Zero-Shot Cross-lingual Phoneme Recognition from Yoruba to English

Aaron Bahr, Nikita L. Beklemishev, Haejin Cho, Kai Seidenspinner and **Ilinca Vandici**
Universität Tübingen

Abstract

In this paper, we pre-train an acoustic phoneme recognition model on the TIMIT dataset and evaluate its performance on the Yoruba portion of the Common voice data. We make use of the CTC architecture, allowing us to forego the need for time-aligned input data, and examine the models performance after transfer through a thorough and linguistically motivated feature weighting metric. Finally, we look at the phone embeddings produced by our model and try to disentangle training errors from transfer errors. All codes are available in our Github.

1 Introduction

Efficiently training an ASR system requires a rich, ideally time-aligned dataset. For low-resource languages, despite efforts towards documentation, exploiting the properties of transfer learning by pretraining on another, high resource, language remains a sensible option. In our case, we focus solely on producing a consistent, generalized phonemic transcription, conditioned only on acoustic segment (no LM model is included). For the purpose of zero-shot evaluation, picking a set of languages with similar phoneme inventories remains the practice yielding the best performance. We chose to work on transferring American English (< West Germanic < Indo-European) to Yoruba (< Volta-Niger < Atlantic-Congo), primarily spoken in Nigeria, whose phoneme inventories overlap in great part, despite different area and *phylum*. Along with obtaining good performance on both languages, we aim to propose an efficient evaluation metric by employing a linguistically sound feature weighted version of PER. Additionally, we also had to tackle the challenge of using non-timed aligned data. While the dataset we used for pretraining offered time-aligned transcriptions, this was not the case for the Yoruba dataset. We opted for an architecture integrating the *Connectionist Temporal Classification Loss*, discarding the time-aligned annotations to make sure we could evaluate on non-time aligned datasets in the future.

1.1 English Yoruba Transfer

Theoretical Capabilities Transfer relies on the knowledge common to language. ... Similarities in inventory, differences in distribution. ... => Learning without an LM CTC. Representing English distinctions instead of Yoruba contrasts. [no work without a kiparsky quote] ...

Empirical Realisation The model is trained to recognize phoneme labels in English. How to relate it to phoneme labels in Yoruba?... . to discussion: theoretical value, other models built on top for practical value. ..

2 Dataset

Training set For training we use the infamous TIMIT ASR corups. (**?**) includes 6300 utterances recorded by 630 speakers from 8 major English dialects across the US. Annotations were done according to the customized IPA convention based on ARPAbet (**?**) (please refer to Appendix **??**). As both English and Yoruba are pluricentric languages, the learning benefits a lot from the variety in the input data. Even though TIMIT confines itself to the 1980s US language roof, it represents the existing dialectal variation well. This phonetic variation is necessary for learning to generalize over acoustically different speech sound, which is particularly relevant for transfer learning. Another reason for TIMIT is the popularity of it in ASR studies, which gives us confidence in yielding baseline results, comparable with other works in the area.

Preprocessing the training set For usage, we first concatenated the train, validation, and test sets and then again randomly split it into train and validation sets with the proportion being 0.25. Thus TIMIT is used only for training and validation in our cross-lingual task. Regarding the audio data, we have extracted log-mel features (dimension 39). The TIMIT alphabet contained 63 unique labels in total. To reduce prediction complexity and to make the phonemic representation of English compatible with that of Yoruba, before mapping to IPA, we merged or split several labels. This included allophones not annotated in the Yoruba corpus: < ax-h > /ə̥/ and /ə/, syllabic sonorants, e.g. < eng > /ŋ̍/ and /ŋ/. Closures < dcl > /d̚ / and the following releases /d/ were joined into one label. In the end, 15 label types were merged. We did not expect systematic unreleased closures in **open-syllable** Yoruba. (**?**) As for the splitting the combinatorially large inventory of English diphthongs, we split them into vowel–glide sequences, keeping the vowels from the IPA convention: thus < oy > /ɔɪ/ became < ao y > /ɔ j/. This step was necessary since Yoruba does not have diphthong vowels. **?** We also concluded that splitting vowels will not perplex the prediction given that the CTC decoding does not need time alignment, and our evaluation ignores word boundaries.

Evaluation set Common Voice Yoruba data was used as a test dataset. Common Voice is a multilingual crowd-sourced corpus aimed for Speech Recognition purposes. (**?**) The audios were recorded by certified native speakers of each language. Annotations are suggested and later validated by other native speaker users via votes.There are 3.4k samples in total and each sample includes an MP3 file, speaker ID and audio transcription written in Yoruba orthography. The dataset also includes data from different dialects. This allows to acknowledge the performance gaps that are shown to arise between standard Yoruba and other dialects in NLP tasks. **?**

Processing the evaluation set The sets of train (1.4k), validation (913) and test (1.1k) were again concatenated and used together for testing. We kept the original threshold of down votes for invalidated samples. Since we need to keep the shape and content of the train and test data identical, 39-dimensional log-mel feature was extracted from Yoruba in the same way. We implemented additional grapheme-to-phoneme conversion, since the model predicts IPA symbols given audio inputs, which means that data should at least have phoneme/phonetic representations as a label. We converted Yoruba sentences line-by-line via `Epitran` Python module. First of all, the word boundaries, as well as the pauses were removed. We also ignored the tone annotation (˧, ˥, ˨), although seeing the transfer abilities could be an interesting branch of research. We have removed the marginally phonemic /ɔ̃, ŋ/ from the inventory, merging with their allophones /ã/ and /n/. (**?**) Another marginal /ɛ̃/ remained. The data also turned out to contain an occurrence of dialectal < ụ > /ʊ/ **?**, which is too small to generalize from, so we removed it as well from the evaluation for clarity. Full and adjusted Yoruba IPA inventory in Appendix **??**.

3 Model and Traning

3.1 ResNet-Bi-LSTM model

Our model is based on that of (**?**), which used ResNet-BiLSTM model for Nepali Speech Recognition. The reason why we chose this model as our base reference was (1) to use pure neural-network-based model so that it is rather easy to train and light-weight in terms of memory, (2) to include residual connection which is largely used in transformers as well as deep neural network models, and (3) to explore whether a model that is trained from scratch can also perform well on zero-shot cross-lingual speech recognition task.

The model starts with 1D-convolution block followed by 1D-Batch Normalization. 5 residual blocks again follows it. This initial CNN layer and ResNet aim to capture locally dependent features and we will call these sets of layers a *ResNet Encoder*. All convolutions are done using identical parameter setting. The kernel size is 15 and the number of channels are 50. Stride is one and dropout rate is 0.2. For residual network encoder, it has 5 residual blocks. Each block consists of 2 unit blocks and one unit block is comprised of initial convolution, Batch Normalization, and PReLU as an activation. Key part of residual block is to add original input to activation layer output. (**?**) have attested that residual connections noticeably stabilize and optimize deep neural network training. It works consistently well with audio task although it was first introduced for image recognition task as can be seen in (**?**). Another important part of the base model is that it does not include pooling layer. Pooling layers aim to reduce spatial dimension in 2D Convolution setting, so width and height of a feature map would be reduced. In our case, pooling in 1D Convolution would down-sample feature map across time-step and it seems like authors of (?) concluded that it is unnecessary to down-sample data across time-step. Bi-LSTM encoder part then follows residual networks. Bi-LSTM is designed to reflect distinctive contextual features in two opposite directions. (?) has two RNN layers with its dimension being both 170. As final layers, two dense layers and ReLU activation takes the output of bi-LSTM and maps 170-dimension input into 66-dimension output data. 66-dimension is set based on the number of labels in Nepali inventory.

We have mostly followed (?)'s model design. However we did make several key changes to prevent overfitting and to keep the depth of our model shallower so that our model can be applicable for cross-lingual inference. Each adjustments is based on our experiments, but unfortunately we did not properly record the difference for each inspections. (1) The first major change is to reduce the depth of ResNet encoder block. Our model has 3 residual blocks while the original model has 5 blocks. Also, one residual block has only one set of unit block while the original residual block has 2 unit blocks. In other words, our ResNet encoder part is 0.3 times depth of that of the base model ResNet encoder. All other settings regarding ResNet encoder have fixed same as (?). (2) We have also diminished hidden dimensions of Bi-LSTM encoder and dense layer. From 170 to 128 as to hidden dimension of RNN layer and from 340 to 256 as to dense layer dimension. This also contributes to better computational efficiency as it corresponds to GPU's natural memory alignment and fetches. (3) Last but not least, we have introduced saliently strong dropout rate for RNN layers. 0.4 was adopted as dropout rate. This was also part of an effort to enable cross-lingual application. Refer to following visualization marks key difference between (?) and our model.

We employed Connectionist Temporal Classification (CTC) loss as our training criterion. Unlike conventional frame-level cross-entropy, CTC does not require pre-aligned input-label pairs, which makes it especially suitable for low-resource languages such as Yoruba, where alignment information is unavailable. The key idea of CTC is to introduce a blank symbol and permit label repetitions so that multiple frame-level alignments can correspond to the same output sequence. During training, CTC computes the negative log-likelihood of the target sequence by summing the probabilities

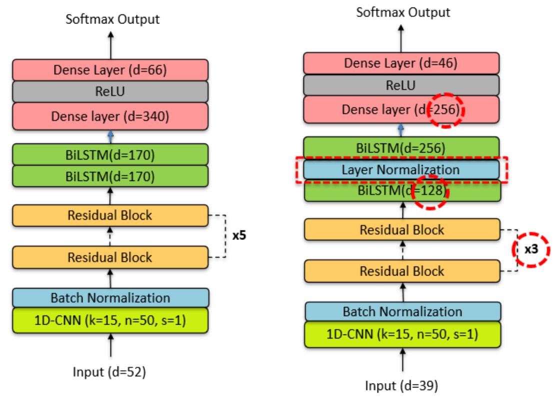

Figure 1: Left: model architecture of (?) , Right: Adjusted model architecture for English-Yoruba cross-lingual phoneme recognition. Changes highlighted with a red circle.

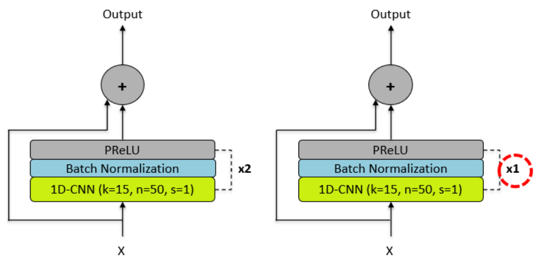

Figure 2: Left: residual block of (?), Right: Adjusted residual block for English-Yoruba cross-lingual phoneme recognition. Changes highlighted with a red circle.

of all valid alignments:

$$L_{CTC} = \sum_{(X,Y)\in D} -\log P_{CTC}(Y \mid X)$$

where the probability of a target sequence Y given the input X is defined as

$$\begin{aligned} P_{CTC}(Y \mid X) &= \sum_{A\in B^{-1}(Y)} P(A \mid X) \\ &= \sum_{A\in B^{-1}(Y)} \prod_{t=1}^{T} p(a_t \mid h_t) \end{aligned}$$

with $A = (a_1, \ldots, a_T)$ being a frame-level alignment, B the collapse function that removes blanks and repeated labels, and h_t the hidden representation at time t. The blank symbol ϵ needs to be manually added to the alphabet so that the model can output blanks during training and inference.

At decoding time, greedy decoding selects the most probable label at each timestep, and the CTC collapse operation removes duplicates and blanks to produce the final prediction sequence $\hat{Y}$.

$$\hat{Y} = \arg\max_{Y} P_{CTC}(Y \mid X).$$

3.2 Train

Train settings were set through multiple experiments. The number of epochs is 50 and batch size was 64. We have adopted Adam as our stochastic gradient optimizer and set weight decay as 1e-4. A plateau-based learning rate scheduler was used and the initial learning rate was 1e-3. The total number of parameters are 0.8 million. With (hyper)parameter settings given above, training was noticeably stable and robust to overfitting. Both training and validation loss and PER consistently decreased and we halted training at the 16th epoch where Train PER reached 0.0223 and Validation PER reached a similar figure, 0.0339.

4 Results

text result - PER, ACC

To evaluate the output of our model, we use Phoneme Error Rate. We use a scheme similar to the Levenshtein algorithm, with deletion, insertion and substitution operations. Special attention is given to the substitution operation, where the cost is determined by a feature-weighting scheme (which we derive from the PanPhon package). To obtain these measures, we convert the ARPA characters to IPA, thereby not directly evaluating on

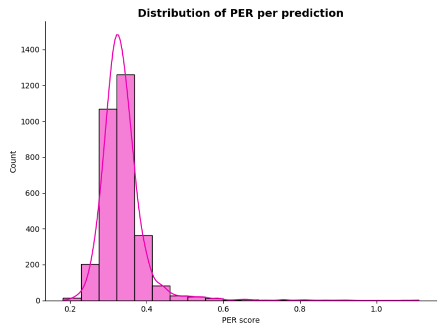

Figure 3: PER Distribution

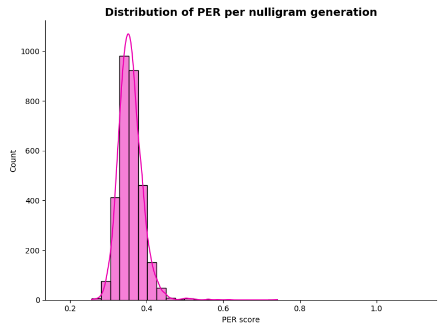

Figure 4: Nulli-gram PER Distribution

raw output. During inference, we observed that our model had a tendency to overgenerate, making the output consistently longer than the gold labels. Lowering the deletion cost from 0.75 to 0.5 helped improve this. However, we still keep this issue in mind for later discussion.

Default PER was 0.34 without any adjustment on deletion, insertion, and substitution operations. PER distribution graph draws positively-skewed bell curve. Together with equi-weighted features and deletion still set as 1.0, PER decreased to 0.35. As mentioned above, adapting deletion cost from 1.0 to 0.5 made the metric even lower to 0.33.

Nulligram PER is further investigated by artifically creating a Yoruba nulligram in which the order of phonemes was randomized. The aim was to examine whether co-occurrence patterns of certain phonemes influence model performance. If the PER of nulli-gram corpus is higher than that of default PER, it suggests that phoneme collocation contributes positively to phoneme recognition. At the very least, it would indicate that the naturally occurring co-occurrence patterns are not so complex as to hinder recognition performance.

Nulligram PER was reported to be 0.36, which is 0.2 higher than default PER. This would in turn suggest that natural sequential distribution of phonemes helps predicting them.

4.1 Inference

4.2 Phone Embeddings Extraction

We later experiment with inference to produce phone embeddings, and attempt to plot these using dimensionality reduction. We collect the final embeddings for each correctly predicted phoneme (whose feature size length equals the vocabulary size) and use UMAP to obtain 2 dimensional embeddings, for which we obtain the mean by phoneme. We settled on UMAP since it maintain the non-linear assumption of t-SNE regarding the data while being less dependent on initialization. Note that we converted the labels obtained by inference to IPA for ease of comparison. While both plots seem to group similar sounds together to an extent (e.g), comparing inference simultaneously on both datasets remains tricky due to 1) the different probability distributions, and, to an extent, the different in dataset sizes 2) while UMAP remains an efficient dimensionality reduction method, it is unclear whether plotting always reflects similarity through clustering. To investigate this further, we decided to run a correlation test between the Euclidian distance between phones (after dimensionality reduction) and feature weight. This shows that there is a significant correlation between the two variables, but it presents a low, negative regression coefficients, showing that phonemes with different features tend to have a relative larger Euclidian value, which goes against what we would initially hope for. Moreover, attempting to map mean Yoruba and English embeddings into the same space and then computing their distance across several trials shows a high degree of variation.

Several implications arise from this. First, we can argue that the UMAP method does not always produce mappable clusters. We took care of fitting on one specific dataset first, before mapping the other onto the obtained space, which should ensure that the dataset size does not disturb the dimensionality reduction process. However, as mentioned before, we only took embeddings for which the alignment was correct and confirmed, meaning that we could have bypassed some relevant examples. Hence, for future studies, it is necessary to look into over-generation patterns to understand how and when these occur.

4.3 Discussions

We obtain posterior probabilities of prediction labels based on gold labels (see Figure 7). At a first glance, we can see predictions close to the gold label tend to have a higher probability. For the majority labels for which the most common prediction is incorrect, it usually still reflects a degree of similarity with the most common label (for instance, i is the most likely prediction for e).

- yoruba phoneme frequency
- Normalized PMI
- confusion entropy

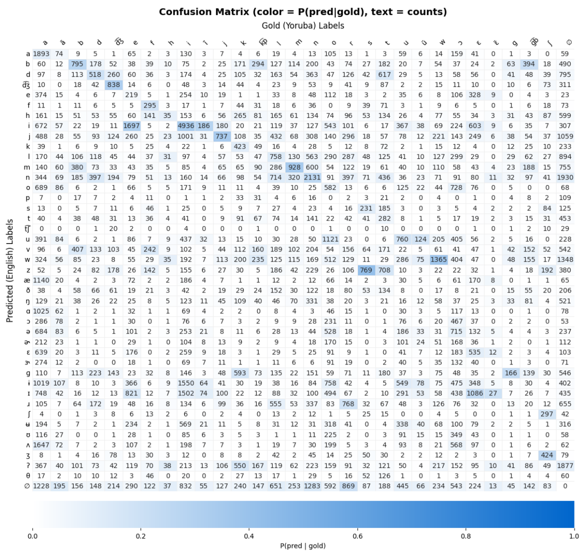

Figure 5: English-Yoruba Confusion Matrix with Posterior Probability

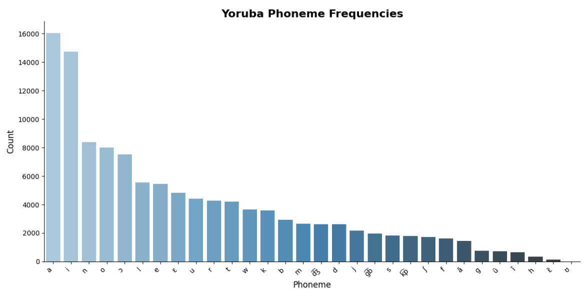

Figure 6: Yoruba Phoneme Frequency

Limitations

For future reference, we believe that our model could be further improved by dedicating more attention to the over-generation problem, and possibly address it at the pre-training stage with a modified pre-training objective, or by investigating whether there are some repeating patterns linked to over-generation. This would then in turn help improve the quality of alignment for the evaluation step. As we did not keep track of the train-test split, the impact that this has on the evaluation step is unclear. Re-producing it with a proper split might

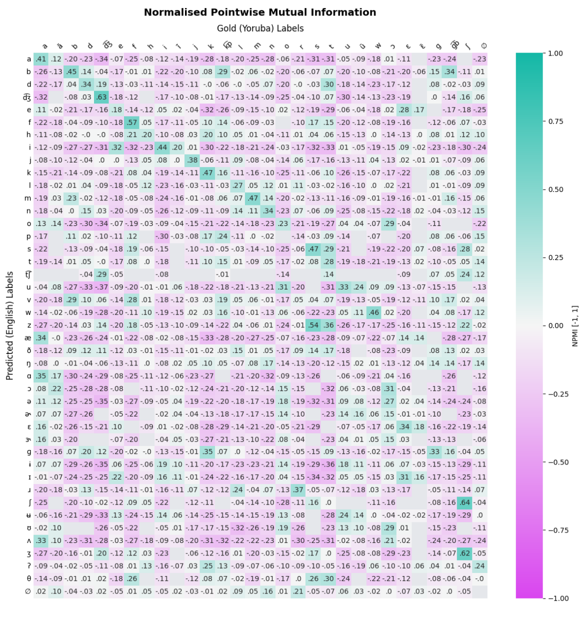

Figure 7: Normalized PMI

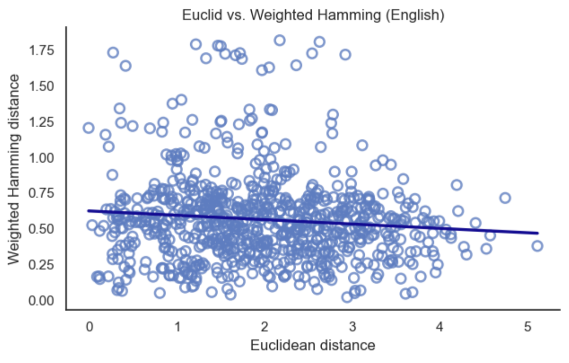

Figure 10: Correlation between Euclidian distance of embeddings and Weighted Hamming distance (English

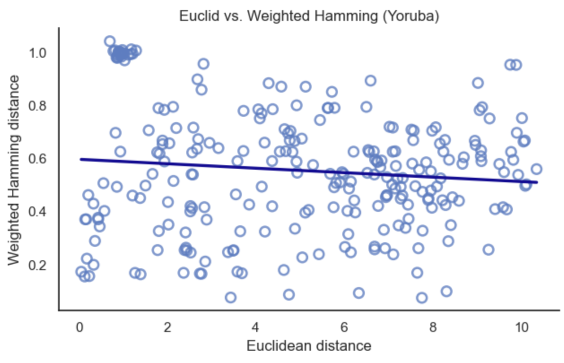

Figure 11: Correlation between Euclidian distance of embeddings and Weighted Hamming distance (Yoruba)

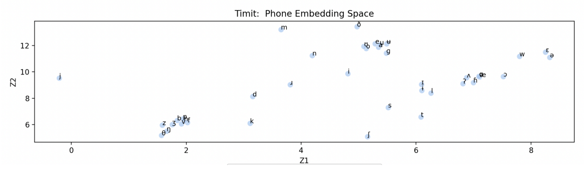

Figure 8: Timit Phone Embeddings

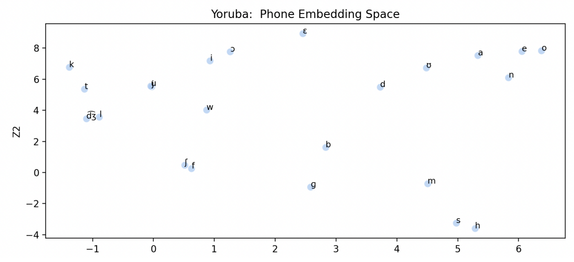

Figure 12: Yoruba Phone Embeddings

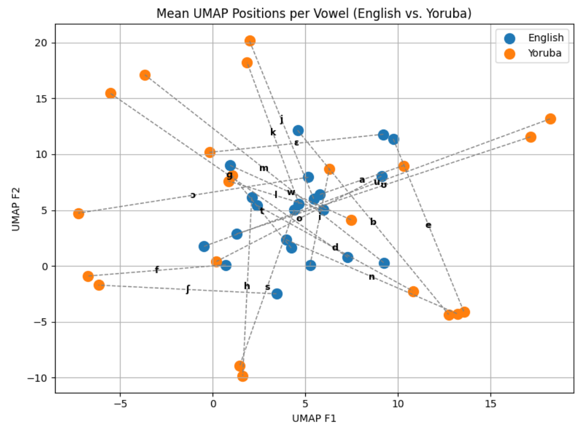

Figure 9: Vowel distances between English and Yoruba

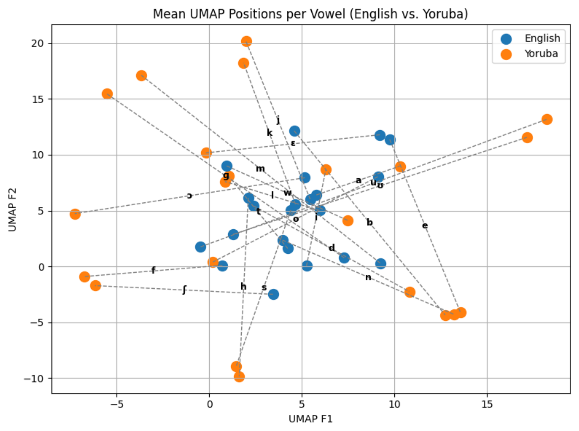

Figure 13: Yoruba and English embeddings plotted in the same space

Table 1: Confusion Entropy

Label	H
∅	4.64
ã	4.55
ɔ	4.53
a	4.52
o	4.41
h	4.33
l	4.26
ũ	4.21
k͡p	4.18
ɡ	4.18
t	4.16
u	4.16
f	4.14
ɡ͡b	4.13
k	4.10
ɛ	4.09
r	4.07
n	4.03
d	4.00
ɛ̃	3.99
ĩ	3.93
j	3.92
e	3.87
w	3.82
i	3.81
ʃ	3.80
b	3.76
d͡ʒ	3.69
m	3.67
s	3.40

yield more comprehensible results, especially when it comes to posterior probabilities. We also encourage the reproduction of this study on different datasets, be it for Yoruba or other languages.

A Appendix

A.1 Appendix. A

A.2 Appendix. B

A.3 Appendix. C

This is a section in the appendix.

Table 2: Pearson correlation between vowel distance and another variable

Language	Correlation Coefficient (r)	p-value
English	-0.1031	0.0044
Yoruba	-0.1067	0.0882

Table 3: Vowel distances between English and Yoruba (sorted by similarity)

Vowel	Distance
	5.193
	5.725
	7.207
	7.546
	8.145
	8.721
	9.537
	9.650
	11.379
	11.421
	11.733
	12.838
	12.973
	14.272
	14.561
	15.947
	15.980
	18.081
	18.406
	19.522
	21.195
	21.895

Table 4: TIMIT to IPA mapping

TIMIT	IPA	IPA (adjusted)
aa	ɑ	ɑ
ae	æ	æ
ah	ʌ	ʌ
ao	ɔ	ɔ
aw	aw	a + w
ay	aj	a + j
ax	ə	ə
axr	ɚ	ə
eh	ɛ	ɛ
er	ɝ	ɝ
ey	ej	e + j
ih	ɪ	ɪ
ix	ɨ	ɨ
iy	i	i
ow	ow	o + w
oy	ɔj	ɔ + j
uh	ʊ	ʊ
uw	u	u
ux	ʉ	ʉ
ax-h	ə̥	ə
bcl	b̚	b
dcl	d̚	d
eng	ŋ̍	ŋ
gcl	ɡ̚	ɡ
hv	ɦ	h
kcl	k̚	k
pcl	p̚	p
tcl	t̚	t
pau	|	–
epi	||	–
h#	/	–
b	b	b
ch	t͡ʃ	t͡ʃ
d	d	d
dh	ð	ð
dx	ɾ	r
el	l̩	l
em	m̩	m
en	n̩	n
f	f	f
g	ɡ	ɡ
hh	h	h
h	h	h
jh	d͡ʒ	d͡ʒ
k	k	k
l	l	l
m	m	m
n	n	n
nx	ɾ̃	n
ng	ŋ	ŋ
p	p	p
q	ʔ	ʔ
r	ɹ	r
s	s	s

Table 5: Yoruba IPA inventory

Yoruba	IPA	IPA (adjusted)
m	m	m
i	i	i
k	k	k
y	j	j
u	u	u
a	a	a
w	w	w
n	n	n
t	t	t
l	l	l
s	s	s
b	b	b
e	e	e
o	o	o
g	ɡ	ɡ
h	h	h
d	d	d
r	ɾ	r
f	f	f
ẹ	ɛ	ɛ
ṣ	ʃ	ʃ
ọ	ɔ	ɔ
j	d͡ʒ	d͡ʒ
´	˦	–
`	˨	–
in	ĩ	ĩ
un	ũ	ũ
gb	ɡ͡b	ɡ͡b
p	k͡p	k͡p
ọn	ɔ̃	ã
ẹn	ɛ̃	ɛ̃
an	ã	ã
–	˧	–
n	ŋ	–
ụ	ʊ	–
ị	ɪ	–