Speech Audio Corrector: using speech from non-target speakers for one-off correction of mispronunciations in grapheme-input text-to-speech

Jason Fong[1]*, Daniel Lyth*[1]*, Gustav Eje Henter*[2]*, Hao Tang*[1]*, Simon King*[1]

[1]The Centre for Speech Technology Research, University of Edinburgh, UK
[2]Division of Speech, Music and Hearing, KTH Royal Institute of Technology, Sweden
{jason.fong, d.s.lyth, hao.tang, simon.king}@ed.ac.uk, ghe@kth.se

Abstract

Correct pronunciation is essential for text-to-speech (TTS) systems in production. Most production systems rely on pronouncing dictionaries to perform grapheme-to-phoneme conversion. Unlike end-to-end TTS, this enables pronunciation correction by manually altering the phoneme sequence, but the necessary dictionaries are labour-intensive to create and only exist in a few high-resourced languages. This work demonstrates that accurate TTS pronunciation control can be achieved *without* a dictionary. Moreover, we show that such control can be performed without requiring any model retraining or fine-tuning, merely by supplying a single correctly-pronounced reading of a word in a different voice and accent at synthesis time. Experimental results show that our proposed system successfully enables one-off correction of mispronunciations in grapheme-based TTS with maintained synthesis quality. This opens the door to production-level TTS in languages and applications where pronunciation dictionaries are unavailable.

Index Terms: speech synthesis, pronunciation control

1. Introduction

The application of neural sequence-to-sequence modelling to text-to-speech (TTS) has enabled an end-to-end training paradigm. End-to-end TTS directly maps graphemes to speech acoustics without the need for auxiliary features such as phonemes or stress markers [1, 2, 3]. Subsequently end-to-end TTS is a straightforward way to build TTS voices, and has the *potential* to scale to all of the world's languages.

However, one major downside of end-to-end TTS is its tendency to make pronunciation errors. This limits its real world applicability. Such errors occur because natural languages typically do not exhibit one-to-one mappings between graphemes and speech sounds [4, 5, 6], which makes the implicit pronunciation prediction task inside end-to-end TTS models error-prone [7, 8, 9]. Consequently, production-quality TTS systems are not trained in an end-to-end manner. Instead, they use a complex linguistic front-end to process graphemes into features that map to audio in a more consistent manner. The most crucial of these are phonemes, which are generated either by dictionary lookup or a grapheme-to-phoneme prediction model. However such front-end resources require significant linguistic and engineering expertise to develop.

Subsequently, in order to realise the full potential of end-to-end TTS, its tendency to make pronunciation errors must first be solved. One possible solution is to increase the variety of word pronunciations encountered during training. However, due to the Zipfian distribution of words in natural language, typical TTS corpora have limited word type coverage in comparison to pronunciation dictionaries (e.g., 14,750 words in LJ Speech vs 135,000 in CMUdict [8]). Therefore end-to-end TTS models learn implicit pronunciation models that are highly unpredictable when compared to the explicit pronunciation resources of fully fledged TTS pipelines [8]. Although one could theoretically improve end-to-end TTS's pronunciations by retraining on vast amounts of text-audio data, this would complicate model deployment and inflate data collection costs to an insurmountable level. Furthermore, and perhaps more crucially, this data-centric approach would not guarantee any practical improvement as it does not solve the unpredictability intrinsic to the mapping between graphemes and speech sounds. As such, simply training with more data is not a viable solution for TTS voice builders.

An alternative and more feasible solution may instead be a model-centric approach where one trains end-to-end TTS with a source of pronunciation information that is simpler to obtain than phonemes. In this work, we pursue this approach and directly use *speech* during training and at synthesis time to both model and control pronunciations. We introduce Speech Audio Corrector (SAC), a novel grapheme-input TTS model which, unlike prior end-to-end models, can utilise a correction query, consisting of just speech, to accurately make one-off corrections of word mispronunciations at synthesis time without requiring any model retraining or fine-tuning.

Due to the added complexity of modelling speech from scratch, SAC uses word aligned speech codes instead of speech waveforms or acoustics to represent pronunciations. Speech codes are discretised acoustic representations extracted from self-supervised learning models trained on large amounts of speech data. Speech codes are a good input representation for speech synthesis as they have been shown to correlate well with phonemes [10, 11, 12] and yet desirably are relatively free of speaker information [13]. Furthermore, speech codes are simple to obtain as self-supervised learning models can be trained on abundant found speech data such as audiobooks or podcasts.

The main contributions of this paper are as follows: We demonstrate that SAC can correct a word's pronunciation using speech obtained from a non-target speaker regardless of their accent, and find that matching the speech correction's accent to the target voice is preferred to using a mismatching accent. We also find that adding SAC's corrective functionality does not heavily impact standard grapheme-input TTS pronunciations.

2. Related work

2.1. Generation from self-supervised acoustic representations

Self-supervised learning (SSL) representations of speech from models such as HuBERT [12], wav2vec2.0 [11], and VQ-VAE [10] have been used for a wide variety of downstream

tasks such as speech recognition [12, 11, 14], speech coding [15, 16], voice conversion [10, 16], natural language generation [17, 18, 13], and speech-to-speech translation [19, 20]. However, SSL acoustic representations have not been used to improve the core functionality of neural TTS. While SSL representations have been used in TTS as intermediate features, their use has been motivated by reducing reliance on engineered features [21], improving duration modelling [22], and increasing prosodic variation [23, 24]. The current work, to the best of our knowledge, is the first to employ the phonetic information contained in discrete speech codes for correcting pronunciation in grapheme-input TTS.

2.2. TTS pronunciation control via speech recordings

Unit selection is a long-standing TTS paradigm in which words are pronounced by selecting from a database and then concatenating a sequence of recorded speech fragments ('units') whose labels match those of the text to be spoken [25]. Importantly, the labels are used only as an index into the database and therefore do not necessarily need to represent fine acoustic detail or even prosody: these come 'for free' by choosing units from an appropriate context. However, concatenating units on the *output* side results in join artefacts that are unresolvable using signal processing. Worse, the database must comprise speech from a single speaker. Incorporating, or adding at a later stage, non-target speaker data is infeasible.

In contrast, our proposed model SAC predicts mel-spectrograms from linguistic and speech features. These are concatenated on the *input* side, and thus SAC synthesises speech with fewer join artefacts. SAC is designed to enable pronunciation control using speech codes from non-target speakers without affecting the identity of the target synthesised voice.

2.3. Multi-modal input for TTS pronunciation control

The pronunciation control aspect of this work is similar to that of the representation mixing paradigm in which graphemes and phonemes are interleaved during training in order to enable synthesis mainly from grapheme input with the option of pronunciation control using phonemes at synthesis time. Both RNNs [26] and CNNs [27] have been used for this.

In SAC, a transformer encoder uses self-attention and word positional information to substitute masked out graphemes with speech codes. SAC's architecture was inspired by the text encoder of PnG BERT [28] which uses self-attention to incorporate phoneme and grapheme sequences yielding synthesised speech with more natural prosody.

Our own previous work has investigated the viability of grapheme-phoneme representation mixing in low-resource scenarios [27]. We found that robust pronunciation control is lost when using a small pronunciation dictionary (i.e., one that only covers a small percentage of the word types seen during training). That is, there is a cost-benefit trade-off associated with representation mixing: Is the cost of phonemically transcribing more word types worth the pronunciation control gained? In contrast, SAC avoids this trade-off because speech codes are available for *all* word types seen during training (i.e., there is an implicit word type-to-speech code 'dictionary' with 100% coverage). This allows SAC to learn a robust back-off model for controlling pronunciation with no cost-benefit trade-off.

3. Speech Audio Corrector (SAC)

At the core of SAC (Figure 1) is Transformer TTS [3], which is trained end-to-end to predict mel-spectrograms from linguistic input – normally graphemes or phonemes. SAC adds the use of speech codes, to enable a multi-modal correction query to control a word's pronunciation at synthesis time. A multi-modal correction query comprises the grapheme sequence of the text to be synthesised, plus a sequence of speech codes for the word(s) whose pronunciation we wish to control, those words being masked out in the grapheme sequence. This reduces to regular grapheme-input TTS when no speech codes are present. Additionally, auxiliary features are summed to the correction query to improve correction robustness at synthesis time. Token and word positional information is added to help the model align masked out graphemes to their speech codes and modality information is added to indicate to the model where grapheme and speech codes sequences are within the correction query.

To form a correction query, a sequence of graphemes are processed by the Correction Query Builder (CQB) module which masks out the words that we wish to correct and retrieves their corresponding speech codes. CQB outputs a sequence of graphemes with mask tokens concatenated to a sequence of speech codes, forming $\mathbf{x_{1:N}}$ where $\mathbf{N}$ is the total number of timesteps of the correction query. The two sequences are each wrapped with start-of-sequence $\langle$sos$\rangle$ and end-of-sequence $\langle$eos$\rangle$ symbols. Each of the symbols in $\mathbf{x_{1:N}}$ is looked up in a shared embedding table, to obtain $\mathbf{e_{1:N}}$.

To the embedded correction query $\mathbf{e_{1:N}}$, we sum modality, token position, and word position features. Word alignment features help SAC align the masked-out graphemes with the corresponding speech codes. We embed token and word positions forming positional embeddings $\mathbf{tpe_{1:N}}$ and $\mathbf{wpe_{1:N}}$. Additionally the word position embeddings are passed through a learned linear projection DENSE to help distinguish them from the token positions. Modality embeddings $\mathbf{mode_{1:N}}$ are also employed to help the model differentiate where the grapheme and speech code sequences are located. Finally, learned weights α, β, and γ are applied to the auxiliary embeddings before summing them to $\mathbf{x_{1:N}}$. These allow the model to choose which features to prioritise. SAC's inputs are summarised in Equation 1. Once the inputs are formed and summed, the architecture and loss to be optimised during training are identical to Transformer TTS.

$$\mathbf{e_{1:N}} + \alpha \cdot \mathbf{tpe_{1:N}} + \beta \cdot \textsc{Dense}(\mathbf{wpe_{1:N}}) + \gamma \cdot \mathbf{mode_{1:N}} \quad (1)$$

4. Experimental setup

In this section, we detail our choices regarding data, model training, model selection, and evaluation in order to test the following hypotheses:

- H_1: speech codes obtained from the speech of a *non-target speaker* can be used to make one-off pronunciation corrections
- H_2: speech codes from a non-target speaker with an accent *matching* that of the target speaker will provide more accurate pronunciation corrections than for a *differing* accent

4.1. Data

To train SAC, we use text and speech from chapters 4 to 50 of the LJ Speech [29] corpus. We use chapter 3 of LJ Speech for

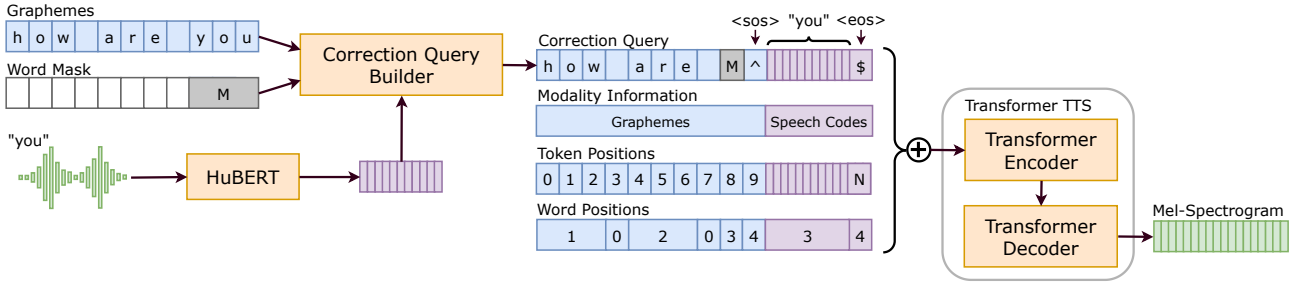

Figure 1: *Architectural overview of Speech Audio Corrector (SAC) performing synthesis time one-off pronunciation correction of grapheme-input TTS using a recorded exemplar encoded as speech codes.*

our development set. LJ Speech contains 24 hours of speech from a single female US speaker. We use the standard normalised text transcription of LJ Speech.

In order to obtain word aligned speech codes from the recorded speech, we first extract 50 Hz layer-6 representations using HuBERT-Base-LS960[1] from 16 kHz speech. We use layer-6 representations as they have been found to correlate well with phone identities [12]. Next we discretise these representations into speech codes using a 100 clusters k-means model[2] which was trained on HuBERT-Base-LS960 representations extracted from librispeech 960h [30]. The authors of [13] demonstrate that the use of 100 clusters provides an appropriate trade-off between phonetic quality (measured by WER) and speaker-agnosticism (measured by speaker probing accuracy). In this work, we choose to not remove duplicate speech codes from the resulting sequences. De-duplicating speech code sequences has been a common choice in recent generative spoken language modelling work [17, 18] as doing so reduces computational load and increases modelling performance. However, we found that de-duplication makes attention-based duration modelling difficult resulting in unreliable pronunciation control when synthesising speech codes obtained from non-target speakers. Finally, we align speech codes at the word-level using Montreal Forced Aligner [31]. From this, we derive a dictionary comprising one-to-many mappings between word types and corresponding speech code sequences.

To test H_1 and H_2 we utilise the multi-speaker, multi-accent VCTK corpus [32]. We obtain word aligned speech codes from VCTK in an identical fashion to LJ Speech, with each word type typically being spoken by multiple speakers across a variety of accents.

4.2. Model

We trained 2 models: a baseline Transformer TTS model (hereafter: TTS), and our proposed model (SAC). We use the Transformer TTS implementation in fairseq [33], and implement SAC on top of this. Subsequently the core model architecture is identical between TTS and SAC and follows that of Transformer TTS [3]: The transformer encoder and decoder both have 6 layers each with 4 attention heads, and a hidden size of 512 between all layers. The embeddings of the correction query, modality information, token position, and word position features are each 512 dimensions. Both TTS and SAC are trained without an attention guide loss [2] since there is no strict monotonic alignment between the correction query and output acoustics when speech codes are included within the query.

Both models are trained in an identical manner, with identical hyper-parameters such as learning rate and batch size. Both models use graphemes rather than phonemes to reflect a low-resource scenario. The training data for SAC comprises randomly-constructed correction queries. For each word in each input grapheme sequence, we mask it out then provide the corresponding speech codes, with a 50% probability. In preliminary training runs we searched over hyper-parameters to find a SAC model that minimised Mel-Cepstrum Distortion (MCD) calculated over the LJ Speech development set when all words are represented by their respective speech codes rather than graphemes. Listening test stimuli were generated from epoch 1000 of TTS and SAC which minimised SAC's development set MCD.

Finally, we use a pretrained HiFi-GAN vocoder [34] to synthesise waveforms from the mel-spectrograms predicted from both TTS and SAC.

4.3. Pronunciation correction evaluation

4.3.1. Listening test design

To evaluate SAC's pronunciation correction capability we ran an AB listening test in which listeners were presented with pairs of samples (each synthesised under a different condition), and asked to choose which pronunciation is preferred; a 'no preference' option was also available. In order to answer H_1 and H_2 we synthesised samples under the following four conditions:

TTS_G: Transformer TTS using grapheme inputs
SAC_G: SAC using grapheme inputs
SAC_{US}: SAC using US female speech code inputs
SAC_{Scot}: SAC using Scottish female speech code inputs

From these we created all 6 possible pairings of conditions, to be presented to listeners. Each synthesised sample consisted of a target word placed within the carrier sentence *'How is ... pronounced?'*. For each condition we used 78 target words to synthesise samples. Each of the target words is: not in LJ-train, mispronounced by SAC_G, and spoken by at least one US and one Scottish female non-target speaker. For SAC_{US} and SAC_{Scot} the target word is represented by speech codes extracted from a randomly-chosen VCTK speaker of that accent. For SAC_{US}, the speaker was chosen from amongst the 18 available US female speakers in VCTK (with each speaker on average contributing 4.3 words), and for SAC_{Scot} from amongst the 13 available Scottish female speakers (on average contributing 6 words each). All samples used in the listening test can be found on our sam-

[1] https://dl.fbaipublicfiles.com/hubert/hubert_base_ls960.pt

[2] https://dl.fbaipublicfiles.com/textless_nlp/gslm/hubert/km100/km.bin

Table 1: *Count matrix obtained from subjective listening tests taken by native US participants. The i,j-th count is the number of times condition i is preferred over condition j. Ties are accounted for by assigning half a win to each item.*

	TTS_G	SAC_G	SAC_{US}	SAC_{Scot}	Combined
TTS_G	-	604.5	263	346	1213.5
SAC_G	565.5	-	215	315.5	1096
SAC_{US}	907	955	-	674.5	2536.5
SAC_{Scot}	824	854.5	495.5	-	2174

ples page[3]. Each stimulus within the listening test comprised a pair of samples, each generated under a different condition. 6 condition pairs $\times$ 78 stimuli $=$ 468 stimuli. Since this is too many for a single listener to judge, we split them into 6 subtests using a Latin square[4]. Each such subtest took a listener approximately 15 minutes to complete. This design ensures that each subtest contains all 6 condition pair and all 78 target words. To mitigate ordering effects, we randomised each subtest's stimuli order and each within-stimulus condition order, differently per listener.

We used Prolific[5] to recruit 90 listeners for our listening test, all of whom were native English speakers and US citizens: this is to test H_2, where we expect to generate more accurate US English pronunciations when using speech codes from a US-accented non-target speaker than a different accents. Each of the 6 subtests was implemented with Qualtrics[6] and was taken by 15 listeners.

4.3.2. *Bradley-Terry statistical analysis*

We used Bradley-Terry model [35] parameter estimation to obtain scores representing the relative listener preference for each condition. Bradley-Terry requires a matrix of pairwise comparison counts $C_{i,j}$ where the i,j-th count is the number of times condition i is preferred over condition j. When the 'no preference' option was chosen by listeners each condition is awarded half a 'win'. We used the choix[7] python package to estimate Bradley-Terry parameters from $C_{i,j}$ by maximum likelihood using the Iterative Luce Spectral Ranking algorithm.

5. Results

5.1. H_1: speech codes enable one-off pronunciation correction

In Figure 2 we observe *considerably* higher scores for SAC_{US} and SAC_{Scot} than for TTS_G and SAC_G. The 'Combined' column in the counts matrix presented in Table 1 shows that both SAC_{US} and SAC_{Scot} are preferred approximately twice as much over the grapheme-input conditions. So, we can conclude that one-off pronunciation correction using speech codes is successful, supporting H_1. We also observe that TTS_G is slightly preferred over SAC_G. Expert listening (by the first author) revealed that TTS_G pronounces 7 of the target words correctly, while all 78 target words are *by design* mispronounced by SAC_G.

[3] https://jonojace.github.io/IS22-speech-audio-corrector

[4] https://cs.uwaterloo.ca/~dmasson/tools/latin_square

[5] https://www.prolific.co/

[6] https://www.qualtrics.com/

[7] https://pypi.org/project/choix/

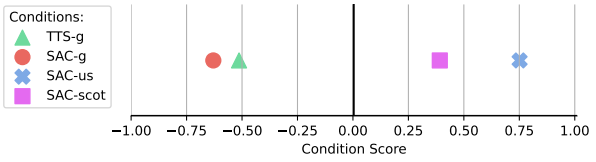

Figure 2: *A visualisation of the Bradley-Terry model parameters estimated by maximum likelihood via the Iterative Luce Spectral Ranking algorithm. These parameter estimates can be regarded as the score or strength of each condition (higher is better) and are calculated from the pairwise comparison count matrix in Table 1. Precise score values are -0.63 for SAC_G, -0.51 for TTS_G, 0.39 for SAC_{Scot} and 0.75 for SAC_{US}.*

5.2. H_2: US speech codes lead to more preferred pronunciations than Scottish ones

We find that SAC_{US} is preferred over SAC_{Scot} by our US listeners, supporting H_2. To further investigate this result we identified the target words generated from SAC_{US} and SAC_{Scot} that were heavily preferred, observing that both conditions can pronounce words more correctly than the other condition. Despite this bilateral effect however, SAC_{Scot} was more likely to incorrectly pronounce words, mispronouncing 19 of the words (24%) versus SAC_{US} mispronouncing 12 words (15%). This may be due to Scottish speech codes being from a different data distribution than the US speech codes encountered during training. Nevertheless we believe that Scottish speech codes worked reasonably well, exhibiting pronunciation performance superior to grapheme input, and do not noticeably affect the identity of the synthesised voice. Furthermore we identified several words where SAC_{Scot}'s pronunciation is correct but SAC_{US}'s rendition was preferred by listeners simply because SAC_{Scot} pronunciations are Scottish. For example SAC_{US} pronounces 'derby' as /ˈdɝː.bi/ but SAC_{Scot} pronounces it as /ˈdɑː.bi/ which would be judged as incorrect by most native US speakers. Other such target words include 'mobile', 'bother', and 'comedy'.

6. Conclusion

In this work, we introduced and demonstrated a novel grapheme-input TTS model 'Speech Audio Corrector' (SAC) that can use speech to make one-off pronunciation corrections. In our experiments, we find that speech from both US and Scottish non-target speakers can correct a SAC system trained on a single target US speaker. We also find that matching the speech accent to the target speaker improves pronunciation correction robustness. SAC's ability to utilise such a simple source of pronunciation information makes it a cost effective paradigm for production use cases, and especially so for low-resource TTS. Furthermore, SAC has the potential to be extended to controlling or correcting non-segmental characteristics such as prosody, also without model retraining or fine-tuning. In future work we wish to make SAC robust to accent, investigate its potential for multilingual code-switching, and remove its reliance on high-resource ASR which we currently use to find word alignments.

Acknowledgements: This work was partially supported by the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation.

7. References

[1] J. Shen, R. Pang, R. J. Weiss, M. Schuster, N. Jaitly, Z. Yang, Z. Chen, Y. Zhang, Y. Wang, R. Skerrv-Ryan *et al.*, "Natural tts synthesis by conditioning wavenet on mel spectrogram predictions," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018, pp. 4779–4783.

[2] H. Tachibana, K. Uenoyama, and S. Aihara, "Efficiently trainable text-to-speech system based on deep convolutional networks with guided attention," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018, pp. 4784–4788.

[3] N. Li, S. Liu, Y. Liu, S. Zhao, and M. Liu, "Neural speech synthesis with transformer network," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, no. 01, 2019, pp. 6706–6713.

[4] X. Marjou, "OTEANN: Estimating the transparency of orthographies with an artificial neural network," *arXiv preprint arXiv:1912.13321*, 2019.

[5] R. Pereira, "Linguistic Map of Orthographic depth," https://linguisticmaps.tumblr.com/post/187856489343/orthographic-depth-languages-have-different-levels, 2019.

[6] A. Perquin, E. Cooper, and J. Yamagishi, "An investigation of the relation between grapheme embeddings and pronunciation for tacotron-based systems," *arXiv preprint arXiv:2010.10694*, 2020.

[7] J. Fong, J. Taylor, K. Richmond, and S. King, "A comparison between letters and phones as input to sequence-to-sequence models for speech synthesis," in *10th ISCA Speech Synthesis Workshop*, 2019.

[8] J. Taylor and K. Richmond, "Analysis of pronunciation learning in end-to-end speech synthesis," in *20th Annual Conference of the International Speech Communication Association: Crossroads of Speech and Language*. International Speech Communication Association, 2019, pp. 2070–2074.

[9] Y. Yasuda, X. Wang, and J. Yamagishi, "Investigation of learning abilities on linguistic features in sequence-to-sequence text-to-speech synthesis," *Computer Speech & Language*, vol. 67, p. 101183, 2021.

[10] A. v. d. Oord, O. Vinyals, and K. Kavukcuoglu, "Neural discrete representation learning," *arXiv preprint arXiv:1711.00937*, 2017.

[11] A. Baevski, H. Zhou, A. Mohamed, and M. Auli, "wav2vec 2.0: A framework for self-supervised learning of speech representations," *arXiv preprint arXiv:2006.11477*, 2020.

[12] W.-N. Hsu, B. Bolte, Y.-H. H. Tsai, K. Lakhotia, R. Salakhutdinov, and A. Mohamed, "HuBERT: Self-Supervised Speech Representation Learning by Masked Prediction of Hidden Units," *arXiv preprint arXiv:2106.07447*, 2021.

[13] E. Kharitonov, J. Copet, K. Lakhotia, T. A. Nguyen, P. Tomasello, A. Lee, A. Elkahky, W.-N. Hsu, A. Mohamed, E. Dupoux *et al.*, "textless-lib: a Library for Textless Spoken Language Processing," *arXiv preprint arXiv:2202.07359*, 2022.

[14] S.-w. Yang, P.-H. Chi, Y.-S. Chuang, C.-I. J. Lai, K. Lakhotia, Y. Y. Lin, A. T. Liu, J. Shi, X. Chang, G.-T. Lin *et al.*, "SUPERB: Speech processing Universal PERformance Benchmark," *arXiv preprint arXiv:2105.01051*, 2021.

[15] E. Dunbar, R. Algayres, J. Karadayi, M. Bernard, J. Benjumea, X.-N. Cao, L. Miskic, C. Dugrain, L. Ondel, A. W. Black *et al.*, "The zero resource speech challenge 2019: TTS without T," *arXiv preprint arXiv:1904.11469*, 2019.

[16] A. Polyak, Y. Adi, J. Copet, E. Kharitonov, K. Lakhotia, W.-N. Hsu, A. Mohamed, and E. Dupoux, "Speech resynthesis from discrete disentangled self-supervised representations," *arXiv preprint arXiv:2104.00355*, 2021.

[17] K. Lakhotia, E. Kharitonov, W.-N. Hsu, Y. Adi, A. Polyak, B. Bolte, T.-A. Nguyen, J. Copet, A. Baevski, A. Mohamed *et al.*, "On generative spoken language modeling from raw audio," *Transactions of the Association for Computational Linguistics*, vol. 9, pp. 1336–1354, 2021.

[18] E. Kharitonov, A. Lee, A. Polyak, Y. Adi, J. Copet, K. Lakhotia, T.-A. Nguyen, M. Rivière, A. Mohamed, E. Dupoux *et al.*, "Text-free prosody-aware generative spoken language modeling," *arXiv preprint arXiv:2109.03264*, 2021.

[19] A. Lee, P.-J. Chen, C. Wang, J. Gu, X. Ma, A. Polyak, Y. Adi, Q. He, Y. Tang, J. Pino *et al.*, "Direct speech-to-speech translation with discrete units," *arXiv preprint arXiv:2107.05604*, 2021.

[20] A. Lee, H. Gong, P.-A. Duquenne, H. Schwenk, P.-J. Chen, C. Wang, S. Popuri, J. Pino, J. Gu, and W.-N. Hsu, "Textless Speech-to-Speech Translation on Real Data," *arXiv preprint arXiv:2112.08352*, 2021.

[21] T. Hayashi and S. Watanabe, "Discretalk: Text-to-speech as a machine translation problem," *arXiv preprint arXiv:2005.05525*, 2020.

[22] Y. Yasuda, X. Wang, and J. Yamagishd, "End-to-end text-to-speech using latent duration based on vq-vae," in *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2021, pp. 5694–5698.

[23] G. E. Henter, J. Lorenzo-Trueba, X. Wang, and J. Yamagishi, "Deep encoder-decoder models for unsupervised learning of controllable speech synthesis," *arXiv preprint arXiv:1807.11470*, 2018.

[24] G. Sun, Y. Zhang, R. J. Weiss, Y. Cao, H. Zen, A. Rosenberg, B. Ramabhadran, and Y. Wu, "Generating diverse and natural text-to-speech samples using a quantized fine-grained vae and autoregressive prosody prior," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 6699–6703.

[25] A. J. Hunt and A. W. Black, "Unit selection in a concatenative speech synthesis system using a large speech database," in *1996 IEEE International Conference on Acoustics, Speech, and Signal Processing Conference Proceedings*, vol. 1. IEEE, 1996, pp. 373–376.

[26] K. Kastner, J. F. Santos, Y. Bengio, and A. Courville, "Representation mixing for TTS synthesis," in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 5906–5910.

[27] J. Fong, J. Taylor, and S. King, "Testing the Limits of Representation Mixing for Pronunciation Correction in End-to-End Speech Synthesis." in *INTERSPEECH*, 2020, pp. 4019–4023.

[28] Y. Jia, H. Zen, J. Shen, Y. Zhang, and Y. Wu, "PnG BERT: Augmented BERT on Phonemes and Graphemes for Neural TTS," *arXiv preprint arXiv:2103.15060*, 2021.

[29] K. Ito and L. Johnson, "The LJ Speech Dataset," https://keithito.com/LJ-Speech-Dataset/, 2017.

[30] V. Panayotov, G. Chen, D. Povey, and S. Khudanpur, "Librispeech: an asr corpus based on public domain audio books," in *2015 IEEE international conference on acoustics, speech and signal processing (ICASSP)*. IEEE, 2015, pp. 5206–5210.

[31] M. McAuliffe, M. Socolof, S. Mihuc, M. Wagner, and M. Sonderegger, "Montreal Forced Aligner: Trainable Text-Speech Alignment Using Kaldi." in *Interspeech*, vol. 2017, 2017, pp. 498–502.

[32] C. Veaux, J. Yamagishi, K. MacDonald *et al.*, "CSTR VCTK corpus: English multi-speaker corpus for CSTR voice cloning toolkit," *University of Edinburgh. The Centre for Speech Technology Research (CSTR)*, 2017.

[33] M. Ott, S. Edunov, A. Baevski, A. Fan, S. Gross, N. Ng, D. Grangier, and M. Auli, "fairseq: A fast, extensible toolkit for sequence modeling," *arXiv preprint arXiv:1904.01038*, 2019.

[34] J. Kong, J. Kim, and J. Bae, "HiFi-GAN: Generative adversarial networks for efficient and high fidelity speech synthesis," *Advances in Neural Information Processing Systems*, vol. 33, pp. 17 022–17 033, 2020.

[35] R. A. Bradley and M. E. Terry, "Rank analysis of incomplete block designs: I. The method of paired comparisons," *Biometrika*, vol. 39, no. 3/4, pp. 324–345, 1952.