshimura, S. Watanabe, T. Toda, K. Takeda, Y. Zhang, and X. Tan, “Espnet-tts: Unified, reproducible, and integratable open source end-to-end text-to-speech toolkit,” in *ICASSP*. IEEE, 2020, pp. 7654–7658.
 - [30] Y. Yasuda and T. Toda, “Analysis of Mean Opinion Scores in Subjective Evaluation of Synthetic Speech Based on Tail Probabilities,” in *Proc. INTERSPEECH 2023*, 2023, pp. 5491–5495.
 - [31] C.-H. Hu, Y. Yasuda, and T. Toda, “Preference-based training framework for automatic speech quality assessment using deep neural network,” in *Interspeech 2023*, 2023, pp. 546–550.