INTERSPEECH 2020
October 25–29, 2020, Shanghai, China

Testing the Limits of Representation Mixing for Pronunciation Correction in End-to-End Speech Synthesis

Jason Fong, Jason Taylor, Simon King

The Centre for Speech Technology Research, University of Edinburgh, United Kingdom
{jason.fong, jason.taylor, simon.king}@ed.ac.uk

Abstract

Accurate pronunciation is an essential requirement for text-to-speech (TTS) systems. Systems trained on raw text exhibit pronunciation errors in output speech due to ambiguous letter-to-sound relations. Without an intermediate phonemic representation, it is difficult to intervene and correct these errors. Retaining explicit control over pronunciation runs counter to the current drive toward end-to-end (E2E) TTS using sequence-to-sequence models. On the one hand, E2E TTS aims to eliminate manual intervention, especially expert skill such as phonemic transcription of words in a lexicon. On the other, a system making difficult-to-correct pronunciation errors is of little practical use. Some intervention is necessary. We explore the minimal amount of linguistic features required to correct pronunciation errors in an otherwise E2E TTS system that accepts graphemic input. We use representation-mixing: within each sequence the system accepts either graphemic and/or phonemic input. We quantify how little training data needs to be phonemically labelled - that is, how small a lexicon must be written - to ensure control over pronunciation. We find modest correction is possible with 500 phonemised word types from the LJ speech dataset but correction works best when the majority of word types are phonemised with syllable boundaries.

Index Terms: speech synthesis, representation mixing, pronunciation control

1. Introduction

The ability to control pronunciation is necessary for text-to-speech synthesis (TTS) in deployment. Predicting pronunciations from text is often difficult for non-standard words like numbers, abbreviations and homographs [1]. More broadly, non-phonemic orthographies (as in English) cause pronunciation prediction to be unreliable.

Traditionally the ambiguities between text and acoustics are tackled by creating a linguistic specification in the front-end. The linguistic specification is an intermediate representation between text and acoustics that contains various levels of linguistic features (phonetic, syllabic, syntactic, prosodic) to enable pronunciation control. Examples of front-end packages include Festival [2], Mary [3] and Sparrowhawk [4].

Developing front-end modules is expensive, especially the pronunciation lexicon. For example, the creation of Unisyn took an estimated 2 person-years [5]. Other available pronunciation lexica for TTS include CMUdict [6] and Combilex [7].

The recent application of sequence-to-sequence modelling (S2S) to TTS has enabled an end-to-end (E2E) paradigm. This is where speech is predicted directly from text without the use of a front-end or linguistic specification. Examples of such models include [8, 9]. The prospect of TTS without a front-end has led to a growing interest in the E2E paradigm, shown most recently by the announcement of the open source ESPnet-TTS research collaboration [10].

However, the need to control pronunciation of output speech runs counter to the current drive towards E2E TTS. A degree of intervention is necessary but should be kept to a minimum.

Motivated by the need to correct pronunciation in E2E TTS systems, *representation-mixing* involves training on a mixture of graphemes and phonemes, with each input *word* represented either graphemically or phonemically [11, 12]. With the option of using phoneme input, it becomes possible to control pronunciations at test time without the need for a complete lexicon of all words in the training data. However, previous research on representation-mixing has not empirically studied the robustness of phoneme correction, nor the quantity of phoneme labelling required.

In this paper, we quantify what the minimal phonemic intervention during training should be for a working pronunciation corrector in a Tacotron 2 model. We analyse 3 factors in pronunciation correction: the number of word types in the training data that are phonemically labelled, whether these word types are selected according to their token frequency, and whether coverage-based selection algorithms can reduce the amount of phoneme labelling needed. Pronunciation correction performance was evaluated by 2 expert linguists.

We find that from as little as 500 word types some correction is possible. As the number of word types labelled during training increases the corrective ability improves. However, 100% accurate pronunciation correction is not possible under representation-mixing. We also find that phoneme correction models are made more robust if syllable boundaries are provided at both training and test time.

2. Linguistic Data

We trained all models using the LJ Speech corpus [13] comprising ~24 hrs of audio from 13,100 sentences from 7 non-fiction LibriVox books read by a single American female speaker. We normalised numbers, ordinals, and monetary units and performed only superficial text pre-processing: removing capitalisation and expanding simple abbreviations such as 'Dr.'. All other non-standard words, such as acronyms, were not normalised.

Training a representation-mixing model involves randomly replacing a word's graphemes with its phoneme sequence. We obtain phoneme sequences from the Unilex pronunciation lexicon [5], as it has wider word type coverage (167,000 entries) and better consistency than open source lexica such as CMUdict [6], which contains entries of uncertain provenance. Unilex uses the Unisyn set of 56 phones (55 of which are found within LJ Speech), and includes both syllable stress and syllable boundary information for each entry, e.g. the entry for 'speech-

http://dx.doi.org/10.21437/Interspeech.2020-2618

less' is /s p ii ch 1 l lw @ s 0/, where digits encode syllable stress and the 'l' symbol represents syllable boundaries.

3. Resource-limited Lexica

As noted earlier, a large lexicon is very expensive to create. To investigate the limits of how well representation-mixing can perform pronunciation correction, we designed a range of pronunciation lexica differing in number and choice of entries. Only the word types contained in a given lexicon are ever randomly phonemicised during training of a system using that lexicon. Three reference lexica were designed as follows:

- `grapheme-only`: an empty lexicon; training a representation-mixing model with this lexicon is equivalent to training with grapheme-only input. This model was used to determine which words are mispronounced by a typical E2E TTS system.
- `oracle-14`: contains 14 word types, which is the smallest possible lexicon that covers all 55 Unisyn phones that occur in LJ Speech at least once. We use this lexicon to discover whether full phoneme coverage is sufficient to enable pronunciation control.
- `full-13049`: all 13,049 word types that co-occur in LJ Speech and Unilex. This lexicon should have the best possible pronunciation correction ability.

We additionally devised 5 word type selection methods that each lead to a lexicon of n entries. We compare models trained with these lexica to determine the most effective size and contents for a resource-limited lexicon. n can range from 0 to N (here, $N = 13049$), and for any $n_2 > n_1$ the lexicon of size n_1 is a strict subset of the lexicon of size n_2.:

- `rand-n`: randomly selects n word types. These lexica help us understand whether a naive selection method can outperform more well-reasoned approaches.
- `freq-n`: selects the top n most frequently occurring word types in LJ Speech. These lexica help answer whether choosing to phonemise words that cover the most tokens during training is the best approach.
- `phone-n`: greedily selects n frequently occurring word types while also trying to achieve wide phoneme coverage. We build these lexica using the greedy Algorithm 1 where the units of concern are phoneme unigrams. This is an oracle condition because it uses pronunciation information from `full-13049`.
- `bigram-n`: identical to `phone-n` but uses character bigrams as the units of concern. These lexica help answer whether wide coverage over graphemic contexts is a potential proxy for wide phoneme coverage and thus may result in decent pronunciation performance. This is a realistic condition to contrast with the previous one.
- `trigram-n`: identical to `bigram-n` but uses character trigrams as the units of concern.

Alongside Algorithm 1 we use a simple score in Equation 1 to build the `phone-n`, `bigram-n` and `trigram-n` lexica. The score weights between frequency and linguistic unit coverage. The score for word type w_i is the frequency of w_i in the speech corpus multiplied by the number of its units that overlap with the current set of unseen units.

$$score_i = freq_i \times num\_overlapping\_units \quad (1)$$

Algorithm 1 Builds a lexicon containing n word types by considering linguistic unit coverage

```
entries ← empty_set()
candidate_words ← all_wordtypes_in_lexicon()
unseen_units ← all_units()
while len(entries) < n do
  while len(unseen_units) > 0 do
    num_unseen_units ← len(unseen_units)
    scores ← score(candidate_words, unseen_units)
    best_word ← max(candidate_words, scores)
    entries.add(best_word)
    candidate_words.remove(best_word)
    unseen_units.remove(get_units(best_word))
    if len(entries) = n then
      return entries
    end if
    if len(unseen_units) = num_unseen_units then
      unseen_units ← all_units()
    end if
  end while
  unseen_units ← all_units()
end while
return entries
```

4. Experiments

We closely followed the representation-mixing training approach detailed in [12]. During training, the graphemes of a word in the input can be randomly replaced by its phoneme sequence. This occurs at a fixed *mixture probability* of p_{mix} only for words in the lexicon being used. We used $p_{mix} = 0.5$ in all experiments. We implemented representation-mixing on top of a Tacotron 2 [8] implementation [14]. The Tacotron 2 model predicts mel-spectrogram frames, from which we use WaveRNN [15] (a single model trained on the LJ Speech corpus is used in all systems) to generate waveforms. Each model is trained on a single Nvidia GTX 1080, with a batch size of 32, learning rate of 0.001 and 350,000 total training steps stopping criterion.

We trained one model for each reference lexicon described in Section 3 – `grapheme-only`, `oracle-14`, `full-13049` – and one model for each combination of word type selection method and value of n in $\{500, 2000, 4000, 6000\}$, for a total of 23 models.

5 supplemental models were trained using the `full-13049` lexicon to answer further questions not related to lexica size or composition:

- `mixprob-up` and `mixprob-down`: linearly vary p_{mix} so that it depends on the frequency rank of the word. `mixprob-up` phonemicises the most common word in LJ Speech with $p_{mix} = 0.5$, and the least common word with $p_{mix} = 0.9$. These values are swapped for `mixprob-down`. These models help investigate whether phonemicising the most frequent or the least frequent words more often can be beneficial for pronunciation correction.
- `stress`, `syllable` and `stress-syllable`: these models additionally include a word's stress and/or syllable information when it is phonemicised during training. These models help investigate whether additional linguistic markup aids pronunciation correction.

After training all the models we then use them to generate our 3 sets of test stimuli[1]. Table 1 shows how many tokens in LJ Speech are covered by each lexicon (expressed as a percentage of the tokens covered by `full-13049`) and therefore could be randomly phonemicised during training.

[1] Samples available at jonojace.github.io/IS20-repmixing-limits

Table 1: *Number of word tokens in LJ Speech covered by each resource-limited lexicon expressed as a percentage of the 223179 tokens covered by* `full-13049`*. Additionally:* `oracle-14` *covers 41 tokens (0.018% of* `full-13049`*).*

n	500	2000	4000	6000
rand	3%	19%	43%	56%
freq	69%	86%	93%	96%
bigram	55%	75%	85%	90%
trigram	44%	49%	65%	72%
phone	66%	81%	90%	94%

5. Evaluation

We developed 3 sets of words, each carried in the sentence "Now we will say ... again." as test stimuli. Each set helps answer a specific question regarding all models:

In-LJ: Phonemic sequences for 50 words that occur in LJ Speech but were never phonemicised during training, and are mispronounced by the baseline `grapheme-only` model: despite being in the training data, they are surprisingly still mispronounced when represented as graphemes. This set investigates which models offer effective phoneme correction for this category of unexpected mispronunciations.

Out-LJ: Phonemic sequences for 50 words that do *not* occur in LJ Speech, and are mispronounced by the baseline `grapheme-only` model. These represent the key challenge of generalising to words without spoken examples.

Phonemicised: Grapheme sequences for 50 words that occur in LJ Speech, *were* phonemicised, and are pronounced *correctly* by the baseline `grapheme-only` model. This test set checks that representation-mixed training preserves correct output from the grapheme-only case.

Judgements of pronunciation correctness require careful listening because errors are not simply a matter of a categorically-different phone being produced but can be subtle and ambiguous. Therefore, we use expert listeners.

We obfuscated the above stimuli and presented them in random order to two expert listeners who independently judged whether each test word was pronounced correctly. The listeners had available to them the intended pronunciation for each stimulus. In cases of disagreement, they discussed and re-listened to reach an agreement.

6. Results

Figure 1 visualises the experimental results.

6.1. Overall Observations

First and foremost we observe that 100% correct pronunciation control is not yet possible with representation-mixing models as described in [12]. Even the topline lexicon `full-13049` achieves a far from perfect score of 38/50 over **Out-LJ**.

The number of *word tokens* phonemicised during training is not the sole factor that determines performance, especially for the most important test set: **Out-LJ**. In other words, the `freq-n` lexica, which cover the most tokens for any given n (Table 1), do **not** lead to the best pronunciation correction. Rather, the choice of *word types* in the lexicon matters more.

Figure 1 also shows a trend across all lexica that increasing size (n) results in better performance over both test sets, which is likely to be simply a result of a greater number and variety of phonemicised word tokens mixed in during training. The results for `oracle-14` support this hypothesis: `oracle-14` phonemicises only 41 tokens during training, and is unable to perform pronunciation control to *any* degree, with scores of 0 on both **In-LJ** and **Out-LJ** (not shown in Figure 1) with even the 14 word types seen during training pronounced incorrectly. It is not sufficient to see each phoneme just a few times in limited contexts to achieve control over pronunciation.

There are a few exceptions to this trend. The performance of `rand-n`, `freq-n`, and `phone-n` can *fall* as n grows: performance from `rand-2000` to `rand-4000`, `freq-4000` to `freq-6000`, and `phone-4000` to `phone-6000`, demonstrate this. Naively using more resources to increase the size of the lexicon doesn't necessarily result in better pronunciation performance. `bigram-n` and `trigram-n` do not exhibit this behaviour, which suggests that the lexicon should contain word types that cover a wide range of grapheme-phoneme contexts. Both `bigram-n` and `trigram-n` – which are both possible in a real application – do this nearly as well as `phone-n` – which, as noted earlier, is an oracle setting.

As expected, performance on **In-LJ** (words that occur in the LJ Speech training audio) is better than on **Out-LJ**.

6.2. Comparison of Lexicon Selection Methods

`rand-n` occasionally outperforms other methods despite covering far fewer tokens in LJ Speech (Table 1): compare `rand-500` with `phone-500`, or compare `rand-6000` with all other $n = 6000$ models. Generally however, `rand-n` performs worse than the other methods when $n \leq 4000$, this along with the fact that `rand-n` isn't a principled or predictable word type selection method means we cannot recommend it over the other methods, especially when resources are scarce.

`phone-n` performs the best for n of either 2000 or 4000, suggesting that a lexicon with balanced phoneme coverage is important. `phone-6000` has surprisingly poor performance and we do not have an explanation for this anomalous result. `phone-500` also has surprisingly poor performance. A potential reason could be that Algorithm 1 takes only 41 words to cover all phoneme unigrams once, 313 words to cover all bigrams once, and 2196 words to cover all trigrams once. Thus, choosing wordtypes by bigram or trigram coverage will tend to select word types with a greater variety of spellings while caring less about token frequency than when covering phoneme unigrams. Greater spelling variety will correspond to greater phonemic context variety, which may be better for **Out-LJ** performance. Since `phone-n` is an oracle setting in which an existing large lexicon is required, it is only intended to provide points of comparison with the other methods.

`freq-n`, **`bigram-n`** and **`trigram-n`** are, in contrast, all feasible in a real limited-resource use case. The simplest method `freq-n` offers poor performance for **Out-LJ** for all n. `trigram-n` almost always outperforms `bigram-n`. The clear conclusion is that a greater variety of grapheme sequences (i.e., spellings) in the lexicon leads to better pronunciation correction performance, as spelling variety will correlate with phoneme context variety.

In summary, considering all of the above results, we recommend the use of the `trigram-n` lexicon selection method when resources are limited. It offers better pronunciation control than any other method for a lexicon with 500 entries (`trigram-500`), falling not far short of the `full-13049` lexicon, despite being less than 4% of the size (i.e., 25 times cheaper to create).

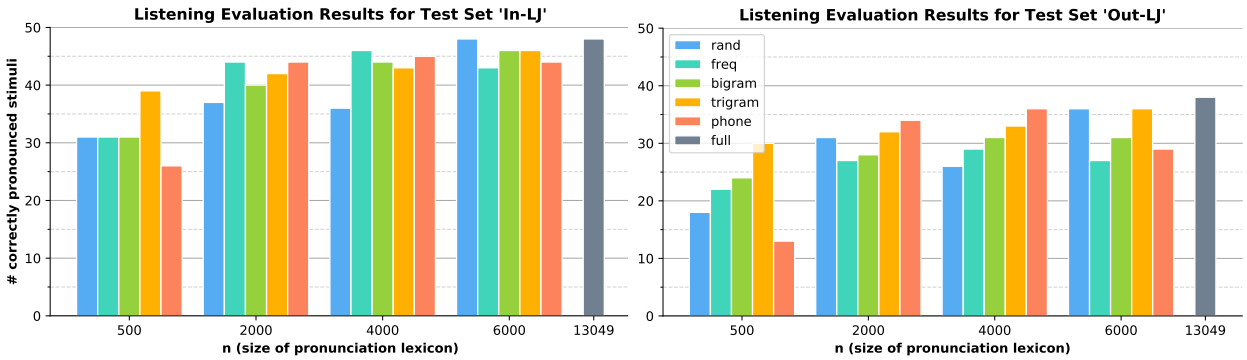

Figure 1: *Listening test results of models trained using resource-limited lexica generated by the word type selection methods.*

6.3. Additional Experiments

The results above indicate that `trigram-n` is the recommended method for creating a lexicon from scratch. In additional experiments, we used the supplemental models listed in Section 4 to explore two further questions.

6.3.1. *Probability of phonemicisation during training*

Results for `mixprob-up` (**In-LJ**:47/50, **Out-LJ**:42/50) demonstrate that phonemicising lower frequency words with a higher p_{mix} during training slightly improves pronunciation control for **Out-LJ** words, compared to the uniform $p_{mix} = 0.5$ across all word types used in the `full-13049` results above. We presume this is because low-frequency words will tend to have more variety in their spelling and/or phoneme sequences, which we know to be beneficial from the results for `trigram-n` and `phone-n` respectively. Results for `mixprob-down` (**In-LJ**:48/50, **Out-LJ**:39/50) are very similar to `full-13049`: phonemicising higher frequency words more often has no benefit.

6.3.2. *Use of syllable information*

`syllable` is the best-performing across all results (**In-LJ**:47/50, **Out-LJ**:48/50), correctly pronouncing 10 more OOV words than the topline reference `full-13049`. To understand the types of words that `syllable` was able to pronounce correctly *because* it exploits syllable boundary information, we analysed the 7 words (input as phoneme sequences) that `syllable` pronounces correctly with syllable information, `full-13049` pronounces incorrectly, and `syllable` pronounces incorrectly when syllable information is absent. These words are almost all morphological compounds containing a stop (e.g., /t/) and then /h/ occurring across the syllable boundary: goatherd /g ou t | h @ @r r d/, loophole /l uu p | h ou lw/, upheld /uh p | h e lw d/, coathanger /k ou t | h a ng | @r r/, plothole /p l aa t | h ou lw/, plughole /p l uh g | h ou lw/, funghi /f uh ng | g ii/. When these words are pronounced incorrectly by `full-13049`, the error occurs over the syllable boundary. For example /l uu p h ou lw/ is mispronounced as /l uu f ou lw/.

Further inspection of the `full-13049` lexicon reveals a potential reason why words containing a stop followed by /h/ are mispronounced without the use of syllable boundaries. The phoneme /h/ occurs mid-word in just 101 word types and only occurs after a stop in 10 of those. These word types have very low token frequencies in LJ Speech, ranging from 1 to 4, which add up to insufficient examples for the model to learn how to correctly pronounce this voiceless glottal fricative mid-word after a stop. Syllable boundary information resolves these cases presumably because they help the model learn to generalise from /h/'s following non-stops to /h/'s following stops.

`stress` did not lead to improved performance. It is worse (**In-LJ**:31/50, **Out-LJ**:33/50) than `full-13049`. `stress-syllable` (**In-LJ**:46/50, **Out-LJ**:45/50) is also inferior to `syllable`. In contrast to syllable information, lexical stress information is not beneficial to pronunciation control.

Most models correctly pronounced 47-50 of the words in the **Phonemicised** test set (Section 5). That is, representation-mixing doesn't negatively impact pronunciations that a `grapheme-only` model already pronounces correctly.

Another potential application of representation-mixing models is to control pronunciations *within* the grapheme representation of a word, making pronunciation correction more accessible to non-expert users. For example, correcting the mispronunciation at the morpheme boundary in 'loophole' by inputting 'loo/p | h/ole', or controlling the pronunciation of the first vowel in the homograph 'gala' by inputting 'g/aa/la' (UK) or 'g/ei/la' (US). We generated various such samples from `syllable`, and informally confirmed that this type of correction is indeed possible despite graphemes and phones never being mixed *within* words during training.

7. Conclusions

This paper shows that representation-mixing models cannot with 100% accuracy correct pronunciations by using phoneme inputs, even when the majority of word types are phonemicised during training by a large pronunciation lexicon. Subsequently future work should look to improve on this result. Despite this, they still control pronunciations relatively well, and we demonstrate via an exploration of principled word type selection methods that performance remains competitive even when using resource-limited lexica as small as just 500 entries. Thus this makes representation-mixing a cost-effective paradigm for correcting mispronunciations made by E2E TTS systems, which is of the upmost importance in low-resource scenarios.

Our experiments show that when developing a lexicon for a representation-mixing TTS system, choosing to phonemically transcribe word types based solely on the number of tokens they cover in the speech corpus is not optimal for pronunciation correction. Instead the *choice* of word types is also important. Our results indicate that choosing words with rich graphemic context helps greatly, possibly as rich graphemic context is a proxy for rich phonemic context.

8. References

[1] R. Sproat *et al.*, "Normalization of non-standard words," *Comput. Speech Lang.*, vol. 15, no. 3, p. 287–333, Jul. 2001.

[2] R. Clark, K. Richmond, and S. King, "Multisyn: Open-domain unit selection for the Festival speech synthesis system," *Speech Communication*, vol. 49, no. 4, pp. 317–330, 2007.

[3] M. Schröder and J. Trouvain, "The German text-to-speech synthesis system MARY: A tool for research, development and teaching," *International Journal of Speech Technology*, vol. 6, no. 4, pp. 365–377, 2003.

[4] P. Ebden and S. R, "The Kestrel TTS text normalization system," *Natural Language Engineering*, vol. 21, no. 3, p. 333–353, 2015.

[5] S. Fitt, "Unisyn lexicon," 2020. [Online]. Available: http://www.cstr.ed.ac.uk/projects/unisyn/

[6] CMU, "The Carnegie Mellon pronouncing dictionary," 2020. [Online]. Available: https://github.com/cmusphinx/cmudict

[7] S. Fitt and K. Richmond, "Redundancy and productivity in the speech technology lexicon - can we do better?" in *Proc. Interspeech*, 2006, pp. 1202–1204.

[8] J. Shen *et al.*, "Natural TTS synthesis by conditioning WaveNet on Mel spectrogram predictions," in *ICASSP*, 2018, pp. 4779–4783.

[9] Y. Wang *et al.*, "Tacotron: Towards end-to-end speech synthesis," in *Proc. Interspeech*, 2017, pp. 4006–4010.

[10] T. Hayashi *et al.*, "ESPnet-TTS: Unified, reproducible, and integratable open source end-to-end text-to-speech toolkit," in *ICASSP*, 2020, pp. 7654–7658.

[11] W. Ping, K. Peng, A. Gibiansky, S. O. Arik, A. Kannan, S. Narang, J. Raiman, and J. Miller, "Deep Voice 3: 2000-speaker neural text-to-speech," in *Proc. ICLR*, 2018.

[12] K. Kastner *et al.*, "Representation mixing for TTS synthesis," in *Proc. ICASSP*, 2019, pp. 5906–5910.

[13] K. Ito, "The LJ speech dataset." [Online]. Available: https://keithito.com/LJ-Speech-Dataset/

[14] Fatchord, "Tacotron and WaveRNN implementation," 2019. [Online]. Available: https://github.com/fatchord/WaveRNN

[15] N. Kalchbrenner *et al.*, "Efficient neural audio synthesis," in *Proc. ICML*, vol. 80, 2018, pp. 2410–2419.