USING EIGENVOICES AND NEAREST-NEIGHBOURS IN HMM-BASED CROSS-LINGUAL SPEAKER ADAPTATION WITH LIMITED DATA

A Dissertation

by

Seyyed Saeed Sarfjoo

Submitted to the
Graduate School of Sciences and Engineering
In Partial Fulfillment of the Requirements for
the Degree of

Doctor of Philosophy

in the
Department of Computer Science

Özyeğin University
August 2017

USING EIGENVOICES AND NEAREST-NEIGHBOURS IN HMM-BASED CROSS-LINGUAL SPEAKER ADAPTATION WITH LIMITED DATA

Approved by:

Assistant Professor Cenk Demiroğlu, Advisor
Department of Electrical and Electronics Engineering
Özyeğin University

Professor Murat Saraçlar
Department of Electrical and Electronics Engineering
Boğaziçi University

Associate Professor Murat Şensoy
Department of Computer Science
Özyeğin University

Associate Professor Ümit Güz
Department of Electrical and Electronics Engineering
Işik University

Associate Professor H. Fatih Uğurdağ
Department of Electrical and Electronics Engineering
Özyeğin University

Date Approved: 15 August 2017

To my family, my friends, and the professors who helped me to become who I am.

ABSTRACT

Cross-lingual speaker adaptation for speech synthesis has many applications, such as use in speech-to-speech translation systems. Here, we focus on cross-lingual adaptation for statistical parametric speech synthesis systems using limited adaptation data. We propose new methods for hidden Markov model (HMM)-based and deep neural network (DNN)-based speech synthesis. To that end, for HMM-based speech synthesis we propose two eigenvoice adaptation approaches exploiting a bilingual Turkish-English speech database that we collected. In one approach, eigenvoice weights extracted using Turkish adaptation data and Turkish voice models are transformed into the eigenvoice weights for the English voice models using linear regression. Weighting the samples depending on the distance of reference speakers to target speakers during linear regression was found to improve the performance. Moreover, importance weighting the elements of the eigenvectors during regression further improved the performance. The second approach proposed here is speaker-specific state-mapping, which performed significantly better than the baseline state-mapping algorithm both in objective and subjective tests. Performance of the proposed state mapping algorithm was further improved when it was used with the intra-lingual eigenvoice approach instead of the linear-regression based algorithms used in the baseline system. In addition to the rapid adaptation algorithm, we propose a new unsupervised adaptation method for DNN-based cross-lingual speech synthesis. In this method, using a sequence of acoustic features from a target speaker, we estimate continuous linguistic features for unlabeled data. Based on objective and subjective experiments, adapted model outperformed the gender-dependent average voice models in terms of quality and similarity.

ÖZETÇE

Ses sentezi için çapraz-dilli konuşmacıya uyarlanma, sesten sese çeviri sistemleri gibi birçok kullanım alanına sahiptir. Bu tezde, sınırlı uyarlama verilerini kullanan istatistiksel konuşma sentezi sistemleri için çapraz-dilli uyarlamaya odaklanılmış ve HMM-/DNN-tabanlı konuşma sentezinde yeni yöntemler önerilmiştir. Bu amaçla, topladığımız iki dilli bir Türkçe-İngilizce konuşma veritabanını kullanarak HMM-tabanlı konuşma sentezi için, iki özses uyarlama yaklaşımı önermekteyiz. Bir yaklaşımda, Türkçe uyarlama verileri ve Türkçe ses modeli kullanılarak çıkarılan özses ağırlıkları doğrusal bağlanım kullanılarak İngilizce ses modelleri için özses ağırlıklarına dönüştürülmüştür. Doğrusal bağlanım esnasında referans konuşmacıların hedef konuşmacılara olan mesafesine bağlı olarak örneklerin ağırlıklandırılmasının performansı arttırdığı gözlemlenmiştir. Dahası, bağlanım sırasında özvektörlerin elemanlarının önem ağırlıklandırılması performansı daha da geliştirmiştir. Burada önerilen ikinci yaklaşım temel sistem olan durum-haritalama algoritmasından hem nesnel hem de öznel testlerde daha iyi performans gösteren konuşmacıya özel durum-haritalamasıdır. Temel sistemde kullanılan doğrusal bağlanım temelli algoritmalar yerine dil içi öz ses yaklaşımı ile birlikte kullanıldığında, önerilen durum-haritası algoritmasının performansı daha da artmıştır. Hızlı uyarlanma yöntemlerinin yanında, çapraz-dilli, DNN-tabanlı konuşma sentezi için bir güdümsüz uyarlama yöntemi önerilmiştir. Bu yöntemde, hedef konuşmacının akustik özellik dizisi kullanılarak, etiketlenmemiş veriler için sürekli dil özellikleri tahmin edilmiştir. Hem nesnel hem de öznel deney sonuçlarında, uyarlanan modelin cinsiyete bağlı ortalama ses modellerini kalite ve benzerlik açısından geçtiği gözlenmiştir.

ACKNOWLEDGEMENTS

I would like to express my gratitude towards my supervisor, Professor Cenk Demiroğlu, for advising me through the learning process of this Ph.D. thesis. His precious guidance and intimate attitude during this period has made me a much more dedicated researcher. I also would like to thank my thesis committee members Professor Fatih Uğurdağ, Professor Ümit Güz, and specially Professor Murat Şensoy and Professor Murat Saraçlar for spending their precious time. During the period of this thesis, many friends have been helpful to color my life. My past and present friends in the speech processing lab made this journey more fun. I have been blessed with a friendly and cheerful group of fellow students in the graduate school. I would also like to thank my parents Nasir and Lina, for their unconditional love and support. Special thanks to staff of our university for giving us an opportunity to work with wonderful people in a warm environment.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER I

INTRODUCTION

Text-to-Speech (TTS), or speech synthesis, refers to converting written text, into spoken language signal, namely, speech. Nowadays, two prominent TTS approaches are concatenative speech synthesis (CSS) and statistical parametric speech synthesis (SPSS). In CSS, speech is segmented into smaller units and these units are stored in a database. During synthesis, the units that best match the input text are selected and an utterance is synthesized by concatenating the selected units [1]. Concatenative speech synthesis systems can create high quality speech but requires a huge speech database during the synthesis time. Statistical parametric speech synthesis (SPSS) has proven to be a promising TTS approach with some advantages compared to the concatenative approach [2]. An important advantage of the SPSS approach is the ability to adapt to a target speaker with limited adaptation data [3]. Compared to CSS systems, SPSS are able to synthesize smoothly-varying speech but with some degradation in quality. SPSS systems also have smaller footprint and use relatively less resources. Decision tree-clustered context-dependent hidden Markov models (HMMs) are typically used in conventional approaches of statistical parametric speech synthesis to represent probability densities of speech parameters given text [4]. This approach is effective but has some limitations, e.g., decision trees are not efficient for modeling the complex context dependencies. An alternative scheme for modeling these dependencies is using deep neural networks (DNNs) [5]. The relationship between input texts and their acoustic realizations is modeled by a DNN. The use of the DNN can address some limitations of the conventional approach. To capture the correlation or co-occurrence information between any two instants in a speech utterance for parametric speech synthesis, Recurrent Neural Networks (RNNs) with Bidirectional Long Short Term Memory (BLSTM) cells are proposed in [6].

Cross-lingual speaker adaptation (CLSA) for statistical parametric speech synthesis is a method for adapting a text to speech for a desired *output* language, given adaptation data (i.e., speech) from the target speaker in a different *input* language. Applications of CLSA include speech-to-speech translation [7, 8].

In a commonly used approach [9–11], a speaker-independent acoustic model (an "Average Voice Model" or AVM) for each of the two languages is required. A mapping between pairs of corresponding states in the two models is constructed, on the basis of the states' acoustic similarity. Then, either the adaptation data itself, or speaker transformation functions, can be mapped from the input language acoustic model to the output language acoustic model.

Mismatch between the two AVMs degrades the output speech quality when mapping transforms [12, 13] since the speaker-specific transformations for states in the input language acoustic model may not actually represent the corresponding state in the output language acoustic model well. To alleviate that problem, a transform mapping using shared decision tree context clustering is proposed in [14] where not only acoustic-similarity but also contextual similarity of states is taken into account during mapping.

DNN-based methods have also been used for training multilingual acoustic models [15–17]. Adaptation in DNN can be done in input, hidden, and output layers. Estimating speaker codes using i-vector [18] or discriminant condition codes (DCC) [19,20] are methods of speaker adaptation using modification in input layer. Learning hidden unit contribution (LHUC) [21] is an example of speaker adaptation using modification in hidden layers. Feature space transformation [22] and multi-speaker modeling [23] do adaptation using modification in output layer.

In this dissertation, we focused on cross-lingual adaptation when only a few utterances are available from a target speaker. In our recent paper [24], to achieve better speaker similarity than existing state-mapping based algorithms under limited data conditions, we proposed two methods. In the first method, eigenvoices were used for rapid adaptation. Eigenvoice weights computed for the input language are linearly transformed into output language weights. The transformation matrix is learned using a bilingual training database which contains English and Turkish speech data from the same speakers.

In the second method, we proposed speaker-specific state-mapping, for which a bilingual database was used. After generating speaker-adapted models for both input and output languages, a speaker-specific state-map is constructed for each speaker in the pool of bilingual speakers. Then, for a previously-unseen target speaker, a nearest-neighbour is found in the pool and the state map of that nearest-neighbour is used for adaptation. Performance for the excitation parameters was found to be significantly better with the proposed method than the baseline generic state-mapping algorithm, in objective and subjective tests.

In our proposed system, during eigenvector transformation, to avoid overfitting and exploit correlations within eigenvector elements, a partial least squares (PLS) approach is used. To further boost the performance, elements of eigenvectors are also weighted using recursive PLS (rPLS). Moreover, in addition to weighting the eigenvectors in a least-squares linear regression approach, as done in [24], eigenvectors are weighted in the proposed PLS and rPLS frameworks leading to weighted-PLS and weighted-rPLS algorithms. The proposed state-mapping algorithm is used for mapping the data in the input language to models in the output language and performing cross-lingual eigenvoice adaptation which enabled significant improvement in the spectral envelope features.

As last novelty, we introduce a new method of unsupervised speaker adaptation using continuous labels. In this method, using BLSTM-RNN model, we estimate the continuous labels (c-lab) for unlabeled adaptation data of target speaker. We used DCC structure for estimating the speaker codes and speaker codes are combined with gender and age codes in input layer. For eliminating the mismatch between binary and continuous labels, AVM model is trained with c-labs. Using another BLSTM-RNN model, we estimated the continuous labels (c-lab) from the binary labels (b-lab). In synthesis time, b-lab is transformed to c-lab before applying to the adapted model. Performance of the proposed method is shown with objective and subjective experiments.

Summary of the research contributions of this dissertation is shown below:

- For spectral envelope features, adaptation using a linear transformation of the eigenvoice weights from input to output language outperformed the baseline data-mapping based state-mapping algorithm.

- Estimation of the eigenvoice weights directly in output language using nearest-neighbor bilingual speaker model from the pool of references, significantly improved the performance of adaptation.

- For excitation features, speaker specific state-mapping outperformed the baseline data-mapping based state-mapping algorithm.

- Using continuous labels in unsupervised DNN-based adaptation, performance of the cross-lingual adaptation was similar to the performance of intra-lingual adaptation.

Publications from research contribution of this dissertation are shown below:

- Mohammadi, Amir, Seyyed Saeed Sarfjoo, and Cenk Demiroglu. "Eigenvoice speaker adaptation with minimal data for statistical speech synthesis systems using a MAP approach and nearest-neighbors." IEEE/ACM Transactions on Audio, Speech, and Language Processing 22, no. 12 (2014): 2146-2157.

- Sarfjoo, Seyyed Saeed, and Cenk Demiroglu. "Cross-Lingual Speaker Adaptation for Statistical Speech Synthesis Using Limited Data." In INTERSPEECH, pp. 317-321. 2016.

- Sarfjoo, Seyyed Saeed, Cenk Demiroglu, and Simon King. "Using eigenvoices and nearest-neighbors in HMM-based cross-lingual speaker adaptation with limited data." IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP) 25, no. 4 (2017): 839-851.

This dissertation is organized as follows. Speech synthesis systems are described in Chapter 2. Intra-lingual speaker adaptation methods are described in Chapter 3. Cross-lingual speaker adaptation methods are described in Chapter 4. Proposed algorithms are discussed in Chapter 5. Experiment results are presented and discussed in Chapter 6. Finally, conclusion is done in Chapter 7.

CHAPTER II

SPEECH SYNTHESIS SYSTEMS

Speech synthesis or text-to-speech (TTS) is a method for generating comprehensible, high quality human-like artificial speech for an input text. Navigation systems in vehicles, e-book readers, screen readers for the visually impaired, and communication devices for the speech impaired are samples samples of applications for speech synthesis systems. Recently applications like speech enabled dialog systems, forthcoming robots, singing style speech synthesizers, and speech-to-speech translation systems are introduced for speech synthesis field.

Speech synthesis system can be fractioned in two main components, natural language processing (NLP) part and digital speech processing (DSP), part which are sometimes called “frontend” and “backend”, respectively. In the natural language processing part, given text will be converted to sequence of linguistic features consisting of linguistic elements like grapheme, phoneme, etc. In the DSP part, raw speech waveforms are generated from the estimated linguistic features. The methods for generating raw speech waveform from input text have progressed from knowledge-based and rule-based methods to data-driven methods. There are number of techniques to create synthetic speech which is known as speech synthesis systems. Two prominent approaches are concatenative speech synthesis (CSS) and statistical parametric speech synthesis (SPSS). Most common method in SPSS was HMM-based speech synthesis (HSS). CSS and HSS come with their own Pros and Cons. Therefore, researchers in speech synthesis also investigated to use of both models together in different hybrid schemes to obtain the advantages of both techniques. An alternative scheme in SPSS for modeling probability densities of speech parameters given texts is using deep neural network (DNN) [5].

2.1 Concatenative Speech Synthesis

Because of lack of resources and processing powers, in the early 1970s, very low dimensional acoustic parameters like formants were used in speech data generation methods. These low dimensional features usually used to model the vocal tract resonances [25]. In the late 1980s, second and first halves of two adjacent phonemes were used as units for the speech waveform generation. This unit in small dataset of phoneme is called "diphones". With concatenating these units based on the sequence of input phonemes using some digital signal processing methods, such as "linear predictive" (LP) analysis, the synthetic speech waveform was generated [26]. With increasing the dataset sizes and processing powers, in the 1990s, for selecting more suitable speech units that cover both phonemes and other "linguistic contexts" like "lexical stress", "pitch accent", and "part-of-speech" information, bigger datasets were collected. These datasets facilitate the generating of human-like and intelligible speech waveform with appropriate prosody. This approach is usually called "unit selection" system and using this system, various commercial systems were develop high quality synthetic speech [27, 28]. This system has been the dominant approach both in the industry and literature for decades. Despite the growing popularity of the SPSS, it is still a major figure in TTS technology. Unit selection based on a very intuitive, yet very successful idea: concatenating the recorded speech units to create synthetic speech. In a general framework and formulation of unit selection systems [27], two costs are defined, target cost $C^T(t_t, u_t)$ and concatenation cost $C^C(u_{t+1}, u_t)$. Then the total cost of unit sequence $U = [u_1, u_2, ..., u_T]$ for a given sentence $S = [t_1, t_2, ..., t_T]$ is given by

$$C(U, S) = \Sigma_{t=1}^{T} C^T(t_t, u_t) + \Sigma_{t=1}^{T-1} C^C(u_{t+1}, u_t), \qquad (1)$$

where $C^T(t_t, u_t)$ is used to measure how suitable the unit u_t is for target t_t. $C^C(u_{t+1}, u_t)$ measures how well the adjacent units u_{t+1} and u_t can be joined. Then a search is applied over all possible sequence to find the optimal sequence $\hat{U}$ which minimizes the total cost $C(U, S)$.

$$\hat{U} = \operatorname*{argmin}_{U} C(U, S) \tag{2}$$

Feature vectors used in the target cost calculations generally consist of phonetic and prosodic contexts. Spectral and acoustic features may be used in concatenation cost calculations to be used with acoustic distance measures. Researchers still look at what features to be used and how to weigh them in cost calculations. Optimal size of the unit is not a resolved issue in unit selection. There is wide variety of unit sizes used in the works. Frame-sized [29], HMM state-sized [30], half-phones [31], diphones [32] and varying sized [33] are the some examples of different unit sizes that are used in unit selection systems. A general observation is that the longer the unit size, the bigger database is required to cover given domain. Overview of unit selection system is shown in Figure 1.

Output speech in the unit selection method, is constrained to be like the original wave files and just a little changing to the selected units of natural speech are usually done. For covering various speaking styles and emotions with unit selection method, we need to record bigger speech database which contains vast speaking styles and emotions. Usually, preparing large speech dataset with rich speaking styles and emotions is expensive and takes significant amount of time [34]. For answering the need for controlling the generated speech in some aspects such as covering singing style, including emotion, or adapting to some specific target speaker, another data-driven method called "statistical parametric speech synthesis" was introduced in the late 1990s.

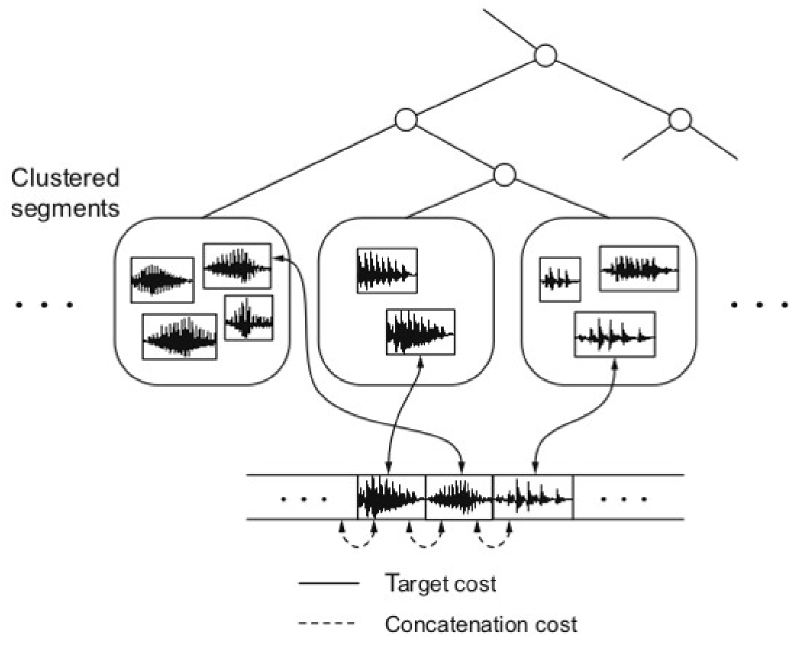


Figure 1: Overview of unit selection system.

2.2 HMM-Based Speech Synthesis

A model for synthetic speech generation in "statistical parametric speech synthesis" usually uses a "hidden Markov model" (HMM) as its generative model. This method of speech synthesis is usually called HMM-based speech synthesis. In this method, in addition of phoneme sequences, HMMs can model several suprasegmental specification just like the unit selection method. In the synthesis time, based on segmental and suprasegmental linguistic features, acoustic speech parameters will be generated from HMMs. These acoustic parameters are input features of a vocoder, which is a source-filter model of speech production. In this model, the speech is generated by applying a filter or vocal tract to source or excitation parameters. Several well-developed and optimized machine learning algorithms, like "BaumWelch", "Viterbi", and clustering algorithms were applied in "automatic speech recognition" (ASR) field [35]. Applying these algorithms to speech synthesis and using several open-source toolkits that contain natural language processing, digital signal processing, and HMMs [36–39], caused the popularity of HMM-based speech synthesis such that it is used globally for both research and commercial purposes.

In recent decade, the quality of HMM-based synthetic speech has been improved, e.g. [40,41] and many techniques for controlling speech variations is proposed, e.g. [42, 43]. In this section, first we will investigate the process of human speech production and simulation model of this production. Details of Hidden Markov Model, parameter generation from HMM model, train and synthesis parts will be discussed sequentially.

2.2.1 Speech Production and Vocoder

Human speech system contains several components such as “lungs”, “larynx”, “pharyngeal cavity” or “throat”, “tracher” or “windpipe”, “oral” or “buccal cavity” (mouth), and “nasal cavity” (nose). The “oral tract” is the system containing the pharyngeal and the oral cavity. Normally, the “nasal cavity” is called the “nasal tract”. Place of these organs in human body is shown in Figure 2. Human speech system in schematic form is shown in Figure 3.

Using signal processing techniques, the process of producing speech in human (Figure 4) can be estimated with a digital “source-filter model” shown in Figure 5. The theory behind this “source-filter model” is based on the similar structure in human voice production [44]. Excitation in the simplest model is white noise for unvoiced and pulse train for voiced data. This source can be filtered with a single filter to model the “spectral envelopes” of the “glottal flow”, “vocal tract resonance”, and “lip radiation effect”. With this structure, components of human speech signal like voiced/unvoiced tags, fundamental frequency (F_0), and “spectral envelope” will be modeled. Spectral envelope can be extracted by mel-cepstral coefficients [45] or line spectral pairs (LSPs) [46], and at the end the sequence of these acoustic parameters will be synthesized to generate the speech waveforms. Using input text, in HSS these vocoder parameters will be predicted. Synthetic voice in frame level will be generated

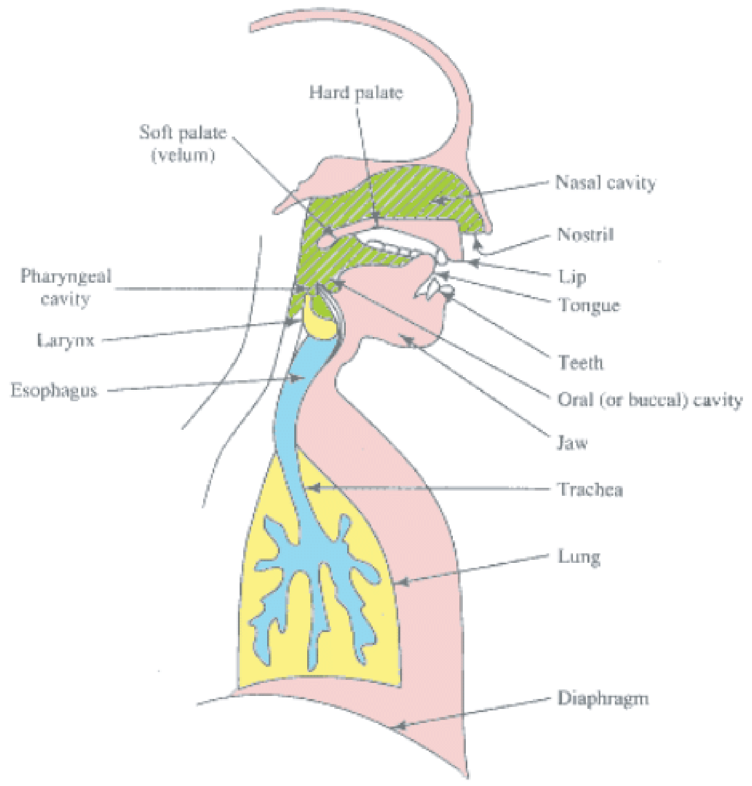


Figure 2: Schematic view of human speech production.

by inverse Fourier transform of multiplying spectral and excitation parameter vectors in frequency domain. In addition to LSP, the mel-cepstral coefficients, and LF0, other spectral and excitation features, such as “mel-generalized cepstral coefficients” [47], and aperiodicities [48]) can also be used.

2.2.2 Hidden Markov Model

Set of parameters λ in an N-state HMM can be modeled by sets of “initial-state probabilities” $\{\pi_i\}_{i=1}^{N}$, “state-transition probabilities” $\{a_{ij}\}_{i,j=1}^{N}$, and “state-output probability distributions” $\{b_i(.)\}_{i=1}^{N}$. Usually, for simplicity the $\{b_i(.)\}_{i=1}^{N}$ are typically assumed to be single “multivariate Gaussian distributions”

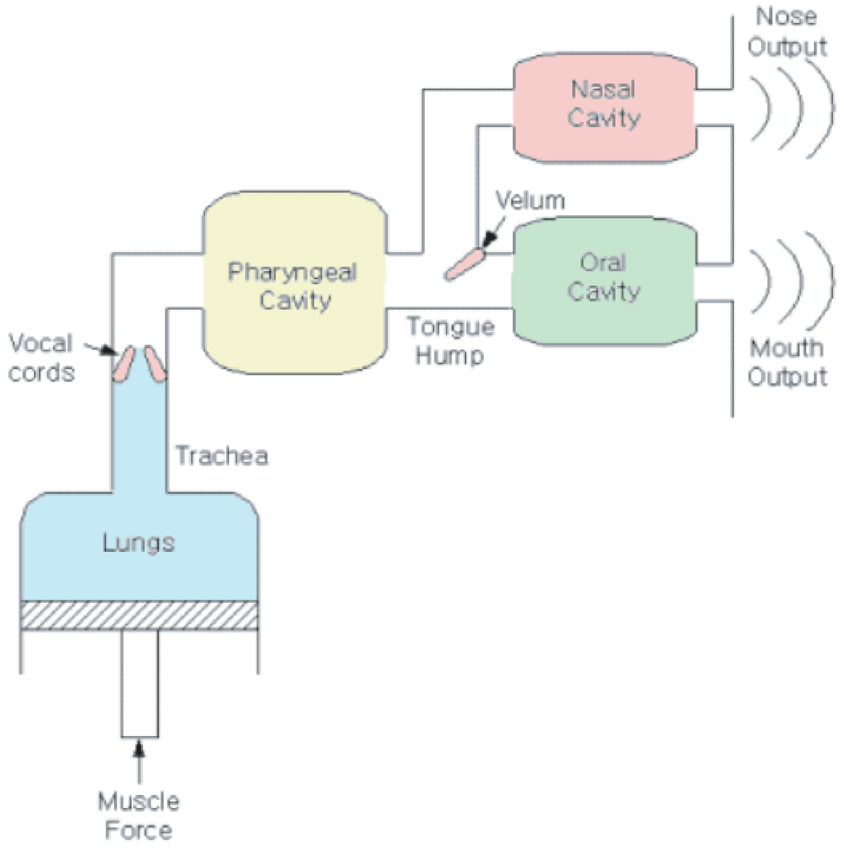


Figure 3: Block model of human speech production.

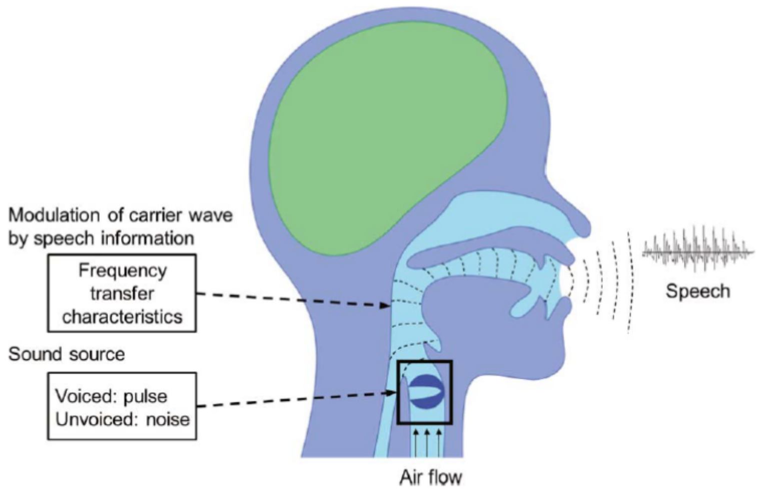

Modulation of carrier wave by speech information
Frequency transfer characteristics
Sound source
Voiced: pulse
Unvoiced: noise
Speech
Air flow

Figure 4: Overview of human speech production process.

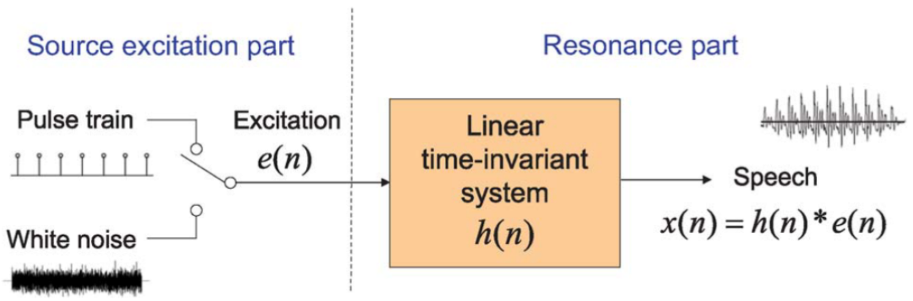


Figure 5: Source-filter model for simulating human speech production.

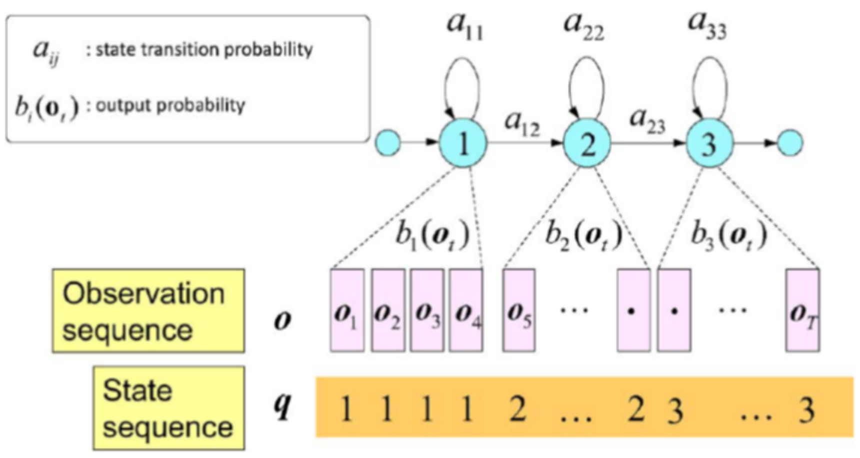

a_{ij} : state transition probability
$b_i(\mathbf{o}_t)$: output probability
a_{11}
a_{22}
a_{33}
a_{12}
a_{23}
1
2
3
$b_1(\boldsymbol{o}_t)$
$b_2(\boldsymbol{o}_t)$
$b_3(\boldsymbol{o}_t)$
Observation sequence
$\boldsymbol{o}$
$\boldsymbol{o}_1$ $\boldsymbol{o}_2$ $\boldsymbol{o}_3$ $\boldsymbol{o}_4$ $\boldsymbol{o}_5$ ... • • ... $\boldsymbol{o}_T$
State sequence
$\boldsymbol{q}$
1 1 1 1 2 ... 2 3 ... 3

Figure 6: Example of a three-state, left-to-right HMM.

$$b_i(\boldsymbol{o}_t) = \mathcal{N}(\boldsymbol{o}_t; \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i) \ = \frac{1}{\sqrt{(2\pi)^d |\boldsymbol{\Sigma}_i|}} exp\left\{ -\frac{1}{2}(\boldsymbol{o}_t - \boldsymbol{\mu}_i)^T \boldsymbol{\Sigma}_i^{-1} (\boldsymbol{o}_t - \boldsymbol{\mu}_i) \right\} \tag{3}$$

where $\boldsymbol{\mu}_i$ and $\boldsymbol{\Sigma}_i$ are mean vector and covariance matrix with the dimension of the acoustic parameters, $d \times 1$ and $d \times d$, respectively. An observation vector is shown with $\boldsymbol{o}_t$, which contains frame level vocoder parameters with frame index t. Simplified sample of a left-to-right HMM with three states, is shown in Figure 6.

The main idea in HMM-based speech synthesis is clear and causes the HMM to be a suitable model for estimating acoustic features. Let $\boldsymbol{O} = [\boldsymbol{O}_1^T, \boldsymbol{O}_2^T, ..., \boldsymbol{O}_T^T]^T$, and W be acoustic speech feature sets and commensurate linguistic features like full-context labels to be used for the training of HMMs, respectively. In this representation for synthesis time, $\boldsymbol{o} = [\boldsymbol{o}_1^T, \boldsymbol{o}_2^T, ..., \boldsymbol{o}_{T'}^T]^T$, and w are acoustic speech features and commensurate linguistic features that are extracted from input text. The training and synthesis parts of HMMs can be formulated as:

$$Training : \lambda_{max} = \operatorname*{argmax}_{\lambda} p(\boldsymbol{O}|\lambda, W) \tag{4}$$

$$p(\boldsymbol{O}|\lambda, W) = \sum_{\forall \boldsymbol{q}} \pi_{q_0} \prod_{t=1}^{T} a_{q_{t-1}q_t} b_{q_t}(\boldsymbol{O}_t) \tag{5}$$

$$Synthesis : \boldsymbol{o}_{max} = \operatorname*{argmax}_{\boldsymbol{o}} p(\boldsymbol{o}|\lambda_{max}, w) \tag{6}$$

where $\boldsymbol{q} = \{q_1, q_2, ..., q_T\}$ is a state sequence.

2.2.3 Speech Parameter Generation From HMM

The most probable estimated acoustic speech feature vector sequence using a set of HMMs and an input text to be synthesized is determined as

$$\boldsymbol{o}_{max} = \operatorname*{argmax}_{\boldsymbol{o}} p(\boldsymbol{o}|\lambda_{max}, w) \tag{7}$$

$$= \operatorname*{argmax}_{\boldsymbol{o}} \sum_{\forall \boldsymbol{q}} p(\boldsymbol{o}, \boldsymbol{q}|\lambda_{max}, w) \tag{8}$$

$$\approx \operatorname*{argmax}_{\boldsymbol{o},\boldsymbol{q}} p(\boldsymbol{o}, \boldsymbol{q}|\lambda_{max}, w) \tag{9}$$

$$= \operatorname*{argmax}_{\boldsymbol{o},\boldsymbol{q}} p(\boldsymbol{o}|\boldsymbol{q}, \lambda_{max}) P(\boldsymbol{q}|\lambda_{max}, w) \tag{10}$$

$$\approx \operatorname*{argmax}_{\boldsymbol{o}} p(\boldsymbol{o}|\boldsymbol{q}_{max}, \lambda_{max}) \tag{11}$$

$$= \operatorname*{argmax}_{\boldsymbol{o}} \prod_{t=1}^{T'} \mathcal{N}(\boldsymbol{o}_t; \boldsymbol{\mu}_{q_{max},t},\ \boldsymbol{\Sigma}_{q_{max},t}) \tag{12}$$

where

$$\boldsymbol{q}_{max} = \operatorname*{argmax}_{\boldsymbol{q}} P(\boldsymbol{q}|\lambda_{max}, w). \tag{13}$$

Using "state-duration probability distributions", the maximization problem of 13 can be solved. The maximization problem of 11 is maximizing $p(\boldsymbol{o}|\boldsymbol{q}, \lambda)$ with respect to $\boldsymbol{o}$ given the predetermined state sequence $\boldsymbol{q}_{max}$.

By choosing the sequence of the mean state vectors as the estimated speech acoustic feature vector sequence, $p(\boldsymbol{o}|\boldsymbol{q}_{max}, \lambda)$ in (3) will be maximized. Sequence of the mean vectors are step-wise. This can be result of the conditional independence of "state-output probabilities" assumed in the HMM. Human speech is continuous but this step-wise speech generation will generate discontinuities at state boundaries in a speech waveform. For alleviating this problem, in speech parameter generation algorithm, dynamic features were used as constraints. Using this constraint, the maximization problem, model the relationship between "static" and "dynamic features". Using first- and second-order time derivatives of speech parameters known as delta and delta-delta features (dynamic features) as an acoustic vector, is a powerful and efficient mechanism for sequence modeling in the HMM framework. This method shows significant improvement in the performance of "HMM-based automatic speech recognizers". We can represent the speech parameter vector $\boldsymbol{o}_t$ with the static c_t and dynamic features Δc_t as

$$\boldsymbol{o}_t = [c_t, \Delta c_t]^T. \tag{14}$$

For simple example, the dynamic feature $\Delta^2 c_t$ is removed from this equation. Dynamic features can be computed from their neighboring static features as regression coefficients, i.e.,

$$\Delta c_t = \sum_{\tau=-L}^{L} w(\tau) c_{t+\tau} \tag{15}$$

where "window coefficients" for calculating the dynamic features are $\{w(\tau)\}_{\tau=-L}^{L}$. The maximum window length L is normally set to 14. For simplicity, the most common case of Δc_t is mentioned as

$$\Delta c_t = c_t - c_{t-1}. \tag{16}$$

The observation vector $\boldsymbol{o} = [\boldsymbol{o}_1^T, ..., \boldsymbol{o}_{T'}^T]^T$ and static feature vector $\boldsymbol{c} = [c_1, ..., c_{T'}]^T$ are correlated and this correlation can be represented in a matrix form.

Using this correlation, maximizing the output probability with respect to $\boldsymbol{o}$ and $\boldsymbol{c}$ are equivalent. This can be shown in

$$\boldsymbol{c}_{max} = \underset{\boldsymbol{c}}{\operatorname{argmax}} \mathcal{N}(\mathbf{W}\boldsymbol{c}; \boldsymbol{\mu}_{\boldsymbol{q}_{max}}, \boldsymbol{\Sigma}_{\boldsymbol{q}_{max}}) \,. \tag{17}$$

Linear equations to determine the most probable static feature vector sequence can be shown by setting the partial derivative of the logarithm of (17) with respect to $\boldsymbol{c}$ to $\mathbf{0}$. This equation can be derived as

$$\mathbf{W}^T \Sigma_{\boldsymbol{q}}^{-1} \mathbf{W}\boldsymbol{c} = \mathbf{W}^T \Sigma_{\boldsymbol{q}}^{-1} \boldsymbol{\mu}_{\boldsymbol{q}} \tag{18}$$

where

$$\boldsymbol{\mu}_{\boldsymbol{q}} = [\boldsymbol{\mu}_{q_1}^T, ..., \boldsymbol{\mu}_{q_{T'}}^T]^T \tag{19}$$

$$\boldsymbol{\Sigma}_{\boldsymbol{q}} = diag[\Sigma_{q_1}, ..., \Sigma_{q_{T'}}] \tag{20}$$

Cholesky decomposition with $O(T^{'})$ operations is a suitable tool for solving the set of linear equations. Simple illustration of speech parameter generation is shown in Figure 7.

The speech trajectory of the zero-th “mel-cepstral coefficient” $c(0)$ in the estimated static and dynamic acoustic features are shown in Figure 7. Each state output is shown in rectangular blocks. In the diagonal covariance matrix condition, each state has separate mean and variance. Mean and standard deviation of each state is shown with dashed line and shaded zone, respectively.

2.2.4 Training Part of HMM-based Speech Synthesis

Overview of steps in an HMM-based speech synthesis system is shown in 9 which contains the training and synthesis parts. Maximum-likelihood estimation of the HMM coefficients using the BaumWelch algorithm is main part of HMM training.

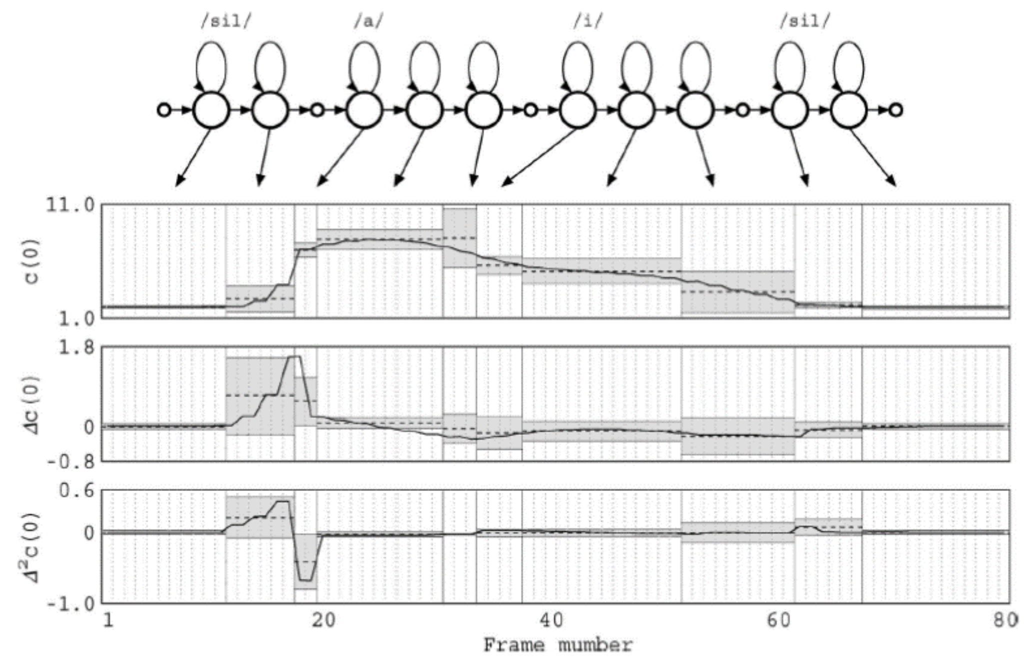


Figure 7: Simple illustration of speech parameter generation from a sentence-level HMM for /a/ and /i/ phonemes. For each state, the mean and standard deviation, are shown with dashed and shading lines, respectively.

Overall steps are similar to the method used for speech recognition. However, we can note some differences between applying HMMs in ASR and TTS systems. As an example of this difference we can mention that HMMs in ASR usually are just based on spectral parameters. These parameters can be modeled by continuous distributions. In TTS systems, HMMs must model both spectral and excitation parameters at the same time. Mel-cepstral coefficients are samples of spectral parameters and F0 can be sample of excitation parameters. Different algorithms have been used for modeling F0 sequences [49–51] and currently for combined modeling of spectral and excitation features, the HSS system uses “multi-space probability distributions” [52]. As an another example, we can mention the difference between estimation of state duration in HMM-based ASR and TTS models. In HMM-based ASR model, with increase of duration in each state, the duration probability will exponentially decrease. In this condition, controlling the temporal structure of the sequence of speech coefficients is straight-forward. On the other hand, “semi-Markov structure” is usually used in HMM-based TTS. In this case, “temporal structure” is estimated by a “Gaussian distribution” [53]. HMM-based ASR uses “phoneme information” as “linguistic contexts”, but HMM-based TTS uses various linguistic information like “lexical stress”, “pitch accent”, “tone”, and “part-of-speech” (POS) information for modeling HMMs [54]. In the HMM-based TTS, there is exponential correlation between increasing the number of contextual factors and their combinations. As a result, robustness and accuracy in estimation of “context-dependent HMM parameters” will be decreased using a few utterances as training data. For alleviating this issue, using “state tying techniques” homogeneous states are clustered and with tying model coefficients among group of context-dependent HMMs, we can increase the robustness in estimation of the model parameters. “Hierarchical tree structure” is used in the “state tying process”. Based on “minimum description length” (MDL) [55], The size of this decision tree will automatically be determined. In HMM-based TTS, using “stream-dependent decision trees” [56], the “spectral”, “excitation”, and “duration parameters” will be clustered separately. Having different “context dependency” for each of these acoustic features is the main reason for using “stream-dependent decision trees”.

2.2.5 Synthesis Part of HMM-based Speech Synthesis

The overview of synthesis part of TTS system is shown in the underneath part of Figure 9. In the first step, sequence of full-context labels are generated from input text. A “sentence-level” HMM can be generated by concatenating “context-dependent” HMMs. Using “state duration probability distribution”, the duration of each state is estimated to maximize its estimated probability. In the next step, with use of the speech parameter generation algorithm, we can estimate the sequence of acoustic features. Acoustic features, usually contain spectral and excitation parameters, are

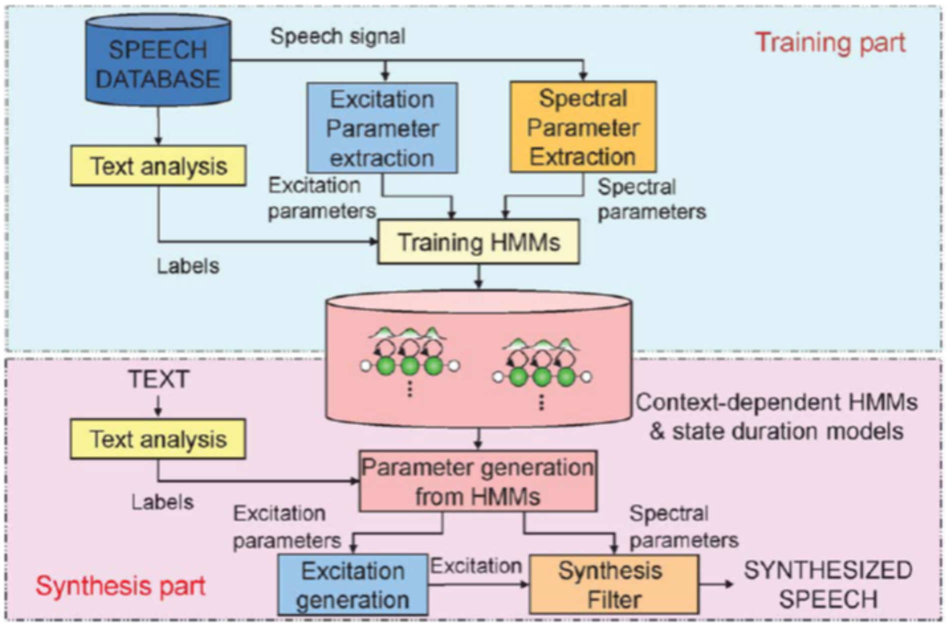


Figure 8: Overview of the HMM-based speech synthesis system.

estimated to maximize the output probability. At the end, using vocoder from the estimated acoustic parameters, synthetic speech waveform will be generated. Based on spectral features, different vocoder will be used. For “mel-cepstral coefficients” features usually the “mel-log spectral approximation” (MLSA) will be used as synthesis filter [57]. For “linear-prediction based” spectral features, all-pole filter will be used [58].

2.3 Hybrid Systems in Speech Synthesis

HSS and CSS systems come with their own pros and cons. Therefore, to use of both models together in different schemes to obtain the advantages of both techniques, TTS researchers also investigated to use both models together. Although, CSS and HSS systems are both data driven approaches, they exhibit different characteristics due to their fundamental difference in foundations. CSS systems are selection based systems. They dont aim at producing a unit if there is no proper one in the database. Hence, for covering the units for given domain, considerable amount of recorded speech must be available to the system. On the other hand, for HSS model, small amount of data is enough. Clustering and statistical estimation causes a much robust system compared to CSS in such situations. As USS systems need all data to be available in the synthesis time, they have bigger footprint with respect to statistical parametric systems.

Most important advantage of HSS is its flexibility. Parametric structure of HSS creates a very suitable infrastructure for changing the voice characteristics, emotion and speaking style. Although small footprint and flexibility are the advantage of HSS systems, because of its statistical nature, quality of synthesized speech is one of the issues in HSS systems. Although using global variance algorithm [40] alleviate the over-smoothing problem in HSS systems, because of using natural speech units, quality of CSS systems when having sufficient amount of data is significantly better than HSS systems. Researchers have been proposed many hybrid schemes and algorithms to combine the strong parts of both systems. HMM guided unit selection systems use HMMs as a guide to selection of units from database. This can be accomplished either using the HMM generated parameters directly as targets [29, 59, 60] or using the HMM likelihoods in the cost calculations [61–63]. In some other methods, for estimating the "state-output distribution" some frame samples in the current states are used. For instance, "frame-sized units" are used in the "unit-selection system" [64]. In this method, for computing the distance between the adjacent features, a "frame-wise dynamic programming" (DP) search method is used. This method is sample of maximum likelihood based approaches for generating the acoustic parameters. For modeling each "state-output distribution" the discrete HMM-based speech synthesis system is proposed in [65]. In this method, using vector quantization the distribution of each state-output will be modeled. For merging USS and HSS systems, a "frame-wise" model of "state-output distribution" was proposed in [66]. Hybrid approaches have some advantages with respect to each of USS or HSS systems. By replacing synthetic samples with natural acoustic samples in estimated samples, an over-smoothing problem is avoided. Using the natural frame-level samples, we can decrease the quality problem caused by vocoding. In addition, context dependencies in each speech parameter can be observed by complicated "cost function" which are designed using the HMM likelihoods. Using hybrid approaches have some disadvantages with respect to the HSS system, like loosing the flexibility in controlling the output voice quality, limitation in applying emotion in output speech, and "small footprint".

2.4 Speech Synthesis Using Deep Neural Network

Using "context-dependent" decision tree for clustering of HMM states is usual method for modeling probability densities of acoustic features. Although this approach is almost effective but has a couple of limitations, e.g., for modeling complex "linguistic context" dependencies, decision trees are inefficient. An alternative scheme using deep neural network (DNN) is proposed in [5]. In DNN-based method, using maximum likelihood approach (MLPG), a sequence of acoustic features will be estimated then using vocoder a speech waveform is generated from the estimated acoustic features. Input text can be converted to linguistic features and the correlation between these features and their acoustic representation can be modeled with DNN. Using DNN can eliminate some constraints of the usual HMM-based method. In DNN, sequence of non-linear activation functions will define deep architecture and causes DNN to learn non-linear correlation between linguistic features and their acoustic representations. Some features of DNN is different from decision trees. Limitation of decision trees for expressing some complicated functions like "XOR", "d-bit parity function", or "multiplex problems" in input layer [67] are samples to show the inefficiency of decision trees. DNNs are suitable structure for representing these complicated functions. Decision trees have poor generalization in parameter estimation that usually caused by using a separate set of features for each terminal node. Word-level or some other "weak" input features will thrown away while constructing decision trees [68]. But because of updating the weights from all training data, DNNs provide better generalization. In addition, in input layer of DNN, high- dimensional or diverse features can be combined. As with respect to decision trees, back-propagation step in training a DNN usually requires vast amount of computation, we need to use larger dataset for training the DNN model. In the prediction step, DNNs need a matrix multiplication at each layer but as decision trees use just a subset of input features, traversing the trees from root to the terminal nodes is enough for prediction. At the end, debugging and interpretation of decision trees are easier than DNNs as weights in hidden layers of a DNN are harder to interpret.

2.4.1 Framework of DNN-based Speech Synthesis

Transforming the information from the linguistic to the acoustic level for producing the speech in humans contains several layers [69]. In speech synthesis systems, for modeling this hierarchical structure, deep DNN-based architecture is applied. A framework of DNN-based speech synthesis is shown in Figure 9. Sequence of linguistic features $\{x_n^t\}$, where $\{x_n^t\}$ denotes the n^{th} input feature at frame t will be generated from input text. Linguistic features in DNN usually contain binary representation of sequence of phonemes (e.g. one-hot vector) and numerical features (e.g. the position of the frame in the current phoneme, the number of words and syllables in the current and adjacent phrases, and durations of the current and adjacent phonemes).

Mean while of training, DNN will find the map between the linguistic input features and their corresponding acoustic output features y_m^t. Common structure for

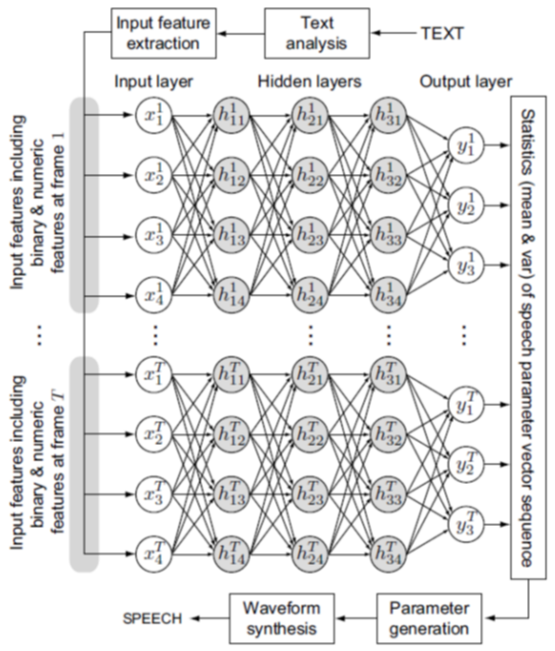

Input feature extraction
Text analysis
TEXT
Input layer
Hidden layers
Output layer
Input features including binary & numeric features at frame 1
Input features including binary & numeric features at frame T
Statistics (mean & var) of speech parameter vector sequence
SPEECH
Waveform synthesis
Parameter generation

Figure 9: Framework of DNN-based speech synthesis.

training a DNN is using feed-forward structure, where y_m^t shows the m^{th} output acoustic feature at frame t. Without using RNN, for modeling the sequence of acoustic features, output features must contains spectral and excitation parameters and their dynamic features. Using linguistic input and acoustic output features as a pair of frame-level feature, the weights of the DNN can be estimated. Using DNN structure, it is possible to generate speech parameters similar to the HMM-based approach. For covering the statistics of both static features and their first and second derivations, variances of acoustic output features will be computed from the train waveforms. In the synthesis time, the estimated output acoustic features will be assumed as mean vector containing both static and dynamic features and according to these features, the speech "parameter generation" algorithm will produce smooth trajectories of speech acoustic features. At the end, using these estimated acoustic features, vocoder will generate the synthetic waveform. Using this DNN based model, with respect to HMM based model, only the map between the "context-dependent" linguistic features and their correlated acoustic features will be changed and other parts in TTS system including text analysis for converting text to linguistic features, duration estimation, speech parameter generation, and vocoder for generating the synthetic waveform will be similar to HMM based model.

2.4.2 Speech Synthesis with BLSTM-RNN

Due to the intrinsic, feed-forward nature in DNN-based modeling, the long time span contextual effect in a speech utterance is not easy to accommodate. To synthesize a smooth speech trajectory, the dynamic features are commonly used to constrain speech parameter trajectory generation in HSS systems. To capture the correlation or co-occurrence information between any two instants in a speech utterance for SPSS, recurrent neural networks (RNNs) with bidirectional long short term memory (BLSTM) cells are proposed in [6]. The speech trajectory generated by the BLSTM-RNN speech synthesis systems is fairly smooth and usually no dynamic constraints are needed.

Recurrent Neural Network (RNN) computes hidden state vector sequence $\boldsymbol{h} = (h_1, ..., h_T)$ and output vector sequence $\boldsymbol{y} = (y_1, ..., y_T)$, for a given input vector sequence $\boldsymbol{x} = (x_1, ..., x_T)$, iterating the following equations from $t = 1$ *to* T:

$$h_t = \mathcal{H}(\mathbf{W}_{xh}x_t + \mathbf{W}_{hh}h_{t-1} + b_h) \tag{21}$$

$$y_t = \mathbf{W}_{hy}h_t + b_y, \tag{22}$$

where $\mathbf{W}$ is the weight matrices, e.g. $\mathbf{W}_{xh}$ is the weight matrix between input and hidden vectors; b is the bias vectors, e.g. b_h is the bias vector for hidden state vectors; and $\mathcal{H}$ is the nonlinear activation function for hidden nodes.

In the conventional RNNs nodes, $\mathcal{H}$ is usually a sigmoid or hyperbolic tangent function. Gradient vanishing problem caused by these activation function prevents RNN from modeling the long-span relations in sequential features. As a result, Long

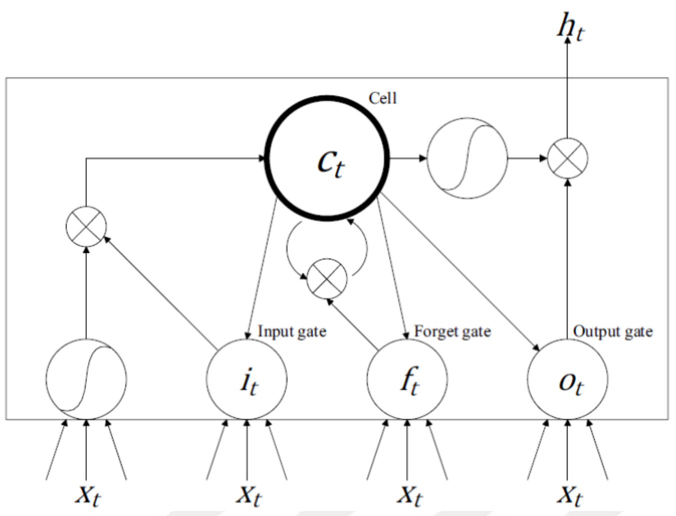


Figure 10: Long short term memory structure.

short term memory (LSTM) network [70], as shown in Figure 10 is used. LSTM which manually build a memory cell inside, can overcome the problems in conventional RNN and can model signals that have a mixture of low and high frequency components. For LSTM $\mathcal{H}$ is implemented with the following functions [71]:

$$i_t = \sigma(\mathbf{W}_{xi}x_t + \mathbf{W}_{hi}h_{t-1} + \mathbf{W}_{ci}c_{t-1} + b_i) \tag{23}$$

$$f_t = \sigma(\mathbf{W}_{xf}x_t + \mathbf{W}_{hf}h_{t-1} + \mathbf{W}_{cf}c_{t-1} + b_f) \tag{24}$$

$$c_t = f_t c_{t-1} + i_t tanh(\mathbf{W}_{xc}x_t + \mathbf{W}_{hc}h_{t-1} + b_c) \tag{25}$$

$$o_t = \sigma(\mathbf{W}_{xo}x_t + \mathbf{W}_{ho}h_{t-1} + \mathbf{W}_{co}c_t + b_o) \tag{26}$$

$$h_t = o_t tanh(c_t), \tag{27}$$

where σ is the sigmoid function; i, f, o, and c are input gate, forget gate, output gate and cell memory, respectively.

Bidirectional RNN as shown in Figure 11 can access both the preceeding and succeeding contexts. It separates the hidden layer into two parts, forward state sequence, $\overrightarrow{h}$, and backward state sequence, $\overleftarrow{h}$. The iterative process is:

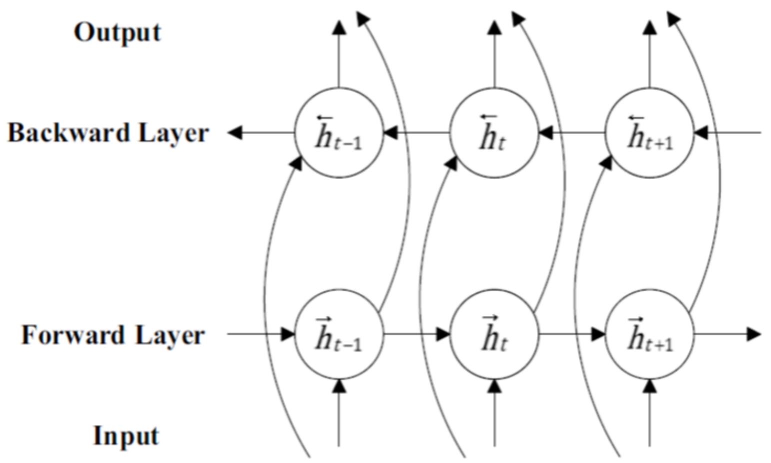

Output
Backward Layer
$\overleftarrow{h}_{t-1}$
$\overleftarrow{h}_{t}$
$\overleftarrow{h}_{t+1}$
Forward Layer
$\overrightarrow{h}_{t-1}$
$\overrightarrow{h}_{t}$
$\overrightarrow{h}_{t+1}$
Input

Figure 11: Bidirectional RNN.

$$\overrightarrow{h}_t = \mathcal{H}(\mathbf{W}_{x\overrightarrow{h}} x_t + \mathbf{W}_{\overrightarrow{h}\overrightarrow{h}} \overrightarrow{h}_{t-1} + b_{\overrightarrow{h}}) \tag{28}$$

$$\overleftarrow{h}_t = \mathcal{H}(\mathbf{W}_{x\overleftarrow{h}} x_t + \mathbf{W}_{\overleftarrow{h}\overleftarrow{h}} \overleftarrow{h}_{t+1} + b_{\overleftarrow{h}}) \tag{29}$$

$$y_t = \mathbf{W}_{\overrightarrow{h}y} \overrightarrow{h}_t + \mathbf{W}_{\overleftarrow{h}y} \overleftarrow{h}_t + b_y \tag{30}$$

Deep bidirectional RNN can be made by stacking multiple RNN hidden layers on top of each other. Each hidden state sequence h^n, is replaced by the forward and backward, $\overrightarrow{h}^n$ and $\overleftarrow{h}^n$, and the iterative process can be formulated as:

$$\overrightarrow{h}^n_t = \mathcal{H}(\mathbf{W}_{\overrightarrow{h}^{n-1}\overrightarrow{h}^n} \overrightarrow{h}^{n-1}_t + \mathbf{W}_{\overrightarrow{h}^n\overrightarrow{h}^n} \overrightarrow{h}^n_{t-1} + b^n_{\overrightarrow{h}}) \tag{31}$$

$$\overleftarrow{h}^n_t = \mathcal{H}(\mathbf{W}_{\overleftarrow{h}^{n-1}\overleftarrow{h}^n} \overleftarrow{h}^{n-1}_t + \mathbf{W}_{\overleftarrow{h}^n\overleftarrow{h}^n} \overleftarrow{h}^n_{t+1} + b^n_{\overleftarrow{h}}) \tag{32}$$

$$y_t = \mathbf{W}_{\overrightarrow{hN_y}} \overrightarrow{h}^N_t + \mathbf{W}_{\overleftarrow{hN_y}} \overleftarrow{h}^N_t + b_y \tag{33}$$

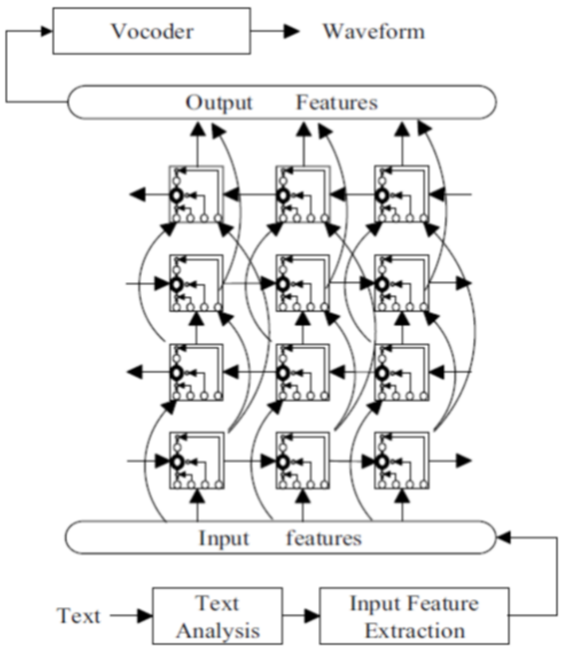


Figure 12: DBLSTM-RNN based TTS synthesis.

Deep bidirectional LSTM (DBLSTM) is the integration of deep bidirectional RNN and LSTM. By taking the advantages of DNN and LSTM, it can model the deep representation of long-span features. Human speech production process can be seen as a process to select spoken words, formulate their phonetics and then finally articulate output speech with setting the place and manner of articulators. As a result, So it is a continuous physical dynamic process. DBLSTM-RNN can simulate human speech production by a layered hierarchical and wide in time scale structure to transform linguistic text information into its final speech output. Using DBLSTM-RNN for TTS synthesis where usually a whole sentence is given as input can have a significant effect in modeling the sequence of output acoustic features. The schematic diagram of DBLSTM-RNN based TTS synthesis is shown in Figure 12.

In DBLSTM-RNN based TTS synthesis, rich contexts are also used as input features, which contain the binary features for categorical contexts, e.g. phoneme labels, part of speech (POS) labels of the current word, and TOBI labels, and numerical features for the numerical contexts, e.g., the number of words or syllable in the current phrase or the position of the current frame of the current phone. The output features are acoustic features like spectral envelope and fundamental frequency. For frame-by-frame time-alignment of input and output features, well-trained HMM model can be used. RNN is also powerful to make it possible to model sequential data where input-output alignment is unknown by Connectionist Temporal Classification [72] and Sequence Transduction [73]. To minimize the errors between the mapped output from the given input and the target output, the weights of DNN are trained by using pairs of input and output features extracted from training data. L_1, mean square error (MSE), or sum square error (SSE) can be used as a cost function in BLSTM-RNN models. Back-propagation through time (BPTT) is the most frequently used algorithm for training BLSTM-RNN. BPTT first unfolds the RNN into feed-forward network through time, and then training the unfolded network with back-propagation. For deep bidirectional LSTM, BPTT algorithm is applied to both forward and backward hidden nodes, and back-propagates layer by layer. In training time, the weights are trained by back-propagation procedure with a mini-batch based SGD algorithm, which select mini-batches of frames randomly from the whole training set. For parallel training, tens of utterances are randomly selected for each updating, then used to update the weights of RNN simultaneously.

In synthesis time, the input text will be converted into the input feature vector through text analysis, then input feature vectors are mapped to output vectors by a trained DBLSTMRNN. In DNN-based TTS, with using the estimated acoustic features as mean vectors and pre-calculated "global variances" of acoustic features from all training set and statistics of static and dynamic features, the speech feature generation can generate smooth trajectories of speech acoustic features, and voiced/unvoiced flag is determined by an empirical threshold of DNN prediction. As

RNN is powerful in sequential modeling, the parameter generation module is implicitly embedded inside the DBLSTM-RNN based TTS synthesis, i.e., the output features of DBLSTM-RNN are only the static features contain spectral envelop, gain, fundamental frequency, and V/U decision and then directly be fed into a vocoder to synthesize final speech waveform. The functions can be compactly represented with a k level deep architecture which can outperform a shallow architecture but with more weights.Sample of shallow architecture is a decision tree, which is generally used in HMM-based TTS for clustering the similar context-dependent state into tied states. Although, both RNN and DNN can overcome the limitation of conditional independence assumption in HMM, DNN only can model the relationship between text and speech on the frame level, take the limited phoneme or HMM state as input, and generate speech through the transition over the finite states. On the other hand, with the same finite states as input, RNN can take the information from neighboring frames, get different hidden states from the same input and break through the limits from input finite states.

CHAPTER III

INTRA-LINGUAL SPEAKER ADAPTATION METHODS

With respect to USS systems, SPSS systems are more flexible for changing the voice characteristics. In addition, changing speaking style like singing style speech synthesis or including emotion meanwhile of keeping small footprint can be assumed as other advantages of SPSS systems. In SPSS systems, for managing the variation in output speech several methods have been introduced like “adaptation”, “interpolation”, “eigenvoice”, and “multiple regression”. Using small amount of data from target speaker adaptation method will generate high-quality synthetic model which is similar to the target speaker’s voice. To that end, existing “speaker independent” (SI) acoustic model will be transformed to match a target speaker [42]. Starting from SI model, this method applies some adaptation methods which showed improvement in adaptation performance in ASR field like “maximum-likelihood linear regression” (MLLR) [74, 75] to adapt the HMMs states of SI model to a target speaker or to a new speaking style or emotion. The SI model is a speaker independent HMM model which is trained using suitable amount of training data. Usually, variation between speakers’ acoustic features can be normalized using “speaker-adaptive training” (SAT) method [76]. In this section, we will review the common methods of adaptation in the intra-lingual SPSS systems.

3.1 MLLR and CMLLR

Mismatch between acoustic features of train and test sets is one of the major issues in ASR systems. Using adaptation technique like MLLR showed significant improvement in performance of ASR systems. In MLLR method, a collection of linear transforms for mapping the density functions from existing SI model to new adapted model will be estimated. These density functions are usually assumed to came from Gaussian distribution. Because of lack of adaptation data from target speaker, Gaussian components will be clustered in a regression class tree. Acoustic similarity of HMM states' component is used as clustering criterion and each cluster shares the same MLLR transform [77]. Similarly, in HSS systems speaker adaptation has significant improvement in performance of adapted model. Using this adaptation technique, HSS systems are able to generate high-quality TTS model for target speaker when small amount of data from the target speaker is available. In this case, small amount of adaptation data (57 minutes) from target speaker is sufficient for generating a personalized HMM-based synthetic voice. Using this feature, for any target speaker, synthetic voice model can be generated with transformation of HMM states' parameters of SI model [78]. Adaptation methods which are used for HSS are almost identical to adaptation methods in ASR. MLLR and "maximum a posteriori" (MAP) estimation [79] are samples of adaptation methods in ASR. In MLLR algorithm, mean vectors of the "state output probability distributions" will transforms linearly in (3)

$$b_i(\boldsymbol{o}_t) = \mathcal{N}(\boldsymbol{o}_t; \boldsymbol{\zeta}_k\boldsymbol{\mu}_i + \boldsymbol{\epsilon}_k,\, \boldsymbol{\Sigma}_i) \tag{34}$$

where $\boldsymbol{\zeta}_k$ is transformation matrix of the k^{th} regression class and $\boldsymbol{\epsilon}_k$ is estimation error of this transformation. In constrained MLLR (CMLLR), in addition of mean vector, covariance matrices of the "state output probability distributions" will be transformed using the same matrices [80] and the total transformation can be formulated as:

$$b_i(\boldsymbol{o}_t) = \mathcal{N}(\boldsymbol{o}_t; \boldsymbol{\zeta}_k\boldsymbol{\mu}_i + \boldsymbol{\epsilon}_k,\, \boldsymbol{\zeta}_k\boldsymbol{\Sigma}_i\boldsymbol{\zeta}_k^T). \tag{35}$$

For clustering the HMMs into regression classes, linguistic similarity of the adaptation data (e.g. vowel, consonant, etc) was used. Kullback-Leibler Divergence

(KLD) [81] in acoustic space would reach to the better result. Number of regression classes are proportional to adaptation data size. Cut is defined in a tree where there is no longer enough data for that branch of tree. Full and diagonal transforms were tested. Full transforms perform better but they need more data to train. Performance of diagonal improved by increasing the number of classes but never reached to the best performance of full transformation. Tree-based MLLR algorithm is illustrated in Figure 13.

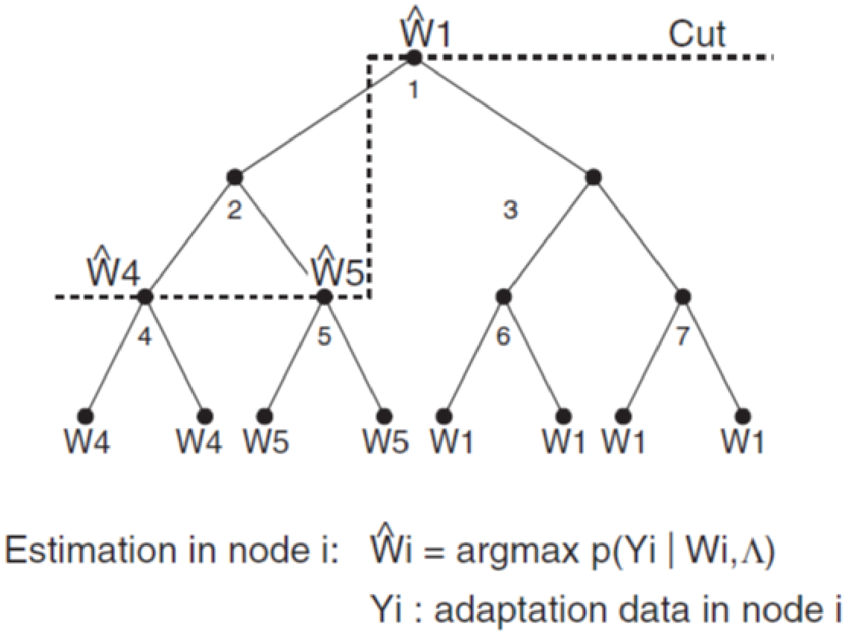

$\hat{W}1$
Cut
1
2
3
$\hat{W}4$
$\hat{W}5$
4
5
6
7
W4 W4 W5 W5 W1 W1 W1 W1
Estimation in node i: $\hat{W}i = \text{argmax } p(Yi \mid Wi, \Lambda)$
Yi : adaptation data in node i

Figure 13: MLLR algorithm with tree structure. Based on minimum size of adaptation data in each leaf node, the threshold for making a regression class will be defined. Each regression class contains all node starting from the threshold line and separate transformation matrix will be defined for each regression class.

3.2 SMAPLR and CSMAPLR

Although, the estimation criterion used to estimate transformation parameters has been mainly based on maximum likelihood (ML) estimation, Bayesian versions of some of the most popular transformation-based adaptation methods like MAPLR, a maximum a posteriori (MAP) based version of the MLLR is proposed to constraint the parameter estimation in order to obtain reliable estimation with a limited amount of data. This prior knowledge, not only prevent overfitting the adaptation data but also allow integration of this knowledge into transformation-based adaptation techniques. When bigger set of adaptation data is used, it is necessary to estimate a larger amount of transformation coefficients. It is also required to define a large number of prior densities for these parameters. Robust estimation of these prior densities is therefore a crucial issue that directly affects the efficiency and effectiveness of the Bayesian techniques. For estimating these priors, "structural maximum a posteriori linear regression" (SMAPLR) algorithm was introduced in [82]. The hierarchical priors, embedded into the tree structure is used to control transformation complexity. Using SMAPLR, for estimating the transformation in each regression cluster, MAP estimation instead of ML estimation can be used. In SMAPLR, usually prior density of the root node $p(\mathbf{W}_1)$ will be estimated as identity transformation, after that, for the first level of child nodes $p(\mathbf{W}_1|\mathbf{Y}_1)$ will be estimated as prior density. This prior/posterior propagation will follow the same rule down to the cut nodes which are defined based on pre-defined threshold. SMAPLR algorithm using tree structure is illustrated in Figure 14.

Similar to SMAPLR, in the CSMAPLR method, a Bayesian approach is used to estimate the constrained linear regression parameters which is especially useful when there is limited amount of training data [42]. In this approach

$$\hat{\Lambda} = \operatorname*{argmax}_{\Lambda} p(x|\lambda, \Lambda)p(\Lambda), \tag{36}$$

where $p(\Lambda)$ is the prior distribution of the transformation parameters Λ, and λ is the parameter set of the Gaussians in the SPSS model. Because prior is taken into account in estimation, parameter over-fitting because of data sparsity can be eliminated with the CSMAPLR algorithm. However, success of the algorithm depends

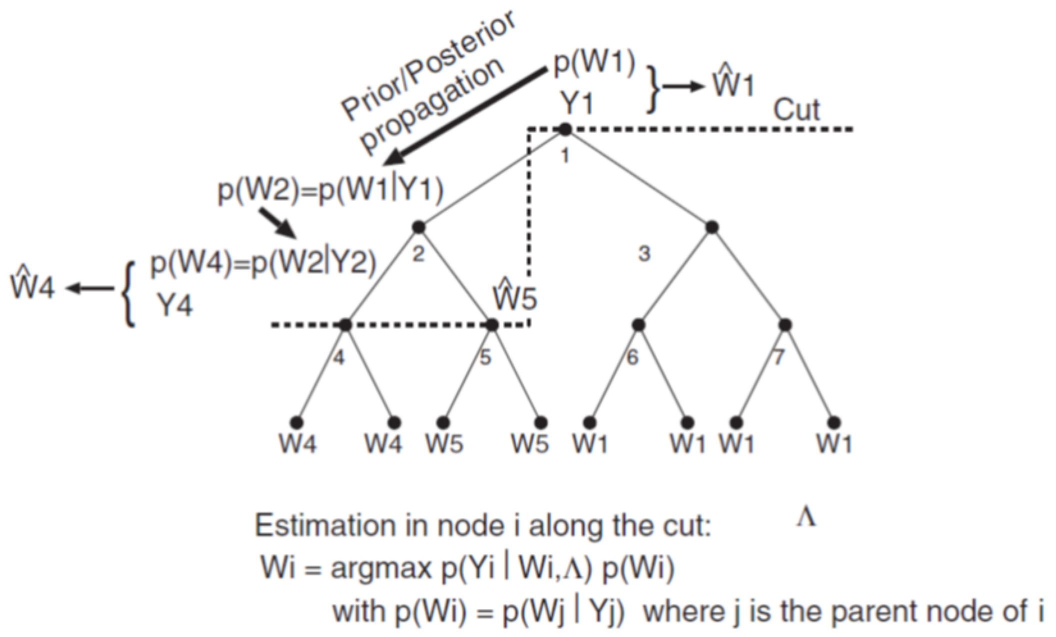
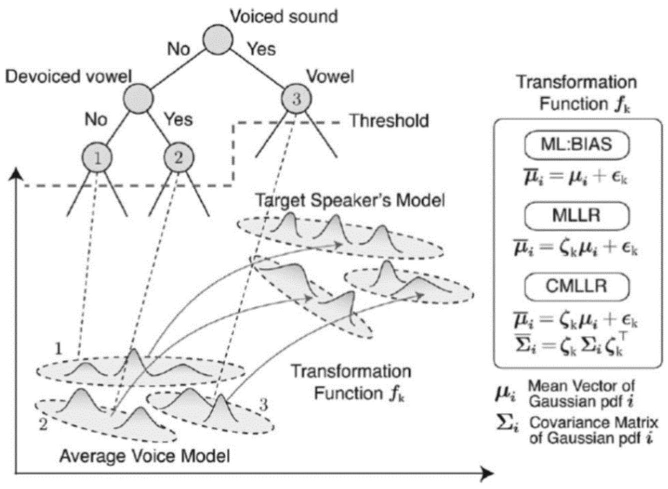

Prior/Posterior propagation
p(W1)
Y1
Ŵ1
Cut
1
p(W2)=p(W1|Y1)
2
3
Ŵ4
p(W4)=p(W2|Y2)
Y4
Ŵ5
4
5
6
7
W4
W4
W5
W5
W1
W1
W1
W1
Estimation in node i along the cut:
Wi = argmax p(Yi | Wi,Λ) p(Wi)
with p(Wi) = p(Wj | Yj) where j is the parent node of i

Figure 14: SMAPLR algorithm using tree structure. For each node i the adaptation data $\mathbf{Y}_i$ is associated with prior density $p(\mathbf{W}_i)$. In the SMAPLR structure, for each child node i of parent node j, the prior density $p(\mathbf{W}_i)$ will be defined as $p(\mathbf{W}_j|\mathbf{Y}_j)$.

on the proper selection of the prior distribution and its hyper-parameters. In the CSMAPLR approach, priors are estimated using a hierarchical approach embedded into a tree structure. Imposing such a structure on prior estimation allows reliable estimation of the prior distribution which is especially important in limited adaptation data case since the posterior relies more on the prior pdf than the likelihood function in that case. Similar to CMLLR, in CSMAPLR adaptation, using the same transformation matrices, in addition to mean vectors, covariance matrices of the state output probability distributions will be adapted. Mean and covariance matrices of speaker adaptation methods in HSS systems are shown in Figure 15.



Voiced sound
No
Yes
Devoiced vowel
Vowel
No
Yes
Threshold
Target Speaker's Model
Transformation Function f_k
ML:BIAS
$\overline{\mu}_i = \mu_i + \epsilon_k$
MLLR
$\overline{\mu}_i = \zeta_k \mu_i + \epsilon_k$
CMLLR
$\overline{\mu}_i = \zeta_k \mu_i + \epsilon_k$
$\overline{\Sigma}_i = \zeta_k \Sigma_i \zeta_k^\top$
Transformation Function f_k
μ_i Mean Vector of Gaussian pdf i
Σ_i Covariance Matrix of Gaussian pdf i
Average Voice Model

Figure 15: Overview of adaptation methods for HSS systems.

3.3 Eigenvoice Adaptation

Eigenvoice approach has been used for rapid adaptation in speech recognition and SPSS [83,84]. The idea is to find a set of R vectors in the high-dimensional space $\mathbb{R}^n$ ($n >> R$) that can be used to approximate a set of vectors in $\mathbb{R}^n$ by optimizing a distance measure. One way to accomplish this is using principal components analysis (PCA) that finds the directions in $\mathbb{R}^n$ where the data has the highest variance and the L_2 norm of the approximation error is minimum after projection. Solution with PCA are the eigenvectors of the sample covariance matrix with the highest eigenvalues.

In the context of SPSS, each eigenvector is called an eigenvoice. The supervector for speaker s can be created by $\boldsymbol{\mu}^{(s)} = [\boldsymbol{\mu}_1^{(s)} \boldsymbol{\mu}_2^{(s)} ... \boldsymbol{\mu}_{N_{st}}^{(s)}]$ where N_{st} is the total number of states in all decision trees in the acoustic model. In the eigenvoice approach, given a set of R eigenvectors $\boldsymbol{e}_r \in \mathbb{R}^n$, the main supervector for speaker s is shown as

$$\boldsymbol{\mu}^{(s)} = \boldsymbol{\mu}_{\text{SI}} + \boldsymbol{E}\boldsymbol{w}_s + \boldsymbol{\epsilon}_s \tag{37}$$

where $\boldsymbol{E} = [\boldsymbol{e}_1 \ \boldsymbol{e}_2 \ ... \ \boldsymbol{e}_R]$, $\boldsymbol{w}_s$ is weight vector of the speaker s, and $\boldsymbol{\epsilon}_s$ is the approximation error. Although $\boldsymbol{E}$ can be found by using the PCA method, it can also be estimated from the training data using a maximum-likelihood (ML) approach. One popular algorithm to do that is the Cluster Adaptive Training (CAT) technique which is really an adaptive training algorithm but can also be used for eigenvoice adaptation. In the case of CAT, columns of $\boldsymbol{E}$ can be seen to represent the clusters in the training data, and the weights for a given speaker are the interpolation factors between those clusters.

An iterative algorithm is proposed in [85] for learning $\boldsymbol{E}$ from a training dataset. In the first step, $\boldsymbol{E}$ is initialized randomly. Then, weights are estimated using a maximum-likelihood approach for each speaker. Using those estimated weights, $\boldsymbol{E}$ is re-estimated and the whole procedure is repeated until convergence.

Although the algorithm in [85] is similar to the Expectation-Maximization (EM) algorithm, it is not EM because posterior distribution of the weights, hence the uncertainty in the weights, are not taken into account in the iterations. This can cause problems especially when there is insufficient data for some of the speakers. The algorithm proposed in [86] solves the problem by offering an exact EM solution. Here, the algorithm proposed in [85] is used for training $\boldsymbol{E}$ since there is sufficient data for each speaker during training.

In the ML-based CAT approach, given some adaptation data
$\chi_a = \{\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, ..., \boldsymbol{x}^{(N_{o,s})}\}$, $N_{o,s}$ is the total amount of observations from speaker s, the likelihood function

$$p(\chi_a|\boldsymbol{w}_s, \boldsymbol{E}) \propto exp(-\frac{1}{2}\sum_{c=1}^{N_{st}}\sum_{i=1}^{N_c^{(s)}}(\boldsymbol{x}_c^{'(i)} - \boldsymbol{E}_c\boldsymbol{w}_s)^T\boldsymbol{\Sigma}_c^{-1}(\boldsymbol{x}_c^{'(i)} - \boldsymbol{E}_c\boldsymbol{w}_s)) \qquad (38)$$

where $\boldsymbol{E}_c$ is the cth block of the $\boldsymbol{E}$ matrix corresponding to state c, $\boldsymbol{x}_c^{'(i)} = \boldsymbol{x}_c^{(i)} - \boldsymbol{\mu}_c$, $\boldsymbol{x}_c^{(i)}$ is ith observation that is aligned with state c, $\boldsymbol{\mu}_c$ and $\boldsymbol{\Sigma}_c$ are the speaker independent mean vector and covariance matrix of the Gaussian emission pdf of state c, and $N_c^{(s)}$ is the number of observations aligned with state c for speaker s.

After removing terms that are independent of $\boldsymbol{w}$ and $\boldsymbol{E}$, the objective function

$$O \;=\; (-\tfrac{1}{2}\textstyle\sum_{c=1}^{N_{st}} S_{xx,c}) + \boldsymbol{w}_s^T \boldsymbol{G}_w^{(s)} \boldsymbol{w}_s - \boldsymbol{w}_{\boldsymbol{s}}^T \tfrac{1}{2} \boldsymbol{k}_w^{(s)} \tag{39}$$

where

$$S_{xx,c} = \sum_{i=1}^{N_c^{(s)}} \boldsymbol{x}_c^{\prime(i)T} \boldsymbol{\Sigma}_c^{-1} \boldsymbol{x}_c^{\prime(i)}. \tag{40}$$

$$\boldsymbol{G}_w^{(s)} = \sum_{c=1}^{N_{st}} N_c^{(s)} \boldsymbol{E}_c^T \boldsymbol{\Sigma}_c^{-1} \boldsymbol{E}_c \tag{41}$$

$$\boldsymbol{k}_w^{(s)} = \sum_{c=1}^{N_{st}} \boldsymbol{E}_c^T \boldsymbol{\Sigma}_c^{-1} \boldsymbol{S}_{x,c}^{(s)} \tag{42}$$

$$\boldsymbol{S}_{x,c}^{(s)} = \sum_{i=1}^{N_c^{(s)}} \boldsymbol{x}_c^{\prime(i)} \tag{43}$$

For maximize the likelihood function, EM method can optimize both $\boldsymbol{E}$ matrix and $\boldsymbol{w}_s$ vector simultaneously. An alternative method is using iterative algorithm [85]. In this method, weight vector of speaker s, $\boldsymbol{w}_s \in \mathbb{R}^R$, is calculated as follows . Firstly, $\boldsymbol{E}$ is fixed to a constant matrix. Then, the objective function is maximized with respect to $\boldsymbol{w}_s$ and

$$\hat{\boldsymbol{w}}_{\text{CAT}} = G_w^{(s)^{-1}} k_w^{(s)} \tag{44}$$

Once the $\boldsymbol{w}_s$ vectors are computed for each speaker, they can be fixed, and the $\boldsymbol{E}$ matrix can be estimated. In this case, the objective function

$$\sum_{s=1}^{S} (-\frac{1}{2} \sum_{c=1}^{N_{st}} S_{xx,c}) + \boldsymbol{w}_s^T \boldsymbol{G}_w^{(s)} \boldsymbol{w}_s - \boldsymbol{w}_{\boldsymbol{s}}^T \frac{1}{2} \boldsymbol{k}_w^{(s)} \tag{45}$$

is maximized with respect to $\boldsymbol{E}_c$, and

$$\hat{\boldsymbol{E}}_{c,\text{CAT}} = \boldsymbol{G}_c^{-1} \boldsymbol{K}_c \tag{46}$$

where

$$\boldsymbol{G}_c = \sum_{s=1}^{S} N_c^{(s)} \boldsymbol{w}_s \boldsymbol{w}_s^T \tag{47}$$

$$\boldsymbol{K}_c = \sum_{s=1}^{S} \sum_{i=1}^{N_c^{(s)}} \boldsymbol{w}_s \boldsymbol{x}_c^{\prime(i)T} \tag{48}$$

and S is the total number of speakers. The new estimate of $\boldsymbol{E}$ can then be used to estimate $\boldsymbol{w}_s$ for all training speakers. Estimates of $\boldsymbol{E}$ and $\boldsymbol{w}_s$ can be improved with more iterations until convergence.

Instead of estimating $\hat{w}$ from maximum likelihood approach, maximum a posteriori estimation is proposed in [87]. In this approach, E matrix is trained using the CAT procedure described above. However, in the BCAT approach, the weight vector for a target speaker s is estimated with the objective function

$$\hat{\boldsymbol{w}}_{\text{BCAT}} = \underset{\boldsymbol{w}}{\text{argmax}}\, p(\chi_a|\boldsymbol{w})p(\boldsymbol{w}) \tag{49}$$

where $p(\boldsymbol{w})$ is the prior distribution and set to $\mathcal{N}(0, \boldsymbol{\Sigma}_w)$ here. Thus,

$$p(\boldsymbol{w}) = \frac{1}{\sqrt{(2\pi)^R|\boldsymbol{\Sigma}_w|}} exp\left(-\frac{1}{2}\boldsymbol{w}^T\boldsymbol{\Sigma}_w^{-1}\boldsymbol{w}\right). \tag{50}$$

The suggested estimator is not "fully Bayesian" because a "point estimate" of $\boldsymbol{w}$ is found. As prior distribution was used in the estimation time, the term "Bayesian" is employed here.

After removing the terms that are independent of $\boldsymbol{w}$ from the objective function, using (38) to replace the likelihood term $p(\chi_a|\boldsymbol{w}_s)$ and with some matrix manipulation, the BCAT objective function becomes

$$\hat{\boldsymbol{w}}_{\text{BCAT}} = \underset{\boldsymbol{w}}{\text{argmax}}\, exp\Bigg(\boldsymbol{w}^T\boldsymbol{E}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{S}_x - \frac{1}{2}\boldsymbol{w}^T\boldsymbol{E}^T\boldsymbol{N}\boldsymbol{\Sigma}^{-1}\boldsymbol{E}\boldsymbol{w}\Bigg) exp\left(-\frac{1}{2}\boldsymbol{w}^T\boldsymbol{\Sigma}_w^{-1}\boldsymbol{w}\right). \tag{51}$$

where the block diagonal $\boldsymbol{\Sigma}^{-1} = diag(\boldsymbol{\Sigma}_1^{-1}, \boldsymbol{\Sigma}_2^{-1}, ..., \boldsymbol{\Sigma}_{N_{st}}^{-1})$, $\boldsymbol{S}_x = [\boldsymbol{S}_{x,1}, \boldsymbol{S}_{x,2}, ..., \boldsymbol{S}_{x,N_{st}}]$, and $\boldsymbol{N} = diag(N_1\boldsymbol{I}_{F\times F}, N_2\boldsymbol{I}_{F\times F}, ..., N_{N_{st}}\boldsymbol{I}_{F\times F})$ where size of the feature vectors are shown with F.

As posterior distribution $p(\boldsymbol{w}|\chi_a)$ is a Gaussian since this Gaussian distribution is the "conjugate prior" of the Gaussian likelihood function with unknown mean in (38) the objective function can be maximized. So, from (49) we can reach to

$$\hat{\boldsymbol{w}}_{\text{BCAT}} = \underset{\boldsymbol{w}}{\operatorname{argmax}}\, exp\left(-\frac{1}{2}(\boldsymbol{w} - \boldsymbol{\mu}_{w|\chi})^T \boldsymbol{\Sigma}_{w|\chi}(\boldsymbol{w} - \boldsymbol{\mu}_{w|\chi})\right) \tag{52}$$

By completing the squares and using (51),

$$\boldsymbol{\Sigma}_{w|\chi} = (\boldsymbol{E}^T \boldsymbol{N} \boldsymbol{\Sigma}^{-1} \boldsymbol{E} + \boldsymbol{\Sigma}_w^{-1}), \tag{53}$$

and

$$\boldsymbol{\mu}_{w|\chi} = \boldsymbol{\Sigma}_{w|\chi}^{-1} \boldsymbol{E}^T \boldsymbol{\Sigma}^{-1} \boldsymbol{S}_x. \tag{54}$$

BCAT estimate of $\boldsymbol{w}$, $\hat{\boldsymbol{w}}_{\text{BCAT}}$, is the mean $\boldsymbol{\mu}_{w|\chi}$ of the posterior distribution. Hence,

$$\hat{\boldsymbol{w}}_{\text{BCAT}} = (\boldsymbol{E}^T \boldsymbol{N} \boldsymbol{\Sigma}^{-1} \boldsymbol{E} + \boldsymbol{\Sigma}_w^{-1})^{-1} \boldsymbol{E}^T \boldsymbol{\Sigma}^{-1} \boldsymbol{S}_x. \tag{55}$$

$\boldsymbol{\Sigma}_w^{-1}$ is the hyperparameter of the prior distribution. It is used to enforce the adaptation algorithm to move more in specific directions compared to other directions. The idea is to learn the typical directions in the speaker space that the speaker-dependent models move during adaptation and use that as a prior information in adaptation when minimal observations are available.

3.4 VTLN and Count Smoothing

One of the techniques for adaptation when little amount of adaptation data is available is Vocal tract length normalization (VTLN) [88]. As major components of VTLN we can mention a "warping function", a "warping factor" and an "optimization criterion". For showing the ratio of the VTL of the target speaker to the mean VTL, a variable called α is defined in "bilinear transform based warping function". For calculating the "warping factor" for each target speaker, an ML-based technique using grid search is used. ML optimization is given by:

$$\hat{\alpha}_s = \underset{\alpha}{\operatorname{argmax}}\, p(\chi_{\alpha_s}|\Theta, w_s) p(\alpha|\Theta) \tag{56}$$

where the features for the speaker s which is shown with χ_{α_s} is warped with the warping factor α_s. The model is shown with Θ parameter and linguistic transcriptions correlated to the target speaker's data are shown with w_s and $\hat{\alpha}_s$ shows the best "warping factor" for the current target speaker. The prior probability of the warping factor α is shown with $p(\alpha|\Theta)$.

Several rapid speaker adaptation methods use prior information to find robust transforms when a few utterance of adaptation data is available. In [89], using some rapid adaptation methods like VTLN or PCMLLR [90], first initial adaptation is done for the target speaker. Then, for smoothing the statistics for computing the transformation matrices in CMLLR, result of the previous rapid adaptation algorithm will be used. This method of statistics smoothing is called "Count Smoothing".

VTLN can be combined with the CSMAPLR algorithm. This approach improves the performance of the CSMAPLR algorithm in case of small adaptation data like 1 utterance. Usually, the top global transformation matrix in CSMAPLR, is calculated either using a maximum likelihood (ML) estimation or using a MAP approach with an identity matrix as a prior. In [91], however, VTLN transformation is used as a prior for the top transformation estimation. The VTLN transform can be viewed as

$$\boldsymbol{x}_\alpha = \mathbf{A}_\alpha \boldsymbol{x} \tag{57}$$

where $\boldsymbol{x}_\alpha = (\hat{x}_1, ..., \hat{x}_M)^T$ and $\boldsymbol{x} = (x_1, ..., x_L)^T$ are the warped and original observation vectors if we truncate them at M-th and L-th dimensions, respectively. $\mathbf{A}_\alpha$ is defined as

$$A_{ml}(\alpha) = \frac{1}{(l-1)!} \sum_{n=max(0,l-m)}^{l} \begin{pmatrix} l \\ n \end{pmatrix} \times \frac{(m+n-1)!}{(m+n-l)!} (-1)^n \alpha^{2n+m-l} \tag{58}$$

where $A_{ml}(\alpha)$ is the m-th row and the l-th column element of warping matrix $\mathbf{A}_\alpha$ and α is the warping factor. $\mathbf{A}_\alpha$ may also be directly applied to the dynamic features, where the transformation matrix is block diagonal with repeating $\mathbf{A}_\alpha$ matrix:

$$\mathbf{B}_\alpha = \begin{bmatrix} \mathbf{A}_\alpha & 0 & 0 & 0 \\ 0 & \mathbf{A}_\alpha & 0 & 0 \\ 0 & 0 & \mathbf{A}_\alpha & 0 \end{bmatrix} \tag{59}$$

In the SMAP criterion in the CSMAPLR estimation, the top transformation matrix, Λ_1, is calculated either using an ML estimation or a MAP estimation with identity matrix as the prior. Then, Λ_1 is used as the $\mathbf{B}$ hyper-parameter in MAP estimation of Λ_2 (refer to (38)). In [91], $\mathbf{B}_\alpha$ is used as the $\mathbf{B}$ hyper-parameter in MAP estimation of the top transformation matrix, Λ_1.

This approach has been tested in matched and unmatched conditions of speaker adaptation and also in both TTS and continuous speech recognition (CSR) setups. This method has also been compared to a cascade algorithm where a VTLN adaptation is done and then its output is used as an SI model for the CSMAPLR algorithm. The cascade algorithm showed no significant difference compared to the original CSMAPLR algorithm. Results showed significant improvements compared to the CSMAPLR with no prior especially in the cases of less than 10 utterances. With respect to using CSMAPLR without the VTLN prior, applying VTLN will improve the naturalness and intelligibility of HSS systems. In addition, this method improved the performance of the HMM-based ASR models. Specially in mismatched conditions, in some aspects like age, gender, and recording environments, applying this method showed significant improvements in performance of adaptation.

3.5 DNN-Based Adaptation

In recent years, using deep neural networks (DNNs) as a mapping function between linguistic and acoustic features, showed significant improvement in performance of SPSS systems. Experimental analysis of DNN-based adaptation for SPSS systems at several layers in DNN is discussed in [22]. In general, speaker adaptation for DNN-based speech synthesis can be done in three different levels. In input layer, in hidden layers, and in output layer. In input layer, speaker codes are augmented as input to neural nets, in the middle layer direct modification of neural network coefficients is applied to generate the adapted model. In the output layer, feature space transformations are performed for adapting to the target speaker. DNN adaptation in three different ways are shown in Figure 16.

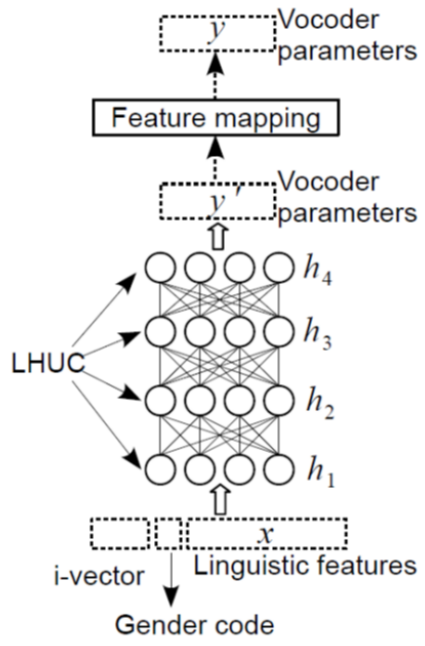


Figure 16: Three ways to do speaker adaptation for DNN-based speech synthesis. LHUC=Learning Hidden Unit Contributions.

3.5.1 Adaptation in Input Layer

One of the methods for adapting to the target speaker is using i-vectors for estimating speaker codes [18]. For showing the identity of each target speaker in a vector while pool of training speakers exists, i-vector is proposed in [18]. Using i-vector significantly improved the equal error rate (EER) performance of "text-independent speaker verification" systems [18]. Using pool of training speakers, first speaker-independent model will be trained. For the target speaker, the corresponding mean supervector s can be shown as,

$$\boldsymbol{s} \approx \boldsymbol{m} + \mathbf{T}\boldsymbol{i}, \quad \boldsymbol{i} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}), \tag{60}$$

where mean supervector of all GMM coefficients of average voice model is shown with $\boldsymbol{m}$ which is called "universal background model" (UBM) which is trained using pool of training speakers, $\boldsymbol{s}$ is the adapted speaker's mean super-vector which is adapted from UBM model with applying the interpolation of $\boldsymbol{i}$ which is the speaker identity vector (i-vector) to the total variability space matrix $\mathbf{T}$ which is estimated from pool of training data. In DNN-based speech synthesis, meanwhile of training average voice model (AVM), frame-level linguistic features will concatenated with estimated i-vector of current speaker. This concatenation is usually done for applying the speaker's identity. In this case, detection of speaker's identity of target or training speakers while the same linguistic context is observed will be possible. After training the model, when we have target speaker's adaptation data, using total variability space $\mathbf{T}$ matrix from (60), target speaker's identity vector (i-vector) will be estimated. Then, for generating target speaker's voice model, the estimated i-vector will be concatenated to frame-level linguistic features which usually contain binary features for representation of sequence of observed phonemes (as one-hot vector) and numerical features for representation of suprasegmental features, and this concatenated vector will be used as input layer to adapt the target speakers voice model. For mapping to the normalized space, "length normalization" is applied on all the i-vectors [92].

Another method for modifying synthetic speech characteristics based on the input codes, is using data-driven numerical encoding sometimes called discriminant condition codes (DCC) [19]. DCCs were initially introduced by Xue et al. [19] for speech recognition. Discriminant codes constitute hidden-unit activations obtained by projecting one-hot speaker codes into K dimensions using a matrix $\mathbf{W}_{DCC} \in \mathbb{R}^{N \times K}$. The DCCs are trained jointly with the weights and biases of a DNN. For distinguishing sentences in synthesis, Watts et al. proposed the DCC based solution [93] and here the equivalent code for distinguishing the speakers was used. In input layer, DCC codes can be combined with gender and age codes [20]. Adding these codes, in addition of improving the adaptation performance, will have some new applications such as morphing two speakers voice and manipulating the speaker codes for generating new speaker. Schematic representation of discriminant codes in speaker adaptation is shown in Figure 17.

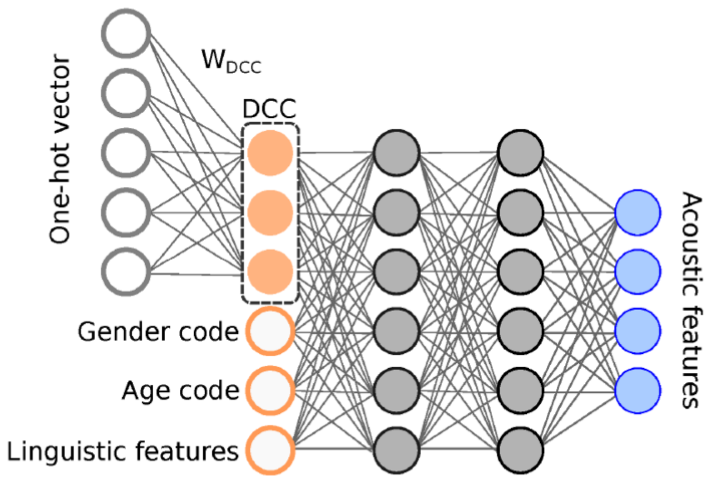


Figure 17: Schematic representation of discriminant codes. W_{DCC} is a projection matrix that reduces one-hot vectors to DCC vectors.

3.5.2 Adaptation in Hidden Layers

When we train an SI model using DNN-based structure, for modeling the non-linear correlation between input and acoustic output layers, hierarchical structure of hidden units with non-linear activation function will be used. In the training step, with monitoring specific patterns in its input, each unit can treat just like a specific function that can be adapted. In the training time, single objective (e.g., MSE, L_1, or L_2 distances), will control the learning steps of the units in all layers of the DNN model. In the training step, because of correlation between the units, all the units in overall structure will be trained and learn the joint representation of the problem that need to be solved. It means, in training time, for explaining different patterns in the training data, all the hidden units are biased and agitated to complement each other. Unfortunately in the adaptation time, this optimality of "relative importance" of specific units for the target speaker's data may no longer be valid. With rescaling the cooperation of some property of hidden units like amplitudes, without changing their feature receptors, LHUC will adapt to the target speaker. For limiting the number of comparisons for evaluation tests amplitude re-parametrization can be eliminated from the learning procedure [22]. As an advantage of LHUC, we can mention the "model-agnostic" property of it. It means, this model can work with any kind of non-linearities and architectures [94]. "Interpolation within pooling regions" is an alternative method for re-weighting the hidden units [95]. But this method needs the certain operations which are differentiable to be implemented in all layers. In other word, this method is "model dependent".

For fast adaptation in DNN, in addition of applying speaker codes in input layer, it is possible to apply it in some hidden layers [96]. Using a multi speaker speech corpus, weights for connection of several layers of DNN will be trained. In synthesis time, first the speaker code must be chosen. If the selected target speaker exists in the training corpus, the speaker code will be computed directly from the training corpus. In the case of unseen target speakers some estimation methods like (i-vector) can be used. After that, with selecting the speaker code and applying it to the DNN layers, target speaker's voice will be generated. The architecture of DNNs using speaker code in hidden layer(s) is shown in Figure 18.

3.5.3 Adaptation in Output Layer

Feature space transformation is one of the adaptation methods using modification in output layer [22]. Output of a DNN can be modified to transform the feature space as

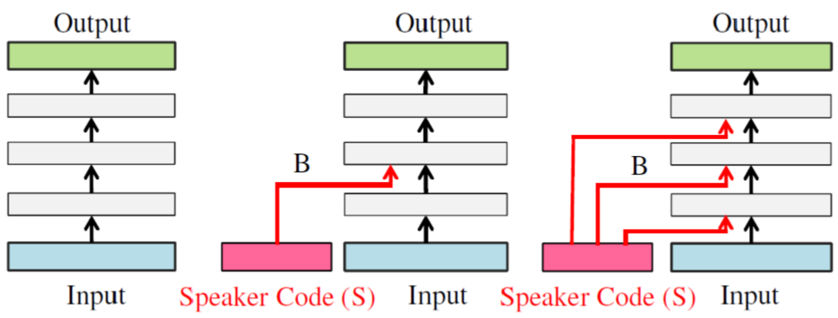


Figure 18: The architecture of DNNs using speaker code in hidden layer(s). (the usual model, model structure with speaker code in a single and all hidden layers are shown in the left, middle, and right sub-figures, respectively.)

$$y \approx \mathcal{F}(y^{'}), \tag{61}$$

where the estimated acoustic feature is shown with $\prime y$, the transformed acoustic features to target speaker is y, and $\mathcal{F}(.)$ is the function for transforming the acoustic features to target speaker's features or "reference vocoder coefficients". For creating the adapted target speaker's voice, first frame-level linguistic features corresponding to the target speaker's data will be transformed to acoustic features using SI model. At the end, using pairs of estimated acoustic features and "reference vocoder coefficients" a transformation function will be trained. For applying the adaptation, in the synthesis time, first using the AVM model the sequence of estimated acoustic features given linguistic features will be generated. After that, for having the target speaker's voice, using the transformation function we will transform the predicted acoustic features. For implementation of this feature space mapping, we can apply any linear or non-linear transforms. State-of-the-art voice conversion method or "joint density Gaussian mixture model" (JD-GMM) [97] is suitable option for implementing this mapping transformation function. For better estimation of correlation between features, full-covariance matrices will be used in the JD-GMM.

Multi-speaker modeling using output layer for speaker adaptation is another method of adaptation in output layer [23]. In this point of view, DNNs hidden layers can be viewed as deep transformation for linguistic features and the output layers as representation of acoustic space to regress the transformed linguistic features to acoustic parameters. The deep-layered architectures of DNN can not only represent highly-complex transformation compactly, but also take advantage of huge amount of training data. In this structure, same hidden layers are shared among different speakers while the output layers are composed of speaker-dependent nodes explaining the target of each speaker. DNN architecture of speech synthesis based on multi-speaker is shown in Figure 19.

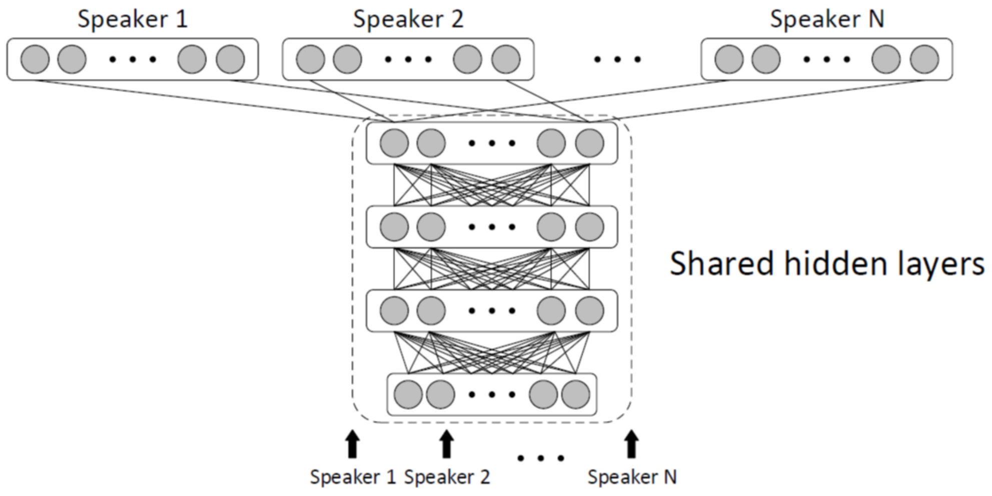


Figure 19: DNN architecture of speech synthesis based on multi-speaker.

CHAPTER IV

CROSS-LINGUAL SPEAKER ADAPTATION METHODS

An adaptation method for “statistical parametric speech synthesis” when using adaptation data of a target speaker in input language, the adapted model in different output language will be generated is called “cross-lingual speaker adaptation” (CLSA). Speech-to-speech translation systems are samples of applications for this kind of adaptation. TC-Star project [98] and “effective multilingual interaction in mobile environments” (EMIME) [99] are samples of projects on spoken language translation (SLT) systems in recent years. Cross-lingual adaptation for SPSS systems can be done in phoneme level [100] or state level [11]. In recent years, for “cross-lingual speaker adaptation”, new DNN-based methods are proposed [101].

4.1 Phoneme-Mapping Based Adaptation

Phoneme mapping based method was the first proposed method for CLSA. Speaker adaptation between English and Chinese languages in phoneme level was investigated in [100]. Mapping the phoneme-level linguistic context, is the main part of the phoneme mapping based adaptation method. Phoneme mapping between English and Chinese languages is sample of this kind of adaptation. After mapping the Chinese phonemes to English phonemes, the mapped data will be assumed as adaptation data in output language (English), and some “intra-lingual speaker adaptation” method like CMLLR or CSMAPLR will be applied to the adaptation data. In this adaptation method, language mapping must be applied for both phonetic and prosodic levels of information. For covering both kinds of linguistic contexts, “full context” labels were used in “HMM-based speech synthesis”. By considering the phonetic definition of Chinese and English phonemes in the international phonetic alphabet (IPA), two rules for mapping between Chinese initial to final and Chinese final phoneme to English phoneme were generated. Because of difference in linguistic contents of two different languages, designing a suitable mapping in prosodic level between two different languages is a challenging task. One solution for this issue can be using triphones instead of full-context labels and making the regression classes and transformation matrices for triphone models. At the end, these models will be applied to fullcontext models [102].

4.2 State-Mapping Based Adaptation

For modeling the similarity of phonemes of two languages in acoustic space phoneme mapping does not have suitable accuracy. For mapping the phonemes between two languages with higher accuracy, state mapping based method for "cross-lingual speaker adaptation" was suggested. In this method, first two "average voice models" (AVMs) in input and output languages are trained, respectively. Based on the list of states in input and output languages, a mapping function is defined to map the states between these two language spaces. At the end, based on the defined mapping information, "cross-lingual speaker adaptation" will be applied. For finding the nearest state between language spaces, "Kullback-Leibler divergence" (KLD) is used as a distance measure for cross-lingual SPSS systems [9]. We can show the models of states in input language with $\Omega^s_k(k = 1, ..., N^s)$, where the maximum number of state models is denoted with N^s. As a coefficient in each state model Ω^s_k, we can mention the "self-transition probability" a^s_k, a mean vector $\boldsymbol{\mu}^s_k$ and a covariance matrix $\boldsymbol{\sigma}^s_k$. These parameters are denoted mostly because here single Gaussian mixture is used. Just like input language, we can define state models in the model space of output language with $\Omega^g_j(j = 1, ..., N^g)$, and the corresponding self-transition probability, mean vector and covariance matrix can be shown with a^g_j, μ^g_j, and $\boldsymbol{\Sigma}^g_j$, respectively.

In the cross-lingual adaptation scenario, we want to find the minimum distance

Ω_j^g which is mostly KLD distance between the state models in output language Ω_j^g and input language Ω_k^s. We will compute this calculation for each state in output language. When modeling the states with single Gaussian mixture, the maximum KLD distance between two models can be computed as

$$D_{KL}(\Omega_j^g, \Omega_k^s) \leq \frac{D_{KL}(G_k^s||G_j^g)}{1-a_k^s} + \frac{D_{KL}(G_j^g||G_k^s)}{1-a_j^g} + \frac{(a_k^s - a_j^g)log(a_k^s/a_j^g)}{(1-a_k^s)(1-a_j^g)} \tag{62}$$

where the “Gaussian distribution” related to the state model Ω_k^s, is shown with G_k^s. The parameters for this distribution are the mean vector $\boldsymbol{\mu}_k^s$ and covariance matrix $\boldsymbol{\Sigma}_k^s$, respectively. In this case, the KLD distance between two Gaussian distributions can be calculated as

$$D_{KL}(G_k^s||G_j^g) = \frac{1}{2}ln\left(\frac{|\boldsymbol{\Sigma}_j^g|}{|\boldsymbol{\Sigma}_k^s|}\right) - \frac{D}{2} + \frac{1}{2}tr(\boldsymbol{\Sigma}_j^{g-1}\boldsymbol{\Sigma}_k^s) + \frac{1}{2}(\boldsymbol{\mu}_j^g - \boldsymbol{\mu}_k^s)^T\boldsymbol{\Sigma}_j^{g-1}(\boldsymbol{\mu}_j^g - \boldsymbol{\mu}_k^s) \tag{63}$$

The KLD between two state models if we ignore the “transition probabilities” effect, can be shown as

$$D_{KL}(\Omega_k^s, \Omega_j^g) \approx D_{KL}(G_k^s||G_j^g) + D_{KL}(G_j^g||G_k^s) \tag{64}$$

For each state model in output language Ω_j^g, the nearest state model in input language $\Omega_{k'}^s$ using KLD distance can be found as

$$k_j^{'} = \underset{k}{\operatorname{argmin}}\, D_{KL}(\Omega_j^g, \Omega_k^s). \tag{65}$$

At the end, these state mappings between two languages can be calculated as

$$\Omega_j^g \Rightarrow \Omega_{k_j^{'}}^s, \quad j = 1, ..., N^g. \tag{66}$$

There are two approaches for using mapping information: data mapping and transform mapping based state mapping.

4.2.1 Data Mapping Based State Mapping

For making a transformation between the adaptation data in the input language and the states in output language data-mapping based approach of CLSA is proposed. In this case, first two average voice models in both input and output languages must be trained. After that, we must find the mapping between the states of input and output state models, for instance for each state of input language find a nearest state in output language. Next, based on the trained state mapping, we will connect the adaptation data of target speaker to the output language voice model. At the end, with assuming the adaptation data as the data in the output language and applying some common intra-lingual speaker adaptation method (e.g., CSMAPLR or CMLLR) on the voice model in output language, we can generate the adapted model in output language. We can assume this data mapping method of cross-lingual adaptation as an extension of phoneme mapping based method. However, in the state mapping based adaptation, the mapping between input and output languages are done in the state level. For each adaptation sentence, the knowledge for mapping in mostly relevant to sequence of states. In this case, in the middle of training, forward-backward algorithm is the part for defining the exact relation between models and data. Overview of the "data mapping based state-mapping" is shown in the Figure 20.

4.2.2 Transform Mapping Based State Mapping

Similar to "data mapping" approach, in "transform mapping" method, we will find the mapping between states of the voice models in input and output languages. Similarly, SI models of both input and output languages must be trained. After that, we will establish state mappings between the output and input state models, for instance for each state model in output language we will find a nearest state model in input language. Next, in input language with use of adaptation data, we will train the transforms for the SI model. At the end, with use of transformation information

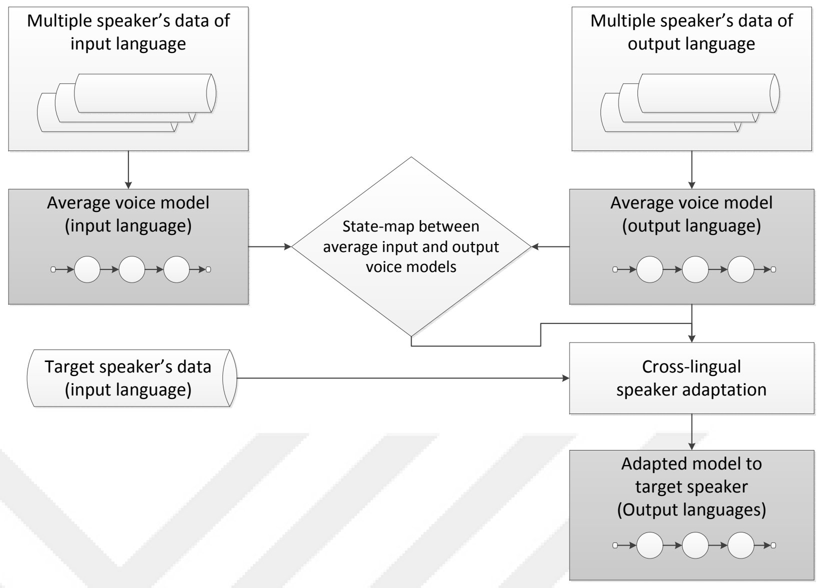

Multiple speaker's data of input language
Multiple speaker's data of output language
Average voice model (input language)
State-map between average input and output voice models
Average voice model (output language)
Target speaker's data (input language)
Cross-lingual speaker adaptation
Adapted model to target speaker (Output languages)

Figure 20: Overview of the "data mapping based state-mapping".

of nearest state in input language, each state of the SI model in output language will be transformed. In this method, adaptation is done in input language and the transformations will be mapped to the output language, because of this reason, there is no direct relation between the output AVM and input adaptation data. For mapping the adaptation transforms from input to output languages, this "transformation mapping-based state-mapping" can be used. At the end, using these transforms the state models in output language can be adapted. Overview of the transform mapping based state-mapping is shown in Figure 21.

For having a successful transform mapping based cross-lingual adaptation, the difference between space of the models in input and output languages must be negligible. For instance, for creating high quality adapted models, we need a bilingual speech database with rich linguistic context to train models in input and output languages. However, recording this kind of bilingual dataset which is suitable for creating average models in input and output languages is cost inefficient and time consuming. Even

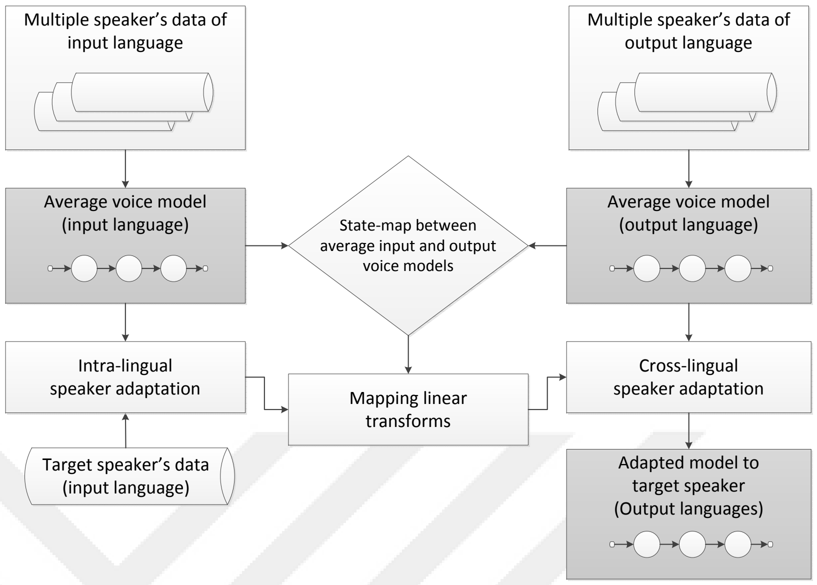

Multiple speaker's data of input language
Multiple speaker's data of output language
Average voice model (input language)
State-map between average input and output voice models
Average voice model (output language)
Intra-lingual speaker adaptation
Mapping linear transforms
Cross-lingual speaker adaptation
Target speaker's data (input language)
Adapted model to target speaker (Output languages)

Figure 21: Overview of the transform mapping based state-mapping.

in the case when using balancing the input and output average voice model datasets based on the number of speakers and gender, we minimize the mismatch between input and output language spaces, still we will have the significant difference between input and output language spaces.

In CLSA, difference between adaptation data in input language and SI model in output language contains both speaker and language characteristics. The ideal condition in cross-lingual adaptation is the condition when the characteristics of the target speaker which is relevant to speaker identity is adapted while the characteristics of output voice model is presumed. In other words, in CLSA with eliminating the effect of language identity of AVM in input language, the language identity of output language must be kept.

Both “Data mapping” and “transform mapping” methods of adaptation for CLSA have some pros and cons. For the “data mapping” method, the adaptation data for target speaker is directly used in output language. As a result, the speaker characteristics of the target speaker can completely be adapted to the output model. But adaptation of input language identity is one of the cons of “data mapping” method. It means, in synthesis time the identity of the input language will be sensible in the output speech waveform. This sensibility is mostly because of partially adaptation of input language identity.

As an advantage for the “transform mapping” method we can mention the elimination of the influence of input language identity meanwhile of cross-lingual adaptation. This elimination is caused by intra-lingual adaptation for “transform mapping” based method. But this model is so sensitive to differences between input and output AVMs. In other word, if there is a mismatch in linguistic and acoustic context of training data for creating average voice model in input and output languages it will cause the degradation in performance of cross-lingual adaptation. In other word, in synthesis time the characteristics of synthesized speech will be different from the target speaker’s voice.

As we mentioned before, mismatch between the two AVMs degrades the quality when mapping transforms [12, 13] since the speaker-specific transformations for states in the input language acoustic model may not actually suit the corresponding state in the output language acoustic model. To alleviate that problem, a transform mapping using shared decision tree context clustering is proposed in [14] where not only acoustic-similarity but also contextual similarity of states is taken into account during mapping.

The AVM can also be trained using data from multiple languages and adapted to a target speaker that speaks one of those languages [103]. However, the adaptation of such a model may hampered by the fact that some leaf nodes of the decision tree might be trained with data from only one language. A speaker and language factorization technique to alleviate this problem is proposed in [104] where Cluster

Adaptive Training (CAT) is used to build an AVM using data from different languages. For a given target language, cluster weights are estimated for building a language-dependent model, before adapting it to a speaker of that language.

For decreasing language dependency and also adapting prosodic information in CLSA, the mapping between languages can be provided by a language-independent space of perceptual characteristics (PC) [105]. This technique relies on two language spaces of speakers' voices in the input and output languages. Each speaker is represented by a mean super-vector. When a new target speaker enters the input language speaker space, it is first projected to the intermediary PC space and, once an appropriate representation for this speaker is found in that space, it is projected to the output language. Finally, speaker interpolation is performed in the output language to reconstruct the super-vector of the target speaker. The perceptual space is constructed using listening tests.

Factor analysis-based CLSA using bilingual speech data is proposed in [106]. In this method, model parameters representing language-dependent acoustic features and factors representing speaker characteristics are simultaneously optimized using a maximum likelihood approach and a single statistical model trained using bilingual speech data. Assuming that the speaker characteristics factors are the same in both languages, performance is expected to improve compared to training each eigenvoice space independently.

A voice conversion algorithm is proposed in [107] for rapid cross-lingual adaptation. An eigenvoice-based conversion model is learned using parallel data between a source speaker and a pool of speakers speaking the same language as the source speaker. Then, that model is adapted to a target speaker that speaks a foreign language using a small amount of data.

4.3 DNN-Based Adaptation

For training the multilingual models, deep neural networks methods have also been investigated [15–17]. Liao [108] investigated supervised and unsupervised adaptation of different weight subsets using a few minutes of adaptation data. Learning hidden unit contribution (LHUC) is unsupervised method which using adaptation data it will rescale the weight coefficients (amplitudes) of each hidden units in the model. In this case, it will not change the hidden units feature receptors [21]. For acoustic modeling, activations at the output layer are normalized by a softmax operation to produce posterior distribution over tied states at time t, s_t:

$$P(s_t|x_t;\theta) = \frac{exp(\mathbf{U}_{s_t}^T)\Phi^L}{\sum_{s'} exp(\mathbf{U}_{s'}^T)\Phi^L}, \tag{67}$$

where x_t is the input data in frame t, θ is model parameters, Φ^L is non-linear transfer function in L (last) hidden layer, and U is output weight matrix. This model is usually trained in a speaker independent (SI) fashion: a set of training speech examples $\{(x_t, s_t)\}_{t=1}^{T}$ produced by some number of distinct speakers is used to train the network. The objective of speaker adaptation is to adjust the parameters such that the acoustic model generalizes better to unseen talkers. This is achieved by using some amount of adaptation data $\{(x_t^m, s_t^m)\}_{t=1}^{T^m}, T^m << T$ for speaker m in order to refine the model such that it better approximates the posterior distribution $P(s_t|x_t^m;\theta^m,\theta)$ for a given speaker. By defining a set of speaker-dependent (SD) parameters for speaker m, $\theta^m = \{r_m^1, ..., r_m^L\}$, where $r_m^l \in \mathbb{R}^{M^l}$ is the vector of SD parameters for the l^{th} hidden layer. If $a(r_m^l)$ is element-wise function that constrains the range of r_m^l, then we can modify the output of hidden layer l in SI

$$h^l = \Phi^l(\mathbf{W}^{lT} h^{l-1}), \tag{68}$$

to define an SD hidden layer output:

$$h_m^l = a(r_m^l) \circ \Phi^l(\mathbf{W}^{lT} h_m^{l-1}), \tag{69}$$

where $\circ$ is an element-wise multiplication. The SD term can be viewed as weighting the hidden unit contributions.

Unsupervised adaptation using multi-speaker DNN structure is done in [101]. In this model, using shared hidden layers of multi-speaker DNN structure, nearest labels for unlabeled acoustic data of target speaker will be estimated. Common linguistic content between training speakers can be modeled using multi-speaker DNN structure. It will improve the robustness of mapping from linguistic to acoustic features. In addition, it will have a positive effect on quality of the synthesized speech. As in cross-lingual adaptation, correct frame-level linguistic features are not available for adaptation, using unsupervised adaptation, for each frame of target speaker's acoustic feature, linguistic label will be estimated. Using multi-speaker DNN-based structure for speech synthesis, optimal estimation of linguistic features l for each frame of acoustic feature o based on some cost function like MSE can be shown as

$$l = \underset{l}{\operatorname{argmin}}\, D(\underset{s}{\operatorname{argmin}}\, \mathcal{F}_s(l), o) \tag{70}$$

where the mapping function from linguistic to acoustic features Using DNN model is shown with $\mathcal{F}_s(.)$. Distance between estimated and real acoustic features for speaker s is shown with $D(.)$. Utilizing rich speech dataset, the accuracy of estimated labels in unsupervised speaker adaptation will be increased. After estimation of the frame-level linguistic features, unsupervised adaptation can be applied similar to supervised adaptation. And in the multi-speaker DNN structure, using some estimation method like least squares, for generating the adapted model, we can estimate the regression layer of DNN.

CHAPTER V

PROPOSED ALGORITHMS

We mostly focused on cross-lingual adaptation when only a few utterances are available from a target speaker. To achieve better speaker similarity than existing state-mapping based algorithms under limited data conditions, we proposed several methods. In the first method, eigenvoices were used for rapid adaptation. Eigenvoice weights computed for the input language are linearly transformed into output language weights. The transformation matrix is learned using a bilingual training database which contains English and Turkish speech data from the same speakers. During eigenvector transformation, to avoid overfitting and exploit correlations within eigenvector elements, a partial least squares (PLS) approach is used. To further boost the performance, elements of eigenvectors are also weighted using recursive PLS (rPLS). Moreover, in addition to weighting the eigenvectors in a least-squares linear regression approach, as done in [24], eigenvectors are weighted in the proposed PLS and rPLS frameworks leading to weighted-PLS and weighted-rPLS algorithms.

In the second method, we proposed speaker-specific state-mapping, for which a bilingual database was used. After generating speaker-adapted models for both input and output languages, a speaker-specific state-map is constructed for each speaker in the pool of bilingual speakers. Then, for a previously-unseen target speaker, a nearest-neighbour is found in the pool and the state map of that nearest-neighbour is used for adaptation. Performance for the excitation parameters was found to be significantly better with the proposed method than the baseline target-speaker-independent state-mapping algorithm, in objective and subjective tests.

As the next novelty, the proposed state-mapping algorithm is used for mapping the data in the input language to models in the output language and performing cross-lingual eigenvoice adaptation which enabled significant improvement in the spectral envelope features.

In recent years, deep neural network has significant effect on statistical parametric speech synthesis (SPSS) field. As the last novelty, for investigating DNN models on cross-lingual adaptation, we proposed new method of unsupervised speaker adaptation using continuous labels. In this method, using BLSTM-RNN based model, we estimate the continuous labels (c-lab) for unlabeled adaptation data of target speaker. For adaptation, DCC structure is used for estimating the speaker codes [20].

5.1 Cross-lingual Eigenvoice Adaptation

Eigenvoice for intra-lingual adaptation is discussed in Section 3.3. For eliminating the effect of overfitting, regularization is done by imposing a zero-mean Gaussian prior, $p(\boldsymbol{w})$, on the weight vector. $\boldsymbol{w}_s$ is then estimated using a maximum a posteriori (MAP) adaptation.

In the MAP approach, the weight vector for a target speaker s is estimated with the objective function

$$\hat{\mathbf{w}}_{\mathrm{s,map}} = \underset{\mathbf{w}}{\operatorname{argmax}}\, p(\boldsymbol{\chi}_a|\mathbf{w})\mathbf{p}(\mathbf{w}) \tag{71}$$

where $p(\mathbf{w})$ is the prior, set to $\mathcal{N}(0, \boldsymbol{\Sigma}_{\mathbf{w}})$ here.

Using (38) to replace the likelihood term $p(\boldsymbol{\chi}_a|\mathbf{w_s})$, removing the terms that are independent of $\mathbf{w}$ from the objective function, and with some matrix manipulation, the MAP objective function becomes

$$\hat{\mathbf{w}}_{\mathrm{s,map}} = \underset{\mathbf{w}}{\operatorname{argmax}} \exp(\mathbf{w^T E^T \Sigma^{-1} S_x} - \frac{\mathbf{1}}{\mathbf{2}}\mathbf{w^T E^T N \Sigma^{-1} E w}) \exp(-\frac{\mathbf{1}}{\mathbf{2}}\mathbf{w^T \Sigma_w^{-1} w}), \tag{72}$$

where the block diagonal $\mathbf{\Sigma^{-1}} = \mathbf{diag}(\mathbf{\Sigma_1^{-1}}, \mathbf{\Sigma_2^{-1}}, ..., \mathbf{\Sigma_{N_{st}}^{-1}})$, $\mathbf{S_x} = [\mathbf{S_{x,1}}, \mathbf{S_{x,2}}, ..., \mathbf{S_{x,N_{st}}}]$, and $\mathbf{N} = \mathbf{diag}(\mathbf{N_1}, \mathbf{N_2}, ..., \mathbf{N_{N_{st}}})$.

The objective function can be maximized by noting that the posterior distribution $p(\mathbf{w}|\boldsymbol{\chi}_\mathbf{a})$ is a Gaussian since the Gaussian distribution is the conjugate prior of the

Gaussian likelihood function with unknown mean in (38). Therefore, (71) can be written as

$$\hat{\mathbf{w}}_{\mathrm{s,map}} = \underset{\mathbf{w}}{\operatorname{argmax}}\, exp(-\frac{1}{2}(\mathbf{w} - \mu_{\mathbf{w}|\chi})^{\mathbf{T}}\mathbf{R}_{\mathbf{w}|\chi}(\mathbf{w} - \mu_{\mathbf{w}|\chi})) \tag{73}$$

where $\mathbf{R}_{\mathbf{w}|\chi}$ is the precision matrix. By completing the squares and using (72),

$$\mathbf{R}_{\mathbf{w}|\chi} = (\mathbf{E}^{\mathbf{T}}\mathbf{N}\mathbf{\Sigma}^{-\mathbf{1}}\mathbf{E} + \mathbf{\Sigma}_{\mathbf{w}}^{-\mathbf{1}}), \tag{74}$$

and

$$\mu_{\mathbf{w}|\chi} = \mathbf{R}_{\mathbf{w}|\chi}^{-\mathbf{1}}\mathbf{E}^{\mathbf{T}}\mathbf{\Sigma}^{-\mathbf{1}}\mathbf{S}_{\mathbf{x}}. \tag{75}$$

MAP estimate of $\mathbf{w}$, $\hat{\mathbf{w}}_{\mathrm{s,map}}$, is the mean, $\mu_{\mathbf{w}|\chi}$, of the posterior distribution. $\mathbf{\Sigma}_{\mathbf{w}}^{-\mathbf{1}}$ is a hyper-parameter of the prior which we set to $\alpha\mathbf{S}^{-\mathbf{1}}$ where α is a scalar (chosen empirically) and $\mathbf{S}$ is the diagonal matrix

$$\mathbf{S} = \mathrm{diag}(\lambda_{\mathbf{1}}, \lambda_{\mathbf{2}}, ..., \lambda_{\mathbf{R}}) \tag{76}$$

where λ_i are the eigenvalues obtained while estimating the E matrix using PCA.

Because adaptation data is available only in the input language, the computations above perform intra-lingual adaption: that is, they result in an estimate for $\boldsymbol{w}_{s,in}$. However, the weight vector for the output language, $\boldsymbol{w}_{s,out}$, is required for cross-lingual adaptation, so that we can compute

$$\boldsymbol{\mu}_{out}^{(s)} = \boldsymbol{\mu}_{\mathrm{SI},out} + \boldsymbol{E}_{out}\boldsymbol{w}_{s,out}. \tag{77}$$

where $\boldsymbol{\mu}_{out}^{(s)}$ is the supervector, $\boldsymbol{E}_{out}$ is the eigenvoice matrix, and $\boldsymbol{\mu}_{SI,out}$ is the speaker-independent supervector for the output language. We have investigated both data-mapping and vector-/space-mapping techniques to estimate $\boldsymbol{w}_{s,out}$. Our proposed techniques are described below.

5.1.1 Algorithms Based on Eigenvector Mapping

Given $\boldsymbol{w}_{s,in}$, computed using intra-lingual adaptation, we can use linear regression to predict $\boldsymbol{w}_{s,out}$. The $\boldsymbol{w}_s$ vectors for a set of bilingual training speakers can be computed for the input and output languages using (44). Then, a linear regression matrix $\boldsymbol{A}$ can be trained such that $\boldsymbol{w}_{s,out} = \boldsymbol{A}\boldsymbol{w}_{s,in} + \boldsymbol{\epsilon}$. In the simplest approach, the least-squares (LS) algorithm is used for training $\boldsymbol{A}$. Once A is trained using the training speaker pool, it can be used to transform the eigenvoice weight vector of a target speaker in input language space into a vector in output language space.

Because the relationship between the input and output vectors is not linear and the number of bilingual speakers is not large, more sophisticated regression techniques are investigated and described below.

5.1.1.1 Speaker-specific Regression of Eigenvoice Vectors

A linear model is chosen because nonlinear methods (e.g., neural networks) require significantly more data, and collection of large bilingual databases is expensive. However, to improve the performance of the linear model, the $\boldsymbol{A}$ matrix can be constructed in a target-specific manner. To that end, we propose a weighted linear regression approach as described below.

Given adaptation data from a target speaker, the speaker-specific $\boldsymbol{A}_{tar}$ matrix is computed using:

$$\boldsymbol{A}_{tar} = \underset{\boldsymbol{A}}{\operatorname{argmin}} \sum_{i=1}^{N_p} \boldsymbol{\epsilon}_{i,tar}^T \boldsymbol{\epsilon}_{i,tar} \tag{78}$$

where N_p is number of training speakers and

$$\boldsymbol{\epsilon}_{i,tar} = L_{tar}(i).(\boldsymbol{w}_{out}(i) - \boldsymbol{A}\boldsymbol{w}_{in}(i)) \tag{79}$$

where $L_{tar}(i)$ is the weight of the i^{th} training speaker, $\boldsymbol{w}_{out}(i)$ is its eigenvoice vector in the output language and $\boldsymbol{w}_{in}(i)$ is its eigenvoice vector in the input language.

The speaker weights $L_{tar}(i)$ are computed as follows. First, intra-lingual adaptation is done and the distance of the target speaker to each of the training speakers is found by using the Euclidean (L_2) distance between the mean supervectors. Then, these distances are compressed and normalized with

$$L_{tar}(i) = 1 - \log_2 \left(\frac{d(i) - d_{min}}{d_{max} - d_{min}} + 1 \right) \tag{80}$$

where $d(i)$ is the distance of the i^{th} training speaker to the target, d_{max} is the maximum and d_{min} the minimum of such distances across all training speakers.

Once $L_{tar}(i)$ and $\boldsymbol{A}_{tar}$ are computed, the eigenvoice weight in the output language is estimated as $\hat{\boldsymbol{w}}_{tar,out} = \boldsymbol{A}_{tar}\boldsymbol{w}_{tar,in}$.

5.1.1.2 Partial Least-squares Regression

Because the number of bilingual speakers is, as already noted, not large, overfitting can occur during linear regression, especially if the eigenvoice vector dimension is large. Correlations between the elements of the eigenvoice vectors, as shown in Figure 22, can be exploited to avoid poor generalization.

When significant co-linearity exists, one way to address the overfitting problem is to use PCA and reduce the dimension of the eigenvoice vectors. However, this is not desirable in our case because the linear regression step is already preceded by a PCA step and further reduction of dimensionality would cause degradation in adaptation performance. Moreover, PCA only minimizes the distortion in the vectors during dimensionality reduction whereas the objective should be to minimize distortion during linear regression.

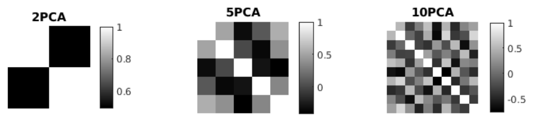
The partial least squares (PLS) linear regression approach is used here to solve the generalization problem. In this approach, the input weight vector is

$$\boldsymbol{w}_{\mathrm{s,in}} = \boldsymbol{\Gamma}\boldsymbol{x}_{\mathrm{s,in}} + \boldsymbol{\epsilon}_{\mathrm{s,in}} \tag{81}$$

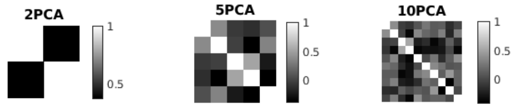
and the output weight vector is

$$\boldsymbol{w}_{\mathrm{s,out}} = \boldsymbol{\Omega}\boldsymbol{x}_{\mathrm{s,in}} + \boldsymbol{\epsilon}_{\mathrm{s,out}} \tag{82}$$

where the regression matrices $\boldsymbol{\Gamma} \in \mathbb{R}^{R \times R_r}$ and $\boldsymbol{\Omega} \in \mathbb{R}^{R \times R_r}$.



(a) Covariance of w_s vectors for spectral envelope (MGC: see chapter 6)



(b) Covariance of w_s vectors for fundamental frequency (LF0: see chapter 6)

Figure 22: Covariances of weight vectors (w_s) for spectral envelope (MGC) and fundamental frequency (LF0) extracted from 88 speakers using 10 utterances per speaker using Eq. 73 are shown. Covariances of the 2, 5, and 10 dimensional weight vectors are shown separately. For an R-dimensional case, an $R \times R$ image is plot where intensity of each pixel is determined by the magnitude of the corresponding element in the covariance matrix.

Because $R_r < R$, the dimensionality of the latent $\boldsymbol{x}_{\text{s,in}}$ vectors is lower than the dimensionality of $\boldsymbol{w}_{\text{s,in}}$ vectors. Thus, dimensionality of $\boldsymbol{w}_{\text{s,in}}$ is reduced in the first equation and a linear regression function is defined between the $\boldsymbol{x}_{\text{s,in}}$ and $\boldsymbol{w}_{\text{s,out}}$ vectors in the second equation. Combining those two equations, the linear regression function becomes

$$\boldsymbol{w}_{\text{s,out}} = \boldsymbol{\Psi} \boldsymbol{w}_{\text{s,in}} + \boldsymbol{\epsilon}_{\text{s}} \tag{83}$$

where $\boldsymbol{\Psi} \in \mathbb{R}^{R \times R}$. The solution with PLS minimizes $\sum_s \|\boldsymbol{\epsilon}_{\text{s}}\|^2$. The SIMPLS algorithm is used to solve the PLS regression problem [109].

5.1.1.3 Recursive Weighted Partial Least-squares Regression (rPLS)

Some of the predictor variables in $\boldsymbol{w}_{s,in}$ are probably more important than others for explaining the observed variables in $\boldsymbol{w}_{s,out}$ through linear regression. One way to handle that in PLS is to use a method such as jack-knife [110] and remove unimportant variables. However, assigning weights to variables depending on their prediction power can lead to a more accurate solution. Recursive PLS (rPLS) algorithm is used here to perform such importance weighting [111].

If the vectors $\boldsymbol{w}_{\text{s,out}}$ and $\boldsymbol{w}_{\text{s,in}}$ are preprocessed to have zero mean and unit variance, then for each element i of $\boldsymbol{w}_{\text{s,out}}$, $\boldsymbol{w}_{\text{s,out}}(i)$, PLS algorithm can be used independently so that

$$w_{\text{s,out}}(i) = \boldsymbol{b}_i^T \boldsymbol{w}_{\text{s,in}}, \tag{84}$$

where $\boldsymbol{b}_i$ is the regression vector for estimating $w_{\text{s,out}}(i)$. After a PLS solution is found, $\boldsymbol{b}_i$ can be used for importance weighting. In that case, the input vectors from the previous iteration are reweighted using

$$\boldsymbol{w}_{\text{s,in}}^{iter} = diag(\boldsymbol{b_i})\boldsymbol{w}_{\text{s,in}}^{iter-1}. \tag{85}$$

where diag($\boldsymbol{b}_i$) is a diagonal matrix where the elements of $\boldsymbol{b}_i$ are on the diagonal. PLS is then used again to re-estimate $\boldsymbol{b}_i$. The PLS and weighting steps are iterated until convergence.

Note that rPLS performs importance weighting for each element of $\boldsymbol{w}_{\text{s,out}}$ independently. Thus, the rPLS model is trained independently for each element of $\boldsymbol{w}_{\text{s,out}}$ which could cause degradation if there is high correlation between the elements of $\boldsymbol{w}_{\text{s,out}}$.

5.1.1.4 Weighted Partial Least-squares Regression (WPLS)

Similar to weighted linear regression, weighted PLS (WPLS) can be used for weighting the eigenvoice vectors depending on their importance, during training. In this approach, the eigenvectors of the training speakers can be weighted such that $\sum_{s=1}^{N_p} \boldsymbol{w}_s ||\boldsymbol{\epsilon}_s||^2$ is minimized, where $\boldsymbol{w}_s$ is the weight for speaker s. In our case, the weights are proportional to the normalized distances of target speakers to training speakers and they can be incorporated into the PLS training algorithm simply by duplicating the training samples in proportion to their weight as described below.

Let the weight of each training speaker i be equal to $L_{tar}(i)$ defined in (80). Then, the data for each training speaker i can be repeated $r_i = round(N_r \times w_i)$ times in the training set where N_r is an integer constant. Those repetitions will approximately increase the size of the training database by a factor of N_r. If the SIMPLS training algorithm is used, the contribution of each sample to the total error $\boldsymbol{\epsilon}$ will be equally weighted. However, because each sample is repeated r_i times and same error $\boldsymbol{\epsilon}_i(r)$ is obtained for each repetition r, total error contributed by speaker i, $\boldsymbol{\epsilon}_i$, is equal to $r_i||\boldsymbol{\epsilon}_i(r)||^2$ where r_i is proportional to w_i if we ignore the round-off effects. Thus, minimization of the total error with the SIMPLS algorithm will minimize weighted errors when samples are duplicated in proportion to their weights.

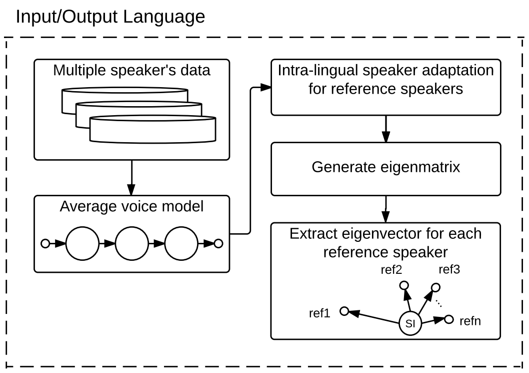

Input/Output Language
Multiple speaker's data
Average voice model
Intra-lingual speaker adaptation for reference speakers
Generate eigenmatrix
Extract eigenvector for each reference speaker
ref1
ref2
ref3
refn
SI

Figure 23: Generation of the eigenspace and extraction of weight vectors for reference speakers. The procedure is done for both input and output languages while performing cross-lingual adaptation using eigenvector mapping.

Because the approach proposed here does not change the training algorithm – it only modifies the training dataset – it can also be used with rPLS, giving us weighted rPLS (WRPLS). Steps for training the AVMs and extracting the eigenvector for each reference speaker in input or output languages are shown in Figure 23. An overview of the various eigenvoice mapping cross-lingual adaptation algorithms is shown in Figure 24.

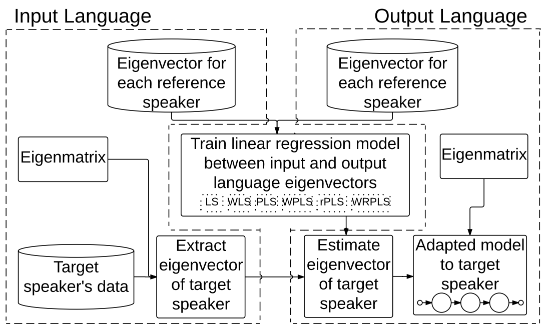

Input Language
Output Language
Eigenvector for each reference speaker
Eigenvector for each reference speaker
Eigenmatrix
Train linear regression model between input and output language eigenvectors
LS
WLS
PLS
WPLS
rPLS
WRPLS
Eigenmatrix
Target speaker's data
Extract eigenvector of target speaker
Estimate eigenvector of target speaker
Adapted model to target speaker

Figure 24: Cross-lingual adaptation of a target speaker to an output language using eigenvector mapping.

5.2 Algorithms Based on Data-Mapping

Using data-mapping structure in state-mapping model of cross-lingual adaptation, two algorithms are proposed. Nearest-neighbour state-mapping and Eigenvoice adaptation using data-mapping.

5.2.1 Nearest-neighbour State-mapping

The baseline algorithm performs state-mapping using the AVMs once and uses the same map for all target speakers. However, data mapping could be more effective if the state-mapping were done in a speaker-specific manner. To that end, separate speaker-dependent models of each reference speaker were adapted for each of the input and output languages.

A cross-lingual state map was learned separately for each of those training speakers, using their speaker-dependent models. As a result, for each bilingual training speaker s_i, a map M_i between that speaker's models for the input and output languages was produced.

Our proposal is to select one of those pre-trained maps to use for adaptation of a (previously unseen) target speaker. Similarity between the target speaker and the training speakers can be used to select the nearest training speaker, S_{nn}, to the target speaker s_{tar}. Euclidean distance, $(\boldsymbol{\mu}_{nn} - \boldsymbol{\mu}_{tar})^T(\boldsymbol{\mu}_{nn} - \boldsymbol{\mu}_{tar})$, is used as the similarity measure, where $\boldsymbol{\mu}_{nn}$ is the supervector of state means in the input language model of nearest training speaker. Similarly, $\boldsymbol{\mu}_{tar}$ is the supervector of the target speaker.

Once S_{nn} is selected, the state-map $\boldsymbol{M}_{nn}$ is used for mapping the adaptation data to output language states. Then, similar to the baseline approach, intra-lingual adaptation is performed.

5.2.2 Eigenvoice Adaptation Using Data-mapping

Cross-lingual Bayesian eigenvoice adaptation (Cross-BEA) can be performed using a data-mapping approach once a state-map $\boldsymbol{M}_{tar}$ is available for the target speaker. Here, the nearest-neighbour based state-mapping algorithm described above is used to find $\boldsymbol{M}_{tar}$.

Once the adaptation data is mapped to the states of the output language, computation of $\hat{\boldsymbol{w}}_{s,out}$ is exactly the same as the intra-lingual adaptation case. The adaptation data-dependent variables $\boldsymbol{S}_{x,c}^{(s)}$ and $N_c^{(s)}$ in (43) are computed by mapping data to output language states using $\boldsymbol{M}_{tar}$. Then, $\boldsymbol{w}_{s,out}$ is estimated using (44). Steps for finding the nearest reference to the target speaker is shown in Figure 25. A diagrammatic overview of all algorithms based on data mapping is in Figure 26.

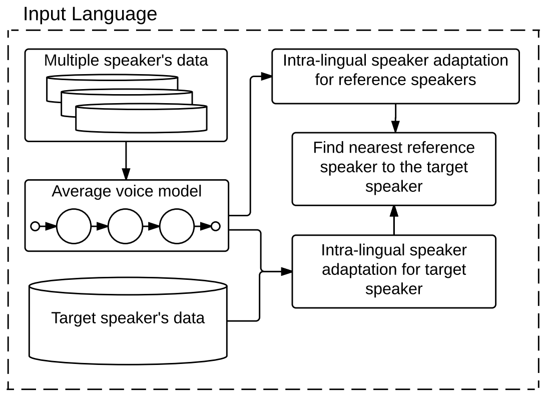

Input Language
Multiple speaker's data
Intra-lingual speaker adaptation for reference speakers
Find nearest reference speaker to the target speaker
Average voice model
Intra-lingual speaker adaptation for target speaker
Target speaker's data

Figure 25: Overview of the algorithm for finding the nearest reference speaker to the target speaker in input language.

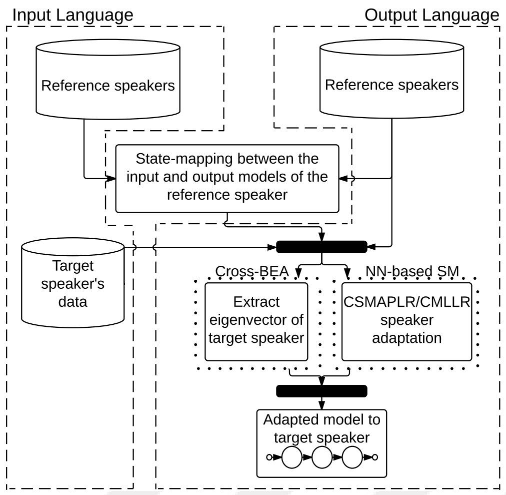

Input Language
Output Language
Reference speakers
Reference speakers
State-mapping between the input and output models of the reference speaker
Target speaker's data
Cross-BEA
NN-based SM
Extract eigenvector of target speaker
CSMAPLR/CMLLR speaker adaptation
Adapted model to target speaker

Figure 26: Overview of the data-mapping algorithms. After state-mapping, proposed eigenvoice adaptation or CSMAPLR/CMLLR adaptations can be done. Those two options are shown between horizontal fork/join bars.

5.3 Unsupervised Speaker Adaptation Using Continuous Labels

In unsupervised adaptation of HSS systems, linguistic labels are mostly generated from ASR [102]. When we use short units like senone, the quality of estimated linguistic features from ASR can be unstable. Even using longer units (e.g. phoneme or word) is also generally depend on the quality of the language model. In ASR mostly shorter linguistic units like senones is common while for generating high quality waveforms, speech synthesis needs richer linguistic content like full context label. For the long-span linguistic features, estimating the rich content linguistic features from estimated ASR labels is problematic. Specially, in the cross-lingual adaptation case, because of difference between linguistic content of input and output languages, tied tri-phone state mapping can not be accurate. For alleviating the problem, using BLSTM-RNN we propose new method for estimating the linguistic features. Instead of multi-speaker DNN [101], for estimating the continuous label for each frame of acoustic data, we used BLSTM-RNN model. Instead of binary label (b-lab) estimation, which is common in DNN-based speech synthesis, with use of sum square error (SSE) as cost function, we estimate continuous labels (c-lab) from the acoustic features. For alleviating the mismatch between binary and continuous labels, TTS model is trained with continuous labels. Then, target speaker is adapted using DCC structure for estimating the speaker codes in input layer [20]. With keeping the DNN weights unchanged in the adaptation time, in addition of the advantage of applying the linguistic features and speaker codes in the same layer, this method doesn't suffer from the gradient vanishing problem. For improving the adaptation performance, age and gender codes are added to the input layer. In the synthesis time, for alleviating the mismatch between binary and continuous labels, another BLSTM-RNN model was trained to convert b-labs to c-labs. Overview of unsupervised adaptation based on continuous labels are shown in Figure 27. As this method of adaptation is not biased to employ any specific language-related feature, it can be applicable for both intra- and cross-lingual adaptation cases.

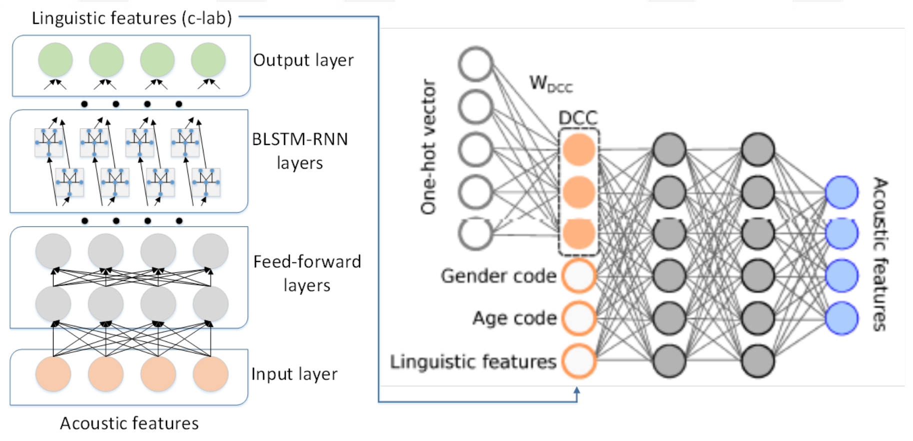


Figure 27: Overview of unsupervised adaptation based on continuous labels.

CHAPTER VI

EXPERIMENTS AND RESULTS

Experimental settings of the proposed methods are shown in Section 6.1. Performance of the proposed methods were measured with both objective and subjective tests. The objective test results are presented in Section 6.2 and the subjective test results are presented in Section 6.3. The first set of objective tests were done to tune the regularization parameter, α, of the eigenvoice adaptation technique discussed in Section 5.1. Then, the objective test results of the proposed data-mapping based algorithms are presented in Section 6.2.2. Performance of the eigenvector-mapping methods LS, WLS, PLS, and WPLS are discussed in Section 6.2.3 and the rPLS algorithm is discussed in Section 6.2.4. The best performing methods proposed HMM-based systems are compared in Section 6.2.5 and the most important findings are summarized in Section 6.2.6. Objective comparison of the proposed DNN-based method is shown in Section 6.2.7.

The subjective test results are presented for the best performing algorithms in Section 6.3. Speaker similarity test results are discussed in Section 6.3.1 and the speech quality test results are discussed in Section 6.3.2. Subjective comparison of the proposed DNN-based method is shown in Section 6.3.3.

6.1 Experimental settings

For the proposed methods experiments on HMM-based and DNN-based systems were done.

6.1.1 HMM-based Systems Experiments

All HMM-based systems in our experiments employed 78 dimensional observation vectors comprising 24 Mel-Generalized Cepstral Coefficients (MGCs), 1 log-energy, 1 log-F0 (LF0) coefficient, and their delta and delta-delta parameters. A 25 msec analysis window with 5 msec frame shift is used for feature extraction. Phonemes are modeled with 5 state Hidden Semi-Markov Models (HSMM).

Turkish is the input language and English is the output language. Two male (bdl and rms) and two female (slt and clb) speakers from the CMU-ARCTIC database (1130 utterances per speaker) were used to train the average voice model (AVM) for English. For training the AVM in Turkish, speech from three female speakers (1100 utterances each) were used. For the purposes of testing the proposed methods, a bilingual Turkish-English database was created, containing speech from 88 female speakers. From 29 speakers, used as targets, 50 Turkish and 50 English utterances were recorded. 10 Turkish and 10 English utterances were recorded by each of the remaining speakers. For better comparison between reference speakers, same sentences were used for all speakers.

For each speaker, a Turkish speaker-dependent model was created using the Turkish AVM and CSMAPLR adaptation followed by MAP adaptation. Similarly, English speaker-dependent models were created using the English AVM for each speaker. A leave-one-out method was used in testing for each of the 29 training speakers in turn. Thus, 87 training speakers were used for each target speaker. The rank hyperparameter of the PLS and rPLS algorithms was tuned using cross-validation. The N_r parameter of the WRPLS algorithm was empirically set to 100. Summary of experimental setup for HMM-based methods is shown in Table 1.

The state-mapping algorithm [11] described in Section 4.2.1 was used as the comparison baseline since in similarity case, it is one of the best performing cross-lingual adaptation techniques available [14, 105].

Table 1: Summary of experimental setup for HMM-based methods.

	Average voice models		Adapted intra-lingual models			
	Number of speakers	Number of utterances	Number of speakers	Number of utterances	Number of target speakers	Number of reference speakers
Input language	3	3217	88	10 (29 with 50 utts)	29	87
Output language	4	4528	88	10 (29 with 50 utts)	-	87

Table 2: Age bands of the training-set of DNN-based method.

Age	Male	Female	Total
10-20	8	8	16
21-30	8	8	16
31-40	8	8	16
41-50	8	8	16
51-60	8	8	16
61-70	8	8	16
71-	8	8	16
Total	56	56	112

6.1.2 DNN-based Systems Experiment

For the DNN-based experiment, For training the average voice model, we used the Japanese Voice Bank corpus, containing studio-quality native Japanese speech uttered by 56 male and 56 female speakers aged between 10 and 89. The training-set speakers were chosen to be equally distributed for each age band (8 speakers for each age band and gender). With approximately 100 utterances per speaker, this yielded a total of 11,170 training-data utterances. Age bands of the training-set are shown in Table 2.

For the adaptation, both intra-lingual and cross-lingual conditions were tested. To this end, speech of one female and one male English-Japanese bilingual speakers were used. This dataset was recorded in National Institute of Informatics (NII) institution in Tokyo, Japan. For the cross-lingual adaptation case, English and Japanese were used as input and output languages respectively. For investigation of the effect of adaptation data size on quality of the synthesized speech, for each speaker, 10, 100, and 200 utterances from English and Japanese languages were used as adaptation data. Description of the target speakers for DNN-based intra- and cross-lingual

Table 3: Description of the target speakers for DNN-based intra-lingual adaptation approach.

Name	**Gender**	**Adaptation Size (utts)**
JpnM00010	Male	10
JpnM00100	Male	100
JpnM00200	Male	200
JpnF00010	Female	10
JpnF00100	Female	100
JpnF00200	Female	200

Table 4: Description of the target speakers for DNN-based cross-lingual adaptation approach.

Name	**Gender**	**Adaptation Size (utts)**
EngM00010	Male	10
EngM00100	Male	100
EngM00200	Male	200
EngF00010	Female	10
EngF00100	Female	100
EngF00200	Female	200

adaptation approaches are shown in Table 3 and Table 4, respectively. To evaluate the speaker adaptation task, for each target speaker, 100 additional utterances in English and Japanese languages were held out as test data.

The speech signal waveforms were sampled at 48 kHz, with 16 bits per sample. World [112] analysis was used to obtain 259-dimensional acoustic feature vectors every 5 ms, each with 25 ms window length comprising 60 mel-cepstral coefficients (with a bilinear frequency warping parameter of 0.77), a linearly interpolated fundamental frequency in the mel scale, and 25-dimensional band aperiodicities, along with their delta and delta-delta counterparts. The last feature was a binary voiced/unvoiced flag. During synthesis time, to produce the most likely speech trajectory sequence, the static and dynamic features were combined as described in [113], based on a forced-alignment against the held-out natural speech (i.e., an oracle duration model from HSMM alignment was used).

For the input linguistic features, Open JTalk[1] was used to perform standard analysis of Japanese text for synthesis, including “grapheme-to-phoneme conversion”, “part-of-speech tagging”, and “morphological analysis” with the MeCab parser [114]. Syntactic and prosodic linguistic information which is extracted from text and calculated quin-phone identity were concatenated into a 389-dimensional mixed discrete and numeric (365 discrete and 24 numeric) vector of linguistic features. As we explained before, we call this linguistic features “binary labels” (b-lab). For eliminating the mismatch between training and unsupervised adaptation data, with used of BLSTM-RNN model, binary labels of training set is converted to continuous labels (c-lab). This continuous labels were then augmented with age, gender, and speaker codes as auxiliary features and used as the input to the neural network speech synthesizer.

This DNN model was feed-forward DNN with five hidden layers of 1024 hidden units per each layer. Sigmoid activation functions were used for all units in the hidden and output layers. The models were initialized randomly and trained to minimize mean square error using AdaGrad [115] as batch optimizer for 10 epochs with the learning rate fixed to 0.05 and mini-batch size set to 256. When adapting to a new speaker, speaker code was estimated using 10 back propagation epochs with learning rate 0.2. For implementation Python wrapper of OpenCL (PyOpenCL)[2] was used and the model was trained on GPU Tesla K80.

For estimating the continuous labels from acoustic features, BLSTM-RNN model was used. This RNN model was two feed-forward layers with 1024 hidden units per each layer followed by two BLSTM layers with 256 hidden units per each. Stocastic gradient descent (SGD) with 0.0004 learning rate with 256 mini-batch size for 40 epochs was used as batch optimizer. 90% of Japanese VoiceBank dataset was

[1]http://open-jtalk.sourceforge.net/
[2]https://mathema.tician.de/software/pyopencl/

used for training and 10% of this dataset was used for testing. For estimating the continuous labels instead of binary labels, sum squared error (SSE) was used as a cost function. CUDA-enabled machine learning library for recurrent neural networks (CURRENNT)[3] was used for implementation and the model was trained on GPU Tesla K80.

For eliminating the mismatch between binary and continuous labels in synthesis time, another BLSTM-RNN model was trained. The structure of this model is just like previous BLSTM-RNN model, but instead of 259 dimension acoustic feature, 389 dimension binary label was used as input feature.

6.2 Objective Measures

Root-mean-square-error (RMSE) is used for objectively measuring the distortion in LF0 features, with respect to natural references. Similarly, Mel-cepstral distortion (MCD) [116] is used for the MGC features. For the HMM-based methods, synthetic speech from speaker-dependent models was played to listeners as the reference samples. Objective performance of the DNN-based method is described in Section 6.2.7. Now we discuss the objective performance of the HMM-based methods.

The duration model of the English AVM was used in all cases [100] and so the duration of reference and test samples is always the same. For each target speaker, adaptation was performed using 2, 5, or 10 utterances of adaptation data. For each adapted model, 40 English sentences from the WSJ1 database were synthesized for testing. Significance of the difference between models was measured with a t-test at 95% confidence interval.

[3]https://sourceforge.net/projects/currennt/

6.2.1 Tuning the Regularization Parameter

The hyper-parameter α that is used in the regularized eigenvoice approach described in Section 5.1 was tuned experimentally for LF0 and MGC features. Tuning was done for Turkish and English voices separately as shown in Figure 28. The values of α used in the experiments are given in Table 5.

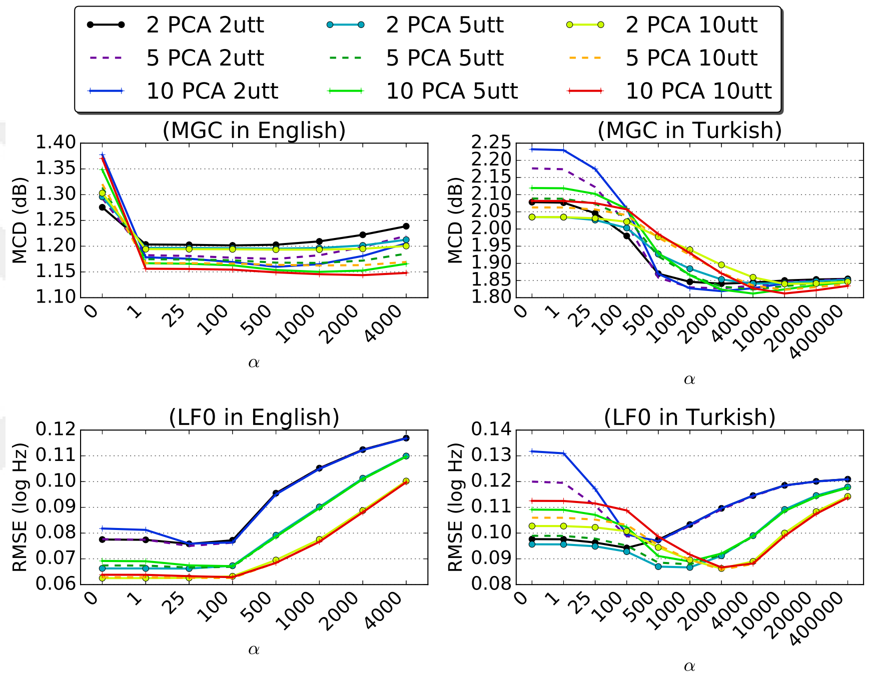


Figure 28: Performance of regularization in intra-lingual adaptation for MGC and LF0 features in English and Turkish with different α values. Note that for the LF0 features, a difference of 0.01 log(Hz) corresponds to 17.3 cents.

Table 5: α values used for 2, 5, or 10 utterances of adaptation data, for English and Turkish.

	English			Turkish		
	2 utt	5 utt	10 utt	2 utt	5 utt	10 utt
MGC	100	100	100	2000	10000	10000
LF0	25	100	100	500	1000	2000

When α increases, the possibility of overfitting decreases. However, if α is too high, then the algorithm does not have enough flexibility to adapt. For Turkish, regularization helped significantly both for LF0 and MGC features.

In the case of English, overfitting did not generally occur. Although a little overfitting occurred for the 10 PCA case, it was not significant for MGC and significant for LF0 only in the 2 or 5 adaptation utterance situations.

There are differences between Turkish and English that explain the differing behaviour regarding regularization. The target speakers are native speakers of Turkish and so their speech is well modelled by the average voice model and their prosodic and pronunciation patterns are consistent when they speak Turkish. For English, this is not the case. Therefore, stronger patterns and higher variability was observed in the case of Turkish, as shown in Figure 29 where the eigenvalues obtained for Turkish and English are shown.

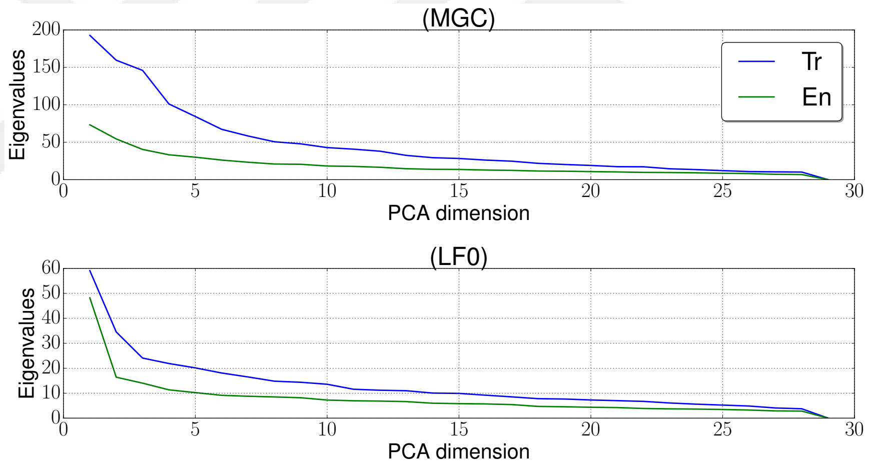


Figure 29: Eigenvalues of reference speakers in Turkish and English languages.

For estimation of distance of average voice models in English and Turkish languages from the reference speakers, Euclidean distance of mean supervector of MGC and LF0 features in English and Turkish languages is illustrated in Figure 30. For MGC features, distance between SI and references in Turkish language is smaller than English language. Using native Turkish speakers for recording the bilingual dataset can be the reason of this observation. But for LF0 features, the observable distance in both languages are similar.

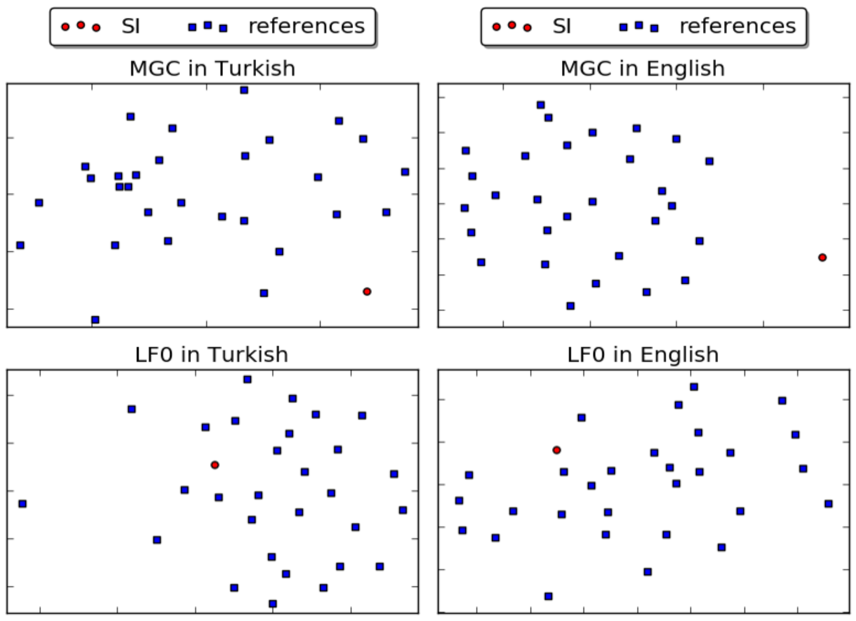


Figure 30: 2D visualization of distance between reference and SI models in English and Turkish data.

6.2.2 Objective Performance of Algorithms Based on Data-mapping

Two algorithms proposed here are based on data-mapping (Section 4.2.1). Performance of the NN-based state-mapping algorithm is compared with the other algorithms in Figure 31. Because CSMAPLR and CMLLR can each be used after data is mapped to output language AVM states, both were tested in combination with the baseline and proposed state-mapping methods. The proposed NN-based state-mapping algorithm significantly outperformed the baseline algorithm both for MGC and LF0 and for all adaptation data sizes. CSMAPLR and CMLLR performed equally well for the MGC features. For LF0, CMLLR performed better than CSMAPLR for the baseline system, and CSMAPLR performed better for the proposed system.

The baseline and NN-based state-mapping algorithms were also compared with the Cross-BEA method in Figure 31 when 2-, 5-, and 10-dimensional eigenspaces were used. The Cross-BEA method substantially improved the performance compared to other techniques, for the MGC features.

For LF0, the Cross-BEA algorithm did not perform as well as NN-based state-mapping. Because the state-mapping accuracy is high when NNs are used, low dimensional LF0 vectors could be adapted well with CSMAPLR. However, performance of the eigenvoice algorithm saturated quickly and it so it does not perform as well as CSMAPLR as the amount of data grows: the performance gap widens with increasing data size.

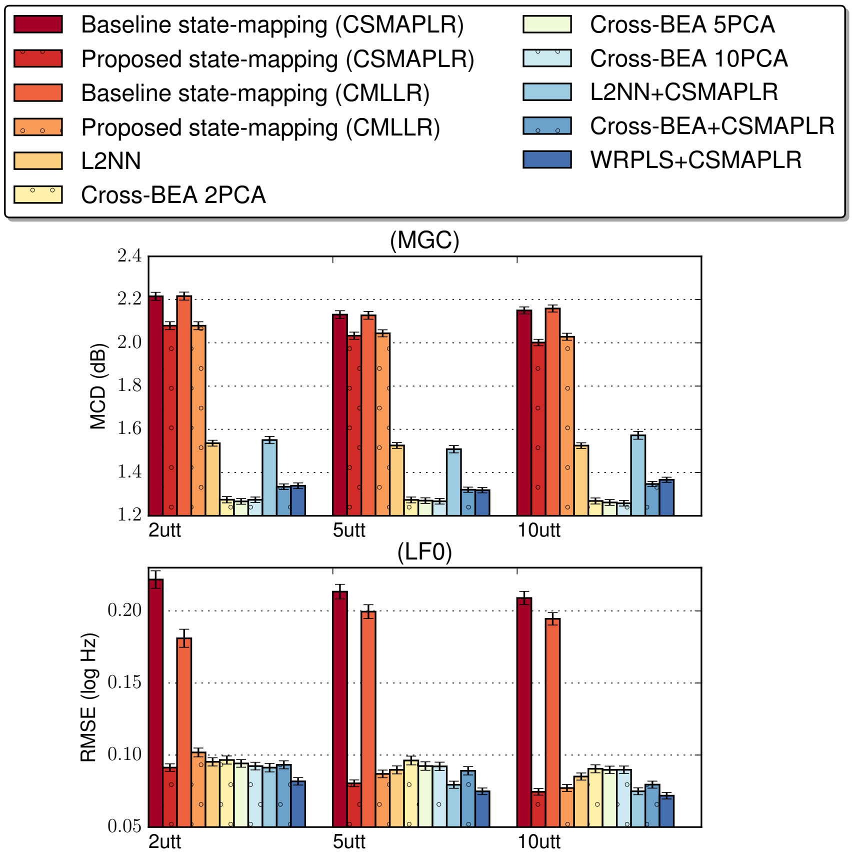


Figure 31: Objective evaluation (RMSE and MCD) of algorithms based on data-mapping for MGC and LF0 features, showing 95% confidence intervals. The groups of results for “2utt”, “5utt” and “10utt” correspond to 2, 5 and 10 utterances of adaptation data. Note that for the LF0 features, a difference of 0.1 log(Hz) corresponds to 173 cents. Cross-BEA+CSMAPLR was done with 10 dimensional PCA. WRPLS+CSMAPLR was done with 2 dimensional PCA.

Algorithms that use data mapping were also compared with the case where speech is synthesized with the nearest-neighbour (L2NN) without any further adaptation. Even though this approach worked well, as shown in Figure 31, it did not perform better than the Cross-BEA algorithm for the MGC features and the NN-based state-mapping algorithm with CSMAPLR for the LF0 features.

Additional CSMAPLR adaptation was done after L2NN, Cross-BEA, and WR-PLS algorithms to investigate if there is opportunity for further improvement with additional adaptation steps. Results are shown in Figure 31. CSMAPLR degraded the performance for the high-dimensional MGC features when applied after L2NN, Cross-BEA, and WRPLS algorithms. Thus, the CSMAPLR algorithm overfit on the adaptation data for the high-dimensional MGC features and that distorted the models. However, it helped improve the performance for the low-dimensional LF0 features.

6.2.3 Objective Performance of Least-squares Algorithms

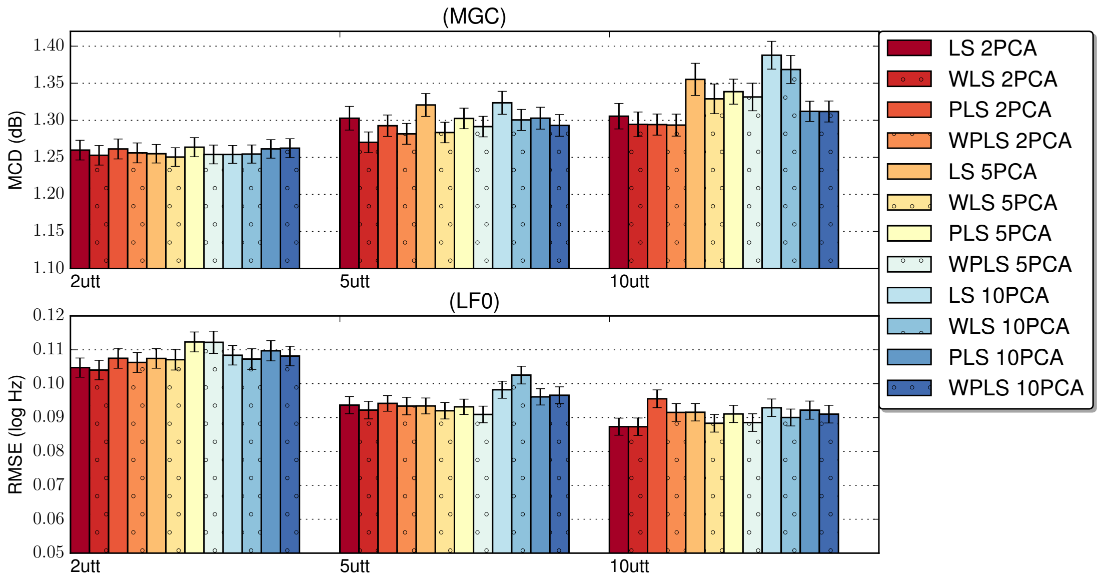


Figure 32: Objective evaluation (RMSE and MCD) of the LS, WLS, PLS, and WPLS algorithms for MGC and LF0 features using 2, 5 and 10 dimensional PCA with 95% confidence intervals. The plots for “2utt”, “5utt” and “10utt” correspond to 2, 5 and 10 utterances of adaptation data. Note that for the LF0 features, a difference of 0.01 log(Hz) corresponds to 17.3 cents.

Performance of the LS, WLS, PLS, and WPLS algorithms for the MGC and LF0 features is shown in Figure 32.

MGC Features: For the 2 utterance case, the differences between the algorithms are not significant. For the 5 utterance case, WLS performed significantly better than LS for all PCA sizes but WPLS is not significantly better than PLS. In the 10 utterance case, for 2 and 5 dimensional PCA, all four algorithms performed equally well. For the 10 dimensional PCA case, PLS and WPLS substantially outperformed the LS and WLS algorithms. This is expected, since the variances of the eigenvectors increase with more data and it becomes harder to predict the English eigenvectors using linear regression. By exploiting correlations between eigenvector elements, PLS is able to do the regression in a lower dimensional space and avoid overfitting.

Note that objectively-measured performance of linear regression algorithms generally becomes worse with increasing data: models deviate further from the AVM. The small number of training speakers and non-linear relationship between input and output eigenvectors cause degradation. Thus, for the MGC features, performance with 2 utterances is actually better than with 5 or 10 utterances.

LF0 Features: Weighting the samples did not generally have a significant effect on performance in the 2 utterance case (except for 5-dimensional PCA), as shown in Figure 32. For 5-dimensional PCA, partial least-squares (PLS, WPLS) is worse than straightforward least-squares (LS, WLS).

For the 5 utterance case, all algorithms performed equally well, except that LS and WLS were significantly worse than the others for 10-dimensional PCA. Similarly to the situation for MGC features, the partial least-squares (PLS) algorithm solved the overfitting exhibited by least-squares (LS, WLS) for the 5 utterance, 10-dimensional PCA case.

In the 10 utterance, 2-dimensional PCA case, least-squares (LS, WLS) outperformed partial least-squares (PLS, WPLS); this is as expected, because the correlations between the elements of the eigenvoice vectors are minimal for the 2-dimensional case, as we saw in Figure 22.

In contrast to MGC features, linear regression for LF0 performed better with more data. That is, the linear regression approach performs better for lower dimensional feature vectors. Moreover, degradation of performance with higher PCA sizes did not occur for LF0, except for the 5 utterance, 10-dimensional PCA case; this can be solved with PLS or WPLS.

6.2.4 Objective Performance of the rPLS Algorithm

The results above show that weighting the samples sometimes improves (and never reduces) the performance of LS and PLS. Hence, the remaining objective evaluations are presented for weighted least-squares (WLS, WPLS) only. In Figure 33, performance of the weighted least-squares (WLS, WPLS) algorithms is compared with the recursive variants (rPLS, WRPLS). Although rPLS did not perform well (at most PCA sizes for the 5 utterance and 10 utterance cases, for MGC features), WRPLS was consistently the best performing algorithm for all amounts of data and at all PCA dimensions. This indicates that weighting is effective and should be speaker-specific.

Note that the rPLS algorithm works independently for each element of the eigenvector in the output language. This means that any correlations between elements of the vector violate the independence assumption and are therefore likely to degrade performance. Covariance matrices for the MGC features is shown in Figure 22: even though the matrix for 2-dimensional PCA is diagonal, substantial covariances can be observed for the 5- and 10-dimensional cases; this explains the relatively poor performance of rPLS for MGC features (Figure 33.)

For LF0, no particular algorithms consistently and significantly outperforms the others. This is probably because of relatively weak correlations between the elements of LF0 eigenvectors (cf. Figure 22).

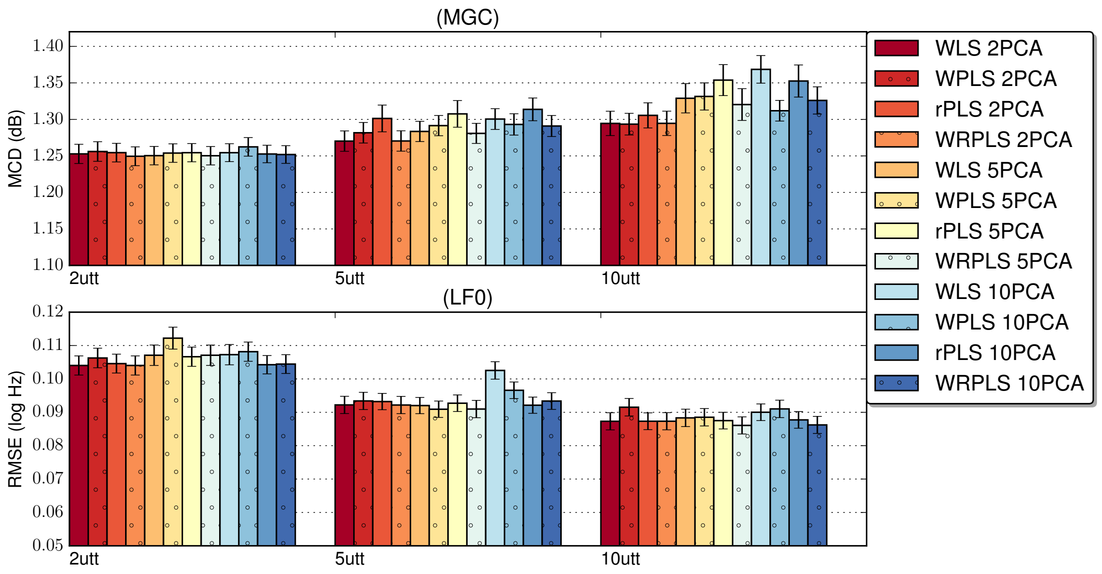


Figure 33: Objective performance (RMSE for LF0 and MCD for MGC features) of the WLS, WPLS, rPLS, and WRPLS algorithms using 2, 5 or 10 dimensional PCA; 95% confidence intervals are shown. Note that for the LF0 features, a difference of 0.01 log(Hz) corresponds to 17.3 cents.

6.2.5 Direct Comparison of the Best Performing Algorithms

WRPLS, which is the best performing linear regression based algorithm, is now compared with the best performing data-mapping algorithm, Cross-BEA. Figure 34 shows the performance of these approaches across different amounts of data and different PCA dimensions. The performance of intra-lingual adaptation is included in the figure, as an upper bound.

For the MGC features, WRPLS outperforms Cross-BEA when only 2 utterances are available; the two algorithms become comparable with 5 utterances, and Cross-BEA outperforms WRPLS algorithm (at all PCA dimensions) when there are 10 utterances. The performance gap between the algorithms increases with PCA dimension.

The situation is reversed for LF0 features. With only 2 utterances, Cross-BEA performs better than WRPLS (at all PCA dimensions). With 5 utterances, WRPLS and Cross-BEA perform similarly, then WRPLS slightly outperforms Cross-BEA when there are 10 utterances.

NN-based state-mapping with CSMAPLR was also compared with WRPLS and Cross-BEA for the LF0 feature and it outperformed them both substantially in the 5 and 10 utterances cases. For those relatively larger data sizes, even though a more accurate state mapping is available, Cross-BEA is not able to exploit the data effectively because its performance has already saturated. In contrast, CSMAPLR performance keeps improving with increasing data (for the LF0 features). This also partly explains why WRPLS outperforms Cross-BEA with increasing data size.

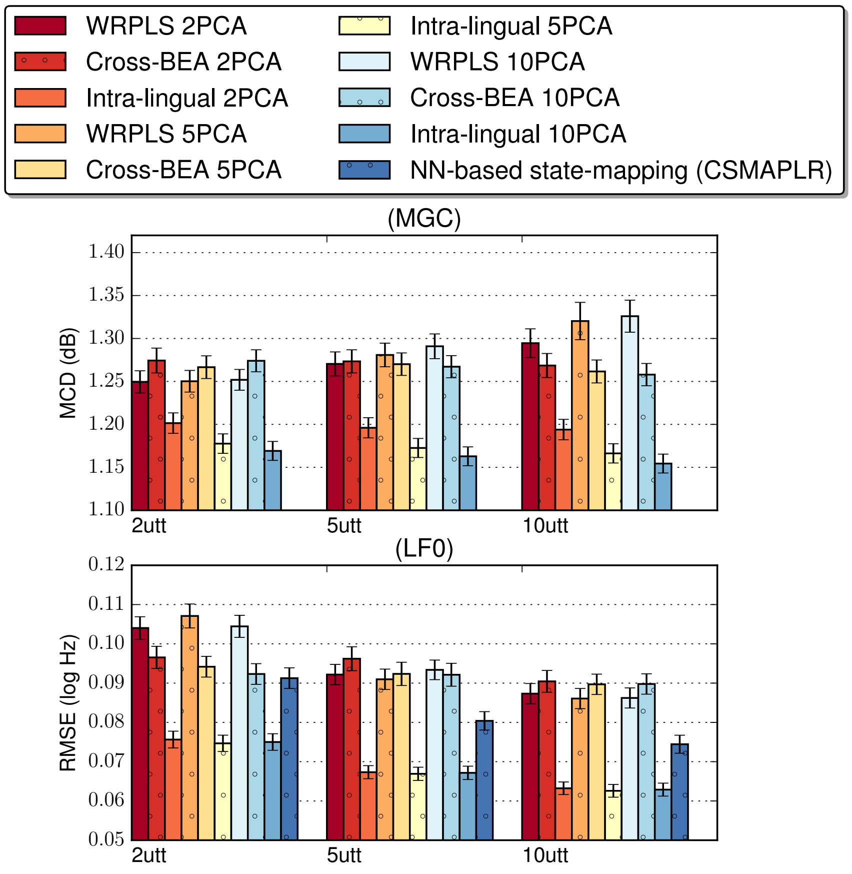


Figure 34: Objective evaluation (RMSE of LF0 and MCD of MGC features) of the best performing algorithms WRPLS and Cross-BEA. NN-based state-mapping is shown for LF0 only. 95% confidence intervals are shown. Intra-lingual adaptation performance is included as an upper-bound. Note that for the LF0 features, a difference of 0.01 log(Hz) corresponds to 17.3 cents.

6.2.6 Summary of Objective Performance

A large number of objective comparison tests have been presented above. The most important findings are:

- NN-based state-mapping outperforms baseline state-mapping for both MGC and LF0 features. This is shown by objective experiments presented in Figure 31. Thus, using speaker-dependent state-mapping was found to be effective compared to speaker-independent state-mapping.

- Cross-BEA performs substantially better than the CSMAPLR algorithm for the MGC features as shown in Figure 31. Hence, the CSMAPLR algorithm could not adapt the high-dimensional MGC features as well as the eigenvoice adaptation algorithm with the limited data.

- Using the nearest-neighbour model without any further adaptation performed significantly better than the baseline system as shown in the Figure 31. This indicates that a nearest-neighbour model trained with intra-lingual adaptation is preferable to a model trained with the baseline algorithm using limited data if the nearest-neighbour sounds similar to the target speaker.

- For the MGC features, eigenvector mapping becomes relatively less effective with increasing adaptation data. Importance weighting and PLS regression improved the performance, although combining them together did not further improve performance as shown in Figure 32 and Figure 33. PLS approach helped reduce the overfit problem because it does regression in a lower dimensional space. Importance weighting addresses the non-linear relationship between the input and output vectors during regression by assuming piecewise linearity. One reason the combination of the two did not further improve the performance could be because of a reduction in non-linearity in the lower dimensional space that the PLS regression operates in.

- For LF0, eigenvector mapping becomes more effective with increasing adaptation data size. Because the feature dimensionality is much lower for LF0, even the basic least-squares (LS) approach performs well, regardless of the amount of adaptation data as shown in Figure 32 and Figure 33.

- rPLS did not perform well, presumably because of correlations in the features. However, weighting remedied this substantially and WRPLS was the best performing algorithm for MGC and LF0, along with WLS as shown in Figure 33.

- Performance degrades significantly with increasing PCA size for all regression algorithms, especially with 5 or 10 utterances, due to overfitting and non-linearities; the issue is more significant for MGC features as shown in Figure 32 and Figure 33.

- For the MGC features, Cross-BEA performs better than the best performing regression method, WRPLS, with the largest amount of adaptation data (10 utterance). WRPLS performs better when only 2 adaptation utterances are available. The converse is true for LF0 as shown in Figure 34. The LF0 features are in a far smaller space compared to the MGC features and 2 utterances are enough for an effective Cross-BEA adaptation whereas larger amount of data is needed for the MGC features. WRPLS performs better than Cross-BEA for the LF0 features with larger data possibly because the relationship between the input and the output eigenvectors is more linear compared to the MGC case.

- NN-based state-mapping with CSMAPLR substantially outperforms both WR-PLS and Cross-BEA for LF0 as shown in Figure 31 and Figure 34. Thus, CSMAPLR can do effective adaptation for the low-dimensional LF0 features with limited amounts of data and eigenvoice based techniques are not necessary if CSMAPLR is used with the NN-based approach.

6.2.7 Objective Performance of the DNN-based Algorithm

Based on the initial experiments, if distribution of linguistic features for train and adaptation speakers be different, performance of adaptation will be degraded. For analyzing the distribution of linguistic features, mean and variance of these features were compared. Comparison between mean and variance of linguistic features in train and cross-lingual adaptation sets are shown in Figures 35 and 36, respectively.

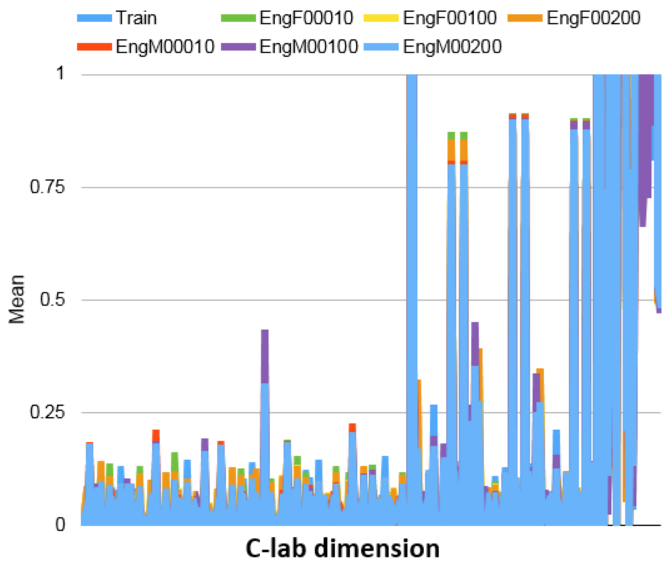


Figure 35: Comparison between mean of linguistic features in train and cross-lingual adaptation sets.

Based on these results, the difference between the distribution of the continuous linguistic features in train and cross-lingual adaptation sets is negligible. As a result, using continuous labels, distribution of estimated linguistic features from acoustic features in input language, will be similar to distribution of estimated linguistic features in output language. In other word, using continuous labels, the difference between linguistic contents of two languages will be negligible and it will enhance the performance of cross-lingual adaptation. Using BLSTM-RNN model for sequence estimation is one of the main reasons for reaching to this result.

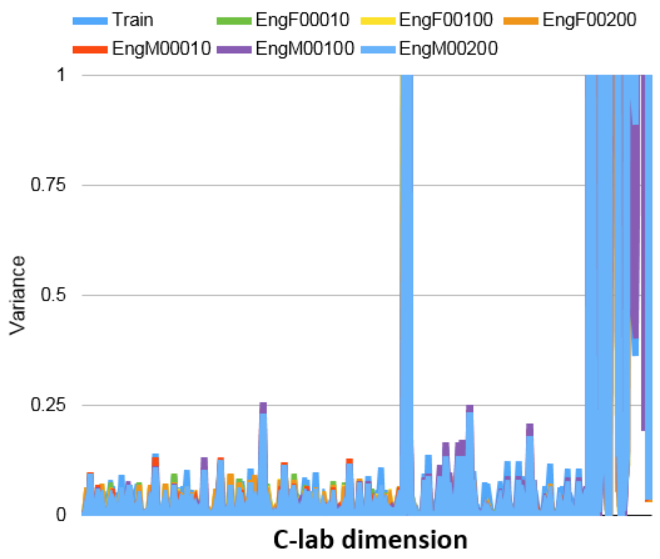


Figure 36: Comparison between variance of linguistic features in train and cross-lingual adaptation sets.

Similar to HMM-based methods, root-mean-square-error (RMSE) is used for objectively measuring the distortion in LF0 features, with respect to natural references. Mel-cepstral distortion (MCD) [116] is used for measuring the distances in MGC and BAP features. As continuous labels are extracted from each frame of acoustic features of target speaker's data, there is no need for duration estimation in adaptation time. As mentioned before, for test set, using HSMM alignment the oracle duration version of time-aligned full context labels were generated. With setting the speaker code to average of all training speaker codes, gender to 0.5, and age to average age (35), the speaker independent (SI) model was generated. For fair comparison between the SI model and adapted models, age of adapted speakers were assumed as average age (35). Objective evaluation (RMSE and MCD) of SI, intra- and cross-lingual adapted models are shown in Figure 37.

Based on objective results, for male speaker, except LF0 feature, the performance of all models for all features is significantly better than SI model. In MGC feature, increasing the adaptation size will improve the performance of adaptation. In this

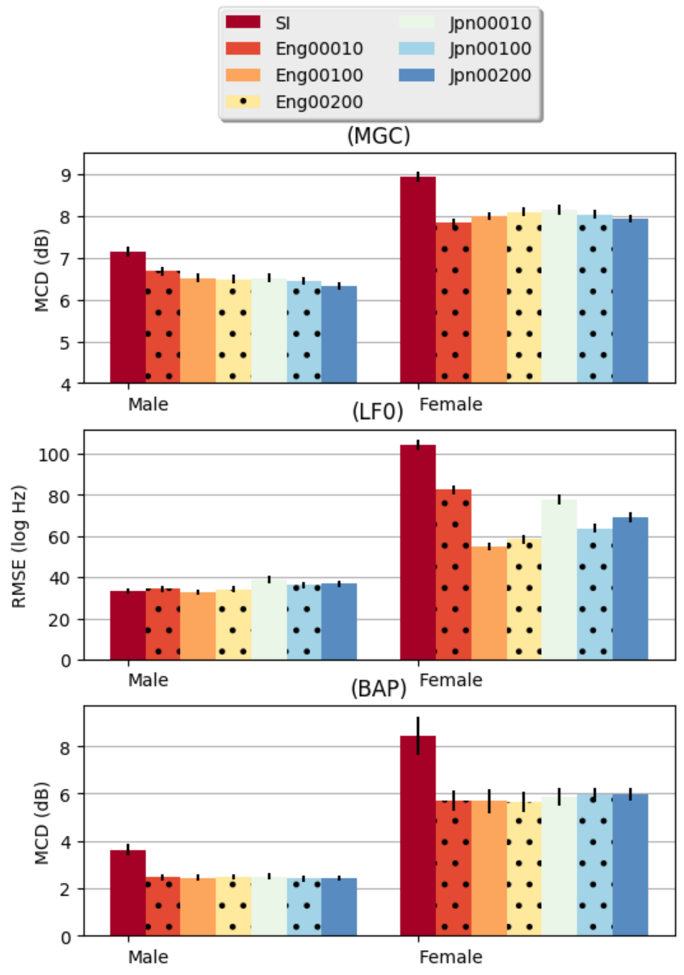


Figure 37: Objective evaluation (RMSE and MCD) of SI, intra- and cross-lingual adapted DNN-Based models with 95% confidence intervals. Description of the target speakers for DNN-based intra- and cross-lingual adaptation approaches are shown in Table 3 and 4, respectively.

case, in average performance of intra-lingual adaptation was better that cross-lingual adaptation. But, when 100 utterances was used for adaptation, the difference between them wasn't significant. For LF0, cross-lingual adaptation outperformed the intra-lingual adaptation models. But it doesn't show improvement with respect to SI model. One reason for this observation can be the difference between channels of train and adaptation sets. For the BAP features, although all the adaptation models outperformed the SI model, increasing the adaptation data from 10 utterances to 200 utterances will not improve the performance of adaptation.

The observed pattern for female speaker was different. The mismatch between recording channels of male and female speakers is one possible reason for this observation. For the female speaker, the performance of all models for all features is significantly better than SI model. For MGC features, increasing the adaptation data size improved the performance of intra-lingual adaptation models, but this increasing degrades the performance of cross-lingual adaptation cases. In average, the difference between intra-lingual and cross-lingual adaptation performance wasn't significant. For LF0 feature, increasing the adaptation data from 10 utterances to 100 utterances significantly improved the performance of adaptation but adding more utterances doesn't improve the adaptation performance. In BAP feature, similar to the male case, although all the adaptation models outperformed the SI model, increasing the adaptation data from 10 utterance to 200 utterances will not improve the performance of adaptation.

6.3 Subjective Evaluation

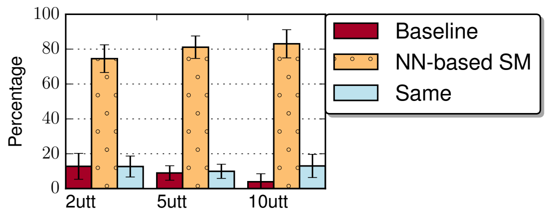
6.3.1 Speaker Similarity Tests in HMM-based Methods

In HMM-based methods, to subjectively measure the similarity of the adapted speaker to the target speaker we employed ABX testing. As with the objective measures, synthetic speech from speaker-dependent models was used as the reference X. Listeners were asked to select which of the speakers of sample A or sample B was more similar to this, or to indicate that samples A and B sounded the same in terms of similarity to X. The A and B samples were synthesized from different adaptation methods randomly. 10 target speakers were selected randomly and, for each speaker, five English sentences from the WSJ1 database were synthesized for each amount of adaptation data (2, 5, or 10 utterances). The tests were done in two phases. In the first phase, 12 native (10 female and 2 male) listeners and 2 non-native male listeners took the tests in soundproof booths and they all listened to one utterance from each speaker. Even though those utterances were different for different speakers, they were the same for all listeners given a speaker. In the second phase, a different set of 12 gender-balanced native English speakers took the tests. In this phase, each listener judged one utterance from each speaker and the utterances were randomly selected out of four utterances synthesized for each speaker. Results from the two phases are combined for analysis.

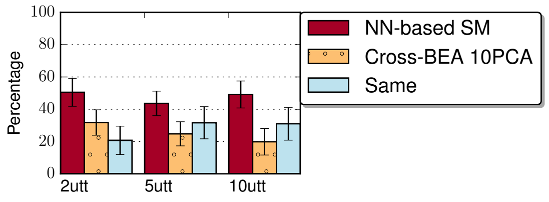
The average age of listeners was 22 years. The stimuli were presented over headphones and listener responses were collected via a simple web browser interface. Listeners could play the A, B and X samples as many times as they desired and they were informed about that before the test. However, they were not encouraged or discouraged to do that. In each test, 30 samples were played to each listener and in average it took 15 minutes to finish the test. The text was the same in A, B and X within a single presentation.

Guided by the objective results, four subjective ABX tests were designed. In the first, the performance of the baseline state-mapping algorithm, generic state-mapping with no information from the target speaker, with CMLLR for LF0 was compared with the proposed NN-based state-mapping algorithm with CSMAPLR; this was the best performing algorithm for LF0 according to objective measures. In both cases, Cross-BEA (10-dimensional PCA) was used to generate the MGC features. The results are shown in Figure 38a. Clearly, the NN-based state-mapping algorithm substantially outperforms the baseline state-mapping algorithm (which uses the same state-map for all speakers).

In the second experiment, the proposed NN-based state-mapping algorithm with CSMAPLR for LF0 was compared with Cross-BEA (10-dimensional PCA). As before, Cross-BEA (10-dimensional PCA) was used to generate the MGC features. Results are shown in Figure 38b. Even though the gap is not as dramatic as in the first experiment, we see that the proposed NN-based state-mapping approach significantly



(a) LF0 generated with either the baseline vs. the NN-based state-mapping (SM).



(b) LF0 generated using the NN-based state mapping vs. the Cross-BEA with 10 dimensional PCA.

Figure 38: Listeners' preferences for the speaker similarity of synthetic speech in which LF0 was generated using different adaptation algorithms. 95% confidence intervals are shown. 2, 5, or 10 utterances were used for adaptation.

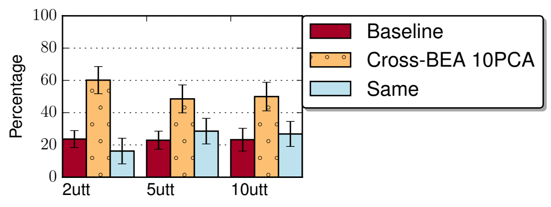
outperformed the Cross-BEA algorithm, for all adaptation data amounts.

In the third experiment, MGC features generated using the baseline state-mapping algorithm with CSMAPLR were compared with those from Cross-BEA (10-dimensional PCA), which was the best performing algorithm for the MGC features according to the objective measure (MCD). The NN-based state mapping algorithm was used to generate LF0 in both cases. Results are shown in Figure 39a where we can see that Cross-BEA is substantially preferred over the baseline system.

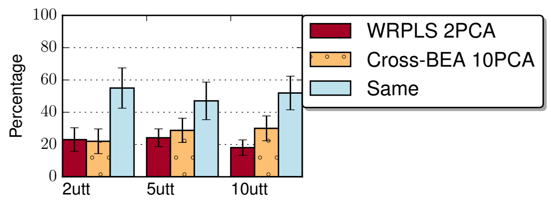
In the final ABX experiment, the WRPLS algorithm was compared with Cross-BEA (10-dimensional PCA) for MGC features. Again, the NN-based state mapping algorithm was used to generate F0. Results are shown in Figure 39b which reveals that listeners had no particular preference for WRPLS or Cross-BEA.

6.3.2 Speech Quality Tests in HMM-based Methods

For evaluation of the speech quality with the proposed methods, the MUSHRA (MUltiple Stimuli with Hidden Reference and Anchor) test was conducted. The samples



(a) Baseline vs. Cross-BEA with 10-dimensional PCA.



(b) WRPLS with 2-dimensional PCA vs. Cross-BEA with 10-dimensional PCA.

Figure 39: Listeners' preferences for the speaker similarity of synthetic speech in which MGC features were generated using different adaptation algorithms. 95% confidence intervals are shown. 2, 5, or 10 utterances were used for adaptation.

were synthesized using the models generated with the best performing proposed adaptation methods and the baseline method in a random order. Five target speakers were selected randomly and, for each speaker, five English sentences from the WSJ1 database were synthesized with the models adapted with 2 and 10 utterances.

14 native (7 female and 7 male) listeners took the tests in soundproof booths. The average age of listeners was 24 years. The test is composed of 25 sets where each set contains 9 stimuli of the same sentence generated by each of the four adaptation systems (baseline, NN-base state-mapping with CSMAPLR, WRPLS, and Cross-BEA) for the 2 and 10 utterance adaptation data cases. Synthetic speech from speaker-dependent models was used as the hidden reference. The listeners were asked to rate each stimulus from 0 (extremely bad in naturalness aspect) to 100 (same as natural speech).

The MUSHRA test results are presented in Figure 40. Paired t-test was used to assess the significance of difference between the systems. For adaptation with 2 utterances, all proposed methods performed significantly better than the baseline system. However, the proposed methods were not found to be significantly different from each other. For adaptation with 10 utterances, the differences between the baseline system, WRPLS 2PCA and Cross-BEA 10PCA methods were not significant but the NN-based state-mapping method performed significantly better than them. Increasing the adaptation data size improved the performances of the baseline and the NN-based state mapping methods. But it does not have a significant effect on the WRPLS 2PCA and the Cross-BEA 10PCA methods.

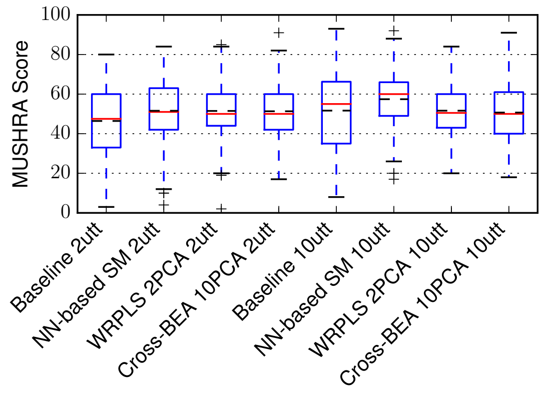


Figure 40: Box plot of MUSHRA result for quality evaluation of the best performing algorithms. The bottom and top of each box are the first and third quartiles, respectively. Ends of the whiskers represent 1.5*IQR (InterQuartile Range)* distances from the first and third quartiles. Outliers are shown with "+" character. Median and mean of each box are shown with solid and dashed lines, respectively.

6.3.3 Speaker Similarity and Quality Tests in DNN-based Method

For subjective evaluation of the DNN-based models in terms of speech quality and speaker similarity, standard 5-point mean opinion score (MOS) and degrading mean opinion score (DMOS) tests were conducted. For fair comparison of the models in similarity aspect, we used speaker dependent (SD) model of each target speaker as a reference model. For male speaker, 1500 utterances were used for training the SD model. For female speaker, because of the limitation in adaptation data, 872 utterances were used for training the SD model. For fair comparison with SI model, with setting the gender code to 0 and 1, gender dependent SI model was generated for

Table 6: Subjective evaluation of DNN-based models. Variance of scores is shown with 95% confidence intervals.

Model	MOS	DMOS
SI_M	2.76 ± 0.05	2.40 ± 0.05
EngM00010	2.86 ± 0.04	2.19 ± 0.04
EngM00100	2.74 ± 0.04	2.14 ± 0.03
EngM00200	2.67 ± 0.03	1.95 ± 0.04
JpnM00010	2.62 ± 0.04	2.12 ± 0.03
JpnM00100	2.78 ± 0.03	2.49 ± 0.04
JpnM00200	2.92 ± 0.04	2.23 ± 0.04
SI_F	2.85 ± 0.05	2.00 ± 0.04
EngF00010	2.99 ± 0.04	2.16 ± 0.05
EngF00100	2.97 ± 0.04	2.29 ± 0.04
EngF00200	2.98 ± 0.03	2.12 ± 0.05
JpnF00010	3.06 ± 0.04	2.21 ± 0.05
JpnF00100	2.81 ± 0.03	2.32 ± 0.03
JpnF00200	2.66 ± 0.03	1.99 ± 0.04

female and male, respectively. For each listener, one gender was selected randomly and 2 random samples was selected out of 100 samples and for each sample, 7 stimuli (3 adaptation size $\times$ 2 adaptation type + gender dependent SI) were played in random order. In each screen, we asked the listener to score the sample from 1 (totally poor quality) to 5 (natural level quality) for MOS test. In addition, for DMOS test, we asked the listener to score the samples from 1 (totally different) to 5 (totally similar) with respect to the reference sample. 789 native Japanese listeners took the test via crowdsourcing[4]. In average, each test took five minutes per listener. The result of subjective test is shown in Table 6.

In average, using continuous labels showed better result in quality rather than similarity. In male speaker, in cross-lingual adaptation case, increasing the adaptation data degrade the quality and similarity of the synthesized speech. However, in intra-lingual case, increasing the data improved the adaptation performance. In cross-lingual adaptation case, just using 10 utterances of adaptation data improved the

[4]https://www.crowdflower.com/

adaptation performance. However, in intra-lingual case, using 100 and 200 of adaptation data showed better performance. In female speaker, in cross-lingual adaptation case, increasing the adaptation data does not have any significant effect of adaptation performance and all cases showed better result with respect to average voice model. In intra-lingual case, increasing the data degrade the adaptation performance, and just using 10 utterance of adaptation data showed improvement in quality of the synthesized speech. In short, using continuous labels, quality and similarity of cross-lingual adaptation can be comparable with intra-lingual adaptation and increasing the adaptation data size will enhance the quality and similarity of the synthesized speech.

CHAPTER VII

CONCLUSION AND FUTURE WORK

We have investigated a variety of cross-lingual speaker adaptation algorithms for HMM-and DNN-based speech synthesis systems, with the specific use case of small amounts of adaptation data from the target speaker. This scenario is motivated by practical applications, in which users are unlikely to be patient enough to provide many minutes or hours of their speech.

In HMM-based systems, we proposed two approaches and compared them objectively and subjectively using a Turkish-English bilingual voice database. In the first proposed approach, a speaker-specific state-mapping is constructed in which the state-map belonging to the nearest-neighbour (NN) speaker to the target speaker is used for adaptation. In the second proposed approach, linear regression is used to relate the eigenvectors of the input and output language acoustic models.

Both approaches performed better than the baseline state-mapping method, both objectively and subjectively. The NN-based state mapping using CSMAPLR adaptation performed the best for LF0. The cross-lingual eigenvoice adaptation technique Cross-BEA performed the best for the MGC feature.

For DNN-based speech synthesis, we proposed a new unsupervised adaptation method using estimation of linguistic features from the sequence of acoustic features. As this method does not assume specific linguistic contents, it can be applied to both intra-lingual and cross-lingual adaptation scenarios. Performance of the proposed method was shown using objective and subjective experiments. Increasing the adaptation data showed improvement in adaptation quality and it outperformed the average voice model.

Even though the algorithms that are proposed here are language-independent, experimenting with them for other language pairs is also interesting and will be investigated in future work. Cross-lingual adaptation between languages that are acoustically more similar to each other than the Turkish-English pair, Spanish and French or Turkic languages for example, will be the focus of our future work. Using continuous labels with other DNN-based adaptation methods like adding i-vectors in input layer can be investigated as future works.

Bibliography

[1] X. Huang, A. Acero, H.-W. Hon, and R. Foreword By-Reddy, *Spoken language processing: A guide to theory, algorithm, and system development*. Prentice hall PTR, 2001.

[2] H. Zen, K. Tokuda, and A. W. Black, "Statistical parametric speech synthesis," *Speech Communication*, vol. 51, no. 11, pp. 1039–1064, 2009.

[3] J. Yamagishi and T. Kobayashi, "Average-voice-based speech synthesis using hsmm-based speaker adaptation and adaptive training," *IEICE TRANSACTIONS on Information and Systems*, vol. 90, no. 2, pp. 533–543, 2007.

[4] K. Tokuda, Y. Nankaku, T. Toda, H. Zen, J. Yamagishi, and K. Oura, "Speech synthesis based on hidden markov models," *Proceedings of the IEEE*, vol. 101, no. 5, pp. 1234–1252, 2013.

[5] H. Ze, A. Senior, and M. Schuster, "Statistical parametric speech synthesis using deep neural networks," in *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pp. 7962–7966, IEEE, 2013.

[6] Y. Fan, Y. Qian, F.-L. Xie, and F. K. Soong, "Tts synthesis with bidirectional lstm based recurrent neural networks," in *Fifteenth Annual Conference of the International Speech Communication Association*, 2014.

[7] S. Matsuda, X. Hu, Y. Shiga, H. Kashioka, C. Hori, K. Yasuda, H. Okuma, M. Uchiyama, E. Sumita, H. Kawai, *et al.*, "Multilingual speech-to-speech translation system: VoiceTra," in *Mobile Data Management (MDM), 2013 IEEE 14th International Conference on*, vol. 2, pp. 229–233, IEEE, 2013.

[8] K. Oura, J. Yamagishi, M. Wester, S. King, and K. Tokuda, "Analysis of unsupervised cross-lingual speaker adaptation for HMM-based speech synthesis using KLD-based transform mapping," *Speech Communication*, vol. 54, no. 6, pp. 703–714, 2012.

[9] H. Liang, Y. Qian, F. K. Soong, and G. Liu, "A cross-language state mapping approach to bilingual (Mandarin-English) TTS," in *Acoustics, Speech and Signal Processing, 2008. ICASSP 2008. IEEE International Conference on*, pp. 4641–4644, IEEE, 2008.

[10] Y.-N. Chen, Y. Jiao, Y. Qian, and F. K. Soong, "State mapping for cross-language speaker adaptation in TTS," in *Acoustics, Speech and Signal Processing, 2009. ICASSP 2009. IEEE International Conference on*, pp. 4273–4276, IEEE, 2009.

[11] Y.-J. Wu, Y. Nankaku, and K. Tokuda, "State mapping based method for cross-lingual speaker adaptation in HMM-based speech synthesis.," pp. 528–531, Interspeech, 2009.

[12] H. Liang and J. Dines, “An analysis of language mismatch in HMM state mapping-based cross-lingual speaker adaptation,” in *Interspeech*, pp. 622–625, 2010.

[13] X. Peng, K. Oura, Y. Nankaku, and K. Tokuda, “Cross-lingual speaker adaptation for HMM-based speech synthesis considering differences between language-dependent average voices,” in *Signal Processing (ICSP), 2010 IEEE 10th International Conference on*, pp. 605–608, IEEE, 2010.

[14] D. Nagahama, T. Nose, T. Koriyama, and T. Kobayashi, “Transform mapping using shared decision tree context clustering for HMM-based cross-lingual speech synthesis,” in *Fifteenth Annual Conference of the International Speech Communication Association*, 2014.

[15] A. Mohan and R. Rose, “Multi-lingual speech recognition with low-rank multi-task deep neural networks,” in *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*, pp. 4994–4998, IEEE, 2015.

[16] N. T. Vu, D. Imseng, D. Povey, P. Motlicek, T. Schultz, and H. Bourlard, “Multilingual deep neural network based acoustic modeling for rapid language adaptation,” in *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*, pp. 7639–7643, IEEE, 2014.

[17] G. Heigold, V. Vanhoucke, A. Senior, P. Nguyen, M. Ranzato, M. Devin, and J. Dean, “Multilingual acoustic models using distributed deep neural networks,” in *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pp. 8619–8623, IEEE, 2013.

[18] N. Dehak, P. J. Kenny, R. Dehak, P. Dumouchel, and P. Ouellet, “Front-end factor analysis for speaker verification,” *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 19, no. 4, pp. 788–798, 2011.

[19] S. Xue, O. Abdel-Hamid, H. Jiang, L. Dai, and Q. Liu, “Fast adaptation of deep neural network based on discriminant codes for speech recognition,” *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)*, vol. 22, no. 12, pp. 1713–1725, 2014.

[20] H.-T. Luong, S. Takaki, G. E. Henter, and J. Yamagishi, “Adapting and controlling dnn-based speech synthesis using input codes,” in *Acoustics, Speech and Signal Processing (ICASSP), 2017 IEEE International Conference on*, pp. 4905–4909, IEEE, 2017.

[21] P. Swietojanski and S. Renals, “Learning hidden unit contributions for unsupervised speaker adaptation of neural network acoustic models,” in *Spoken Language Technology Workshop (SLT), 2014 IEEE*, pp. 171–176, IEEE, 2014.

[22] Z. Wu, P. Swietojanski, C. Veaux, S. Renals, and S. King, "A study of speaker adaptation for dnn-based speech synthesis.," in *INTERSPEECH*, pp. 879–883, 2015.

[23] Y. Fan, Y. Qian, F. K. Soong, and L. He, "Multi-speaker modeling and speaker adaptation for dnn-based tts synthesis," in *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*, pp. 4475–4479, IEEE, 2015.

[24] S. S. Sarfjoo and C. Demiroglu, "Cross-lingual speaker adaptation for statistical speech synthesis using limited data," *Interspeech*, pp. 317–321, 2016.

[25] D. H. Klatt, "Software for a cascade/parallel formant synthesizer," *the Journal of the Acoustical Society of America*, vol. 67, no. 3, pp. 971–995, 1980.

[26] E. Moulines and F. Charpentier, "Pitch-synchronous waveform processing techniques for text-to-speech synthesis using diphones," *Speech communication*, vol. 9, no. 5-6, pp. 453–467, 1990.

[27] A. J. Hunt and A. W. Black, "Unit selection in a concatenative speech synthesis system using a large speech database," in *Acoustics, Speech, and Signal Processing, 1996. ICASSP-96. Conference Proceedings., 1996 IEEE International Conference on*, vol. 1, pp. 373–376, IEEE, 1996.

[28] R. E. Donovan and E. Eide, "The ibm trainable speech synthesis system.," in *ICSLP*, 1998.

[29] T. Hirai and S. Tenpaku, "Using 5 ms segments in concatenative speech synthesis," in *Fifth ISCA Workshop on Speech Synthesis*, 2004.

[30] R. E. Donovan and P. C. Woodland, "Improvements in an hmm-based speech synthesiser," in *Eurospeech Proceedings: 4th European Conference on Speech Communication and Technology*, vol. 1, pp. 573–576, 1995.

[31] M. Beutnagel, A. Conkie, J. Schroeter, Y. Stylianou, and A. Syrdal, "The at&t next-gen tts system," in *Joint meeting of ASA, EAA, and DAGA*, vol. 1, pp. 18–24, Berlin, Germany, 1999.

[32] A. W. Black and P. A. Taylor, "Automatically clustering similar units for unit selection in speech synthesis.," 1997.

[33] H. Segi, T. Takagi, and T. Ito, "A concatenative speech synthesis method using context dependent phoneme sequences with variable length as search units," in *Fifth ISCA Workshop on Speech Synthesis*, 2004.

[34] A. W. Black, "Unit selection and emotional speech.," in *Interspeech*, 2003.

[35] L. R. Rabiner and B.-H. Juang, "Fundamentals of speech recognition," 1993.

[36] "HTK. [Online]. Available: http://htk.eng. cam.ac.uk/."

[37] "Festival. [Online]. Available: http://www.festvox.org/festival/."

[38] "HTS. [Online]. Available: http://hts.sp. nitech.ac.jp/."

[39] "SPTK. [Online]. Available: http://sp-tk. sourceforge.net/."

[40] T. Toda and K. Tokuda, "A speech parameter generation algorithm considering global variance for hmm-based speech synthesis," *IEICE TRANSACTIONS on Information and Systems*, vol. 90, no. 5, pp. 816–824, 2007.

[41] Y.-J. Wu and K. Tokuda, "Minimum generation error training by using original spectrum as reference for log spectral distortion measure," in *Acoustics, Speech and Signal Processing, 2009. ICASSP 2009. IEEE International Conference on*, pp. 4013–4016, IEEE, 2009.

[42] J. Yamagishi, T. Kobayashi, Y. Nakano, K. Ogata, and J. Isogai, "Analysis of speaker adaptation algorithms for HMM-based speech synthesis and a constrained SMAPLR adaptation algorithm," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 17, no. 1, pp. 66–83, 2009.

[43] K. Kazumi, Y. Nankaku, and K. Tokuda, "Factor analyzed voice models for hmm-based speech synthesis," in *Acoustics Speech and Signal Processing (ICASSP), 2010 IEEE International Conference on*, pp. 4234–4237, IEEE, 2010.

[44] G. Fant, *Accoustic theory of speech production: with calculations based on X-ray studies of Russian articulations.* Mouton & Company, 1970.

[45] T. Fukada, K. Tokuda, T. Kobayashi, and S. Imai, "An adaptive algorithm for mel-cepstral analysis of speech," in *Acoustics, Speech, and Signal Processing, 1992. ICASSP-92., 1992 IEEE International Conference on*, vol. 1, pp. 137–140, IEEE, 1992.

[46] F. Itakura, "Line spectrum representation of linear predictor coefficients of speech signals," *The Journal of the Acoustical Society of America*, vol. 57, no. S1, pp. S35–S35, 1975.

[47] K. Tokuda, T. Kobayashi, T. Masuko, and S. Imai, "Mel-generalized cepstral analysis-a unified approach to speech spectral estimation.," in *ICSLP*, vol. 94, pp. 18–22, 1994.

[48] H. Kawahara, I. Masuda-Katsuse, and A. De Cheveigne, "Restructuring speech representations using a pitch-adaptive time–frequency smoothing and an instantaneous-frequency-based f0 extraction: Possible role of a repetitive structure in sounds," *Speech communication*, vol. 27, no. 3, pp. 187–207, 1999.

[49] G. Freij and F. Fallside, "Lexical stress recognition using hidden markov models," in *Acoustics, Speech, and Signal Processing, 1988. ICASSP-88., 1988 International Conference on*, pp. 135–138, IEEE, 1988.

[50] U. Jensen, R. K. Moore, P. Dalsgaard, and B. Lindberg, "Modelling intonation contours at the phrase level using continuous density hidden markov models," *Computer Speech & Language*, vol. 8, no. 3, pp. 247–260, 1994.

[51] K. Ross and M. Ostendorf, "A dynamical system model for generating f0 for synthesis," in *The Second ESCA/IEEE Workshop on Speech Synthesis*, 1994.

[52] K. Tokuda, T. Masuko, N. Miyazaki, and T. Kobayashi, "Multi-space probability distribution hmm," *IEICE TRANSACTIONS on Information and Systems*, vol. 85, no. 3, pp. 455–464, 2002.

[53] H. Zen, K. Tokuda, T. Masuko, T. Kobayasih, and T. Kitamura, "A hidden semi-markov model-based speech synthesis system," *IEICE transactions on information and systems*, vol. 90, no. 5, pp. 825–834, 2007.

[54] K. Tokuda, H. Zen, and A. W. Black, "An hmm-based speech synthesis system applied to english," in *IEEE Speech Synthesis Workshop*, pp. 227–230, 2002.

[55] K. Shinoda and T. Watanabe, "Mdl-based context-dependent subword modeling for speech recognition," *Acoustical Science and Technology*, vol. 21, no. 2, pp. 79–86, 2001.

[56] T. Yoshimura, K. Tokuda, T. Masuko, T. Kobayashi, and T. Kitamura, "Simultaneous modeling of spectrum, pitch and duration in hmm-based speech synthesis," in *Sixth European Conference on Speech Communication and Technology*, 1999.

[57] S. Imai, K. Sumita, and C. Furuichi, "Mel log spectrum approximation (mlsa) filter for speech synthesis," *Electronics and Communications in Japan (Part I: Communications)*, vol. 66, no. 2, pp. 10–18, 1983.

[58] B. Atal and J. Remde, "A new model of lpc excitation for producing natural-sounding speech at low bit rates," in *Acoustics, Speech, and Signal Processing, IEEE International Conference on ICASSP'82.*, vol. 7, pp. 614–617, IEEE, 1982.

[59] H. Kawai, T. Toda, J. Ni, M. Tsuzaki, and K. Tokuda, "Ximera: A new tts from atr based on corpus-based technologies," in *Fifth ISCA Workshop on Speech Synthesis*, 2004.

[60] S. Krstulovic, J. Latorre, and S. Buchholz, "Comparing qmt1 and hmms for the synthesis of american english prosody," in *Proc. Speech Prosody*, pp. 67–70, 2008.

[61] H. Hon, A. Acero, X. Huang, J. Liu, and M. Plumpe, "Automatic generation of synthesis units for trainable text-to-speech systems," in *Acoustics, Speech and Signal Processing, 1998. Proceedings of the 1998 IEEE International Conference on*, vol. 1, pp. 293–296, IEEE, 1998.

[62] T. Okubo, R. Mochizuki, and T. Kobayashi, "Hybrid voice conversion of unit selection and generation using prosody dependent hmm," *IEICE TRANSACTIONS on Information and Systems*, vol. 89, no. 11, pp. 2775–2782, 2006.

[63] Z.-H. Ling and R.-H. Wang, "Hmm-based hierarchical unit selection combining kullback-leibler divergence with likelihood criterion," in *Acoustics, Speech and Signal Processing, 2007. ICASSP 2007. IEEE International Conference on*, vol. 4, pp. IV–1245, IEEE, 2007.

[64] Z.-H. Ling, "Hmm-based unit selection using frame sized speech segments," *Proc. Interspeech (ICSLP), 2006*, 2006.

[65] J. Yu, M. Zhang, J. Tao, and X. Wang, "A novel hmm-based tts system using both continuous hmms and discrete hmms," in *Acoustics, Speech and Signal Processing, 2007. ICASSP 2007. IEEE International Conference on*, vol. 4, pp. IV–709, IEEE, 2007.

[66] P. Taylor, "Unifying unit selection and hidden markov model speech synthesis," in *Ninth International Conference on Spoken Language Processing*, 2006.

[67] S. Esmeir and S. Markovitch, "Anytime learning of decision trees," *Journal of Machine Learning Research*, vol. 8, no. May, pp. 891–933, 2007.

[68] K. Yu, F. Mairesse, and S. Young, "Word-level emphasis modelling in hmm-based speech synthesis," in *Acoustics Speech and Signal Processing (ICASSP), 2010 IEEE International Conference on*, pp. 4238–4241, IEEE, 2010.

[69] D. Yu and L. Deng, "Deep learning and its applications to signal and information processing [exploratory dsp]," *IEEE Signal Processing Magazine*, vol. 28, no. 1, pp. 145–154, 2011.

[70] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[71] F. A. Gers, N. N. Schraudolph, and J. Schmidhuber, "Learning precise timing with lstm recurrent networks," *Journal of machine learning research*, vol. 3, no. Aug, pp. 115–143, 2002.

[72] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber, "Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks," in *Proceedings of the 23rd international conference on Machine learning*, pp. 369–376, ACM, 2006.

[73] A. Graves, “Sequence transduction with recurrent neural networks,” *arXiv preprint arXiv:1211.3711*, 2012.

[74] C. J. Leggetter and P. C. Woodland, “Maximum likelihood linear regression for speaker adaptation of continuous density hidden markov models,” *Computer Speech & Language*, vol. 9, no. 2, pp. 171–185, 1995.

[75] P. C. Woodland, “Speaker adaptation for continuous density hmms: A review,” in *ISCA Tutorial and Research Workshop (ITRW) on Adaptation Methods for Speech Recognition*, 2001.

[76] J. Yamagishi, M. Tamura, T. Masuko, K. Tokuda, and T. Kobayashi, “A training method of average voice model for hmm-based speech synthesis,” *IEICE transactions on fundamentals of electronics, communications and computer sciences*, vol. 86, no. 8, pp. 1956–1963, 2003.

[77] M. Gales and S. Young, “The application of hidden markov models in speech recognition,” *Foundations and trends in signal processing*, vol. 1, no. 3, pp. 195–304, 2008.

[78] J. Yamagishi, B. Usabaev, S. King, O. Watts, J. Dines, J. Tian, Y. Guan, R. Hu, K. Oura, Y.-J. Wu, *et al.*, “Thousands of voices for hmm-based speech synthesis–analysis and application of tts systems built on various asr corpora,” *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 18, no. 5, pp. 984–1004, 2010.

[79] J.-L. Gauvain and C.-H. Lee, “Maximum a posteriori estimation for multivariate gaussian mixture observations of markov chains,” *IEEE transactions on speech and audio processing*, vol. 2, no. 2, pp. 291–298, 1994.

[80] M. J. Gales, “Maximum likelihood linear transformations for HMM-based speech recognition,” *Computer speech & language*, vol. 12, no. 2, pp. 75–98, 1998.

[81] P. Liu, F. K. Soong, and J.-L. Thou, “Divergence-based similarity measure for spoken document retrieval,” in *Acoustics, Speech and Signal Processing, 2007. ICASSP 2007. IEEE International Conference on*, vol. 4, pp. IV–89, IEEE, 2007.

[82] O. Siohan, T. A. Myrvoll, and C.-H. Lee, “Structural maximum a posteriori linear regression for fast hmm adaptation,” *Computer Speech & Language*, vol. 16, no. 1, pp. 5–24, 2002.

[83] R. Kuhn, J.-C. Junqua, P. Nguyen, and N. Niedzielski, “Rapid speaker adaptation in eigenvoice space,” *Speech and Audio Processing, IEEE Transactions on*, vol. 8, pp. 695 –707, nov 2000.

[84] K. Shichiri, A. Sawabe, T. Yoshimura, K. Tokuda, T. Masuko, T. Kobayashi, and T. Kitamura, "Eigenvoices for HMM-based speech synthesis," in *Seventh International Conference on Spoken Language Processing*, pp. 1269–1272, 2002.

[85] M. J. F. Gales, "Cluster adaptive training of hidden Markov models," *Speech and Audio Processing, IEEE Transactions on*, vol. 8, no. 4, pp. 417–428, 2000.

[86] N. Dehak, P. Kenny, R. Dehak, P. Dumouchel, and P. Ouellet, "Front-end factor analysis for speaker verification," *Audio, Speech, and Language Processing, IEEE Transactions on*, vol. 19, pp. 788–798, may 2011.

[87] A. Mohammadi, S. S. Sarfjoo, and C. Demiroglu, "Eigenvoice speaker adaptation with minimal data for statistical speech synthesis systems using a MAP approach and nearest-neighbors," *Audio, Speech, and Language Processing, IEEE/ACM Transactions on*, vol. 22, no. 12, pp. 2146–2157, 2014.

[88] L. Saheer, J. Dines, P. N. Garner, and H. Liang, "Implementation of vtln for statistical speech synthesis," tech. rep., Idiap, 2010.

[89] C. Breslin, K. Chin, M. J. Gales, K. Knill, and H. Xu, "Prior information for rapid speaker adaptation," in *Eleventh Annual Conference of the International Speech Communication Association*, 2010.

[90] M. J. Gales and R. C. van Dalen, "Predictive linear transforms for noise robust speech recognition," in *Automatic Speech Recognition & Understanding, 2007. ASRU. IEEE Workshop on*, pp. 59–64, IEEE, 2007.

[91] L. Saheer, J. Yamagishi, P. N. Garner, and J. Dines, "Combining vocal tract length normalization with hierarchical linear transformations," *IEEE Journal of Selected Topics in Signal Processing*, vol. 8, no. 2, pp. 262–272, 2014.

[92] D. Garcia-Romero and C. Y. Espy-Wilson, "Analysis of i-vector length normalization in speaker recognition systems.," in *Interspeech*, vol. 2011, pp. 249–252, 2011.

[93] O. Watts, Z. Wu, and S. King, "Sentence-level control vectors for deep neural network speech synthesis," in *Sixteenth Annual Conference of the International Speech Communication Association*, 2015.

[94] O. Abdel-Hamid and H. Jiang, "Rapid and effective speaker adaptation of convolutional neural network based models for speech recognition.," in *Interspeech*, pp. 1248–1252, 2013.

[95] P. Swietojanski and S. Renals, "Differentiable pooling for unsupervised speaker adaptation," in *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*, pp. 4305–4309, IEEE, 2015.

[96] N. Hojo, Y. Ijima, and H. Mizuno, "An investigation of dnn-based speech synthesis using speaker codes.," in *INTERSPEECH*, pp. 2278–2282, 2016.

[97] T. Toda, A. W. Black, and K. Tokuda, “Voice conversion based on maximum-likelihood estimation of spectral parameter trajectory,” *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 15, no. 8, pp. 2222–2235, 2007.

[98] D. Sündermann, H. Höge, A. Bonafonte, H. Ney, and J. Hirschberg, “Tc-star: Cross-language voice conversion revisited,” 2006.

[99] “Emime project: http://www.emime.org.”

[100] Y.-J. Wu, S. King, and K. Tokuda, “Cross-lingual speaker adaptation for HMM-based speech synthesis,” in *Chinese Spoken Language Processing, 2008. ISCSLP’08. 6th International Symposium on*, pp. 1–4, IEEE, 2008.

[101] Y. Fan, Y. Qian, F. K. Soong, and L. He, “Unsupervised speaker adaptation for dnn-based tts synthesis,” in *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on*, pp. 5135–5139, IEEE, 2016.

[102] S. King, K. Tokuda, H. Zen, and J. Yamagishi, “Unsupervised adaptation for HMM-based speech synthesis,” ISCA, 2008.

[103] J. Latorre, K. Iwano, and S. Furui, “New approach to the polyglot speech generation by means of an HMM-based speaker adaptable synthesizer,” *Speech Communication*, vol. 48, no. 10, pp. 1227–1242, 2006.

[104] H. Zen, N. Braunschweiler, S. Buchholz, M. J. Gales, K. Knill, S. Krstulovic, and J. Latorre, “Statistical parametric speech synthesis based on speaker and language factorization,” *Audio, Speech, and Language Processing, IEEE Transactions on*, vol. 20, no. 6, pp. 1713–1724, 2012.

[105] V. d. F. Oliveira, S. Shiota, Y. Nankaku, and K. Tokuda, “Cross-lingual speaker adaptation for HMM-based speech synthesis based on perceptual characteristics and speaker interpolation,” in *Interspeech*, pp. 983–986, 2012.

[106] T. Yoshimura, K. Hashimoto, K. Oura, Y. Nankaku, and K. Tokuda, “Cross-lingual speaker adaptation based on factor analysis using bilingual speech data for HMM-based speech synthesis,” in *8th ISCA Speech Synthesis Workshop*, pp. 317–322, 2013.

[107] M. Charlier, Y. Ohtani, T. Toda, A. Moinet, and T. Dutoit, “Cross-language voice conversion based on eigenvoices,” in *Interspeech*, pp. 1635–1638, 2009.

[108] H. Liao, “Speaker adaptation of context dependent deep neural networks,” in *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pp. 7947–7951, IEEE, 2013.

[109] S. De Jong, “SIMPLS: an alternative approach to partial least squares regression,” *Chemometrics and intelligent laboratory systems*, vol. 18, no. 3, pp. 251–263, 1993.

[110] B. Efron, *The jackknife, the bootstrap, and other resampling plans*, vol. 38. Philadelphia, Pa: Society for Industrial and Applied Mathematics, 1982.

[111] Å. Rinnan, M. Andersson, C. Ridder, and S. B. Engelsen, "Recursive weighted partial least squares (rPLS): an efficient variable selection method using PLS," *Journal of Chemometrics*, vol. 28, no. 5, pp. 439–447, 2014.

[112] M. Morise, F. Yokomori, and K. Ozawa, "World: A vocoder-based high-quality speech synthesis system for real-time applications," *IEICE TRANSACTIONS on Information and Systems*, vol. 99, no. 7, pp. 1877–1884, 2016.

[113] K. Tokuda, T. Yoshimura, T. Masuko, T. Kobayashi, and T. Kitamura, "Speech parameter generation algorithms for hmm-based speech synthesis," in *Acoustics, Speech, and Signal Processing, 2000. ICASSP'00. Proceedings. 2000 IEEE International Conference on*, vol. 3, pp. 1315–1318, IEEE, 2000.

[114] T. Kudo, K. Yamamoto, and Y. Matsumoto, "Applying conditional random fields to japanese morphological analysis.," in *EMNLP*, vol. 4, pp. 230–237, 2004.

[115] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization," *Journal of Machine Learning Research*, vol. 12, no. Jul, pp. 2121–2159, 2011.

[116] R. F. Kubichek, "Mel-cepstral distance measure for objective speech quality assessment," in *Communications, Computers and Signal Processing, 1993., IEEE Pacific Rim Conference on*, vol. 1, pp. 125–128, IEEE, 1993.

VITA

Seyyed Saeed Sarfjoo received the B.Sc. degree in information technology from Isfahan University, Isfahan, Iran, in 2009 and the M.Sc. degree in information technology from Qom University, Qom, Iran, in 2012. In 2009, he joined Asr Gooyesh Pardaz Co, Tehran, Iran, as a Software Developer/Researcher, and worked there till 2013. During that time, he worked on Persian text-to-speech synthesis, Persian speech recognition, and Persian interactive voice response systems. Currently, he is a Ph.D. student in the Computer Science Department of Özyeğin University, Istanbul, Turkey. He is also a graduate research assistant at the Speech Processing Lab of Özyeğin University. In the Ph.D. period, his research mostly focused on intra-lingual and cross-lingual speaker adaptation in statistical parametric speech synthesis.