Forced Phonetic Alignment Using Deep Neural Networks

Arlie Coles
Supervisor: Morgan Sonderegger

April 16, 2018

1 Introduction

1.1 Forced Alignment

Phonetic alignment is the task of placing time boundaries between phones in a sequence in an audio recording of speech. Forced phonetic alignment, hereafter *forced alignment*, is a technique for accomplishing this task automatically. In addition to the audio recording, an orthographic transcription of the text in the recording and a pronunciation dictionary, which lists possible word-to-phone mappings for each word, are necessary inputs for this technique. In this light, forced alignment can also be thought of as the alignment of a speech recording and its orthographic transcription at the phone level. An example of the input and output of the task can be seen in Figure 1.

PLEASE SAY 'BUS' AGAIN

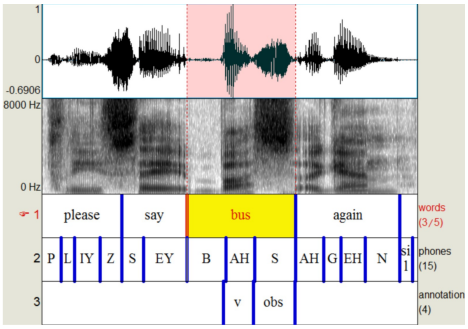

Figure 1: From [25]; left, an example input transcription of an audio recording of speech. Right, the audio recording force-aligned on the phonetic level, as generated by the Montreal Forced Aligner.

Forced alignment is a widely used technique in the language sciences, constituting a critical piece in the classical architectures of automatic speech recognition (ASR) and speech synthesis pipelines [37, 6]. Since its ultimate result identifies the position of phonemes in a recording, it can be directly applied to a number of downstream tasks that require locating or isolating certain speech sounds or spoken words of interest. For example, forced alignment has been used by language researchers and speech scientists to investigate topics such as speech reduction and historical sound change of particular languages [1, 22]. Its other applications outside of ASR and speech synthesis include facilitating simultaneous audiobook reading and listening for language learners and providing time cues for facial animation of speaking characters [2, 10].

Given the numerous applications for forced alignment, any significant improvements in alignment performance is likely to have a wide range of beneficial consequences. To these ends, a number of automatic forced aligners have been developed, notably including FAVE, (Web)MAUS, Prosodylab-Aligner, Gentle, and the Montreal Forced Aligner [35, 21, 13, 28, 26].

The Montreal Forced Aligner (MFA), the Kaldi-based successor to the Prosodylab-Aligner on which this paper is based, distinguished itself from previous aligners in its trainability (it is shipped with pre-trained models to speed up computation, but can also be retrained on new data) and its architecture (which used more elaborate modelling techniques than its predecessors). MFA outperformed Prosodylab-Aligner and FAVE in producing alignments closest to human-annotated gold standards on the Buckeye and Phonsay corpora [26].

1.2 Objective

These results were achieved by MFA's existing Hidden Markov Model-Gaussian Mixture Model (HMM-GMM) architecture, which uses Gaussian Mixture Models (GMMs) to approximate probability distributions over acoustic features. In this work, we seek to augment the current functionality of MFA by replacing the GMM component with a Deep Neural Network (DNN), which, when given HMM-GMM-produced alignments as training data, may be able to learn a yet more accurate model for the test data and thus produce better alignments.

There is reason to conjecture, from the domain of ASR, that the resulting new HMM-DNN architecture might produce better alignments than the older HMM-GMM version. ASR systems such as Kaldi consist of pipelines that produce alignments at an intermediate stage, and can do so via an HMM-GMM method or an HMM-DNN method [19]. Kaldi systems which use even simple DNNs report large decreases in Word Error Rate (WER) at the end of the ASR pipeline over those which use GMMs [33]; it is therefore possible that some of this improvement in performance could be attributed to better alignments generated at the intermediate stage, and desirable to investigate to what degree this is the case.

Specifically, the effectiveness of an HMM-DNN architecture is of interest in several ways: (1) does it produce alignments that are more accurate than the HMM-GMM architecture relative to human-annotated gold standards, on the phone- and word-levels, as well as around silence boundaries? (2) If the accuracy of the HMM-GMM architecture is not surpassed, what configurations of the DNN parameters themselves lead to the best possible performance? And, (3) how might varying the data on which different components of the aligner are trained impact alignment performance?

2 Background & Previous Work

2.1 HMM-GMM Approaches

2.1.1 Standard Architecture

A standard approach to forced alignment is to follow a typical ASR pipeline until a forced alignment is generated, and then terminate. Existing aligners making use of this general framework include FAVE, Prosodylab-Aligner, and MFA [35, 13, 26]. Intuitively, this approach frames the problem in a way that

allows us to determine probabilities of chunks of speech, given that we have heard a certain sound (such as /a/). The two typical broad stages of this approach are:

i. A *feature-processing* stage, where the acoustic signal is transformed into a set of spectral features per window of time. These features represent the "chunk of speech" whose probability we want to determine, and are manipulated as multi-dimensional vectors throughout the rest of the pipeline.

ii. An *acoustic-modelling* stage, where we compute the likelihood of seeing these multi-dimensional vectors given a *senone* (a context-dependent phone-like unit, such as a monophone, triphone, or subphone) [18]. The senone represents the underlying "sound" that we have heard. To capture the sequential nature of the data, then, we model our speech as a Hidden Markov Model, with the sequence of spectral features as the observations and the sequence of senones as the hidden states. We then choose the underlying sequence of senones with beginning and ending points that, when given, results in the highest probability of seeing the sequence of spectral features, and label each window of time with its respective senone.

In the classical HMM-GMM model, GMMs are used in the acoustic-modelling stage to compute the likelihood, given a senone, of seeing the spectral features generated by the feature-processing stage. GMMs are particularly appropriate for acoustic modelling, since their mixture of multiple Gaussian components lends itself to model development accommodating different realizations of phones (such as breathy versions, reduced versions, etc.). Often some form of speaker adaptation is added, typically by transforming the acoustic feature space, in order to make the system robust to speaker variation. Finally, once these likelihoods have been computed, an *alignment* is generated by computing the overall most likely beginning and ending points for each senone in the sequence, given the sequence of spectral features. A schematic diagram of the HMM-GMM probabilistic model underlying this approach can be found in Figure 2.

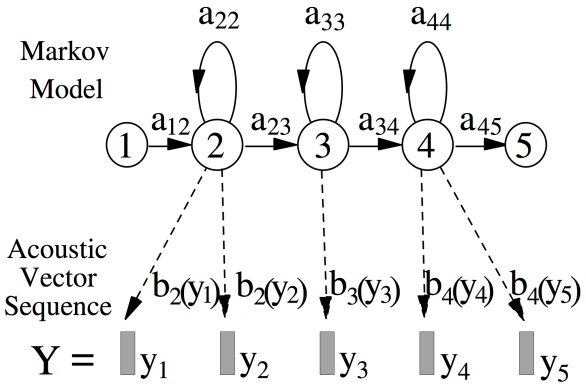

Figure 2: From [12]; a schematic diagram of an HMM-GMM model. In this case, the y_i vectors represent spectral feature vectors taken from a window of the speech recording. Each Hidden Markov Model state j (in circles) represents a subpart of a triphone (e.g. "/a/ between /t/ and /p/"), and the $b_i(y_j)$'s represent GMMs modeling the probability distribution over the observed spectral feature vectors.

2.1.2 The Montreal Forced Aligner

The Montreal Forced Aligner is a standalone utility that performs forced alignment given an audio recording, an orthographic transcription, and a pronunciation dictionary [26]. It is built using binaries from the Kaldi speech recognition toolkit and makes use of the classic HMM-GMM architecture outlined above, using several passes to generate a more complex acoustic model on each pass, with the previous pass' results as a starting point for the next [34]. A schematic diagram of MFA's HMM-GMM architecture can be seen in Figure 3.

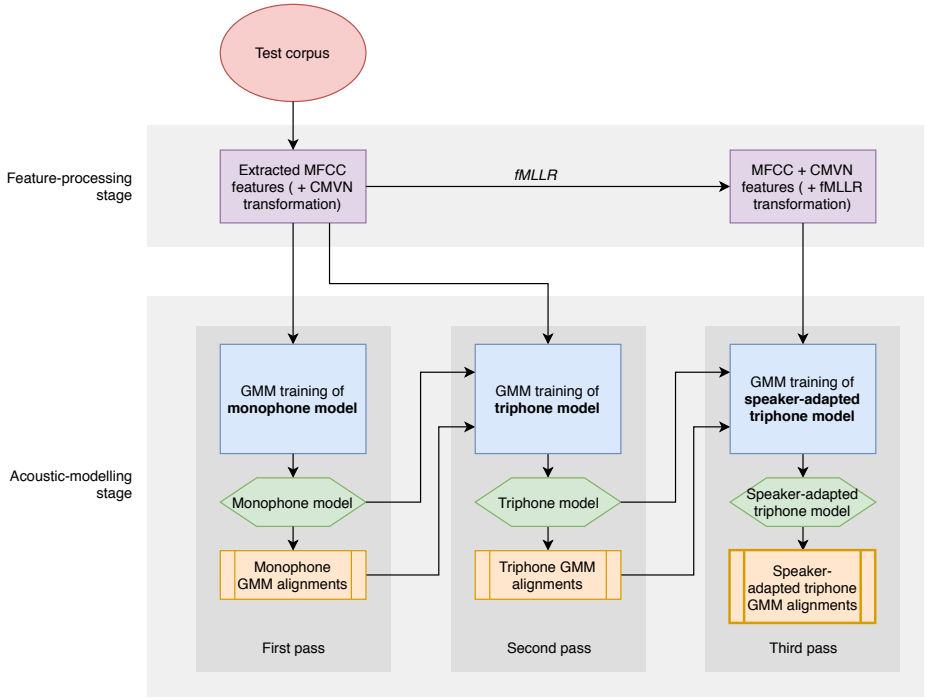

Figure 3: A schematic diagram of the HMM-GMM architecture of the Montreal Forced Aligner. The aligner passes through monophone, triphone, and speaker-adapted triphone GMM training stages before generating a final alignment. Red circles represent corpora, purple boxes represent feature vectors, blue boxes represent training periods, green hexagons represent acoustic models, and orange boxes represent alignments.

In the feature-processing stage, Mel-frequency cepstral coefficients (MFCCs) are calculated for each window in the audio recording, including delta and delta-delta features (first and second derivatives from the original MFCCs, respectively) from surrounding frames. Cepstral mean and variance normalization (CMVN) is then applied to the features to make them more resistant to audio variability between recordings.

The acoustic-modelling stage's several passes initialize Gaussian Mixture Models (GMMs) and fit them to MFCC features using Expectation Maximization [27]. Each GMM models the probability distribution over the MFCCs given a particular hidden state of the HMM. The first pass concerns itself only with *monophones*, and not with any surrounding phones, by building and training HMMs for each phone (treating its beginning, middle, and end as separate states) independently of its context. It thus builds a *monophone acoustic model* for the data and then produces a first alignment.

The second pass takes into account the surrounding phones on either side of the phone in question,

these three constituting a *triphone*, when building its HMMs. To account for the fact that not all possible triphones will appear in the dataset, the aligner constructs a decision tree to group the triphones into a smaller amount of acoustic units to avoid sparsity issues. Each node in the decision tree thus constitutes a "tied HMM state", over each of which a GMM is trained. The resulting *triphone acoustic model* is used to produce a second alignment. This is useful, since phonemes can be realized in different ways according to their context and this allows the aligner to account for these variations.

The third and lass pass also uses triphones, but on newly-computed features, transformed using Feature-Space Maximum Likelihood Linear Regression (fMLLR) [32]. This normalizes for speaker, in efforts to produce a speaker-independent acoustic model, by applying a transform to the features to bring them in line with some adaptation data extracted from the speaker's recording [23]. This *speaker-adapted triphone acoustic model* is then used to produce the final alignment. Hereafter, the term *GMM model* will be used to refer to this final speaker-adapted triphone acoustic model, and *GMM alignment* will be used to refer to the alignment it produces.

2.2 HMM-DNN Approaches

2.2.1 Motivation

GMMs are highly powerful in that enough of them can model almost any sort of distribution function. However, this also means that they can be inefficient for modelling certain distributions. Hinton et al. cite the example that a sphere has few parameters, and therefore a sphere model would efficiently capture data points that lie on or near a sphere, but modeling the surface of a sphere with GMMs would require an extremely large number of them. Since speech is produced by an apparatus with a relatively low number of parameters, its underlying structure might be simpler than the many-GMMs models might suggest, and might be better approximated with a different model [17].

Moreover, while the classic HMM-GMM architecture is the accepted state of the art for the task of forced alignment, the field of ASR, with which the task shares methodology, has largely moved on to DNNs over the past decade [15, 14, 24]. As alluded to in Section 1, ASR systems using DNNs for their acoustic modelling component also achieve better performance than those using GMMs.

Hence, we are motivated to examine deep neural networks, which may be much better at learning data that lie on a non-linear "surface", as an alternative forced alignment architecture. We train a DNN using the previously output GMM alignments as a new "gold standard", in hopes that the DNN will learn a better acoustic model (despite being trained to make the same predictions) since the nature of a DNN allows more complex dependencies to be learned. Using this new, hopefully improved *DNN acoustic model*, better alignments may be able to be generated for new data.

Essentially, this HMM-DNN approach replaces the GMM component of the classical architecture with a DNN, so that the probability of acoustic features given a senone is computed via the DNN instead of by GMMs. A schematic diagram of this setup can be found in Figure 4, and contrasted with the classic HMM-GMM setup shown in Figure 2.

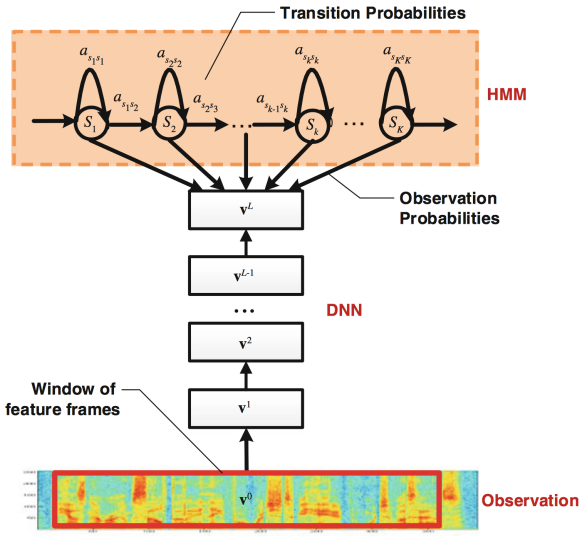

Figure 4: From [38]; a schematic diagram of a generic HMM-DNN architecture. Here, acoustic features are input to the net, and probabilities of particular senones given those features are output. Then, these probabilities are normalized to give probabilities of acoustic features given a senone, which are used later to generate an alignment.

2.2.2 Kaldi's `nnet2`

We model our neural network training architecture from the existing recipes, collected under `nnet2`, included in the Kaldi ASR toolkit [34]. This framework requires training an HMM-GMM system as described above, and then using the produced alignments as new "gold standards" to compare against when training a neural network [33]. `nnet2` as distributed is highly configurable, so a schematic diagram of our chosen architecture and implementation decisions can be found in Figure 5 to accompany the following explanation.

First, an HMM-GMM system is trained on the data to be aligned as described above, and the generated GMM alignments are saved. Next, a transform is applied to the MFCC features which de-correlates them, since neural networks have trouble learning from correlated input [16]. This transform can be likened to a Linear Discriminant Analysis (LDA) transform that does not reduce dimensionality, but reduces the variance of the dimensions of the output feature for which between-class variance is low. The matrix of this transformation is also saved. Then, a subset of the data (including the corresponding MFCCs and GMM alignments) is separated out, to be used as training data for the neural network. With all these components, the neural network is then ready to be initialized with a single hidden layer.

The neural network takes as input the MFCC features and the matrix from the LDA-like transform and uses a p-norm activation function, a dimension-reducing nonlinearity, with parameter $p = 2$ as described in [39]:

$$y = ||x||_p = (\sum_i |x_i|^p)^{1/p}$$

This function takes x to be a small group of inputs and thus achieves a kind of pooling, where each p-norm output is the result of pool over several hidden layer inputs. This helps to reduce computational load (as the

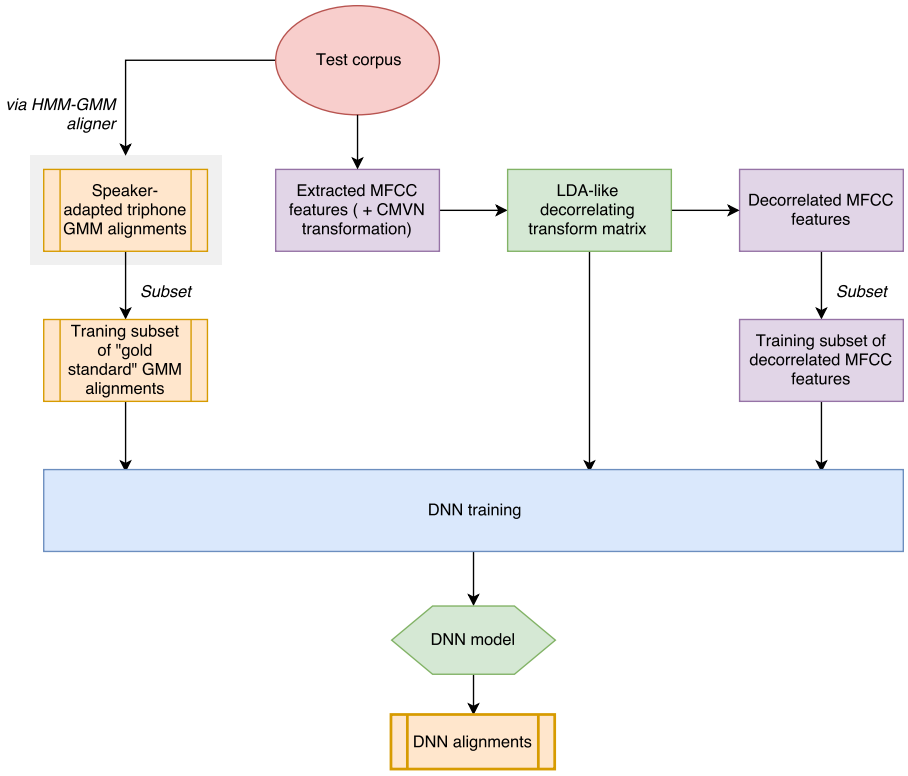

Figure 5: A schematic diagram of the proposed HMM-DNN architecture for the Montreal Forced Aligner. Red circles represent corpora, purple boxes represent feature vectors, green boxes represent transformation matrices, blue boxes represent training periods, green hexagons represent acoustic models, and orange boxes represent alignments.

number of parameters and weights to manage is reduced) and to control overfitting. Weights and biases are also treated with online preconditioning, a method that modifies the conventional stochastic gradient descent to use a matrix-valued learning rate instead of a scalar-valued one. This mitigates parameters changing too quickly in any one direction by reducing the learning rate in dimensions where the derivatives have a high variance.

The output component of the network is a softmax nonlinearity, which produces the probability of being in a given HMM state (a senone) given the MFCC input features. These probabilities are normalized to yield the final normalized probabilities of the input features given being in a senone.

In the main training loop, we train for 30 iterations, and add hidden layers regularly until the network has 3 hidden layers. The learning rate begins at 0.32 and steadily decreases to 0.032, remaining there for the final 5 iterations. About halfway through training, we "mix up": by analogy with the "mixture" inherent to GMMs, whose multiple Gaussian components allow the acoustic model to develop a probability distribution that can account for different realizations of the same phone, we copy components of the weight matrix and allow the copies to independently develop probability representations of these different realizations. This network "mixes up" to 12,000 components. The transition matrix for the HMM is trained on the "gold-standard" training data (the selected GMM alignments) and the prior probabilities of each HMM state are computed and fed back into the model. This main training loop, working via stochastic gradient descent,

trains and updates the parameters of the network through back-propagation. After the training loop has completed its 30 iterations, an alignment is generated.

To reiterate, we use alignments from the HMM-GMM architecture to train the DNN, since these alignments are already state-of-the-art. The potential for improvement upon this alignment for future data being input into the DNN lies in the fact that DNNs might be able to learn complex dependencies and parameters that the HMM-GMM model cannot, and able to do so more efficiently than an HMM-GMM model, resulting in a better acoustic model that may result in a more accurate alignment for new data.

2.2.3 I-Vectors as Speaker Adaptation

The HMM-GMM model uses fMLLR to account for high variability in speaker when generating alignments (which transforms the feature space in order to normalize for speaker). The default HMM-DNN architecture does not perform fMLLR, notably since DNNs prefer learning from decorrelated feature input [16]. Hence, we are motivated to include speaker adaptation in the HMM-DNN architecture in another way, namely by computing a low-dimensional speaker representation in feature space, appending this representation onto the acoustic feature inputs, and feeding these inputs into the DNN, letting it learn the adaptation as needed. To this end, we make use of *i-vectors* as suggested in the `nnet2` recipes.

I-vectors ("speaker identity vectors") were first introduced by Dehak et al. as part of a way to condense sequences of feature frames down into a low-dimensional vector [8]. Ultimately, this can model the variability of the training data that is due to speaker differences, since each i-vector represents a separate speaker, whose characteristics have been condensed into a low-dimensional representation. In this way, i-vectors can be thought of as "speaker embeddings", somewhat analogous to word embeddings common to neural network literature, and they have become state-of-the-art in the speaker recognition field [9]. We integrate them into our neural network in the hopes of achieving a more robust speaker adaptation. A schematic diagram of the i-vector mechanism as it relates to this neural network can be found in Figure 6 to accompany the following explanation.

First, LDA and a Maximum Likelihood Linear Transform (MLLT) are applied to the features of a separate corpus. Then, a diagonal Universal Background Model (UBM) is generated from several GMMs that are fit to the means, covariances, and weights of the separate corpus' features. An "i-vector extractor" is then trained from the corpus data, which more precisely is a parameter matrix T, following from this equation:

$$s = m + Tw$$

where s represents the speaker-dependent GMM supervector, m represents the UBM-GMM (speaker-independent) mean supervector, T is the i-vector extractor, and w gives the i-vector after taking its MAP point estimate. This equation represents the crux of the i-vector method: we assume a linear dependence between the speaker-dependent vector s and the speaker-adapted vector w. The parameter matrix T is tuned by Expectation Maximization, and then stored away as an i-vector extractor for future use [7, 36]. When it is time to perform training and testing with our DNN, we extract i-vectors from our data by using the inverse of the pretrained extractor T, and concatenate them to the input MFCC features.

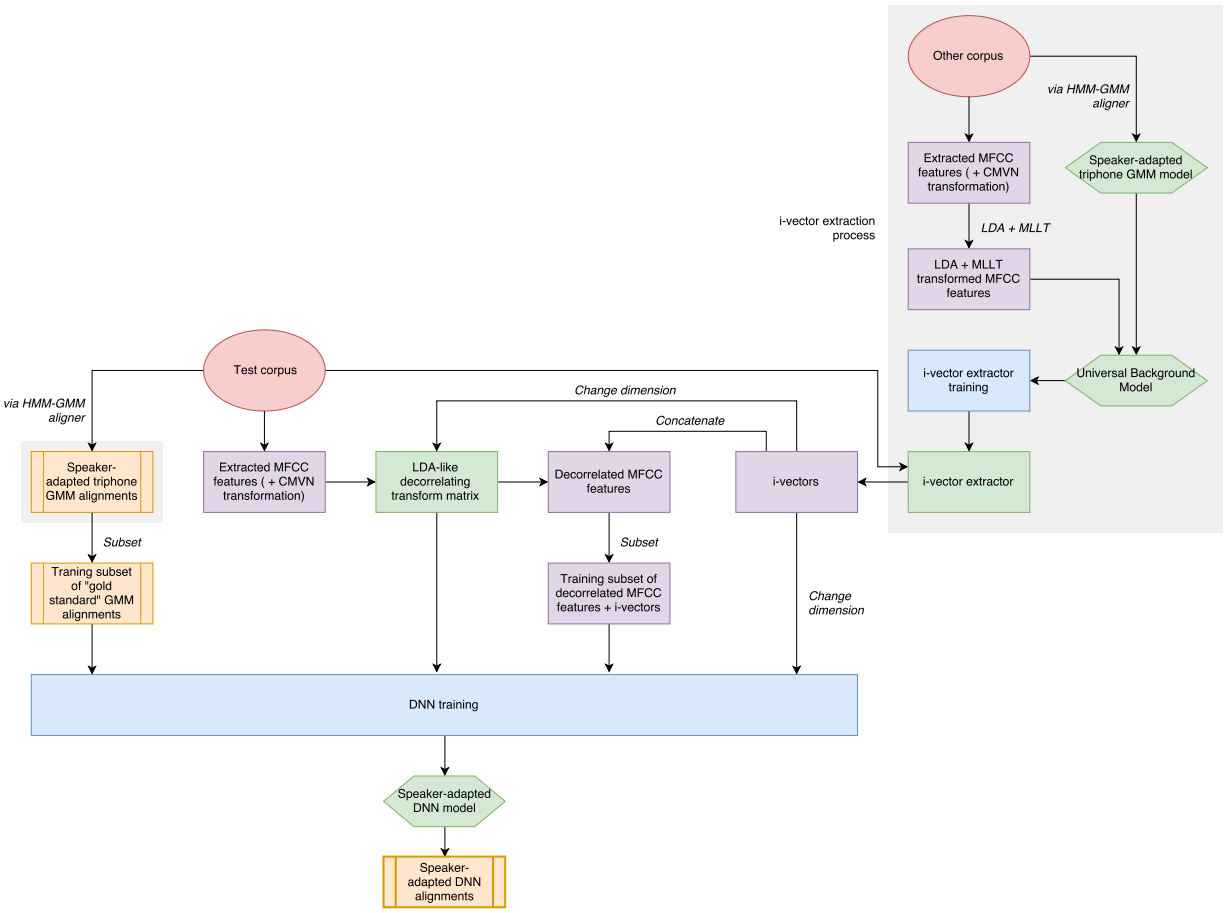

Figure 6: A schematic diagram of the proposed speaker-adapted HMM-DNN architecture, utilizing i-vectors, for the Montreal Forced Aligner. Red circles represent corpora, purple boxes represent feature vectors, green boxes represent transformation matrices, blue boxes represent training periods, green hexagons represent models, and orange boxes represent alignments.

3 Experiments

In assessing the new HMM-DNN architecture of MFA, we address three main questions: (1) how accurate are its produced alignments relative to human-annotated gold standards and to HMM-GMM-generated alignments, (2) what configurations of the DNN result in best performance, and (3) what is the effect of varying the corpora on which different aspects of the DNN are trained? To these ends, we examine the accuracy of several HMM-DNN versions of MFA in detecting alignment boundaries. We also split measurement cases further into detection of phone- vs. word-boundaries, adjacent to vs. non-adjacent to silence, since any systematically differing performance between these cases would indicate where to direct future development efforts.

3.1 Datasets

To evaluate the HMM-DNN version of MFA, we make use of two corpora that come with "gold-standard" human-annotated alignments: Buckeye and Phonsay [31, 26]. Buckeye contains approximately 40 hours of conversational speech from 40 speakers, and its gold-standard alignments (on word and phone levels) are generated by automatic forced alignment followed by hand-correction. For the purposes of this evaluation, the Buckeye phone set has been matched to the phone set of our pronunciation dictionary (since its default phone set has more sub-phonemic detail than we need). Phonsay contains approximately 1.5 hours of lab speech (from McGill University's ProsodyLab) from 73 participants. Its speakers say words in the frame *"Please say _____ again"*, where target words contain a vowel followed by a consonant. Phonsay's gold-standard alignments of this vowel-consonant sequence (on the phone level) are produced only from hand-annotation. All test cases in this work examine alignment performance on monosyllabic CVC words. For Phonsay test cases, the particular words examined are simply the test words in the frame, while for Buckeye, words examined come from the list given in [11].

Buckeye and Phonsay are appropriate corpora for this investigation for two main reasons. First, their size difference (Phonsay being much shorter than Buckeye) allows us to address the question of the effect of dataset size on accuracy measures. Second, since Buckeye uses interview-style speech and Phonsay uses isolated-word lab speech, we are able to explore the effects of our system on two distinct and commonly-used types of speech in linguistic research, one of the main uses of forced alignment.

3.2 Conditions

The new HMM-DNN architecture of MFA itself has several parameters of interest that may be varied. In these experiments, we examine the effects of varying:

i. The corpus used to train the i-vector extractor. We test models constructed using i-vector extractors trained from the same corpus as the test corpus (i.e. Buckeye or Phonsay), on a 40-hour 132-speaker subset of the Librispeech corpus we call *Librispeech Medium*, and on a 100-hour 251-speaker subset of the Librispeech corpus we call *Librispeech 100* [29].

ii. The corpus used to train the acoustic model. We test models trained on the same corpora as for the i-vector extractor variations, i.e. Buckeye or Phonsay, Librispeech Medium, and Librispeech 100.

The use of Librispeech Medium, particularly for the Buckeye test cases, allows us to isolate and examine the effect of speech quality for speaker adaptation (i-vector) and acoustic modelling purposes, since it contains clean audiobook speech (while Buckeye is conversational) and is the same length as Buckeye. Similarly, the use of Librispeech 100 allows us to see the effects of a corpus length much longer than the test corpus. Both of these choices follow the intuition from [26] that either matched data or more data might improve results under some training conditions. Note also for that acoustic models that are not the test corpus, we test only i-vector extraction corpora that match either the test corpus or the acoustic model corpus, in order to imitate a common use case (Google Groups correspondence with the authors of `nnet2` suggest that i-vector extraction corpora ought to be very large, on the order of several thousand hours of speech; it is unlikely therefore that the user would use an i-vector extraction corpus smaller than the acoustic model corpus, save to examine any effect of using the test corpus itself, as we do here).

We remark also that, as is usual when using neural networks, hyperparameter values can have a large effect on performance, as can choice of nonlinearity. These network-specific design choices were arrived at by conducting a set of trials using Buckeye as a test corpus. The combination of a p-norm nonlinearity and online preconditioning of weights and biases was found to substantially increase performace over a tanh nonlinearity. With regard to hyperparameters, tuning over a range of values yielded best results using the values described in Section 2.2.2, with the number of iterations and the learning rate responsible for the most improvement. The hyperparameter values from the testing on Buckeye are those used throughout all experimental conditions in this work. (It may be the case that independently tuning the hyperparameters for each separate corpus tested and used for training an acoustic model could result in better performance for those test corpora, but we approach performance evaluation from the principle that as MFA is a distributable software, a desirable condition for the user would be that hyperparameters are generalizable and could yield good results without the user tuning hyperparameters on his own.)

3.3 Evaluation

In Buckeye's case, the input to the aligner included an orthographic transcription and utterance boundaries only, in order to simulate the common-use case of a forced aligner. We remark also that there is no need to split any data into training and testing subsets, as there is no previously-unseen data on which to "test" in any use case of a forced aligner, which uses entire corpora both to train acoustic models and to perform alignment.

Our test uses the GMM and DNN aligners as above described on the Buckeye and Phonsay corpora to generate alignments on the phone level. We then derive word boundaries in addition to the phone-boundaries and compare them to each corpus' "gold standard" as appropriate (Buckeye contains manual word- and phone-level annotations, while Phonsay only contains those at the phone level). Accuracy of the generated alignments is measured by absolute distance (in milliseconds) of the force-aligned boundary to the gold-standard boundary.

4 Results

We apply our findings to assess the performance of the HMM-DNN version of MFA (1) compared to hand-annotated gold standards and the HMM-GMM version, (2) when DNN parameters are varied, and (3) when trained on different corpora. The HMM-GMM aligner architectures used are those described in [26]. All error bars in the following plots represent a 95% confidence interval.

4.1 Alignment quality relative to gold-standard

We evaluate the HMM-DNN approach by examining the aligning performance of several HMM-DNN aligners. Different HMM-DNN aligners are constructed by varying the corpora used to train the i-vector extractor and the acoustic model, between Librispeech Medium, Librispeech 100, and the test corpus (Buckeye or Phonsay, respectively).

4.1.1 Aligning Buckeye

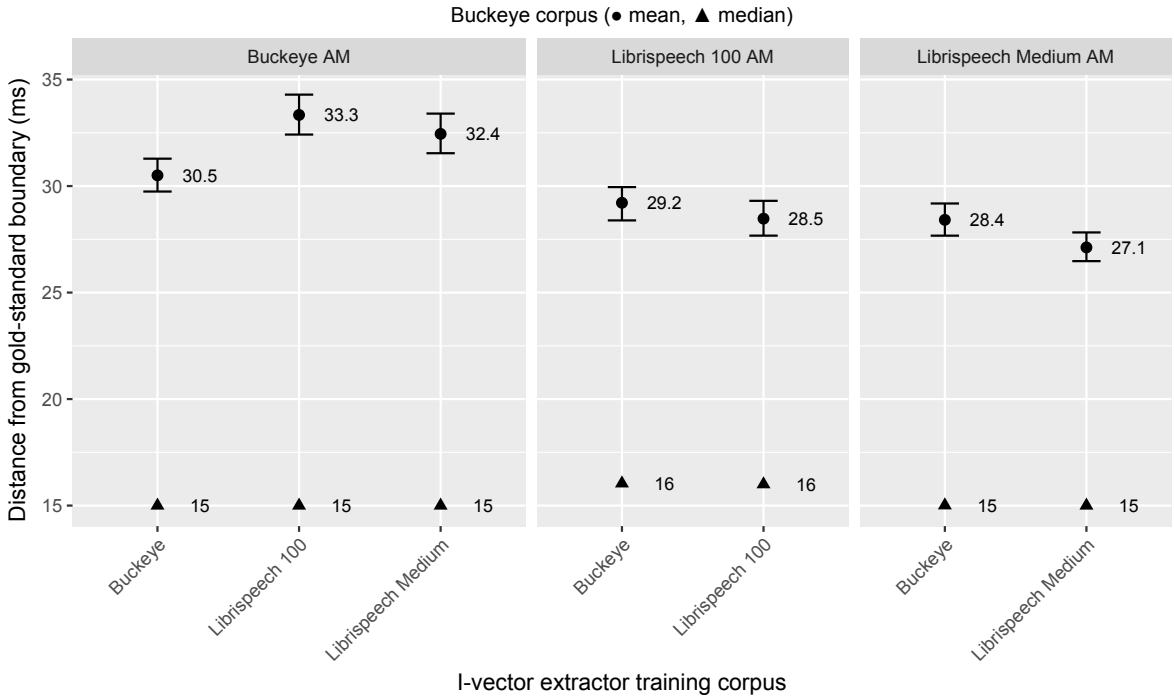

Figure 7: Performance of HMM-DNN aligners on Buckeye at detecting word boundaries, as measured by mean and median absolute differences in milliseconds between aligner-generated and gold-standard boundaries. Data are split into panels by corpus used to train their acoustic model.

Figures 7 and 8 show the performance on Buckeye of the HMM-DNN aligners in comparison to gold-standard alignments at detecting word and phone boundaries respectively. Across HMM-DNN aligners, mean distances are lower for determining phone boundaries than word boundaries. Median distances also tend to be lower at phone boundaries than at word boundaries, although by a smaller margin than mean distances. Additionally, median distances are absolutely lower than mean distances in all cases, suggesting distributions that are generally right-skewed.

When Buckeye, the test corpus, is used to train the acoustic model, using Buckeye to also train the i-vector extractor gives lower mean distances than using other corpora to train the i-vector extractor. However, when other corpora are used to train the acoustic model, using Buckeye to train the i-vector extractor gives higher mean distances than using other corpora. Corresponding median distances, however, do not always follow in the same direction as the means. Generally, using Librispeech Medium to train the acoustic model results in better performance (lower distances) than using Buckeye itself or Librispeech 100, although by a smaller margin for the latter than the former.

To further explore the difference in performance at word boundaries vs. phone boundaries, it is also instructive to split the word boundary measurements from Buckeye into cases where the word boundary is adjacent to silence vs. adjacent to another word. Figure 9 shows the measurements split in this way.

Performance at detecting word boundaries is consistently better, by both mean and median measurements, when the word boundary in question is adjacent to another word than to silence. Similar relative

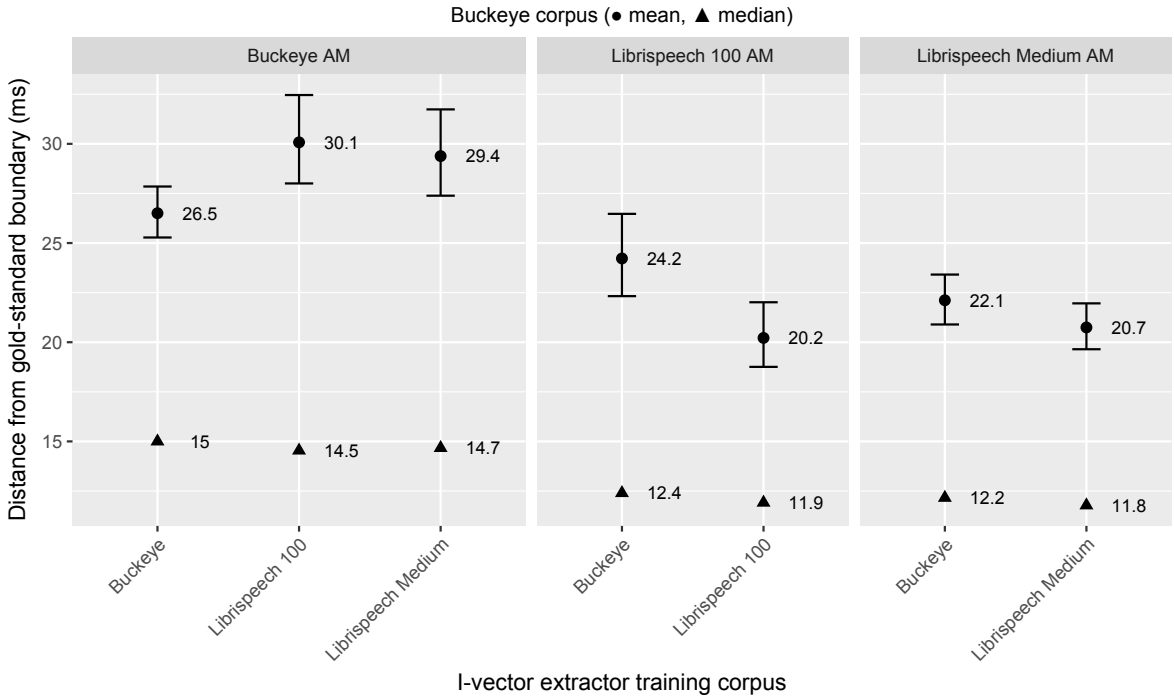

Figure 8: Performance of HMM-DNN aligners on Buckeye at detecting phone boundaries, as measured by mean and median absolute differences in milliseconds between aligner-generated and gold-standard boundaries. Data are split into panels by corpus used to train their acoustic model.

trends to those observed in the above case (which is collapsed over silence and non-silence boundaries) hold: training the acoustic model on corpora other than Buckeye produces smaller distances from the gold standard, and the effects of training the i-vector extractor on corpora other than Buckeye are not in a consistent direction with respect to mean vs. median measurements.

These data mostly suggest that best performance occurs when the i-vector extractor and the acoustic model are trained on the same corpus. However, this is not the case at silence boundaries.

4.1.2 Aligning Phonsay

Figure 10 shows the performances on Phonsay of the HMM-DNN aligners in comparison to gold-standard alignments at detecting phone boundaries. Magnitudes of both medians and means are generally higher than those for Buckeye, although the same general trends hold: using Librispeech Medium to train the acoustic model appears to produce the best results, and use of other corpora than Phonsay to train the i-vector extractor does not move the median and the mean in a consistent direction.

These data do not corroborate the pattern suggested by the Buckeye tests that best performance may occur when the i-vector extractor and the acoustic model are trained on the same corpus, as the only case where matching the i-vector extractor and acoustic model corpora yields the best performance among the Phonsay tests is the Librispeech 100 case.

These results taken together, the HMM-DNN aligner that uses Librispeech Medium to train the i-vector extractor and Librispeech Medium to train the acoustic model performs the best, achieving the lowest

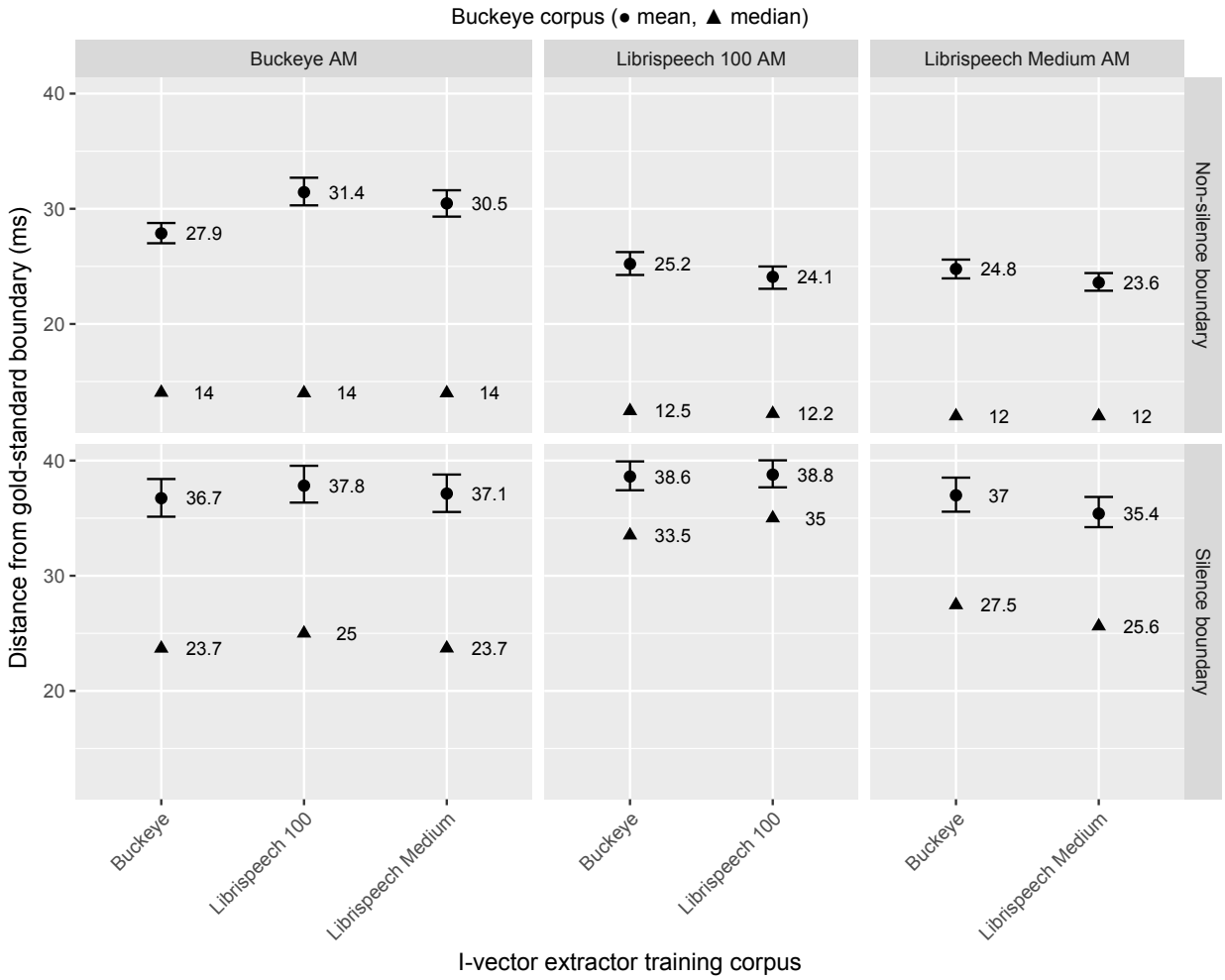

Figure 9: Performance of HMM-DNN aligners on Buckeye in detecting word boundaries adjacent to silence vs. non-silence, as measured by mean and median absolute differences in milliseconds between aligner-generated and gold-standard boundaries. Data are split into panels by corpus used to train their acoustic model, and by silence vs. non-silence adjacency.

distance from the gold-standard alignments produced by human annotators of the test corpora. Hereafter, this aligner will be referred to by the abbreviation *MFA-DNN*.

4.2 Alignment quality relative to HMM-GMM version

Next we examine the performance of the best HMM-DNN aligner, MFA-DNN, vs. that of the HMM-GMM versions produced by [26].

4.2.1 Aligning Buckeye

Figure 11 shows the performance of MFA-DNN on Buckeye against two HMM-GMM aligners developed in [26]: *MFA-Retrained*, wherein a GMM acoustic model is trained from the test corpus, and *MFA-LS*,

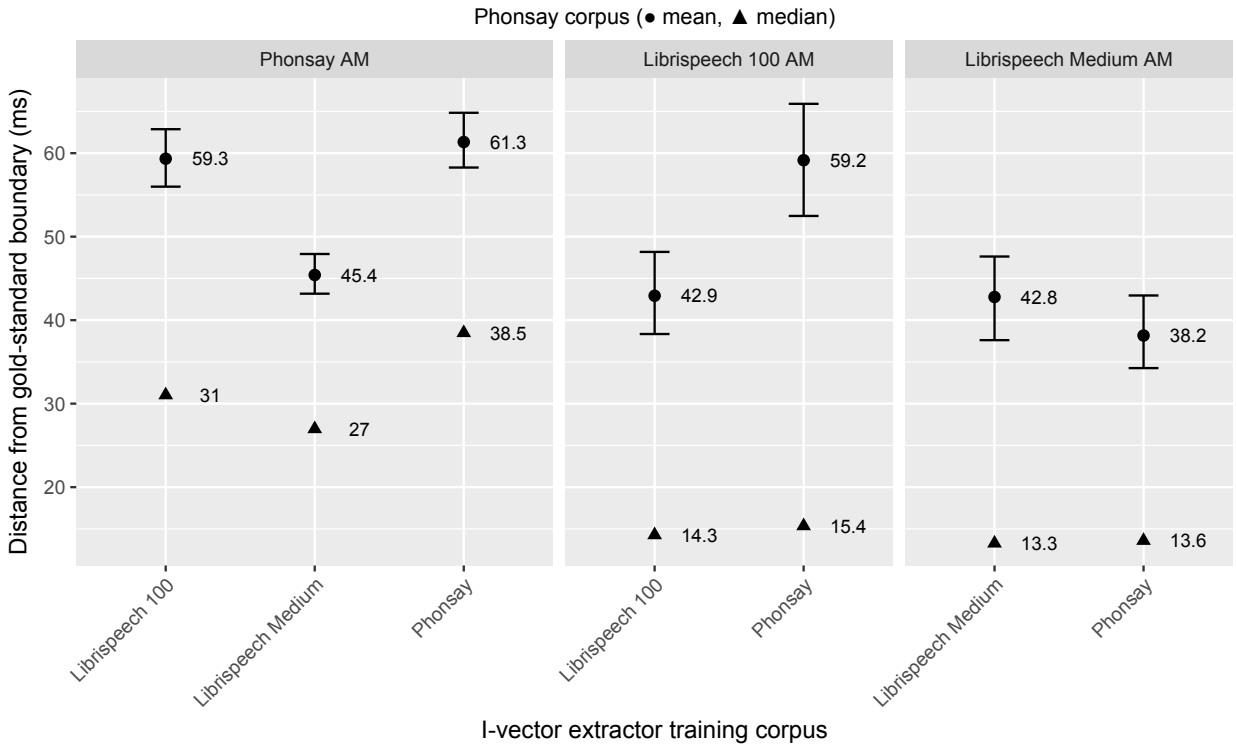

Figure 10: Performance of HMM-DNN aligners on Phonsay at detecting phone boundaries, as measured by mean and median absolute differences in milliseconds between aligner-generated and gold-standard boundaries. Data are split into panels by corpus used to train their acoustic model.

wherein an acoustic model is trained from the full Librispeech corpus. In both word- and phone-boundary cases, MFA-DNN produces higher mean values for distance between aligned and gold-standard boundary, although its median is on par with that of MFA-Retrained, and in the word-boundary case, lower than that of MFA-LS.

Similarly to the above section, we examine the distances when detecting word boundaries for Buckeye in order to better explain this by splitting them further into silence-boundary and non-silence-boundary cases, shown in Figure 12. This reveals that, as in the above cases, performance at silence boundaries tends to be worse than at non-silence boundaries. In the non-silence boundary case, it is still true that MFA-DNN's mean is higher than the HMM-GMM aligners', and that its median is competitive with theirs. However, in the silence-boundary case, MFA-DNN outperforms MFA-LS with respect to both median and mean; its median is also on par with that of MFA-Retrained.

4.2.2 Aligning Phonsay

Examining the performance of MFA-DNN against the HMM-GMM aligners on Phonsay, as shown in 13, shows a somewhat different effect: MFA-DNN still gives the highest mean, but its median is firmly below that of MFA-Retrained. This suggests that while MFA-Retrained may have a smaller spread of its distribution, and that gross alignment errors are more common with MFA-DNN, a larger portion of MFA-DNN's alignments have a lower distance to the gold-standard than of the MFA-Retrained alignments.

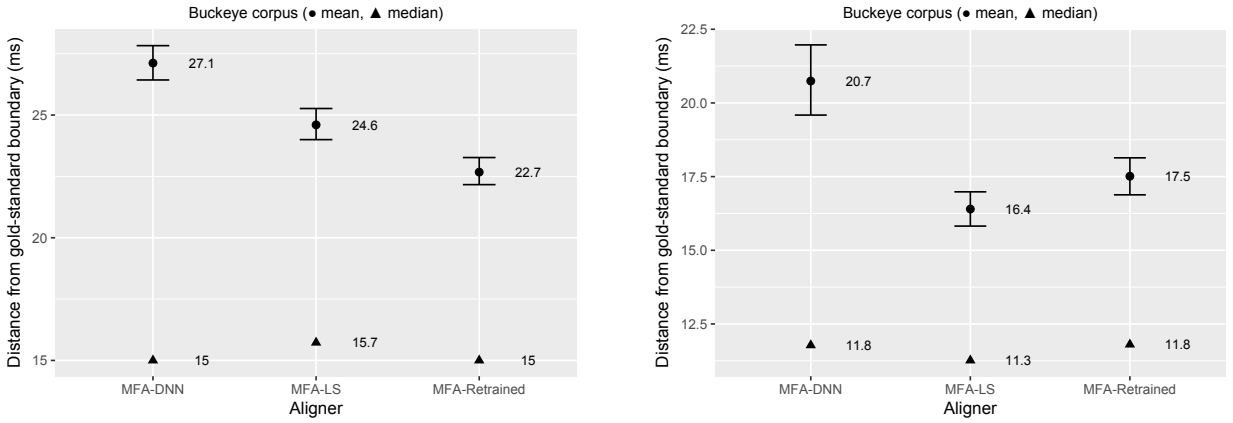

Figure 11: Performance of HMM-GMM aligners (MFA-LS and MFA-Retrained) [26] vs. MFA-DNN on Buckeye, as measured by mean and median absolute differences in milliseconds between aligner-generated and gold-standard word (left) and phone (right) boundaries.

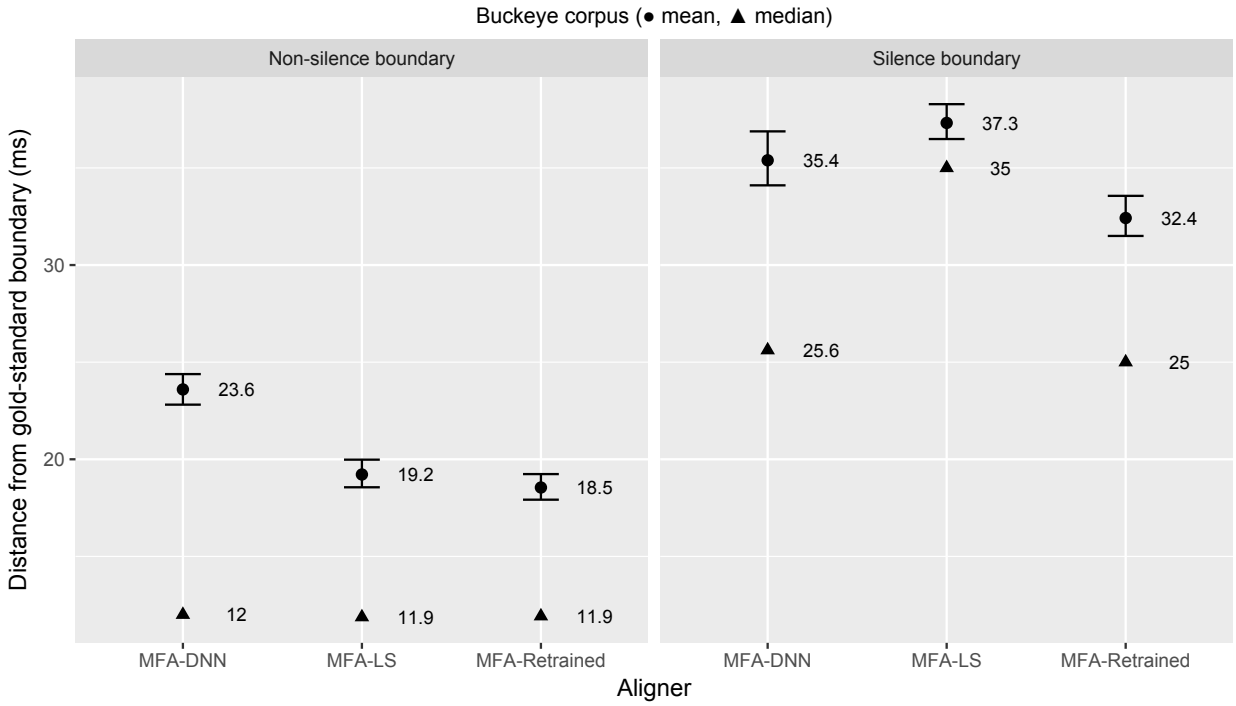

Figure 12: Performance of HMM-GMM aligners [26] vs. MFA-DNN on Buckeye in detecting word boundaries adjacent to silence vs. non-silence, as measured by mean and median absolute differences in milliseconds between aligner-generated and gold-standard boundaries

4.3 Other effects

4.3.1 Silence boundaries

We find that across conditions, performance at detecting word boundaries is better when the boundary is adjacent to another word than to silence. To further examine this effect, which is somewhat present in the

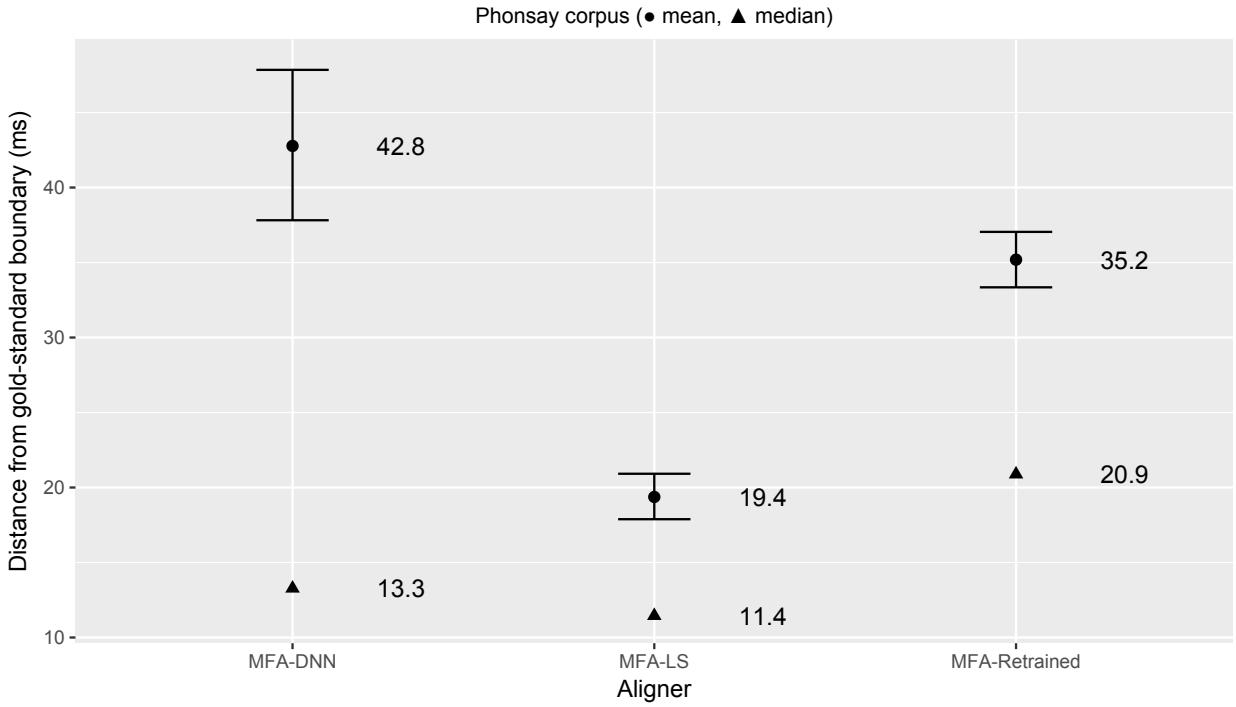

Figure 13: Performance of HMM-GMM aligners [26] vs. MFA-DNN on Phonsay in detecting phone boundaries, as measured by mean and median absolute differences in milliseconds between aligner-generated and gold-standard boundaries.

HMM-GMM versions, but exists to a greater degree in the HMM-DNN version, we analyzed 4000 alignments produced from the unmodified Kaldi `nnet2` recipe on the approximately-90-hour Wall Street Journal (WSJ) corpus, which also uses p-norm nonlinearities and online preconditioning but with hyperparamaters presumably tuned to that corpus [30].

This analysis revealed that the mean absolute difference in milliseconds between the placement of a boundary by the HMM-GMM version vs. the HMM-DNN version of the Kaldi recipe on WSJ was of a larger magnitude when the boundary in question was at a silence boundary than when it was not. Median values, however, do not reflect a difference between placement of silence vs. non-silence boundaries, both cases yielding the same value of 10 ms. (For the sake of clarity, we reiterate that these differences are those between the HMM-GMM boundary placement vs. the HMM-DNN boundary placement, *not* the absolute distances from the gold-standard by each aligner as in the previous figures.) Table 1 shows these results.

The higher mean absolute difference between HMM-GMM and HMM-DNN boundary placement at silence boundaries than at non-silence boundaries suggests that in the course of learning, the DNN may treat silence boundaries differently than a) non-silence boundaries and b) than how GMMs may treat them. Given that WERs are lower (a better performance) at the end of an ASR pipeline (for which these alignments are intermediate steps) for the HMM-DNN setup than the HMM-GMM setup, it may be the case that the ability of DNNs to accurately model realizations of non-silence phones outweighs any deficiencies they may have in modeling silence phones. In other words, silence boundary placement may have a lesser relative importance for end-of-pipeline WER.

It is possible that the absolute differences between HMM-GMM and HMM-DNN Kaldi aligners in Table 1 are in fact in an undesirable direction, and may be overcome by the second half of an ASR pipeline. In this case, they would be in line with our Buckeye and Phonsay data which show more undesirable performance at silence boundaries. Otherwise, it may be the case that the absolute differences are in a desirable, accurate direction, where the WSJ Kaldi recipe examined here is moving silence boundaries by a larger magnitude than non-silence boundaries in the HMM-DNN version which contributes to the lower WER at the end of an ASR pipeline. In that case, the absolute differences in the table would be in a desirable direction, and would not be in line with our Buckeye and Phonsay data. This would suggest given the near-identical architecture of MFA-DNN and the Kaldi recipe that the hyperparameters and/or corpora themselves are responsible for MFA-DNN's undesirable performance at silence boundaries. For further discussion, see Section 6.1.

Silence boundary		Non-silence boundary	
mean	med	mean	med
50.3	10.0	38.2	10.0

Table 1: Absolute mean and median differences in milliseconds in boundary placement at silence boundaries and non-silence boundaries by the Kaldi `nnet2` recipe in generating HMM-GMM vs. HMM-DNN alignments.

4.3.2 Test corpus choice

Means (and typically medians, but to a much lesser degree) of differences between HMM-DNN aligned boundaries and gold-standard boundaries are generally higher for Phonsay than for Buckeye.

4.4 Qualitative error analysis

A manual examination of some MFA-DNN alignments featuring the grossest boundary placement error from the gold standard did not reveal any systematic pattern such as a common speaker. In some Buckeye cases, problems particularly arose around word-final nasals, placing their beginning boundary too far back so as to intrude into the preceding vowel. This may be a by-product of the DNN's handling of consonants produced in unclear or uncommon ways (such as stopped fricatives, although why this should have such a far-reaching effect as the terminal consonant is unclear), or its handling of silence between words. An example of such a misalignment can be found in Figure 14. Neither examination of Phonsay MFA-DNN nor of Buckeye MFA-Retrained gross misalignments revealed a similar pattern.

5 Discussion

The motivation for using DNNs for forced alignment was predicated on the idea that, while trained on GMM alignments and thus presumably trained to make the same predictions that would lead to those GMM alignments, perhaps the DNN would be able to learn some more overarching, complex dependencies during training, resulting in a potentially better alignment. The sign of success, if this were true, would be an

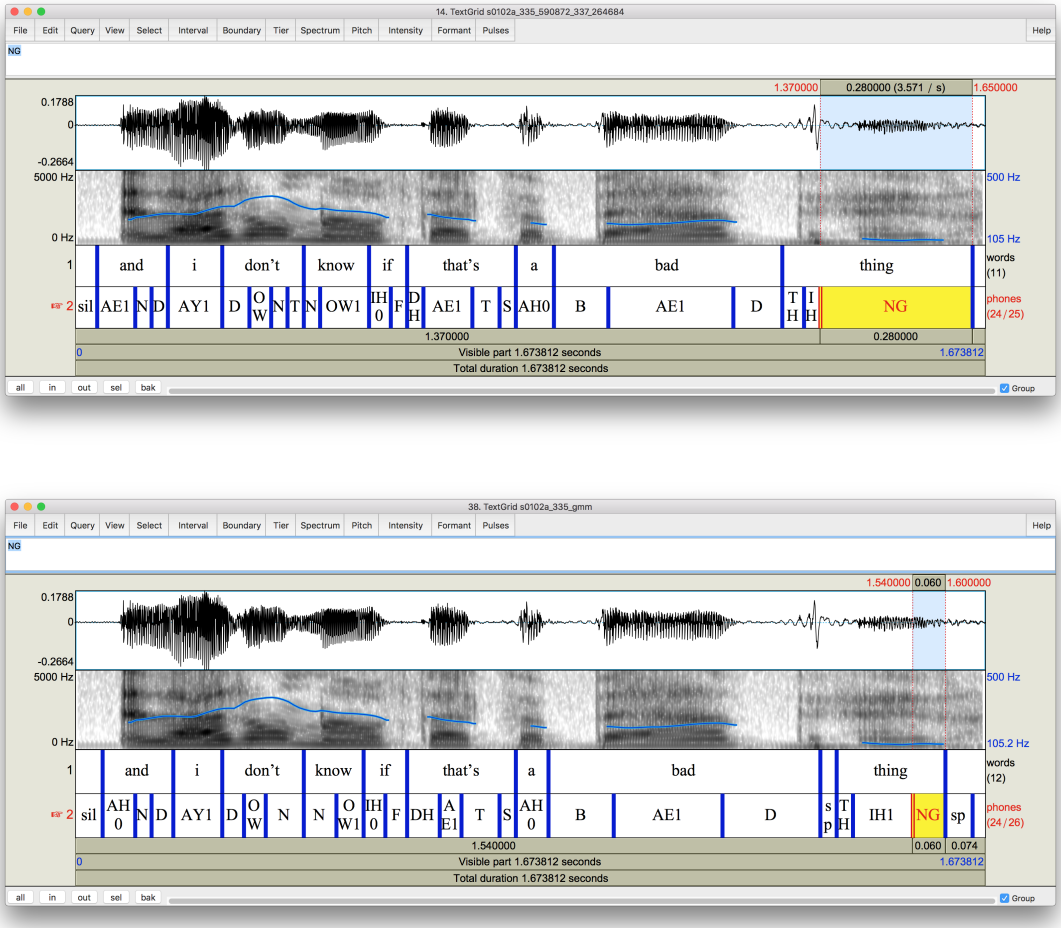

Figure 14: Above, a gross misalignment generated by MFA-DNN. Below, the corresponding alignment generated by MFA-Retrained.

HMM-DNN aligner that produced alignments closer to gold-standard alignments than those produced by the HMM-GMM aligners. We only see this situation under one condition tested: our best HMM-DNN aligner, MFA-DNN, outperforms MFA-Retrained when aligning Phonsay (Figure 13). Other than this single condition, MFA-DNN was unable to surpass the performance of the HMM-GMM aligners. MFA-DNN used Librispeech Medium as the corpus both for training the i-vector extractor and the acoustic model and generally achieved the lowest absolute differences among HMM-DNN aligners from gold-standard alignment boundaries for both Buckeye and Phonsay.

5.1 Acoustic model corpora

The result that an aligner using Librispeech Medium as its acoustic model corpus produced the best HMM-DNN results is somewhat counter-intuitive, as one might expect (as is often the case with DNNs) that the more data one gives to the DNN, the more the DNN will learn, and the better the final result – one might expect an aligner using Librispeech 100 for its acoustic model to yield even better performance. However,

it is worth noting that the hyperparameters used throughout training and testing in this work were tuned to be maximally effective where Buckeye was used as the acoustic model corpus. Therefore, there is no particular reason to expect these hyperparamaters to be equally applicable to Librispeech 100, especially since Librispeech 100 is a much longer corpus than Buckeye (at around 100 vs. 40 hours); it may well be the case that the Librispeech 100 DNN is not finished training when given the same number of iterations as Librispeech Medium as a hyperparameter. On the other hand, Librispeech Medium is the same size as Buckeye, so their DNNs are likely to be finished training at similar numbers of iterations.

The fact that Librispeech Medium produces better alignments than Buckeye when used as the acoustic model corpus, then, is informative. Their two DNNs are likely to be equally trained, and they are equal in length; the main difference between the two corpora is the quality of speech, where Librispeech Medium's audiobook speech is much cleaner than Buckeye's conversational speech. Therefore, all else being equal, training an acoustic model on cleaner speech is likelier to yield better alignments when using an HMM-DNN aligner than less-clean speech, even if the less-clean speech comes from the test corpus. In other words, it is better to train an acoustic model on an external clean-speech corpus than to retrain on the test corpus. (This is generally the opposite of what was observed with HMM-GMM aligners, where retraining on the test corpus produced results among the best [26].)

5.2 I-vector extractor corpora

Interpretation of results regarding the effect of corpus choice used to train the i-vector extractor is more difficult, since any trends also seem to be functions of the acoustic model as well as of the test corpus. For cases using Buckeye as the test corpus, using the same corpus for the acoustic model and the i-vector extractor yields the best results. For cases using Phonsay as the test corpus, using another corpus other than Phonsay for the i-vector extractor yields the best results when Phonsay or Librispeech 100 are used as the acoustic model, while using Phonsay for the i-vector extractor gives the best performance when Librispeech Medium is used for the acoustic model.

5.2.1 Buckeye

For the Buckeye test cases, when Librispeech Medium and Librispeech 100 are used as acoustic model corpora, results are best when Librispeech Medium and Librispeech 100 are used as the i-vector extractor corpora respectively (Figures 7 and 8). We remark that both Librispeech Medium and Librispeech 100 have more speakers than Buckeye. This is a desirable characteristic when training an i-vector extractor; since the extractor will have seen data from a larger number of speakers, it will be able to construct more distinct low-dimensional representations of speakers at the time of testing. Intuitively, this translates into a "speaker embedding" that more accurately sums up the quality of the speaker.

Under this interpretation, it is expected that the Librispeech corpora give better performance under these conditions than Buckeye. (We may also expect cases where Librispeech 100 is used as the i-vector extractor corpus to outperform cases where Librispeech Medium is used, since Librispeech 100 has more speakers than Librispeech Medium. This is indeed the case in Figure 8, where the Librispeech 100 i-vector extractor and acoustic model condition slightly outperforms the corresponding Librispeech Medium condition at determining phone boundaries. However, it is not the case when determining word boundaries Figure 7,

potentially due to confounding acoustic model effects as the Librispeech 100 acoustic model may not be fully trained; see previous subsection for discussion.)

This interpretation would also suggest that in the case where Buckeye is used as the acoustic model corpus, where using Buckeye also as the i-vector extractor corpus gives the lowest distances from the gold standard despite having fewer speakers than the Librispeech corpora, that the presence of "dirtier" speech muddles the beneficial effect of a large number of speakers. It may be the case that given less-clean speech in the acoustic model, the benefit of training the i-vector extractor on speech of similar quality trumps the benefit of training it on a larger number of speakers, although this would require more systematic testing over corpora of varying speech quality and number of speakers to confirm.

5.2.2 Phonsay

The Phonsay test cases require careful treatment as there are many moving parts present. First, the DNN used for testing has hyperparamaters tuned to Buckeye, a corpus of significantly longer length than Phonsay. Second, Phonsay is a corpus of lab speech and is clean in quality, where Buckeye is not. Third, Phonsay has more speakers than Buckeye, but fewer than Librispeech Medium or Librispeech 100.

When Librispeech 100 is used as the acoustic model corpus (Figure 10), we see the expected effect: using Librispeech 100 as the i-vector extractor corpus results in better performance than using Phonsay, presumably since Librispeech 100 has many more speakers than Phonsay. When Librispeech Medium is used as the acoustic model corpus, we see the opposite effect: Phonsay as the i-vector extractor corpus gives a better result than Librispeech medium. This may be due to a similar effect to that described in the previous section: Librispeech Medium is the acoustic model corpus with the highest chance of being amenable to the hyperparameters chosen due to its similarity in length with Buckeye, but due to Phonsay's very different length and quality from Buckeye, making the i-vector extractor train on Phonsay might result in speaker embeddings that can compensate for the difference between the two corpora. Finally, in cases where Phonsay itself is used as the acoustic model corpus, we see that using Phonsay as the i-vector extractor corpus gives the worst results, followed by Librispeech 100 and Librispeech Medium. Remarking again that Phonsay is a radically different type of corpus than Buckeye, it may be the case that training an acoustic model on Phonsay with Buckeye's hyperparameters is too incongruous a task and that previously established relationships between i-vector extractor, acoustic model, and test corpora should not be expected to be seen.

5.3 Performance relative to HMM-GMM version

The only test condition where our best HMM-DNN aligner outperformed the HMM-GMM aligner was when aligning Phonsay, where MFA-DNN resulted in lower median differences from gold-standard alignments than MFA-Retrained (Figure 13). While typically, MFA-Retrained resulted in better alignments than MFA-LS [26], it is still worth noting the caveat that MFA-DNN did not outperform MFA-LS under this particular condition. Also worthy of note is the fact that MFA-DNN's mean under this condition was substantially higher than that of MFA-Retrained, suggesting that the gross misalignments of MFA-DNN are worse in magnitude than those of MFA-Retrained under this condition.

In all other cases, MFA-DNN medians are competitive with (equal to, or within one millisecond of) HMM-GMM medians, with consistently higher means. This consistent right-skew indicates that though the

most extreme misalignments of MFA-DNN are worse than those of the HMM-GMM aligners, a large portion of the alignments generated by MFA-DNN are of comparable quality to those generated by the HMM-GMM aligners.

The presence of the one condition where the MFA-DNN median was lower than an HMM-GMM aligner median suggests that while it may be difficult for a DNN with this architecture to improve on performance from an HMM-GMM aligner, it is not impossible. Examining corpus and model properties further elucidates when this improvement is possible: MFA-DNN uses hyperparameters tuned to Buckeye, a much longer corpus than Phonsay, and trained on Librispeech Medium for its acoustic model, a corpus of similar length to Buckeye (i.e. the DNN is unlikely to not be finished training, and has the advantage of more data to train on than the test corpus) and of similar quality to Phonsay (i.e. clean speech). MFA-DNN also uses Librispeech Medium for its i-vector extractor corpus, which contains more speakers than Phonsay.

This all taken together, a speculative picture of the ideal configuration for an HMM-DNN aligner that surpasses median performance of an HMM-GMM aligner emerges: the acoustic model should be on a corpus similar in quality and longer in length than the test corpus, to which the hyperparameters should be well-tuned, and the i-vector extractor corpus should contain more speakers than the test corpus. While we have consistently used one set of hyperparameters throughout testing in this work, their configuration should eventually be open to the user of the software, allowing them to potentially achieve better results with an HMM-DNN aligner over an HMM-GMM one more often than these results indicate (see Section 6.3 for further discussion).

6 Future work

6.1 Silence boundaries

In the aligned-vs.-gold-standard comparisons, we consistently observe that the aligners' placement of word boundaries adjacent to silence is less accurate than their placement of word boundaries not adjacent to silence; likewise, there are differences on average between Kaldi's WSJ-based HMM-GMM and HMM-DNN aligners' placement of boundaries at silence vs. non-silence boundaries. The fact that these mean differences are variable yet the median differences are highly invariable (Table 1) seems to indicate that the biggest boundary-placement changes from the HMM-GMM to the HMM-DNN when at a silence boundary are larger yet than the biggest changes when not at a silence boundary. Since WSJ has no gold-standard alignments, it is unknown whether these changes are in the direction that would produce a more accurate alignment or not.

The magnitudes of mean difference between these two passes also seem high, especially in comparison to the differences between aligned and gold-standard measurements (Figures 11 and 13). It is uncertain whether this discrepancy in magnitude is due to the corpora and/or hyperparameters used to train Kaldi's neural network, due to the test corpus, or some combination thereof.

It would be desirable, in order to better understand this, to extract a trained acoustic model from the Kaldi recipe and apply it to one of our corpora with gold-standard alignments, such as Buckeye. This would permit us to say more clearly whether the large discrepancies in silence-boundary treatment between the HMM-GMM and HMM-DNN passes are a) helpful in seeking a better alignment and under which

circumstances they might be helpful, and/or b) particular to the training corpus and/or hyperparameters used to train the model, or general to the nature of DNNs. However, the task of pulling out a model generated by Kaldi's `nnet2`, which is designed particularly with WSJ in mind, and using it to align another corpus is not trivial, as it would require substantial modifications to the Kaldi recipe, to MFA, or to both. However, this could be a fruitful future extension, as it would allow the clearer separation of variables in trying to determine the root of differing silence-boundary treatment.

6.2 Realignment

As the schematic diagrams in Section 2 indicate, the HMM-DNN version of MFA only produces one alignment, and this takes place at the end of all neural network training. In development, a strategy for "realignment" was attempted, in efforts to bring MFA's DNN architecture into analogy with both the GMM architecture (where new alignments are generated in each pass from monophone to triphone to triphone with speaker adaptation) and the traditional neural network conception of learned data, after a set of *iterations*, being fed back into the training loop for a new *epoch* of training, allowing for bootstrapping on the learned material. New alignments were thus produced after every given number of training iterations and piped into the training loop themselves, replacing the triphone-speaker-adapted GMM alignments with which the network had been initialized. However, this strategy was unhelpful, with the log-probability of the network losing some ground before regaining it at each realignment.

Modifying this strategy to instead reinitialize the network altogether at the production of new alignments may alleviate this problem, although it would require subsequent careful testing of the best times to realign (potentially as a function of corpus size and other factors). Other strategies could include re-initializing with new alignments at particular layers (perhaps the last one) and/or resetting the learning rate to its default value at certain respective iterations. Fleshing out these various possibilities would constitute a substantial domain for future work, ad-hoc though they are.

6.3 Automatic hyperparameter tuning

The evaluation in this work was approached from the principle that as MFA is a distributable software, a desirable condition for the user would be that hyperparameters are generalizable and could yield good results without the user tuning hyperparameters on his own. All conditions were thus tested using hyperparameters tuned specifically to the Buckeye corpus and evaluated with respect to that principle. However, results revealed that these hyperparameters may not be generalizable, particularly in cases where the corpus used to train the acoustic model and the corpus used to tune the hyperparameters are of different lengths; for example, Librispeech 100 acoustic model conditions failed to outperform other conditions despite a larger wealth of data to learn from, possibly since the hyperparameters tuned to Buckeye, a shorter corpus, cut off its model training before it was finished.

Since users will likely often want to train their own acoustic models using DNNs, it will be desirable in future development to embrace non-generalizability of hyperparameters by opening their up their configuration to the user. In order to cut out significant trial-and-error time from the tuning process, a method for automatic tuning could be developed and integrated into MFA, which could search over hyperparameter configuration space to efficiently find a useful combination without manual user involvement. Such a method

could be implemented via gradient search (following [3]), via random search (following [5]), or via Bayesian optimization (following [4]).

6.4 Performance and software structure

MFA is built to support multiprocessing, allowing the training process to be split into jobs and distributed over several cores as to increase speed. This is essential to software of this type, as audio corpora can be quite large and computationally expensive to treat to the point of becoming prohibitive. It was discovered during development that when running the DNN components of MFA, increasing the number of jobs resulted in significantly worse performance on measures of alignment accuracy. As such, all DNN tests reported were run using a low number of three jobs (the default setting for GMM passes).

As of now it is not known whether this worsening effect is unique to the DNN components of MFA, or whether the GMM components suffer from them to a similar degree. A set of systematic tests examining the effect of number of jobs vs. performance for both the DNN and the GMM components would be helpful in determining the cause of this issue and thus what steps could be taken to relieve it, and in the interim to notify users of the precise nature of the trade-off between speed and performance when using MFA. For the purposes of bolstering this research, it could also be helpful to rerun all tests on only one job, so that performance differences could be more meaningfully attributed to solely the GMM vs. DNN architecture without the interfering variable of number of jobs.

With regard to the DNN components, they may particularly suffer with increasing jobs due to the way that Kaldi (and thus MFA) deals with acoustic models over multiple cores. During training, each core learns a separate model at each iteration, having been allocated each a separate subset of training data, and combines them together at the end of the iteration, in MFA's case by averaging them together. Splitting over a too-high number of cores may therefore allocate too little data to any one core, leading it to learn a model that is not sufficiently representative of the entire dataset. Since MFA's GMM training uses multiprocessing in a similar way, it is possible that the different nature of parameters learned by the DNN than by GMMs also plays a role in how models may be effectively combined, which also merits further investigation.

It may be possible to mitigate this problem by varying, according to some to-be-determined parameter, the way that models are passed between iterations, whether by combining all learned models by averaging, by assessing which model is best and selecting that one, or by some combination thereof. Other strategies might include fully implementing Kaldi's splitting of training examples into training and validation sets (which is likely to have a larger effect with larger corpora) or permitting some overlap between the subsets of data allocated to each core.

Lastly, an important next step in the development of MFA is the implementation of CUDA/GPU functionality, which is already integrated into the original Kaldi recipes. This would allow users to leverage even more of their available computational power to process larger datasets in less time, since GPUs are designed to handle exactly the kind of parallelization that large audio corpora benefit from being treated with. However, the way to reconcile the issue of decreasing performance with increasing number of jobs with even higher parallelization is not immediately clear. It is possible that running MFA with GPUs will yield a better result than with typical CPUs due to the design of GPUs to handle tasks specifically in parallel, rather than to handle sequential tasks like CPUs, but this would require further study to substantiate [20].

References

[1] Martine Adda-Decker and Natalie D Snoeren. "Quantifying temporal speech reduction in French using forced speech alignment". In: *Journal of Phonetics* 39.3 (2011), pp. 261–270.

[2] Xavier Anguera. "Multimodal Read-Aloud eBooks for Language Learning". In: *Sixteenth Annual Conference of the International Speech Communication Association*. 2015.

[3] Yoshua Bengio. "Gradient-based optimization of hyperparameters". In: *Neural computation* 12.8 (2000), pp. 1889–1900.

[4] James S Bergstra et al. "Algorithms for hyper-parameter optimization". In: *Advances in neural information processing systems*. 2011, pp. 2546–2554.

[5] James Bergstra and Yoshua Bengio. "Random search for hyper-parameter optimization". In: *Journal of Machine Learning Research* 13.Feb (2012), pp. 281–305.

[6] Robert AJ Clark, Korin Richmond, and Simon King. "Multisyn: Open-domain unit selection for the Festival speech synthesis system". In: *Speech Communication* 49 (2007), pp. 317–330.

[7] Najim Dehak. "Discriminative and generative approaches for long-and short-term speaker characteristics modeling: application to speaker verification". PhD thesis. École de technologie supérieure, 2009.

[8] Najim Dehak et al. "Cosine similarity scoring without score normalization techniques." In: *Odyssey*. 2010, p. 15.

[9] Najim Dehak et al. "Front-end factor analysis for speaker verification". In: *IEEE Transactions on Audio, Speech, and Language Processing* 19.4 (2011), pp. 788–798.

[10] Zhigang Deng et al. "Expressive facial animation synthesis by learning speech coarticulation and expression spaces". In: *IEEE transactions on visualization and computer graphics* 12.6 (2006), pp. 1523–1534.

[11] Susanne Gahl, Yao Yao, and Keith Johnson. "Why reduce? Phonological neighborhood density and phonetic reduction in spontaneous speech". In: *Journal of memory and language* 66.4 (2012), pp. 789–806.

[12] Mark Gales and Steve Young. "The application of hidden Markov models in speech recognition". In: *Foundations and trends in signal processing* 1.3 (2008), pp. 195–304.

[13] Kyle Gorman, Jonathan Howell, and Michael Wagner. "Prosodylab-aligner: A tool for forced alignment of laboratory speech". In: *Canadian Acoustics* 39.3 (2011), pp. 192–193.

[14] Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. "Speech recognition with deep recurrent neural networks". In: *Acoustics, speech and signal processing (icassp), 2013 ieee international conference on*. IEEE. 2013, pp. 6645–6649.

[15] Alex Graves et al. "Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks". In: *Proceedings of the 23rd international conference on Machine learning*. ACM. 2006, pp. 369–376.

[16] Søren Halkjær and Ole Winther. "The effect of correlated input data on the dynamics of learning". In: *Advances in neural information processing systems*. 1997, pp. 169–175.

[17] Geoffrey Hinton et al. "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups". In: *IEEE Signal Processing Magazine* 29.6 (2012), pp. 82–97.

[18] Mei-Yuh Hwang and Xuedong Huang. "Subphonetic modeling with Markov states-Senone". In: *Acoustics, Speech, and Signal Processing, 1992. ICASSP-92., 1992 IEEE International Conference on*. Vol. 1. IEEE. 1992, pp. 33–36.

[19] Dan Jurafsky and James H Martin. *Speech and language processing*. Vol. 3. Pearson London: 2014.

[20] David Kirk et al. "NVIDIA CUDA software and GPU parallel computing architecture". In: *ISMM*. Vol. 7. 2007, pp. 103–104.

[21] Thomas Kisler, Florian Schiel, and Han Sloetjes. "Signal processing via web services: the use case WebMAUS". In: *Digital Humanities Conference 2012*. 2012.

[22] William Labov, Ingrid Rosenfelder, and Josef Fruehwald. "One hundred years of sound change in Philadelphia: Linear incrementation, reversal, and reanalysis". In: *Language* 89.1 (2013), pp. 30–65.

[23] Christopher J Leggetter and Philip C Woodland. "Maximum likelihood linear regression for speaker adaptation of continuous density hidden Markov models". In: *Computer Speech & Language* 9.2 (1995), pp. 171–185.

[24] Liang Lu, Xingxing Zhang, and Steve Renais. "On training the recurrent neural network encoder-decoder for large vocabulary end-to-end speech recognition". In: *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on*. IEEE. 2016, pp. 5060–5064.

[25] Michael McAuliffe et al. *Montreal Forced Aligner: an accurate and trainable aligner using Kaldi.* Presented at the 91st Annual Meeting of the Linguistic Society of America, Austin, TX. 2017.

[26] Michael McAuliffe et al. "Montreal Forced Aligner: trainable text-speech alignment using Kaldi". In: *Proceedings of interspeech*. 2017.

[27] Todd K Moon. "The expectation-maximization algorithm". In: *IEEE Signal processing magazine* 13.6 (1996), pp. 47–60.

[28] R. M. Ochschorn and M. Hawkins. *Gentle forced aligner [computer program]*. `https://github.com/lowerquality/gentle`. 2017.

[29] Vassil Panayotov et al. "Librispeech: an ASR corpus based on public domain audio books". In: *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*. IEEE. 2015, pp. 5206–5210.

[30] Douglas B Paul and Janet M Baker. "The design for the Wall Street Journal-based CSR corpus". In: *Proceedings of the workshop on Speech and Natural Language*. Association for Computational Linguistics. 1992, pp. 357–362.

[31] Mark A Pitt et al. "Buckeye corpus of conversational speech (2nd release)". In: *Columbus, OH: Department of Psychology, Ohio State University* (2007).

[32] Daniel Povey and George Saon. "Feature and model space speaker adaptation with full covariance Gaussians." In: *INTERSPEECH*. 2006.

[33] Daniel Povey, Xiaohui Zhang, and Sanjeev Khudanpur. "Parallel training of DNNs with natural gradient and parameter averaging". In: *arXiv preprint arXiv:1410.7455* (2014).

[34] Daniel Povey et al. "The Kaldi speech recognition toolkit". In: *IEEE 2011 workshop on automatic speech recognition and understanding*. EPFL-CONF-192584. IEEE Signal Processing Society. 2011.

[35] Ingrid Rosenfelder et al. "FAVE (forced alignment and vowel extraction) program suite". In: *URL http://fave. ling. upenn. edu* (2011).

[36] George Saon et al. "Speaker adaptation of neural network acoustic models using i-vectors." In: *ASRU*. 2013, pp. 55–59.

[37] Colin W Wightman and David T Talkin. "The aligner: Text-to-speech alignment using Markov models". In: *Progress in speech synthesis*. Springer, 1997, pp. 313–323.

[38] Dong Yu and Li Deng. *Automatic speech recognition: A deep learning approach*. Springer, 2014.

[39] Xiaohui Zhang et al. "Improving deep neural network acoustic models using generalized maxout networks". In: *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*. IEEE. 2014, pp. 215–219.