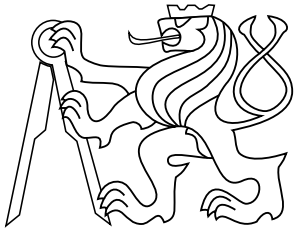
master's thesis

DNN-HMM Based Multilingual Recognizer of Telephone Speech

Bc. Jiří Fiala



May 2016

advisor: Doc. Ing. Petr Pollák, CSc.
second advisor: Ing. Petr Mizera

Czech Technical University in Prague
Faculty of Electrical Engineering, Department of Circuit Theory

Czech Technical University in Prague
Faculty of Electrical Engineering

Department of Cybernetics

DIPLOMA THESIS ASSIGNMENT

Student: Bc. Jiří F i a l a

Study programme: Cybernetics and Robotics

Specialisation: Robotics

Title of Diploma Thesis: DNN-HMM Based Multilingual Recognizer of Telephone Speech

Guidelines:

1. Meet the principles of speech recognition with a special focus on systems based on DNN-HMM architecture.
2. Create multilingual acoustic models for languages in available databases of telephone speech. Realize the system using KALDI toolkit. Based on established conventions, create scripts ("recipes") for defined task which will allow its further usage by the research community in the field of speech recognition.
3. Analyze recognition accuracy on the basis of tasks of phoneme recognition as well as large vocabulary continous speech recognition. Compare achieved results with results obtained using acoustic models created purely for particular languages.

Bibliography/Sources:

[1] J. Psutka, L. Müller, J. Matoušek, V. Radová. Mluvíme s počítačem česky. Academia, 2006.
[2] J. Uhlíř a kol. Technologie hlasových komunikací. Nakladatelství ČVUT, Praha, 2007.
[3] T. Schulz. Multilingual Speech Processing. Academic Press, 2006.
[4] G. Hinton et al, "Deep Neural Networks for Acoustic Modeling in Spech Recognition" The Shared Views of Four Research Groups," IEEE Signal Processing Magazine, vol. 29, no. 6, pp. 82-97, 2012.
[5] D. Povey et al, The Kaldi Speech Recognition Toolkit. In Proc. of IEEE 2011 ASRU, Hawai, US, 2011. Note. Project WEB-page http://kaldi.sourceforge.net/.

Diploma Thesis Supervisor: doc. Ing. Petr Pollák, CSc.

Valid until: the end of the summer semester of academic year 2016/2017

L.S.

prof. Dr. Ing. Jan Kybic
Head of Department

prof. Ing. Pavel Ripka, CSc.
Dean

Prague, December 11, 2015

Acknowledgement

I would like to thank my advisor Doc. Ing. Petr Pollák, CSc. for his guidance and all provided information and also, I would like to thank all the members of the CTU Speechlab for their support and the valuable comments.

Declaration

I declare that the presented work was developed independently and that I have listed all sources of information used within it in accordance with the methodological instructions for observing the ethical principles in the preparation of university theses.

Prague, date _____ _____
signature

Abstract

Tato práce se zabývá problematikou multilingválního akustického modelování založeného na sdíleném fonetickém inventáři pro 5 východoevropských jazyků: češtinu, slovenštinu, polštinu, maďarštinu a ruštinu, které jsou dostupné v souboru databází telefonních signálů SpeechDat-E. Jelikož dostupné výslovnostní slovníky v jednotlivých databázích obsahují výslovnosti za použití SAMPA abeced s neustálenou konvencí napříčí jazyky, není použit pro stejné hlásky v různých jazycích stejný symbol. Byla proto vytvořena jednotící reprezentace na úrovni fonémů pro všechny použité jazyky na bázi X-SAMPA abecedy a tím zajištěna jednotná reprezentace hlásek napříč jazyky. Přínos multilingválního akustického modelu byl analyzován na úloze rozpoznávání spojité řeči. Byla provedena analýza dvou realizací akustického modelování v LVCSR: tj. byl použit standardní GMM-HMM (Gaussian Mixture Model-Hidden Markov Model) a DNN-HMM architektura. Vlastní experimenty byly provedeny pro LVCSR s akustickým modelem pro jednotlivé jazyky a pro mutilingvální systém. Jednotlivé systémy automatického rozpoznávání řeči byly realizovány pomocí nástrojů Kaldi. Jedním z cílů této práce je poskytnout základní návod pro používání Kaldi a vytvořit vzorovou implementaci (angl. *recipe*) s databázemi z řady SpeechDat. V závislosti na jazyku se nejlepší dosažená úspěšnost GMM-HMM sytému pohybovala v rozmezí 18%-28% *WER*. DNN-HMM systém přinesl zlepšení v průměru o 4% *WER*. Pro multi-lingvální HMM systém se pak výsledky pohybovaly v rozmezí od 25%-37% *WER*. Použití DNN přineslo nakonec další výrazné snížení *WER*, v případě multilingválního systému v průměru o 9% pro všech 5 jazyků.

Klíčová slova

rozpoznávání spojité řeči; LVCSR; GMM-HMM systém; DNN-HMM systém; multilingvální systém; akustické modelování; IPA; SAMPA; X-SAMPA; Kaldi

Abstract

This thesis deals with the multilingual acoustic modeling problem based on the shared global phones inventory for five East Eurpoean languages: Czech, Russian, Hungarian, Slovak and Polish which are available within SpeechDat-E, i.e. the set of telephone speech databases. Because the SAMPA with unnormalized convention is used to represent the phonetic content of the particular languages and different symbols are in several cases representing the same phone, the mapping to the general X-SAMPA phonetic alphabet was proposed in the first step. The impact of a multilingual acoustic modeling was analyzed on the basis of a continuous speech recognition. The analysis of the acoustic modeling in the LVCSR task was performed for the GMM-HMM system and for the DNN-GMM approach. The experiments were performed for the LVCSR with the language specific acoustic model same as for the multilingual system. The particular recognizers were implemented via the Kaldi toolkit. One of this thesis goals is to provide a tutorial-style description of the Kaldi usage and create the recipe for the SpeechDat databases. Depending on the language, the best obtained accuracy of HMM recognizers was 18%-28% *WER*. DNN-HMM improved the results about 4% *WER* on average. The results for the multilingual HMM system reached the values from 25%-37% *WER*. The DNN approached had significant impact on the speech recognition accuracy for the multilingual system as well and it reduced the *WER* about 9% on average.

Keywords

continuous speech recognition; LVCSR; GMM-HMM system; DNN-HMM system; multilingual system; multilingual acoustic modeling; IPA; SAMPA; X-SAMPA; Kaldi

Contents

Abbreviations

This is the list of the used abbreviations with their explanation:

abbrv.	explanation
ASR	Automatic Speech Recognition
LM	Language Model
MAP	Maximum A Posteriori probability
MFCC	Mel-Frequency Cepstrum Coefficients
DCT	Discrete Cosine Transform
STFT	Short-Time Fourier Transform
PLP	Perceptual Linear Predictive analysis
IDFT	Inverse Discrete Fourier Transform
LDA	Linear Discriminant Analysis
MLLT	Maximum Likelihood Linear Transform
HMM	Hidden Markov Model
GMM	Gaussian Mixture Model
MMI	Maximal Mutual Information
SGMM	Subspace Gaussian Mixture Model
fMLLR	feature-spaced Maximum Likelihood Linear Regression
ANN	Artifiacial Netural Network
DNN	Deep Neural Network
DBN	Deep Belief Network
WFST	Weighted Finite State Transducers
IPA	International Phonetic Alphabet
SAMPA	Speech Assessment Methods Phonetic Alphabet
X-SAMPA	Extended SAMPA
HTK	Hidden Markov Toolkit

1 Introduction

With the progress of the machine learning algorithms and the hardware computational performance, the speech technologies are becoming vastly popular and are being deployed in a still growing amount of applications. It reflects the fact, that the speech is the most natural way of a human communication, so the technology is desired to be able to understand the human speech. The speech technologies can be found in many applications such as dictation, automatic subtitles generation or archiving the various types of spoken recordings. In general, the speech recognition covers the area from simple commands recognition to the sophisticated dialog systems and artificial personal assistants. Also, the human voice is consider as one of the biometric identifier and can be used for a speaker identification or verification.

The most noticeable progress in the field of a speech recognition brought the introduction of the HMM (Hidden Markov Models). Such acoustic modeling method allowed the new capabilities of recognizing the continuous speech with a large vocabulary and it became the baseline of speech recognition systems. In the recent years, the ANN (Artificial Neural Networks) have provided noticeable results. Their output is often used in the systems based on the TANDEM features or they are a part of the hybrid HMM-DNN recognizers. The ANN variant denoted as deep neural network has proven the promising results for an acoustic modeling and such approach is also investigated in this thesis.

Hand in hand with the increasing speech recognition performance and capabilities, the requirements of such systems are also growing. They have to perform well in the various conditions e.g. the noisy car environment or with the low quality telephone data. Another requirement for such systems is often the need to provide their functionality for various languages. In general, the statistical speech recognition system is a language dependent, so its setup for another language often means to obtain the appropriate training set and reinitiate the training process. But some parts of such system can be shared even for the new language. One such part can be the acoustic model. Phones sharing in the acoustic modeling can result in the need of the smaller training data set. Thus, this can allow to deploy the system for the new language with less effort and cost. With respect to this summary, the multilingual speech modeling is the main topic of this work.

The content is organized as follows. The general principles and approaches of continuous speech recognition are described in the chapter 2. The general speech recognition theory is expanded with used multilingual acoustic modeling approach in the chapter 3. The used phonetic alphabet and a phone inventory unification is also described there. Chapter 4 describes the Kaldi toolkit and provides a tutorial-like manual. The individual experiments are introduced in chapter 5 as well as the discussion of experiment results. The complete thesis evaluation is stated in the last chapter.

2 Continuous Speech Recognition

The first speech recognition attempts were to recognize the isolated words and expressions. The principle of the first recognizers was a template matching. To evaluate and compare two utterances, the dynamic programing was used to model the nonlinear variations in the speech speed of one of the utterances. Such approach is called the dynamic time warping and it was the most used classification method in the 70s and early 80s. During the 80s, the statistical classification methods were introduced and laid down the base for continuos speech recognition. The statistical approach of continuos speech recognition is described in this chapter.

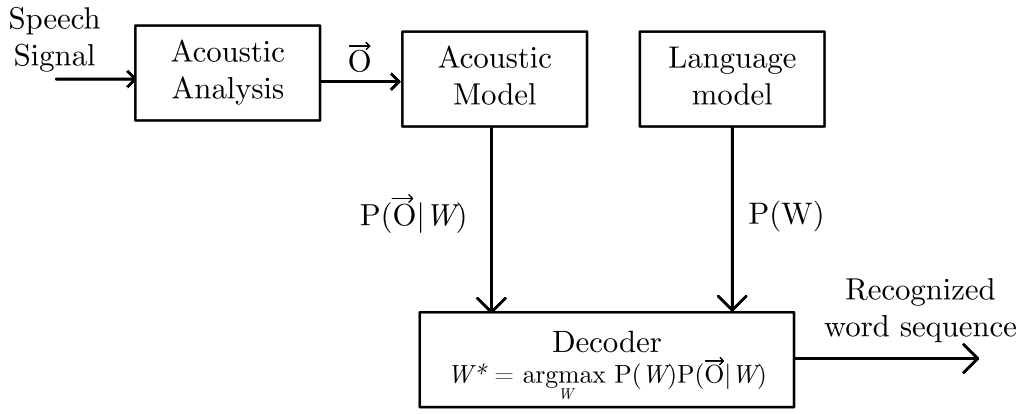


Figure 1 The principle of a statistical large vocabulary speech recognition approach.

An acoustic analysis of the input speech signal performs two main subtasks. The first is the signal processing itself. It can include denoising, echo cancellation, pre-emphasis and other modifications to clean and normalize the input speech audio signal. The main function of the acoustic analysis is to extract the sequence of features that is processed and recognized by a decoder. The elements of this sequence represent the feature vectors in the individual time steps t. Let's denote this sequence as $\mathbf{O} = (\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \ldots, \mathbf{x}_t)$. Let's assume the sequence $W = (w_1, w_2, w_3, \ldots, w_n)$ of n words. Then the sequence of the acoustic observations $\mathbf{O}$ generates W with the probability $P(W|\mathbf{O})$. The key task of the decoder is to find such sequence W^* which maximizes the probability $P(W|\mathbf{O})$, written as

$$W^* = \underset{W}{\text{argmax}}\, P(W|\mathbf{O})\,. \tag{1}$$

Thus, it is a decoding with the maximum a posteriori probability (MAP). The Eq. 1 can be rewritten with Bayes' rule to the form

$$W^* = \underset{W}{\operatorname{argmax}} \frac{P(W)P(\mathbf{O}|W)}{P(\mathbf{O})}. \tag{2}$$

The a priori probability P(W) is the probability that a speaker will say the sequence of words W. $P(\mathbf{O}|W)$ is the probability, that the feature vector sequence is produced when the W sequence is pronounced. The a priory probability of the observation feature sequence $\mathbf{O}$ can be omitted, since it is constant under the max operation which results in

$$W^* = \underset{W}{\operatorname{argmax}} P(W)P(\mathbf{O}|W). \tag{3}$$

As it can be seen, the decoding problem can be decomposed into the evaluation of a two probabilities. These probabilities are independent which means that they can be trained separately. The probability $P(W)$ is called, or represented by, the Language Model (LM), that reflects the semantic and(or) syntactic constrains of the given language. $P(\mathbf{O}|W)$ is determined by an acoustic model. It needs to be stated, that the evaluation of Eq. 3, thus obtaining the W^* for observed $\mathbf{O}$ over all possible sequences W, involves enormous number of operations and it is computationally very expensive. The sophisticated decoding techniques has to be applied to obtain the desired sequence W^*.

To conclude, the statistical continuous speech recognition task can be formulated in the form of the following problems:

- The acoustic processing problem. Signal processing in the time domain to remove or reconstruct missing information. Then, the proper features are needed to be extracted out of the speech signal. It means to find such feature vectors with as low dimension as possible while keeping sufficient amount of information.
- To train appropriate model to evaluate the probability $P(\mathbf{O}|W)$. It means to decide which acoustic units are to be modeled and what evaluation mechanism should be used (HMM, ANN, ...)
- Train the language model and evaluate the probability $P(W)$.
- Obtain the sequence W^* in acceptable time by using the proper methods.

2.1 Acoustic Analysis

Depending on the various conditions like the environment, quality of communication canal or with respect to the nature of a human speech production, the speech signal often suffers from information loss or abundance of misleading information, which is inappropriate for further processing. Between common preprocessing methods belongs

the pre-emphasis, that compensate the energy loss proportionate to the increasing frequency. Regarding the human speech production system, the speech signal is considered as stationary in short time intervals around 10-30 ms during which the current state of production system is being kept. This state corresponds to the sound unit that is then recognized. A further speech signal processing therefore requires the short time analysis both for the time and spectral domain. So, the next step is a signal segmentation. Many experiments proved 25 ms to be an optimal segment width with 10 ms shift. Several windows with different characteristics are used for this purpose. Namely rectangular window, Hanning or the most used Hamming window. The complete description of short-time analyses methods and their principles can be found for example in [14].

Since the time domain reflects every aspects of a signal production and transmission channel, the time steps themselves are not suitable for the direct classification and modeling. The key function of the acoustic analysis is to provide a proper and robust features. The widely used are the Mel-Frequency Cepstrum Coefficients (MFCC) and features based on Perceptual Linear Predictive analysis (PLP) [16] also provided promising results in many speech applications. These methods of the feature computation are briefly described further.

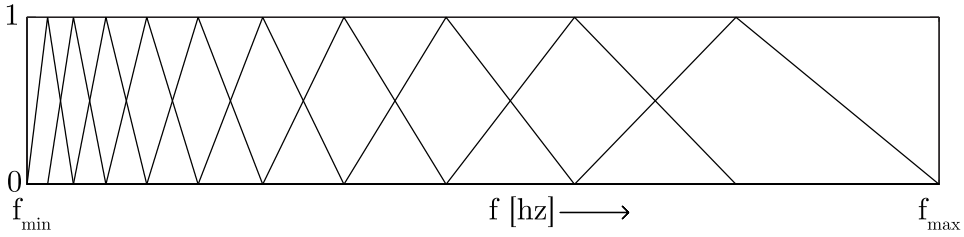
2.1.1 MFCC

The MFCC features are designed with respect to the human audio perception. The human ear does not perceive the frequencies in linear scale but in a logarithmic one. This property is simulated by the application of a filter-bank in the frequency domain. The filter bank consists of triangular filters designed in the Mel-scale and is illustrated in Fig. 3a. The conversion from a linear to Mel-scale is given by

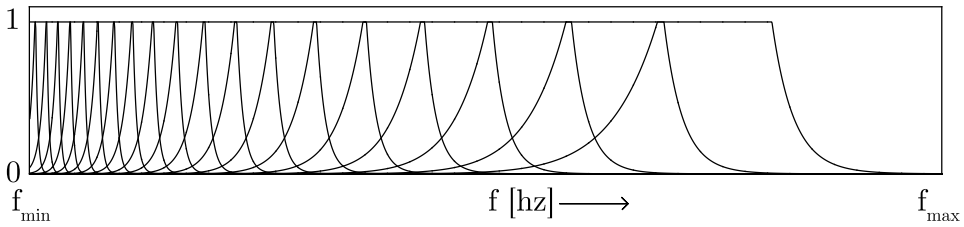
$$f_{mel} = 2595 \log_{10} \left(1 + \frac{f}{700}\right).$$

The procedure of MFCC computation is following. The signal is pre-emphasized and the short-time analysis is then performed. It means that the magnitude frequency spectrum is computed and filtered via the Mel Filter Bank that is designed with respect to the requirements and the signal properties. Then the logarithm of filter outputs is computed, that allows to divide the convolution channel distortion. The Discrete Cosine Transformation (DCT) decorrelates the output coefficients, which is desired for further statistical classifier. The MFCC features tries to emulate the human perception of

Figure 2 The principle of MFCC computation.



(a) Triangular-shaped filter bank used for MFCC computation.



(b) Filter bank with trapezoidal filters applied in PLP analysis.

Figure 3 Comparison of the filter-banks used in MFCC and PLP features computation. The x-axis is scaled to Hz from mel and bark scale.

sound frequencies and thus they try to increase the recognition accuracy. The principle of MFCC computation is illustrated in the Fig. 2.

2.1.2 PLP

The LP analysis approximates the signal spectrum well for all the frequencies of the analyzed band. Such approximation does not involve the principles of a human sound perception, such as frequency sensitivity or non-linear perception of frequencies and intensity. The Perceptual Linear Predictive (PLP) analyses was designed to account the properties of a human hearing. The block diagram of the individual steps of PLP analysis is in Fig. 4

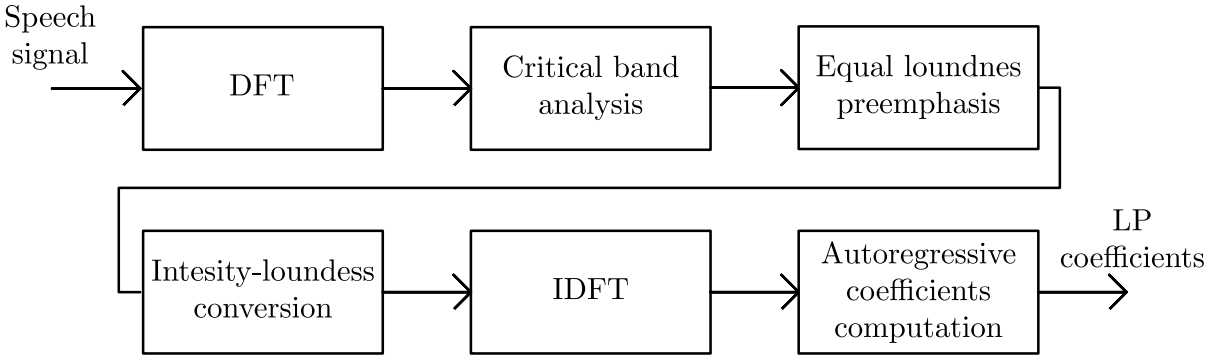


Figure 4 The individual steps of PLP analysis.

In the first step, the short time spectrum $S(\omega)$ is computed out of the windowed signal frame via the FFT. Then the power spectrum is computed as a squared amplitude spectrum. As for the MFCC computation, the filter bank is then applied. Since the PLP tries to incorporate the nonlinear human sound perception, the Bark-scaled filter bank with trapezoid-shaped filters is applied and the output of each filter is pre-emphasized by a simulated equal-loudness curve. The law of hearing is used and finally the approximation by the spectrum of an all-pole model is performed. Often, the resulting prediction coefficients $a_0, a_1, \ldots, a_p$ are transformed to the cepstral coefficients as

$$c(0) = 1 \tag{4}$$

$$c(1) = -a_1 \tag{5}$$

$$c(k) = -a_k - \sum_{i=1}^{k-1} \left(\frac{i}{k}\right) c(i) a_{k-1} \ , \quad \text{for } 2 \leq k \ \leq p \tag{6}$$

$$c(k) = -\sum_{i=1}^{p} \left(\frac{k-i}{k}\right) c(k-i) a_i \ , \quad \text{for } k = p+1, p+2, \ldots \tag{7}$$

2.1.3 Further Features Procesing

To support a robustness and discrimination of the features, the further processing and adaptation methods are applied in a current approaches resulting in better classification results. Let's introduce some feature transformations used in the experimental part.

Dynamic coefficients ($\Delta + \Delta\Delta$)

The individual cepstral coefficients obtained by methods described above are called static, since they are computed only out of the current frame. To include some context information, the first and second derivative is computed to obtain the delta ($\Delta \mathbf{c}_m$) and delta-delta ($\Delta^2 \mathbf{c}_m$) dynamic features that are appended to the static ones. These dynamic coefficients are determined via the equations

$$[\Delta c_m(j)]_n = \frac{\sum_{\kappa=-L_1}^{L_1} \kappa [c_m(j)]_{n+\kappa}}{\sum_{\kappa=-L_1}^{L_1} \kappa^2}, \qquad [\Delta^2 c_m(j)]_n = \frac{\sum_{\kappa=-L_2}^{L_2} \kappa [\Delta c_m(j)]_{n+\kappa}}{\sum_{\kappa=-L_2}^{L_2} \kappa^2}, \tag{8}$$

out of $2L_{1,2} + 1$ consecutive frames, where typically $L_1 = L_2 = 1$. These dynamic coefficients are appended to the static ones and altogether represents the final feature vector.

Linear Discriminant Analysis (LDA)

LDA [17] reduces the dimension of n-dimensional feature vector into a m-dimensional space where $(m < n)$. It clusters the elements of the individual classes closer to each

other so the class separability is maximum. Also, lowering the features dimension can lead to overcoming the curse of dimensionality problem. Let's suppose the $D > K$ dimensional input vector $\mathbf{x}$, where K is the number of classes. Then the vector $\mathbf{x}$ can be reduced to $D' > 1$ features

$$y_k = \mathbf{w}_k^T \mathbf{x}, \quad \text{for } k = 1, \ldots, D'. \tag{9}$$

where $\mathbf{w}_k^T$ is the weight vector. We can combine the equations 9 as $\begin{bmatrix} y_1 & y_2 & \ldots & y_{D'} \end{bmatrix} = \begin{bmatrix} \mathbf{w}_1 & \mathbf{w}_2 & \ldots & \mathbf{w}_{D'} \end{bmatrix}^T \mathbf{x}$ and write down the matrix equation of the form

$$\mathbf{y} = \mathbf{W}^T \mathbf{x}. \tag{10}$$

The individual components of matrix $\mathbf{W}$ can be selected in a such way, that the projection to the D' dimensional space maximizes the class separation [22]. The mean vector of the class $\mathcal{C}_k$, for $k = 1, \ldots, K$ can be computed as

$$\mathbf{m}_k = \frac{1}{N_k} \sum_{n \in \mathcal{C}_k} \mathbf{x}_n \tag{11}$$

where N_k is the number of elements in class $\mathcal{C}_k$. The within-class covariance matrix is given by

$$\mathbf{S}_W = \sum_{k=1}^{K} \mathbf{S}_k \tag{12}$$

where $\mathbf{S}_k$ is computed as

$$\mathbf{S}_k = \sum_{n \in \mathcal{C}_k} (\mathbf{x}_n - \mathbf{m}_k)(\mathbf{x}_n - \mathbf{m}_k)^T. \tag{13}$$

For the total number of N data points the mean $\mathbf{m}$ of the whole data set is

$$\mathbf{m} = \frac{1}{N} \sum_{k=1}^{K} N_k \mathbf{m}_k \tag{14}$$

and the between-class covariance matrix is then

$$\mathbf{S}_B = \sum_{k=1}^{K} N_k (\mathbf{m}_k - \mathbf{m})(\mathbf{m}_k - \mathbf{m})^T. \tag{15}$$

The columns of the weight matrix $\mathbf{W}$ are represented by eigenvectors of matrix

$$\mathbf{S}_W^{-1} \mathbf{S}_B, \tag{16}$$

t corresponding to the D' larges eigenvalues.

Maximum Likelihood Linear Transform (MLLT)

As it will be described in the following section in more detail, each state of HMM has associated the observation symbol probability distribution expressed as the Gaussian Mixture Model (GMM) with parameters $\Theta = \{c_j, \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j\}_{j=1}^{M}$ given by

$$p(\mathbf{x}|\Theta) = \sum_{j=1}^{M} c_j \cdot \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j). \tag{17}$$

M is the number of components in a mixture, c_j is the weight of j-th component satisfying $c_j \geq 0$ and $\sum_{j=1}^{M} c_j = 1$. Then, $\boldsymbol{\Sigma}_j$ is the j-th component square covariance matrix of the rank n and finally $\boldsymbol{\mu}_j$ is the mean value of j-th component. The MLLT imposes a different form of the covariance matrix, that allows to share the full covariance matrices by all components of one GMM [18]. MLLT decomposes the inverse covariance matrices as:

$$\boldsymbol{\Sigma}_j^{-1} = \mathbf{W}\boldsymbol{\Lambda}_j\mathbf{W}^{\mathrm{T}}$$

so each covariance matrix consists of two elements, the diagonal matrix $\boldsymbol{\Lambda}_j$ and non-singular linear transformation matrix $\mathbf{W}$, that is shared across all GMM components. The EM algorithm with maximum likelihood approach is usually used the estimate the model parameters.

2.2 Acoustic modeling

As it was stated at the beginning of this chapter, the acoustic model evaluates the probability $P(\mathbf{O}|W)$, thus the probability that sequence of the output feature vectors $\mathbf{O}$ is observed, when the sequence of words W is spoken. The training conditions and the conditions of a real speech recognition application are completely different in the most cases. The speaker variability, distortions in the acoustic channel, pronunciation variations and other differences emphasize the importance of robustness and flexibility of the acoustic model. The HMM proved to meet such requirements and this acoustic modeling method became the standard in statistical speech recognition and was not overcome for a long period of time. In the recent years, DNN as acoustic modeling method has shown better results than conventional HMM approach. These two acoustic modeling methods are the main topics of the following sections.

2.2.1 The Hiden Markov Model (HMM)

When the speech is produced, the human articulatory system is in one of the finite set of states that results in a desired phone. This state can't be observed directly, the listener only hears the acoustic output. So it can be viewed as a system with

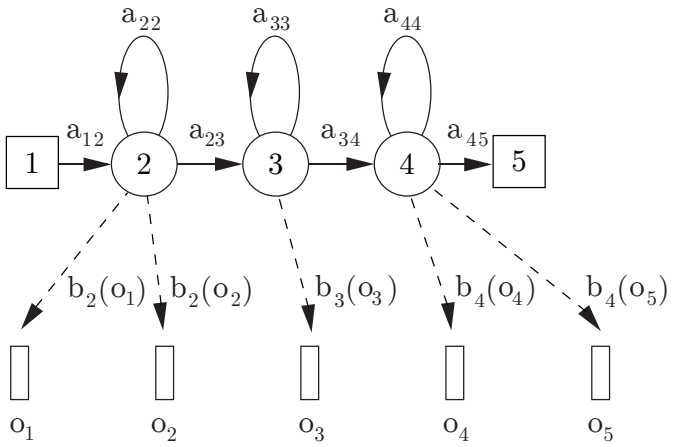


Figure 5 The HMM illustration.

hidden states that produces the sequence of observations. Such system can be modeled via the HMMs, which are stochastic finite-state automata generating the sequence of observations vectors with a hidden sequence of states. In the speech applications, the left-right models are used, since they are suitable to model processes progressing through the time. Based on the set of transition probabilities a_{ij}, the state transition is preformed from the state s_i to the state s_j at every discrete time step. When the state s_j is reached, the actual vector of observations $\mathbf{o}_t$ generates the probability based on the symbol probability distribution $b_j(\mathbf{o}_t)$.

HMM structure can be various and it can model the whole words or sub-word units like monophones, diphones and others. The practical applications showed, that the triphones are the best compromise between the the unit size to be modeled and between the context information involved. Modeling whole words requires the vast training dataset, since several acoustic realizations of one word is needed, therefore, the sub-word unit models are used, mainly the triphones represented by 5-state HMM as shown in Fig. 5. The first and the last state are non-emitting, thus no probability is produced when reaching this state. These states are used to composing individual HMMs together. Let's review the parameters of HMM. The observation symbol probability distribution $b_j(\mathbf{o})$ for emitting state j has mostly the form of Gaussian Mixture Model given by:

$$b_j(\mathbf{o}) = \sum_{m=1}^{M} c_{jm} \cdot \mathcal{N}(\mathbf{o}, \boldsymbol{\mu}_{jm}, \mathbf{C}_{jm}) \tag{18}$$

where the meaning of particular symbols is as follows. M is the number of Gaussian mixtures, c_{jm} is the weight of individual mixtures and $\mathbf{C}_{jm}$ represents the covariation matrix for j-th state and m-th mixture component with mean value $\boldsymbol{\mu}_{jm}$ For the model given in Fig. 5, which includes only the forward state transitions, the transition matrix

A has the form:

$$A = \begin{bmatrix} 0 & a_{12} & 0 & 0 & 0 \\ 0 & a_{22} & a_{23} & 0 & 0 \\ 0 & 0 & a_{33} & a_{34} & 0 \\ 0 & 0 & 0 & a_{44} & a_{45} \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

And it holds, that sum over all transition probabilities a_{ij} is equal to 1. To use the HMM in real applications, the three following problems has to be solved:

1. Evaluation problem: denoting λ as the HMM, what is the probability $P(\mathbf{O}|\lambda)$, that given observation sequence $\mathbf{O}$ was generated by the model λ?
2. Decoding problem: how to estimate the state sequence $S = \{s_0, s_2, \ldots, s_{T+1}\}$ (where $s(0)$ and $s(T+1)$ denotes the non-emitting boundary states) of model λ corresponding to the given $\mathbf{O}$?
3. Training problem: How to tune the model parameters to maximize $P(\mathbf{O}|\lambda)$?

Solution for problem 1.Let's consider for a while, that the state sequence for a given observation is known. Then solution of the first problem for the example given in Fig. 5 is

$$P(\mathbf{O}|\lambda) = a_{12}b_2(\mathbf{o}_1)a_{22}b_2(\mathbf{o}_2)a_{23}b_3(\mathbf{o}_3)a_{34}b_4(\mathbf{o}_4)a_{44}b_4(\mathbf{o}_5).$$

But in real application, the state sequence is not known and has to be estimated as sum over all possible sequences of states

$$P(\mathbf{O}|\lambda) = \sum_S P(\mathbf{O}, S|\lambda) = \sum_S a_{s(0)s(1)} \prod_{t=1}^{T} b_{s(t)}(\mathbf{o}_t) a_{s(t)s(t+1)} \tag{19}$$

Such direct computation requires to evaluate enormous amount of operations. To compute the probability $P(\mathbf{O}|\lambda)$ more efficiently, the iterative forward-backward procedure can be used [23].

Solution for problem 2. To determine the optimal state sequence for a given observations, the dynamic programming based method called the Viterbi algorithm can be used. The detailed description can be found in several publications e.g. in [24].

Solution for problem 3. In many applications, the standard approach to estimate the model parameters is based on Maximum Likelihood (ML) criterion. Let's consider the individual HMM parameters as $\lambda \equiv \{a_{ij}, c_{jm}, \boldsymbol{\mu}_{jm}, \mathbf{C}_{jm}\}$ for $1 \leq i, j \leq N$ and $1 \leq m \leq M$ where N is the number of states. Next, let's consider the set of E known utterances $\{\mathbf{O}^e\}_{e=1}^{E}$ that is used to estimate to estimate the model parameters λ. Then the desired maximum of ML criterion is

$$\hat{\lambda} = \underset{\lambda}{\operatorname{argmax}} \sum_{e=1}^{E} \log P(\mathbf{O}^e|\lambda). \tag{20}$$

To obtain $\hat{\lambda}$ the iterative procedure called Baum-Welch (sometime referred as EM - Expectation Maximization) algorithm can be used. ML criterion is often used in many current application, since it provides sufficiently accurate model whose training is not computationally demanding. Let's state, that if certain assumptions hold, no other criteria will out-perform this one [25]. These assumptions are: the unlimited training data set, individual observation vectors are independent and observation vectors are really generated by HMM, where modeled density function corresponds to the real one [26].

Discriminative training - Maximal Mutual Information (MMI) Criterion

The ML assumptions are not met in the real applications, so obtaining the optimal model parameters is not guaranteed. This problem try to solve the discriminative training methods. ML uses the correct observation vectors to train the individual parameters of HMM state, by the correct observations vector are meant vectors, that belong to the state. In a discriminative approach, both correct (positive) and wrong (negative) samples are used for training. This state is trained not only with the correct vectors, but it is trained also with respect to the vectors that are negative for this state, thus belong to the different states. The discriminative training is used for the majority of classification methods including neural networks or linear classifiers [26]. Such discriminative HMM training method is MMI. Let's consider again the training utterances $\mathbf{O}^e$ and the model parameters λ, the objective function to be maximized is

$$\mathcal{F}_{MMI}(\lambda) = \sum_{e=1}^{E} \log \frac{P_\lambda(\mathbf{O}^e|M_{W_r})P(W_r)}{\sum_{\hat{W}} P_\lambda(\mathbf{O}^e|M_{\hat{W}})P(\hat{W})} \tag{21}$$

where w_r is the reference transcription of the utterance O_e and M_{W_r} is HMM representing the w_r. $M_{\hat{W}}$ is the model corresponding to the transcription $\hat{W}$ and $P(W)$ is the probability of a word sequence $\hat{W}$ determined by a language model such as unigram. To maximize 21, the numerator has to be increased while the denominator is decreased. The first part $P_\lambda(\mathbf{O}^e|M_{W_r})$ of numerator is identical with MLE criterion. So MMI tries to maximize the likelihood of all observations given the training transcriptions. The denominator can be minimized by reducing the probabilities of other possible word sequences. So the MMI performs maximization of correct hypothesis probability while reducing the probability of false hypothesis. But the following problems are connected with MMI training:

- A maximization of the given objective function is difficult. The conventional Baum-Welch algorithm cannot be used, so its modified version known as extended Baum-Welch algorithm [27] needs to be applied.
- It is computationally expensive to maximize the objective function since the denominator represents a sum over all possible word sequences in the language model.

- Poor generalization performance of testing data.

With respect to these drawbacks, in the current applications, the standard ML training is used to train the whole model whose parameters are then updated with several iterations of MMI training algorithm. Also, the other version of MMI training that boosts the likelihoods in the denominator lattice with higher phone error was introduced in [28] and it is used in this work.

Subspace Gaussian Mixture Model (SGMM)

An alternative method of an acoustic modeling is a Subspace Gaussian Mixture Model (SGMM). In a SGMM, the model parameters are derived from the globally shared model subspace with a very low dimensional state-dependent vector [32], thus, each state is described by a low dimensional vector and there is globally shared mapping from this vector, denoted as state vector, to the means and weights of the state's GMM [34]. The probability model $p(\mathbf{x}|j)$ of state j is in SGMM defined as

$$p(\mathbf{x}|j) = \sum_{k=1}^{K_j} c_{jk} \sum_{i=1}^{I} w_{jki} \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_{jki}, \boldsymbol{\Sigma}_i) \tag{22}$$

$$\boldsymbol{\mu}_{jki} = \mathbf{M}_i \mathbf{v}_{jk} \tag{23}$$

$$w_{jki} = \frac{\exp \mathbf{w}_i^T \mathbf{v}_{jk}}{\sum_{i'=1}^{I} \exp \mathbf{w}_{i'}^T \mathbf{v}_{jk}} \tag{24}$$

where $\mathbf{x}$ is the feature, j is the HMM state, k is a sub-state, where each state j has K_j sub-states with own mixture weight c_{jk} and sub-state vector $\mathbf{v}_{jk}$. I is the number of Gaussian components in a sub-state each with weight w_{jki}, mean $\boldsymbol{\mu}_{jki}$ and covariance matrix $\boldsymbol{\Sigma}_i$. $\mathbf{M}_i$ is the mean projection matrix and $\mathbf{w}_i$ is the weight projection vector. This model is actually a mixture of mixtures of Gaussians. This approach can be further extended by introducing the speaker vectors $\mathbf{v}^{(s)}$ describing the speaker s. Then the model becomes:

$$p(\mathbf{x}|j,s) = \sum_{k=1}^{K_j} c_{jk} \sum_{i=1}^{I} w_{jki} \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_{jki}^{(s)}, \boldsymbol{\Sigma}_i)$$

$$\boldsymbol{\mu}_{jki} = \mathbf{M}_i \mathbf{v}_{jk} + \mathbf{N}_i \mathbf{v}^{(s)}$$

$$w_{jki} = \frac{\exp \mathbf{w}_i^T \mathbf{v}_{jk}}{\sum_{i'=1}^{I} \exp \mathbf{w}_{i'}^T \mathbf{v}_{jk}}$$

when $\mathbf{N}_i \mathbf{v}^{(s)}$ is the speaker specific offset to mean. The SGMM acoustic modeling provides better results then the conventional GMM structure [34]. SGMM reduces the total number of model parameters to be estimated that is crucial when only limited amount of training data is available. By introducing the speaker vectors, it also incorporates the speaker adaptation.

2.2.2 Adaptation Methods

In general, the two types of ASR systems can be distinguished. The first system is a speaker independent which means that it was trained on the large group of speakers. Such system would perform slightly worse for a particular speaker when compared with a case, when the whole system is trained specifically for this speaker. We call these systems a speaker dependent. But to train a speaker dependent system requires a lot of training data recorded by this speaker and it can be costly from many angles of view. The idea of adaption methods is to adapt the speaker independent system for a particular speaker without the need of huge amount of speaker training data and thus improve the performance of a recognition. This is referred as speaker adaptive traininng (SAT) and it can be performed at the level of feature space or at the level of the acoustic model parameters. Between adaptation methods belongs Vocal Tract Length Normalization (VTLN), Maximum A posteriori Probability (MAP) and last but not least Maximum Likelihood Linear Regression (MLLR) or its feature spaced variant fMLLR [19], that is widely used in many current applications.

feature-spaced Maximum Likelihood Linear Regression (fMLLR)

MLLR method reduces the number of the model parameters by concatenating the acoustically similar Gaussian mixture components to the individual classes C_n which are further processed with the same transformation [3]. Consider the adaptation data $\mathbf{O}^e = \{\mathbf{o}^e(1), \mathbf{o}^e(2), \ldots, \mathbf{o}^e(T_e)\}, e = 1, \ldots, E$. The mean value linear transform is given by

$$\hat{\boldsymbol{\mu}}_{jm} = \mathbf{A}_{(n)}\boldsymbol{\mu}_{jm} + \mathbf{b}(n), \tag{25}$$

where $\hat{\boldsymbol{\mu}}_{jm}$ is the the adapted mean value. The regression matrix $\mathbf{A}_{(n)}$ and additive vector $\mathbf{b}(n)$ are both connected with the class C_n and $\boldsymbol{\mu}_{jm}$ is the original mean value of m-th component of GMM in j-th model state. After the substitution

$$\boldsymbol{\xi}_{jm}^T = [\boldsymbol{\mu}_{jm}^T, 1], \quad \mathbf{W}(n) = [\mathbf{A}(n), \mathbf{b}(n)] \tag{26}$$

is it possible to rewrite the equation (25) as

$$\hat{\boldsymbol{\mu}}_{jm} = \mathbf{W}(n)\boldsymbol{\xi}_{jm}^T \tag{27}$$

The transformation of a covariance matrix can be expressed in the form of

$$\hat{\mathbf{C}}_{jm} = \mathbf{H}(n)\mathbf{C}_{jm}\mathbf{H}(n)^T, \tag{28}$$

where $\mathbf{H}(n)$ is the transformation matrix for class C_n and $\mathbf{C}_{jm}$ is the original covariance matrix. The individual transformation matrices can be found by estimating the optimum

of the function

$$Q(\lambda,\bar{\lambda}) = const - \frac{1}{2}\sum_{b_{jm}\in\lambda}\sum_{e=1}^{E}\sum_{t=1}^{T_e}\gamma_{jm}(t)(c_{jm}+\log|\hat{\mathbf{C}}_{\mathbf{jm}}|+(\mathbf{o}^e(t)-\hat{\boldsymbol{\mu}}_{jm})^T\hat{\mathbf{C}}_{jm}^{-1}(\mathbf{o}^e(t)-\hat{\boldsymbol{\mu}}_{jm})). \tag{29}$$

The description of further steps is provided in [33]. The fMLLR method performs the linear adaptation of feature vector $\mathbf{O}$ instead of acoustic model parameters that results in a lower computational cost. The transformed feature vector $\hat{\mathbf{o}}^e$ is given by

$$\hat{\mathbf{o}}^e = \mathbf{A}_{(n)}\mathbf{o}^e(t) + \mathbf{b}_{(n)} = \mathbf{A}_{(n)c}^{-1}\mathbf{o}^e(t)\mathbf{A}_{(n)}^{-1}\mathbf{b}_{(n)c} = \mathbf{W}(n)\boldsymbol{\xi}^e(t) \tag{30}$$

where $\mathbf{W}(n) = [\mathbf{A}(n), \mathbf{b}(n)]$ is the transformation matrix, $\boldsymbol{\xi}^e(t) = [\mathbf{o}^e(t), 1]$ is the extended feature vector and $\mathbf{A}(n)c, \mathbf{b}(n)c$ are matrices for equivalent transformation of an acoustic model parameters

$$\hat{\boldsymbol{\mu}}_{jm} = \mathbf{A}_{(n)c}\mu_{jm} - \mathbf{b}_{(n)c}, \tag{31}$$

$$\hat{\mathbf{C}}_{jm} = \mathbf{A}_{(n)c}\mathbf{C}_{jm}\mathbf{A}_{(n)c}^T \tag{32}$$

The optimization function has the form

$$Q(\lambda,\bar{\lambda}) = const - \frac{1}{2}\sum_{b_{jm}\in\lambda}\sum_{t,e=1}^{T_E}\gamma_{jm}^e(t)(c_{jm}+\log|\hat{\mathbf{C}}_{\mathbf{jm}}|-\log(|\mathbf{A}_{(\mathbf{n})}|^2)+(\hat{\mathbf{o}}^e(t)-\boldsymbol{\mu}_{jm})^T\hat{\mathbf{C}}_{jm}^{-1}(\hat{\mathbf{o}}^e(t)-\boldsymbol{\mu}_{jm})). \tag{33}$$

2.2.3 Deep Neural networks (DNN)

In the hybrid DNN-HMM acoustic model, the DNN outputs provides pseudo-likelihoods for the states of HMM, thus DNN replaces the GMM and emulates the observation symbol probability function $b()$. The usage of DNN in acoustic model is schematically illustrated in Fig. 6. The optimization of DNN training and setup is not the main task of this work, therefore the recommended values are used and only brief description of DNN principle is provided. A DNN is a such type of feed-forward artificial neural network, which has more than one hidden layer between its input and output layers. The output of j-th neuron in a hidden layer, with m inputs, is given by

$$y_j = \phi\left(\mathrm{b}_j + \sum_{i=1}^{m} x_{ij}w_{ij}\right) = \phi(z), \tag{34}$$

where b_j is the bias of j-th neuron, w_{ij} is the weight of the input x_ij and ϕ is the transfer function with a well-behaved derivative like hyperbolic tangent or the mostly used sigmoid function

$$\phi(z) = \frac{1}{1+e^{-z}}. \tag{35}$$

The DNN outputs have the probabilistic character defined by the softmax transfer function for neurons in its output layer. Thus, output of j-th neuron in this last layer is

$$y_j = \frac{e^{z_j}}{\sum_k e^{z_k}}, \tag{36}$$

where k is the index over all classes. The error back-propagation method can be used

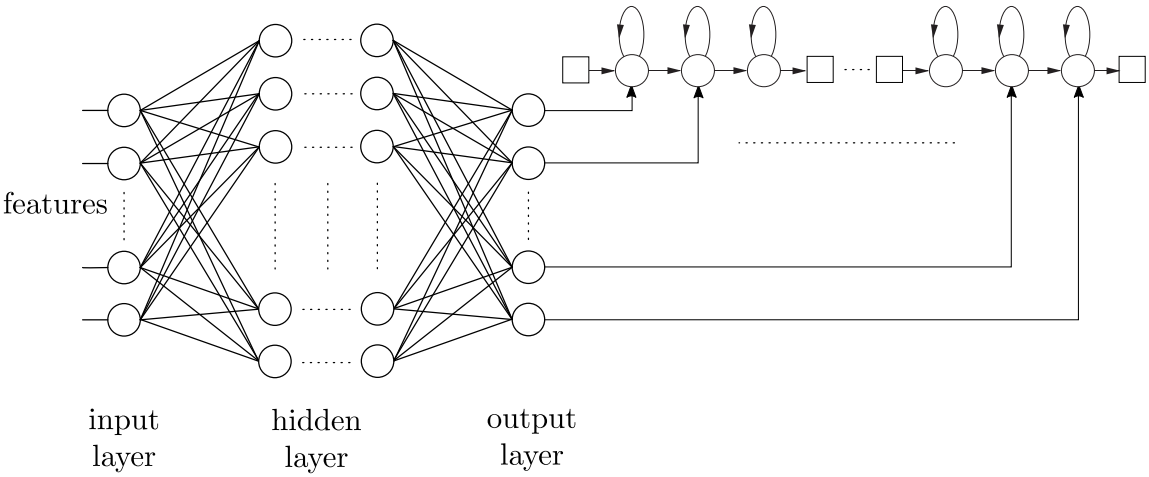


Figure 6 DNN-HMM hybrid approach to acoustic modeling.

to train the DNN and recent experiments [13] proved, that DNN pre-training provides better results in many task. In this case, the training procedure has two stages. In the first stage, the restricted Boltzmann machines (RBM) are trained. GRBM is trained to model feature window at first. The states of its hidden units are then used to train a RBM. The process is repeated until the desired number of layers is obtained. Finally, these RBM are stacked together and converted into the Deep Belief Network (DBN). The resulting DBN-DNN is created by adding the output layer with softmax transfer function. Process of creation of pre-trained DBN-DNN with three hidden layers is illustrated in the Fig. 7. After this, the alignment and lattices, evaluated by GMM-HMM system from training data are used to discriminatively train the DBN-DNN to predict the individual HMM states. The complete description of the training process can be found in several publications, e.g. [13] and [29].

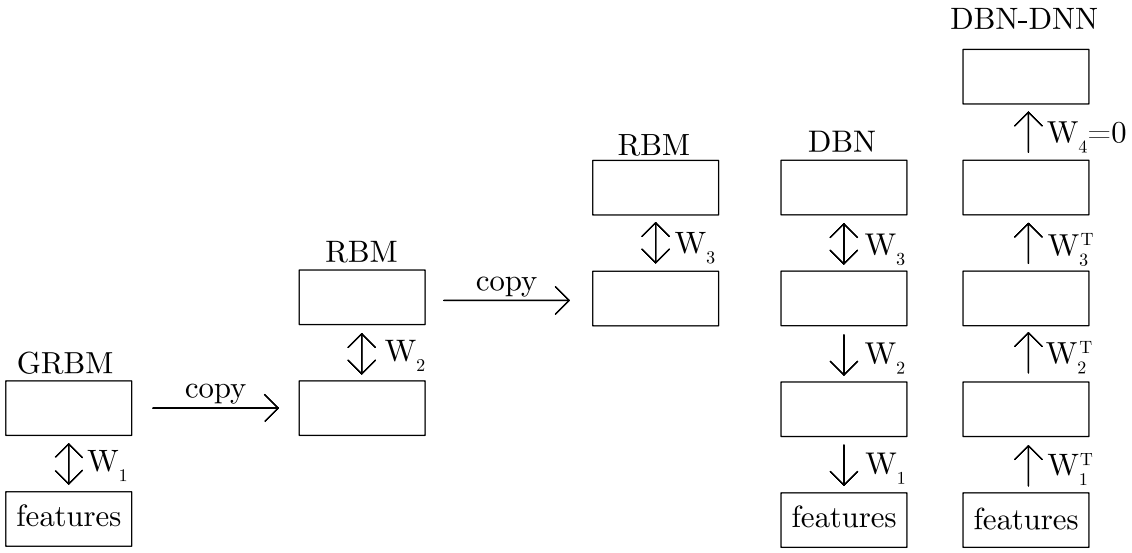


Figure 7 DNN pretraining process. *Taken from [13].*

2.3 Language Modeling

A stochastic language model estimates the probability $P(W)$ for every word sequence W with K words as

$$\begin{aligned} P(W) &= P(w_1^K) \\ &= P(w_1, w_2, \ldots, w_k) \\ &= P(w_1)P(w_2|w_1)P(w_3|w_1w_2)\ldots P(w_K|w_1w_2\ldots w_{k-1}) \\ &= \prod_{i=1}^{K} P(w_i|w_1, w_2, \ldots, w_{i-1}), \end{aligned}$$

where the probability $P(w_i|w_1, w_2, \ldots, w_{i-1})$ is conditioned only by the previous word sequence. But it is impossible to evaluate the $P(w_1^K)$ for all the possible word sequences of an arbitrary length K. Therefore, the approximation is performed by estimating the probability only for the preceding $N-1$ words as

$$P(W) \approx \prod_{i=1}^{K} P(w_i|w_{i-N+1}, w_{i-N+2}, \ldots, w_{i-1}). \tag{37}$$

These stochastic models are referred as N-grams. Depending on the N value, we distinguish unigrams (N=1), bigrams(N=2) and trigrams (N=3), which are used by the most of applications. N-gram models are especially suitable for languages with a strict word order like for example English. The individual probabilities of N-gram model can be estimated by the relative frequency of occurrence of the word sequences in the training data. For trigram model holds

$$\bar{P}(w_k|w_{k-2}w_{k-1}) = \frac{N(w_{k-2}, w_{k-1}, w_k)}{N(w_{k-2}, w_{k-1})},$$

where $N(w_{k-2}, w_{k-1}, w_k)$ is the number of occurrences of trigram w_{k-2}, w_{k-1}, w_k in the training data and $N(w_{k-2}, w_{k-1})$ denotes the number of training data occurrences of bigram w_{k-2}, w_{k-1}).

To evaluate the quality of a stochastic language model, the perplexity value is defined. It is also possible to use the word error rate, but this would required working speech recognition system. Perplexity PP is defined as

$$PP = \frac{1}{\sqrt[K]{P(w_1, w_2, \ldots, w_K)}},$$

where $P(W) = P(w_1, w_2, \ldots, w_K)$ is the estimate of probability of occurrence of the word sequence $w_1, w_2, \ldots, w_K$ provided by a language model. The lower the perplexity is, the more the language is predictable. The perplexity can be also interpreted as an estimate of the size of the word list that recognizer must choose from.

2.4 Decoding

The aim of the decoder is to obtain the sequence of words W out of the observation sequence $\mathbf{O}$. This can be expressed as:

$$\hat{W} = \underset{W}{\operatorname{argmax}}\, P(W)P(\mathbf{O}|W).$$

Both of these probabilities are known. $P(\mathbf{O}|W)$ is estimated by an acoustic model and probability P(W) is computed by a language model. Especially in the large vocabulary speech recognition tasks, it is infeasible to search for all possible word sequences due the vast amount of computations. But the real time response is required by the ASR system in real applications, therefore several techniques to solve the decoding problem have been proposed.

In the recent years, the weighted finite state transducers (WFST) are being used in the task of speech recognition. Since the HMMs, that are finite state automata, are usually used as the AM and language model can be expressed as probabilistic finite state automaton, the WFSTs are the natural representation of of these two part of ASR system as well as the pronunciation dictionary, context dependency and even further. Thus, these representations can be combined together and flexibly and efficiently optimized.

2.4.1 WFST

A finite state automaton whose state transitions are labeled with the input and output symbols is called the finite state transducer. A weighted finite state transducer (WFST) is created by adding the weight on its transitions. Thus, the path through the transducer encodes a mapping from an input symbol sequence to an output symbol sequence. The

formal definition is as follows. A WFST is a 8-tuple $T = (\Sigma, \Omega, Q, E, i, F, \lambda, \rho)$ over the semiring $\mathbb{K}$ where Σ is an input alphabet and Ω is an output alphabet. Q represents a finite set of states and $E \subseteq Q \times (\Sigma \cup \{\epsilon\}) \times (\Omega \cup \{\epsilon\}) \times \mathbb{K} \times Q$ is the final set of transitions. $i \in Q$ denotes the initial state, $F \subseteq Q$ set of final states, λ an initial weight and ρ is a final weight function. Transition t is 5-tuple $t = (p[t], l_i[t], l_o[t], w[t], n[t])$ given by source state $p[t]$, destination state $n[t]$, input label $l_i[t]$, output label $l_o[t]$ and weight $w[t]$. The weights in a speech recognition task mostly represent probabilities, so the corresponding semiring is called the tropical semiring $(\mathbb{R}, +, ., 0, 1)$. An example of WFST representing the pronunciation lexicon is in Fig. 8. The common set of operations

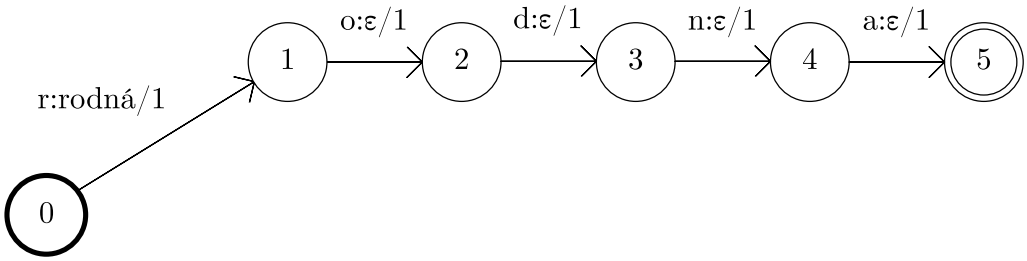


Figure 8 An example of WFST representing the pronunciation lexicon. An initial state is represented by bold circle while a finite state by double circles.

for weighted transducers like composition, intersection or determinization defined in [30] supports the ASR applications. In speech recognition, the WFST are used to compose the HCLG recognition graph:

$$HCLG \equiv H \circ C \circ L \circ G$$

where H specifies the HMM structure. As an input, it takes the probability density function labels. Its output is the context dependent phone. C is the context dependency transducer that takes the context dependency phones and returns a phone. L is the lexicon that pairs any sequence of vocabulary words to their pronunciations. Finally, the G is the language model. To optimize this integrated transducer, the determinization (det), minimization (min) and factoring operations are performed which can be expressed by the formula

$$N = \pi_\epsilon(det(\tilde{H} \circ det(\tilde{C} \circ det(\tilde{L} \circ G)))).$$

The tilde symbol is used to denote the slightly modified transducers that arise from the original ones by using other techniques in real implementations like deploying the auxiliary symbols for denoting the homonyms and so on. π_ϵ is operation which replaces the auxiliary symbols with ϵ's.

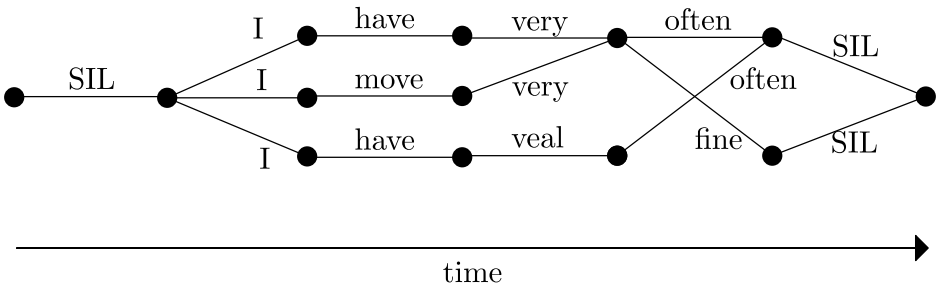


Figure 9 A word lattice example.

2.4.2 N-best results, word lattice

The desired output of the decoding is the most likely word sequence for an obtained sequence of observations. In the practical applications, it is useful to generate the N-best set of hypotheses. N-best results can be used in multi-pass search of a state space, when the decoding process is divided into the several stages. In every pass, the set of hypothesis is reduced so it allows to use the more sophisticated and computationally demanding methods thanks the smaller state space to be searched. Also, the set of N-best results can be used for re-scoring the new language or acoustic models without the need of decoding the whole graph [3]. The N-best results are typically represented by a word lattice that consists of a set of nodes and spanning arcs. The individual nodes represent word ends with a given transition time to another word and arcs represent the words with an evaluation of the form of negative logarithm of the uttering probability of the given word in the time interval given by the surrounding nodes. An example of a word lattice is given in Fig. 9.

3 Multilingual Acoustic Modeling

One approach in the multilingual acoustic modeling is to find a global phonetic inventory. That means to find an acoustic similarities across all languages which leads to such phonetic inventory that is detailed enough to model all acoustic variations but which is not too big, to require huge amount of training data. To find such global phonetic inventory, it is necessary to determine the appropriate representation of the acoustic units. Therefore, the upcoming analyses in this chapter mainly deal with phonetics that is essential for this task.

3.1 Phonetic Inventory Unification

Phonetic alphabets allow to represent the utterances in the form of pronunciation that is essential for recognition. This process is called the phonetic transcription and it assigns the unique graphic symbol from the phonetic alphabet to an every distinguishable speech sound in the phonetic inventory of a given language. For this purpose, several phonetic alphabets like IPA or SAMPA were introduced.

SpeechDat

SpeechDat-E database [21], that serves as data source for all the experiments, provides an appropriate resource for telephone speech multilingual experiments. It consists of the five Eastern Europe languages namely Russian, Czech, Slovak, Polish and Hungarian. Every database includes several types of utterances, the read and also spontaneous ones. The SpeechDat pronunciation dictionaries use the SAMPA for phonetic transcription, which definition differs for every language. Therefore the proper unification is investigated in this section.

3.1.1 Phonetic alphabets

International phonetic Alphabet (IPA)

IPA represents a standard for the phonetic transcription. This alphabet provides symbols for every distinct sound of every known language in the world. It was created in 1888 by the International Phonetic Association and was last updated in 2015. The IPA

includes 107 letters, 52 diacritics and 4 prosodic marks. The individual symbols are represented by Roman alphabet letters and specially defined characters. The complete IPA is in Fig. 10.

Speech Assessment Methods Phonetic Alphabet (SAMPA)

SAMPA is machine readable phonetic alphabet, that actually allows coding of IPA characters in ASCII symbols. It was initially introduced for six European languages (Danish, Dutch, English, French, German,Italian) and lately extended for other ones. In 2006, the SAMPA was officially defined for 20 languages [3]. Despite the fact, that the SAMPA usage recommendations has been proposed, the SAMPAs are defined for every language individually, so this alphabet loses its unifying purpose.

Extended SAMPA (X-SAMPA)

X-SAMPA proposes machine readable coding for the entire IPA, so it is possible to use ASCII symbols for phonetic transcription of every possible language. It is an extension to SAMPA and its complete definition can be found in [1]. Let's summarize the main conventions:

- The IPA lower case alphabet symbols remain the same in X-SAMPA.
- Apostrophe symbol is redefined for palatalization diacritics.(cʲ[IPA] = `c'`[X-SAMPA])
- Reverse apostrophe stands for r-colouring and retroflex consonants.(ɚ[IPA] = `@\`[X-SAMPA], ɳ[IPA] = `n\`[X-SAMPA])
- The backslash extends, or changes, the meaning of preceding character, thus it has to be interpreted together with the character which immediately precedes it. Symbols with backslash represent other individual characters of IPA and backlash symbol used alone has no meaning. (e.g. X-SAMPA letter `B` corresponds to the β in IPA, while `B\` stands for voiced bilabial trill denoted in IPA with the symbol ʙ)
- Underscore implies diacritics. Thus, the symbol that follows underscore has different interpretation as in case of the standalone character.

3.1.2 Conversion to XSAMPA

For the multilingual speech recognition is essential to unify the phonetic transcription throughout the used languages. As previously described, since the IPA is not machine readable and SAMPA is not defined in general, the X-SAMPA is used to normalize the phonetic transcription of the given languages. The individual SpeechDat databases use SAMPA for phonetic transcription. Its unification, the conversion to X-SAMPA, is proposed further. The following notation is used. Letters in [ ] represent X-SAMPA

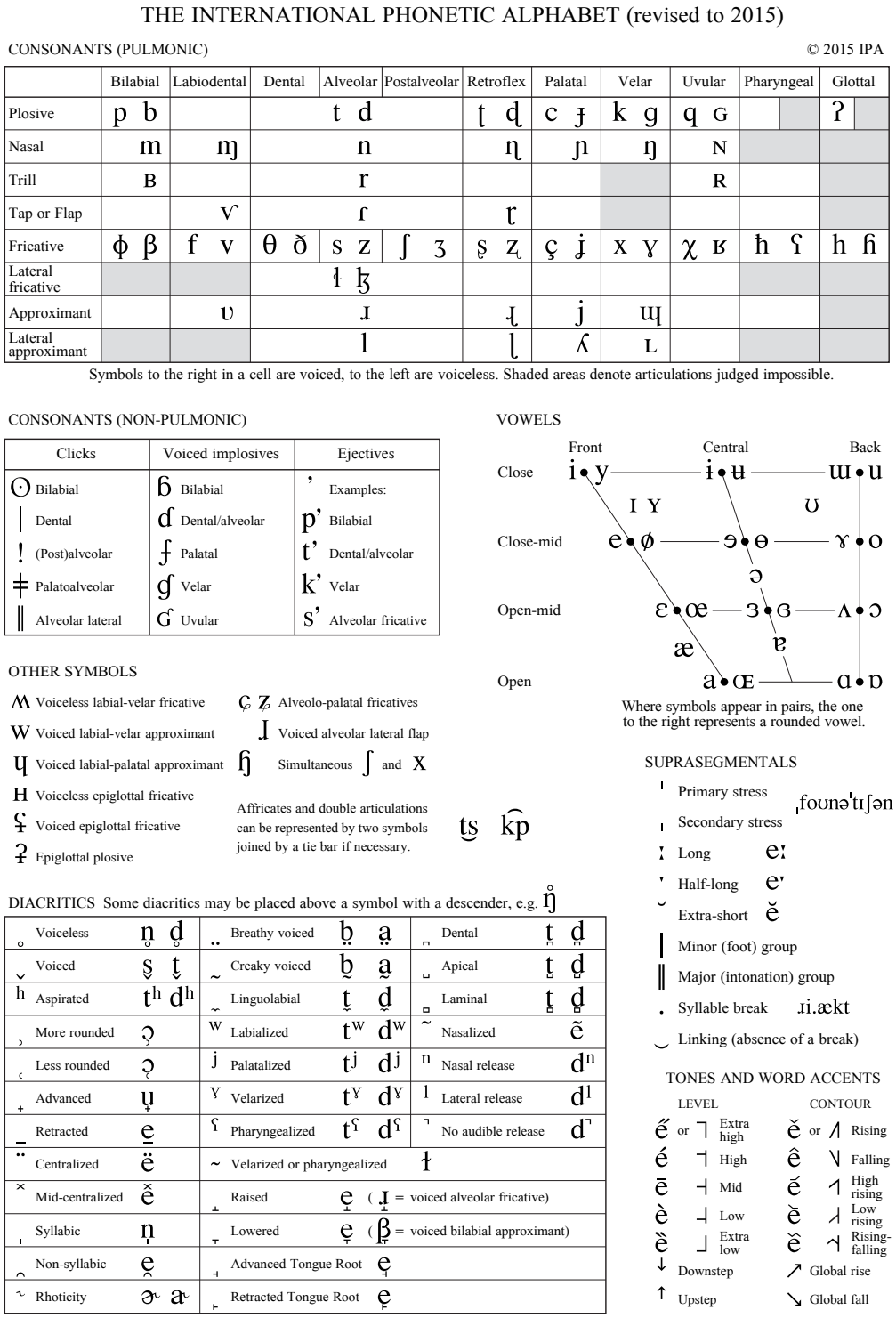


Figure 10 International phonetic alphabet. Source: *www.internationalphoneticassociation.org*

symbols, the individual SAMPA characters are written in / / and italic font is used to mark the phones themselves.

Czech SpeechDat SAMPA

The complete list of phones that occur in the Czech SpeechDat database is in Tab. 1. This notation corresponds to the definition in [4] and the following changes are proposed. The long and short variant of vowel *a* should be, in X-SAMPA, denoted as [a] and [a:]. Vowels *e* correspond to the open-mid variants ɛ, ɛː in IPA, which is rewritten as [E], [E:]. The long variant of *i* is close front, thus [i]. The short *i* is pronounced as near-close near-front variant [I] in Czech language. Vowels *o*'s are open-mid variants [o], [o:]. The last tuple of vowels *u*'s are close back variants [u], [u:] for now, but recent research has been shown the change in pronunciation to the [U] and [u:] variants. Diphthongs /a_u/, /e_u/ and /o_u/ should be [ou], [au] and [Eu]. Note that /2:/ and /y:/ may occur when spelling the foreign words, especially German.

Plosive consonants remain the same as in SAMPA, thus [p, b, t, d, c, J\, k] and [g]. With respect to the [2], the affricates /t_s,t_S,d_z/ and /d_Z/ are denoted as [ts, tS, dz] and [dZ]. Fricatives /f,v,s,S,Z,j,x,h\/ remain the same, but /P\/, representing voiced *ř* (e.g. in the word řeka), is in IPA called as alveoral trill fricative and marked with symbol r̝. This symbol is encoded as [r_r] in X-SAMPA. The official Czech SAMPA definition also distinguishes unvoiced variant r̝̊ written in X-SAMPA as [r_r_0]. The rest of consonants has the same symbols as in SAMPA. The complete conversion chart between individual alphabets for Czech language is in Tab. 2.

Slovak SpeechDat SAMPA

The list of SAMPA symbols used in Slovak SpeechDat is in Tab. 3. With respect to the definition of the Slovak SAMPA [5], the Slovak vowels `a,e,i,o,u` and their long variants should be rewritten as [a,e,i,o,u,a:,e:,i:,o:,u:]. Also, the Slovak diphthongs denoted as i͡a, i͡e, i͡u, i͡o in IPA remain the same as in SAMPA, thus [i_^a], [i_^e], [i_^u], [i_^o].

The small sonorant letters remain the same too. The syllabic mark `=` does not require the underscore according to the X-SAMPA definition. But its usage with underscore does not lead to any collision and since it is diacritics symbol, I suggest to stick to the definition and use the underscore character. This leads to [r_=], [l_=]. The length mark [:] represents the standalone symbol, so it is written without the underscore. For the long variants of these two sonorants we get [r_=:] and [l_=:]. There are three pronunciation variants of letter *v* in the Slovak language. As it can be seen from the table Tab. 3, these are the sonorants `v`, `u_^` and fricative `w`. In the official Slovak SAMPA definition, the following examples can be found:

Vowels		**Consonants**					
num	sampa	plosives		fricatives		nasals	
1	**a**	num	sampa	num	sampa	num	sampa
2	**e**	14	**p**	26	**f**	38	**m**
3	**i**	15	**b**	27	**v**	39	**n**
4	**o**	16	**t**	28	**s**	40	**N**
5	**u**	17	**d**	29	**z**	41	**J**
6	**a:**	18	**c**	30	**P**	additional allophones	
7	**e:**	19	**J**	31	**S**	num	sampa
8	**i:**	20	**k**	32	**Z**	42	**F**
9	**o:**	21	**g**	33	**j**	foreign phones	
10	**u:**	affricates		34	**x**	num	sampa
diphthongs		22	**t_s**	35	**h**	43	**E:**
num	sampa	23	**d_z**	liquids		44	**2:**
11	**a_u**	24	**t_S**	num	sampa	45	**y:**
12	**e_u**	25	**d_Z**	36	**r**		
13	**o_u**			37	**l**		

Table 1 The Czech SAMPA used in the SpeechDat database.

num	SAMPA	IPA	XSAMPA	num	SAMPA	IPA	X-SAMPA
1	a	a	a	24	t_S	tʃ	tS
2	e	ɛ	E	25	d_Z	dʒ	dZ
3	i	ɪ	I	26	f	f	f
4	o	o	o	27	v	v	v
5	u	u	u	28	s	s	s
6	a:	a	a:	29	z	z	z
7	e:	ɛː	E:	30	P\	r̝	r_r
8	i:	iː	i:	31	S	ʃ	S
9	o:	o	o:	32	Z	ʒ	Z
10	u:	uː	u:	33	j	j	j
11	a_u	au	au	34	x	x	x
12	e_u	ɛu	Eu	35	h\	ɦ	h\
13	o_u	ou	ou	36	r	r	r
14	p	p	p	37	l	l	l
15	b	b	b	38	m	m	m
16	t	t	t	39	n	n	n
17	d	d	d	40	N	ŋ	N
18	c	c	c	41	J	ɲ	J
19	J\	ɟ	J\	42	F	mŋ	F
20	k	k	k	43	E:	.	.
21	g	ɡ	g	44	2:	.	.
22	t_s	ts	ts	45	y:	.	.
23	d_z	dz	dz				

Table 2 Conversion table from Czech SpeechDat SAMPA to X-SAMPA.

vowels		consonants					
num	SAMPA	sonorants		plosives		affricates	
1	i	num	SAMPA	num	SAMPA	num	SAMPA
2	e	15	r	32	p	48	ts
3	a	16	r=	33	b	49	tS
4	o	17	r=:	34	t	50	dz
5	u	18	l	35	c	51	dZ
6	i:	19	l=	36	d		
7	e:	20	l=:	37	J\		
8	a:	21	L	38	k		
9	o:	22	m	39	g		
10	u:	23	F	fricatives			
diphthongs		24	n	num	SAMPA		
num	SAMPA	25	N\	40	f		
11	i_^a	26	N	41	w		
12	i_^e	27	J	42	s		
13	i_^u	28	v	43	z		
14	u_^o	29	u_^	44	S		
		30	i_^	45	Z		
		31	j	46	x		
				47	h		

Table 3 The Slovak SAMPA used in the SpeechDat database.

`v` - slovo (en. word), transcription: `slovo`

`u_^` - kov (en. metal), transcription: `kou_^`

`w` - vdova (en. widow), transcription: `wdova`

It has been decided to write the /u_^/ as [w] and the remaining variants as [v] and [w]. The character `N\`, representing uvular n, is absent in Slovak language. Therefore I suggest to map in onto the **`n`** symbol. There is also more opened variant of phoneme j, that is in SpeechDat SAMPA alphabet defined as `i_^`. This variant should be the IPA symbol j̞ and thus **`j_r`** in XSAMPA. The remaining consonant should be the same in X-SAMPA as in SAMPA. The complete mapping to X-SAMPA is in Tab. 4.

Polish SpeechDat SAMPA

The complete list of SAMPA symbols used in Polish SpeechDat is in Tab. 5. There is one change for vowel /I/, that should correspond to ɨ, written in X-SAMPA as [1]. Other vowels can be directly rewritten as [i], [e], [a], [o] and [u]. The nasalization symbol /~/ is kept in the X-SAMPA, so the remaining vowels are [e~] and [o~]. As it is written in the Polish SAMPA definition, the 'symbol stands for palatalization as well as in X-SAMPA. But consonants /s'/ and /z'/ can be rewritten as [z\] and [z\], since they correspond the IPA symbols ɕ and ʑ. Thus, the affricates /ts'/ and /tz'/ are changed the same way.

num	SAMPA	IPA	X-SAMPA	num	SAMPA	IPA	X-SAMPA
1	i	i	i	27	J	ɲ	J
2	e	e	e	28	v	v	v
3	a	a	a	29	u_^	w	w
4	o	o	o	30	i_^	j̯	j_r
5	u	u	u	31	j	j	j
6	i:	iː	i:	32	p	p	p
7	e:	eː	e:	33	b	b	b
8	a:	aː	a:	34	t	t	t
9	o:	oː	o:	35	c	c	c
10	u:	uː	u:	36	d	d	d
11	i_^a	i͡a	i_^a				
12	i_^e	i͡e	i_^e	37	k	k	k
13	i_^u	i͡u	i_^u	38	g	ɡ	g
14	u_^o	u͡o	u_^o	39	f	f	f
15	r	r	r	40	w	w	w
16	r=	r̩	r=	41	s	s	s
17	r=:	r̩ː	r=:	42	z	z	z
18	l	l	l	43	S	ʃ	S
19	l=	l̩	l=	44	Z	ʒ	Z
20	l=:	l̩ː	l=:	45	x	x	x
21	L	ʎ	L	46	h	h	h
22	m	m	m	47	ts	ts	ts
23	F	mɱ	F	48	tS	tʃ	tS
24	n	n	n	49	dz	dz	dz
25	N\	n	n	50	dZ	dʒ	dZ
26	N	ŋ	N				

Table 4 Conversion table for Slovak SAMPA

Since the main goal is to unify the transcription, the consonant /n'/ is mapped as [J]. This is the description of changes for the Polish phonetic alphabet.

Hungarian SAMPA

The Hungarian is the phonetically richest language used in this work. Its phonetic inventory (according to the SAMPA) contains 68 phonemes. The complete phonetic alphabet is in Tab. 7. Let's review the vowels at first. Hungarian official SAMPA [7] has a different definition for the long version of the phone *a*. Compared with Hungarian IPA in [2], it is decided to write it as `a:` in X-SAMPA. As well as the /O/ vowel is rewritten as [A]. Other consonants are identical in both alphabets.

The palatalized /t'/ and /d'/ with their long variants occur in other languages. It is valuable to unify them as [c] and [J\]. Consonants 64-68 should represent the salient allophones for Hungarian. The character /x/ denotes voiceless velar fricative, that is

vowels		consonants			
num	SAMPA	num	SAMPA	num	SAMPA
1	i	9	p	24	ts
2	I	10	b	25	dz
3	e	11	t	26	tS
4	a	12	d	27	dZ
5	o	13	k	28	ts’
6	u	14	g	29	dz’
7	e~	15	f	30	m
8	o~	16	v	31	n
		17	s	32	n’
		18	z	33	N
		19	S	34	l
		20	Z	35	r
		21	s’	36	w
		22	z’	37	j
		23	x		

Table 5 Polish SAMPA used in SpeechDat.

num	SAMPA	IPA	X-SAMPA	num	SAMPA	IPA	X-SAMPA
1	i	i	i	20	Z	ʒ	Z
2	I	ɨ	1	21	s’	ɕ	s\
3	e	e	e	22	z’	ʑ	z\
4	a	a	a	23	x	x	x
5	o	o	o	24	ts	ts	ts
6	u	u	u	25	dz	dz	dz
7	e ~	ẽ	e~	26	tS	tʃ	tS
8	o~	õ	o~	27	dZ	dʒ	dZ
9	p	p	p	28	ts’	tɕ	ts\
10	b	b	b	29	dz’	dʑ	dz\
11	t	t	t	30	m	m	m
12	d	d	d	31	n	n	n
13	k	k	k	32	n’	ɲ	J
14	g	g	g	33	N	ŋ	N
15	f	f	f	34	l	l	l
16	v	v	v	35	r	r	r
17	s	s	s	36	w	w	w
18	z	z	z	37	j	j	j
19	S	ʃ	S				

Table 6 Polish SAMPA to X-SAMPA conversion table.

vowels		consonants					
num	SAMPA	num	SAMPA	num	SAMPA	num	SAMPA
1	i	15	p	33	ts:	51	m:
2	i:	16	p:	34	dz:	52	n
3	E	17	b	35	tS	53	n:
4	e:	18	b:	36	tS:	54	J
5	O	19	t	37	dZ	55	J:
6	A:	20	d	38	f	56	r
7	o	21	t’	39	f:	57	r:
8	o:	22	d’	40	v	58	l
9	2	23	t’:	41	v:	59	l:
10	u	24	t:	42	s	60	j
11	u:	25	d’:	43	s:	61	j:
12	y	26	d:	44	z	62	h
13	y:	27	k	45	z:	63	h:
14	2:	28	k:	46	S	64	F
		29	g	47	S:	65	N
		30	g:	48	Z	66	x
		31	ts	49	Z:	67	h\
		32	dz	50	m	68	x’

Table 7 Hungarian SAMPA.

used for example in the Czech for denoting the pronunciation of *ch* letter, so it is proposed to map this character to the [x]. Symbols /F/ and /N/ are used for nasals *m* and *n* so they are kept as well as the symbol /h\/.The character /x'/ is in the Hungarian SAMPA used for the phoneme j. It is demonstrated as the example of the transcription of the word *kapj*, and thus transcribed as /kOpx'/ It is proposed to map it on [C]. The complete mapping of the Hungarian SAMPA to X-SAMPA with changes discussed above is in Tab. 8.

Russian SAMPA

The last used language is Russian. It’s SAMPA is in Tab. 9. The vowels are distinguished with respect to the stress. Accented vowels are marked with ” symbol. This symbols fulfills the X-SAMPA definition and all the vowels are mapped to X-SAMPA without a change. As in the case of Hungarian SAMPA, the palatalized /t’, d’, s’, z’, x’, m’/ and /n’/ are rewritten as [c,J\,zs\,z\,F] and [N]. Sounds separated by a morpheme boundary are marked with [-] symbol. It has been decided not to distinguish between such cases and write these symbols as [tS’] and [ts]. The remaining symbols have the same meaning in both phonetic alphabets.

num	SAMPA	IPA	X-SAMPA	num	SAMPA	IPA	X-SAMPA
1	i	i	i	35	tS	tʃ	tS
2	i:	iː	i:	36	tS:	tʃː	tS:
3	E	ɛ	E	37	dZ	dʒ	dZ
4	e:	eː	e:	38	f	f	f
5	O	ɑ	A	39	f:	fː	f:
6	A:	aː	a:	40	v	v	v
7	o	o	o	41	v:	vː	v:
8	o:	oː	o:	42	s	s	s
9	2	ø	2	43	s:	sː	s:
10	u	u	u	44	z	z	z
11	u:	uː	u:	45	z:	zː	z:
12	y	y	y	46	S	ʃ	S
13	y:	yː	y:	47	S:	ʃː	S:
14	2:	øː	2:	48	Z	ʒ	Z
15	p	p	p	49	Z:	ʒː	Z:
16	p:	pː	p:	50	m	m	m
17	b	b	b	51	m:	mː	m:
18	b:	bː	b:	52	n	n	n
19	t	t	t	53	n:	nː	n:
20	d	d	d	54	J	ɲ	J
21	t’	c	c	55	J:	ɲː	J:
22	d’	ɟ	J\	56	r	r	r
23	t’:	cː	c:	57	r:	rː	r:
24	t:	tː	t:	58	l	l	l
25	d’:	ɟː	J\:	59	l:	lː	l:
26	d:	dː	d:	60	j	j	j
27	k	k	k	61	j:	jː	j:
28	k:	kː	k:	62	h	h	h
29	g	g	g	63	h:	hː	h:
30	g:	gː	g:	64	x	x	x
31	ts	ts	ts	65	F	ɱ	F
32	dz	dz	dz	66	N	ŋ	N
33	ts:	tsː	ts:	67	h\	ɦ	h\
34	dz:	dzː	dz:	68	x’	ç	C

Table 8 Hungarian X-SAMPA conversion table.

vowels		consonants					
num	SAMPA	plosives		fricatives		sonants	
1	"1	num	SAMPA	num	SAMPA	num	SAMPA
2	"a	13	p	29	f	42	m
3	"e	14	p’	30	f’	43	m’
4	"i	15	b	31	v	44	n
5	"o	16	b’	32	v’	45	n’
6	"u	17	t	33	s	46	l
7	i	18	t’	34	s’	47	l’
8	1	19	d	35	z	48	r
9	e	20	d’	36	z’	49	r’
10	a	21	k	37	S	50	j
11	o	22	k’	38	S’:		
12	u	23	g	39	Z		
		24	g’	40	x		
		africates		41	x’		
		num	SAMPA				
		25	ts				
		26	tS’				
		27	t-S’				
		28	t-s				

Table 9 Russian SAMPA

3.1.3 Summary

Let’s summarize the results of proposed SAMPA to X-SAMPA mapping. Let’s denote Υ_N as the phonetic set for language N. Then there are $|\Upsilon_{CS}| = 45$ phonemes in Czech, $|\Upsilon_{HU}| = 67$ in Hungarian, $|\Upsilon_{PL}| = 37$ in Polish, $|\Upsilon_{RU}| = 48$ in Russian and $|\Upsilon_{SK}| = 48$ in Slovak. The unified global set for all the five languages consists of $|\Upsilon| = \Upsilon_{CS} \cup \Upsilon_{HU} \cup \Upsilon_{PL} \cup \Upsilon_{RU} \cup \Upsilon_{SK} = 111$ units. Then we can determine the share factor sf_5 [31], that indicates the percentage of data that can be shared to train acoustic models across known languages as

$$sf_5 = \frac{|\Upsilon_{CS}| + |\Upsilon_{HU}| + |\Upsilon_{PL}| + |\Upsilon_{RU}| + |\Upsilon_{SK}|}{|\Upsilon|} = \frac{245}{111} = 2.21 \tag{38}$$

On average, each phoneme of the unified global set is shared by two languages. That is 40 % sharing rate for 5 languages. There total number of 46 polyphonemes, thus such phonemes shared by ≤ 2 languages and 65 monophonemes that are distinct for particular languages. The exact counts an of phonemes shared by different numbers of languages can be found in Tab. 11. As it can be seen, the language with the highest number of unique phonemes is Hungarian. On the other hand, Polish is the best covered by the global set.

num	SAMPA	IPA	X-SAMPA	num	SAMPA	IPA	X-SAMPA
1	"1	ˈɨ	"1	26	tS’	tʃʲ	tS’
2	"a	ˈa	"a	27	t-S’	tʃʲ	tS’
3	"e	ˈe	"e	28	t-s	ts	ts
4	"i	ˈi	"i	29	f	f	f
5	"o	ˈo	"o	30	f’	fʲ	f’
6	"u	ˈu	"u	31	v	v	v
7	i	i	i	32	v’	vʲ	v’
8	1	ɨ	1	33	s	s	s
9	e	e	e	34	s’	ɕ	s\
10	a	a	a	35	z	z	z
11	o	o	o	36	z’	ʑ	z\
12	u	u	u	37	S	ʃ	S
13	p	p	p	38	S’:	ʃʲː	S’:
14	p’	pʲ	p’	39	Z	ʒ	Z
15	b	b	b	40	x	x	x
16	b’	bʲ	b’	41	x’	ç	C
17	t	t	t	42	m	m	m
18	t’	c	c	43	m’	mʲ	F
19	d	d	d	44	n	n	n
20	d’	ɟ	J\	45	n’	nʲ	N
21	k	k	k	46	l	l	l
22	k’	kʲ	k’	47	l’	lʲ	l’
23	g	ɡ	g	48	r	r	r
24	g’	ɡʲ	g’	49	r’	rʲ	r’
25	ts	ts	ts	50	j	j	j

Table 10 Russian X-SAMPA

polyphonemes		
Number of languages	shared phoneme count	phonemes
5	22	N,S,Z,b,d,f,g,j,k,l,m,n,o,p,r,s,t,ts,u,v,x,z
4	8	F,J,a,c,dZ,dz,i,tS
3	6	J\,a:,e,i:,o:,u:
2	10	1,2:,C,E,e:,h,s\,w,y:,z\
monophonemes		
language	count	phonemes
CS	8	E:,Eu,I,au,h\,ou,r__r,r__r__0
HU	27	2,A,J:,J\:,S:,Z:,b:,c:,d:,dz:,f:,g:,h:,j:,k:, l:,m:,n:,p:,r:,s:,t:,tS:,ts:,v:,y,z:
RU	16	"1,"a,"e,"i,"o,"u,S’:,b’,f’,g’,k’,l’,p’,r’,tS’,v’
SK	10	L,i_^a,i_^e,i_^u,j_r,l=,l=:,r=,r=:,u_^o
PL	4	e~,o~,ts\,tz\

Table 11 Table with details about shared and unique phonemes.

4 KALDI

4.1 Introduction

There are several toolboxes for the speech system development and experiments. The Hidden Markov Model Toolkit (HTK) [12] can be considered as the baseline. Between other toolkits belong for example CMU Sphinx open recognition toolkit [8] or Open-Source Large Vocabulary CSR Engine Julius [9]. In the 2009, the Kaldi project [11] was introduced. Kaldi is a progressively developing speech recognition toolkit with excellent support provided by its authors and the wide base of users. According to its development status, Kaldi allows to implement the state of the art speech recognition systems. One aim of this thesis is to provide the tutorial-style description of Kaldi usage and principles.

After the main Kaldi distribution is downloaded, the content of its directory called `kaldi-master`, is as shown in 11.

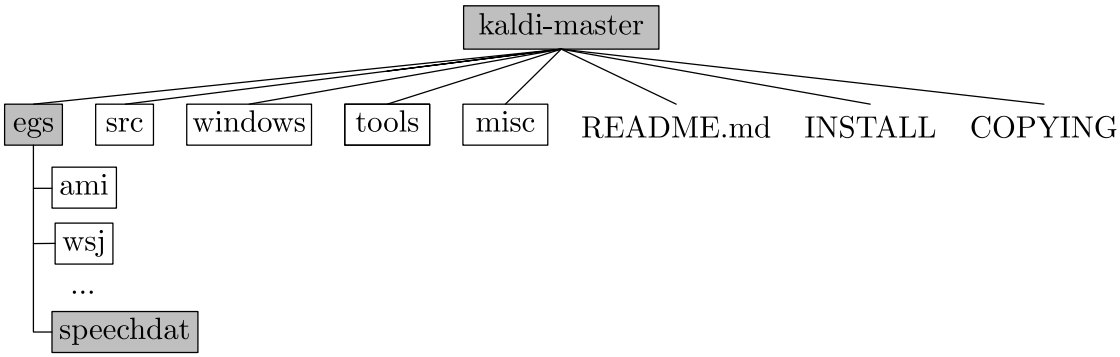


Figure 11 The content of a Kaldi distribution.

Kaldi is compiled against two main external libraries - `BLAS/LAPACK` for linear algebra and `OpenFST` for weighted finite state transducers. These libraries and other external tools that Kaldi may use are located in the **`tools/`** directory. Kaldi compilation begins in this directory by following the instructions in INSTALL file located here.

After the content of a tools/ directory is compiled, the next step is to continue with compilation in the **`src/`** directory. The own code files of Kaldi are located here divided in to the several subdirectories. When the compilation is finished, the subdirectories with *bin* suffix will contain the executable files. To get information about usage, just

invoke the desired tool with no option in the command line.

The **misc/** directory includes several scripts for model conversion back and forth between Kaldi and HTK, logos, paper and other supportive stuff.

The content of **windows/** directory is essential for working with Kaldi on Windows operating system.

The **egs/** directory should be the main subject of interest when someone wants to begin to work with Kaldi. It contains several examples, referred as recipes, mainly for experiments performed with various databases. This directory can be viewed as a source of tutorials. Note, that some database not for free and require special memberships to get them. The `wsj` recipe can be consider as base-line recipe. In the following section, the `speechdat` recipe will further be described.

4.2 SpeechDat recipe

The SpeechDat-E database is the data source for this thesis. Its recipe was published in [35] and can be downloaded from `http://noel.feld.cvut.cz/speechlab/start.php?page=download&lang=cz`. Therefore, it is used to demonstrate the Kaldi usage. Kaldi recipes usually have the structure given in Fig. 12

```
kaldi-master/egs/speechdat/s5
  |-- conf/
  |-- local/
  |-- steps/
  |-- utils/
  |-- cmd.sh
  |-- path.sh
  \ -- run.sh
```

Figure 12 Kaldi examples directory.

Kaldi is distributed not only with executable command line tools, but also with bunch of script files, that encapsulate the individual Kaldi tools or perform several tasks to ensure proper functionality and make the work with Kaldi even more efficient. These global script files, shared across all Kaldi recipes, are located in the **utils** and **steps** directories. There is no documentation with the detailed function description of the individual scripts, but the names themselves indicate what the scripts do. It is necessary to open the script, where the short description of usage is given and individual parameters listed. It is useful to search these directories when working with Kaldi tools and data files at first, to find out, if the desired functionality is already provided by some of these scripts. The usage message can be obtained when some of these scripts is called without a single parameter. As example, it can be named `steps/make_mfcc.sh`

for feature extraction, `utils/best_wer.sh` that returns the best obtained WER and so on.

The **conf** directory contains various configuration files read by the individual tools. These files are not necessary, since they actually contain the individual tools options, that can also be provided in the command line. But these files allow to immediately see the used setting and when the same configuration is needed in some other task, it can be easily set up with these files without the need of a manual writing. This is an example of the configuration file for mfcc computation typically stored in the `conf` directory:

```
kaldi-master/egs/speechdat/s5/conf/mfcc.conf
--sample-frequency=8000
--window-type=povey
--frame-length=25
--frame-shift=10
...
```

Figure 13 Example of the configuration file for MFCC computation.

The `local` directory includes scripts, that are specific to the given experiment. It contains especially data preparation files, since every database uses different structure. So everything, what is written specifically for the given task and cannot be used in general, is located here. In our speechdat recipes its content is:

```
kaldi-master/egs/speechdat/s5/local/
|-- nnet/
|-- create_LM.sh
|-- make_fea_ctucopy.sh
|-- speechdat_create_trans.pl
|-- speechdat_data_prep.sh
|-- speechdat_format_lms.sh
|-- speechda_prepare_dict.pl
|-- speechdat_prepare_dict_CS.pl
        ...
- speechdat_prepare_dict_SK.pl
```

Figure 14 The content of a speechdat recipe local directory.

Besides the directories described above, there are also come script files:

- **`path.sh`**: sets up the path to Kaldi root directory, the `KALDI_ROOT` variable has to point to the directory, where the Kaldi is compiled, denoted as kaldi-master by default. If you want to get immediate access to the Kaldi executables, just write `../path.sh` to the command line and than you call all the Kaldi tool without the need of providing full path to kaldi-master directory.
- **`cmd.sh`**: used to configure the variables for parallel computing. With respect to the individual cluster setup, the parameters for particular queues has to be modified.

- **run.sh**: summarizes the whole experiment. Described further in detail.

4.2.1 Description of run.sh script

This is the key file. All the required steps are included in this script. In general, it is not recommended to run this file at once, but run the individual step separately to see what is going on. The script was written with respect to that and allows to choose which step to be run by defining the `stage` variable. As illustrated in 15, it is also possible to select

```
# Set the path to the root speecon database directory
speechdat=/data/SPEECHDAT

# Choose language
lang="CS"
#       - CS HU RU PL SK

# stage of run.sh
stage=0;
#      0 - Data & Lexicon & Language Preparation
#      1 - Feature Extration
#      2 - AM training
#      3 - Create HCLG.fst
#      4 - Decode

tool="ctucopy"
#    - compute-mfcc # KALDI mfcc feature extraction tool
#    - ctucopy      # CtuCopy tool implementing speech enhancement and feature extraction algorithms.

# Use the following to map to X-SAMPA phoneset
xsampa_map=true
```

Figure 15 run.sh header with configuration.

which language will be processed via the `lang` variable and if the original SAMPA or the X-SAMPA should be used. There is also possibility to choose the parametrization tool. It can be the Kaldi's default `comput-mfcc` or the `CtuCopy` tool. CtuCopy tool was implemented by members of speech processing group at CTU in Prague. Its main function is to perform the speech enhancement methods and feature extraction. The speech enhancement is based on spectral subtraction with VAD, that is also implemented internally. Otherwise the extended spectral subtraction can be used. CtuCopy allows user to choose several types of filter banks and various filter shapes. When used as feature extraction tool, the common features like PLP, MFCC, LPC or TRAP-DCT can be computed. CtuCopy supports an output format compatible with HTK or KALDI standards. Detailed information can be found in official documentation provided on the web pages of CTU SpeechLab (`http://noel.feld.cvut.cz/speechlab/start.php?page=download&lang=cz`). Now we will go through the individual stages.

stage 0

The data preparation script. As it can be see in Fig. 16, the majority of the scripts, that are called here, is located in the local/ directory. Let's assume, that the chosen

language is CS.

```
if [ $stage == 0 ]; then
 echo "Data & Lexicon & Language Preparation"
 local/speechdat_data_prep.sh --tool $tool $speechdat $lang
 local/speechdat_prep_dict.sh --xsampa_map $xsampa_map $speechdat $lang
 utils/prepare_lang.sh data/$lang/local/dict "!sil" data/$lang/local/lang_tmp data/$lang/lang || exit 1;
 local/create_LM.sh  --smoothing "-wbdiscount" data/$lang/train/text.lm data/$lang/local/lm || exit 1;
 local/speechdat_format_lms.sh --lm-srcdir "data/$lang/local/lm" data/$lang/lang
fi
```

Figure 16 Stage 0 of the run.sh script.

`speechdat_data_prep.sh` generates the `data/` directory containing the individual data files that are required by Kaldi and that are desired to perform the experiment with. The majority of this directory content is crucial for Kaldi to run and it has to be created manually. The structure of this directory is illustrated in Fig. 17

```
../s5/CS/data/
|-- dev
|-- local
|-- test
\-- train
    | -- wav.scp
    | -- spk2utt
    | -- utt2spk
    | -- spk2gender
    \ -- text
```

Figure 17 Data directory content illustration.

The **`train`**,**`test`** and **`dev`** subdirectories includes the data for test, train and dev sets. Each of this subdirectory contains:

- `wav.scp` - the list of audio files. Wav.scp serves as reference file with the location of individual audio files and its mapping to an utterance id, which could by an arbitrary string that will represent the corresponding audio file in the experiment. This file has the following structure:

  `<utterance id1> <audio file1>`

  `<utterance id2> <audio file2>`

  `...`

  For example

  `A30000A1 /data/SPEECHDAT/CS/FIXED3CS/BLOCK00/SES0000/A30000A1.wav`

  `...`

  It is necessary to note, that official Kaldi feature extraction tool requires the audio files in the wav format. If the audio files are in different format, it is possible to use some conversion tool, like sox, to establish pipe for the extraction tool. Then the wav.scp would have the following format:

```
A30000A1  sox -t raw data/A30000A1.CSA -t wav  - |
...
```

(The sox parameters has to be appropriate to the used audio format).

- `text` - the transcription file. It has the format of utterance id and text, e.g.:

  ```
  A30000A1 one two four seven nine
  ...
  ```

- `utt2spk` - mapping between utterance and the speaker. Again, the speaker can be an arbitrary text to distinguish, which utterance belongs to the corresponding speaker. The format is <utterance id> <speaker>:

  ```
  A30000A1 0000
  ...
  ```

  `spk2utt` - this file contains the mapping between speaker and all his utterances. The format is <speaker id> <utterance id 1> <utterance id 2> ...

  ```
  0000 A30000A1A30000A2 A30000A3 ...
  ...
  ```

  It is not needed to create this file manually, since there are scripts `utils/utt2spk_to_spk2utt.pl` (`utils/spk2utt_to_utt2spk.pl`) that convert these files between each other.

- `spk2gender` - mapping between speaker and gender. The format is <speaker ID> <gender>

  ```
  0000 f
  0002 m
  ...
  ```

`speechdat_prep_dict.sh` generates the `dict` directory inside `../s5/CS/data/local`. Again, all the following content of the `dict` directory has to be created manually.

- `lexicon.txt` - it contains the pronunciation of individual words that occur in transcription.

  ```
  !SIL sil
  aby a b I
  adam a d a m
  ...
  ```

- `silence_phones.txt` - the list with silence phones

  ```
  sil
  ..
  ```

- `optional_silence.txt` - optional silence phones.
- `nonsilence_phones.txt`
- `extra_questions.txt`

The third invoked tool is the default Kaldi tool `utils/prepare_lang.sh`. It takes the content of `dict` directory and generates `../s5/data/CS/lang` folder with the files for training and testing, including the L.fst, which is the transducer representing the pronunciation lexicon. The next tool is the `create_LM.sh`. Its output is the arpa-format language model estimated from the provided text corpus. Finally this language model is processed with `speechdat_format_lms.sh`, transformed into G.fst transducer and the `../s5/data/CS/lang_test_lm_tg` is created. The content of this directory is used for generating the resulting HCLG.fst graph. Let's list the `lang_test_lm_tg` to summarize the output of the previous three scripts:

```
..s5/data/CS/lang_test_lm_tg/
|-- G.fst
|-- L.fst
|-- L_disambig.fst
|-- oov.int
|-- oov.txt
|-- phones/
|-- phones.txt
|-- tmp/
|-- topo
`-- words.txt
```

`word.txt` and `phones.txt` consists of mapping from text corpus words to integers as well as from phones to integers, which is the representation used by Kaldi. `oov.txt` contains the out-of vocabulary words.

stage 1

Feature extraction step. For all the signals in data/train, test and dev directories, the features are computed with the selected tool.

```
if [ $stage == 1 ]; then
 echo "Feature Extration & CMN stats for Training and Test set"
 mfccdir=/scratch/speechdat/mfcc/$lang
 for x in train dev test; do
   if [ $tool == "ctucopy" ]; then
     local/make_fea_ctucopy.sh --cmd "$train_cmd" --nj 32 data/$lang/$x exp/mfcc/$lang/$x $mfccdir/$x
   else
     steps/make_mfcc.sh --cmd "$train_cmd" --nj 32  data/$lang/$x exp/mfcc/$lang/$x $mfccdir || exit 1;
  fi
   steps/compute_cmvn_stats.sh data/$lang/$x exp/mfcc/$lang/$x $mfccdir/$x || exit 1;
 done
fi
```

Figure 18 Stage 1 of the run.sh script.

The Features are located in the directory specified with `mfccdir` variable in the form of ark files. Ark file includes all data like features, transformation matrices etc. in the form of text or binary files. To print the content of an ark file, the `copy-feats` or

`copy-matrix` can be used. The the cepstral mean variance normalization is performed by `steps/compute_cmvn_stats.sh`. In this step the `s5/exp/CS` directory is created. This directory serves for storing the experiments results including the log files. When this stage is finished, the log files with progress of MFCC ann CMVN are located here and the scp files pointing to the features and cmvn statistics are generated to the data directory.

```
exp/mfcc/CS/
|-- dev
    |-- cmvn_dev.log
    |-- make_mfcc_dev.1.log
                ...
|-- train
        ...
\-- test
        ...
```

The number associated with every log with corresponds to the number of parallel jobs.

stage 2

An acoustic model training. This stage performs the training of acoustic model in several steps.

```
if [ $stage == 2 ]; then
 echo "MonoPhone Training & Decoding"
 steps/train_mono.sh  ...

 echo "tri1 : Deltas + Delta-Deltas Training & Decoding"
 steps/align_si.sh ...
 steps/train_deltas.sh ...

 echo "tri2 : LDA + MLLT Training & Decoding"
 steps/align_si.sh ...
 steps/train_lda_mllt.sh ...

 echo "tri3 : LDA + MLLT + SAT Training & Decoding"
 steps/align_si.sh ...
 steps/train_sat.sh ...

 echo "SGMM2 Training & Decoding"
 steps/align_fmllr.sh ...
 steps/train_ubm.sh ...
 steps/train_sgmm2.sh ...

 echo "MMI + SGMM2 Training & Decoding"
  steps/align_sgmm2.sh ...

   steps/make_denlats_sgmm2.sh ...
   steps/train_mmi_sgmm2.sh ...
fi
```

Figure 19 Stage 2 of the run.sh script.

At first, the monophones are trained by `steps/train_mono.sh`. It requires the data/train and data/lang directories to be run and provides the final model in the `s5/exp/CS/mono` directory together with its log file. In the next step, the triphone system (*tri1*) is trained with delta and delta delta features. Then LDA and MLLT transformations are applied to train the model denoted as *tri2* followed with speaker adaptive training performed by applying the fMLLR to obtain the *tri3* model. feature Based on this model, other setting are trained including triphones, various transformations and other methods. Finally, the subspace gaussian mixture models are trained followed by discriminative MMI training.

stage 3

Creating the HCLG graph. Based on the resulting acoustic models, G.fst and L.fst transducers the HCLG graph is created by default `utils/mkgraph.sh`. This script

```
if [ $stage == 3 ]; then
   echo "Create HCLG.fst graphs"
   (
   $train_cmd JOB=1:1 exp/$lang/mono/log/mkgraph.lm_tg.log \
      utils/mkgraph.sh ...
   )&

   for x in tri1 tri2 tri3 sgmm2_4a; do
     (
      $train_cmd JOB=1:1 exp/$lang/$x/log/mkgraph.lm_tg.log \
         utils/mkgraph.sh data/$lang/lang_test_lm_tg exp/$lang/$x exp/$lang/$x/graph_lm_tg || exit 1;
     )&
   done
   wait
fi
```

Figure 20 Stage 3 of the run.sh script.

generates the HCLG.fst to the folder in the same location, where the final.mdl acoustic model is stored. Thus in case of monophones, it generates the resulting graph and copies the necessary words.txt and phones.txt to the `../s5/exp/CS/mono/graph_lm_tg` directory. Now everything is prepared for decoding.

stage 4:

Decoding. Since there are HCLG graphs and all required models available in the exp directory, the decoding can be performed.

Depending on the acoustic model, it is done by `steps/decode.sh`, `steps/decode_fmllr.sh`, `steps/decode_sgmm2.sh` and `steps/decode_sgmm2_rescore.sh`. Individual results are stored in the `s5/exp/<acoustic model>/decode_test_lm_tg` directory, where `/<acoustic model>` is substituted by mono, tri1 and so on. The output of this stage are decoded lattices which are post-processed with acoustic rescoring with various acoustic scale. The resulting WER values are stored in the files denoted as wer_#,

```
if [ $stage == 4 ]; then
   echo "Decode"
   for eval in test dev; do
     (
      steps/decode.sh --nj "$decode_nj" --cmd "$decode_cmd" \
                      exp/$lang/mono/graph_lm_tg \
                      data/$lang/${eval} \
                      exp/$lang/mono/decode_${eval}_lm_tg || exit 1;

                        ............
                        ............

      steps/decode_sgmm2.sh --nj "$decode_nj" --cmd "$decode_cmd" \
                      --transform-dir exp/$lang/tri3/decode_${eval}_lm_tg \
                      exp/$lang/sgmm2_4a/graph_lm_tg \
                      data/$lang/${eval} \
                      exp/$lang/sgmm2_4a/decode_${eval}_lm_tg || exit 1;

      for iter in 1 2 3 4; do
        (
        steps/decode_sgmm2_rescore.sh --cmd "$decode_cmd" \
                      --iter $iter \
                      --transform-dir exp/$lang/tri3/decode_${eval}_lm_tg \
                      data/$lang/lang_test_lm_tg \
                      data/$lang/${eval} \
                      exp/$lang/sgmm2_4a/decode_${eval}_lm_tg \
                      exp/$lang/sgmm2_4_mmi_b0.1a/decode_${eval}_lm_tg$iter || exit 1;
        )&
      done
    )&
   done$WER$
fi
```

Figure 21 Stage 4 of the run.sh script.

where # stand for the value of an acoustic scale. To obtain the best result, it is very useful to use the `utils/best_wer.sh` as given in the following example:

```
grep WER exp/CS/mono/decode_*test/wer_* | best_wer.sh
```

stage 5 (optional) In this last step, usually the DNN training and decoding is performed by run_dnn.sh located in the `../s5/local/nnet` folder. More information about DNN usage in Kaldi can be found on the page `http://kaldi-asr.org/doc/dnn1.html`. The results of DNN step are stored e.g in the exp/CS/dnn5b_pretrain-dbn and exp/CS/dnn5b_pretrain-dbn_dnn_smbr.

Based on this recipe, the multilingual system was implemented. The individual stages are almost identical except for the data preparation part in which the the lists are merged to train the models on all the data. Also for the decoding steps, the separate scripts were created to decode the GMM and DNN AM for every language individually. All these scripts are included as the part of the CD content.

5 Experiments

This chapter describes the individual experiments. The resulting recognition accuracy was observed with respect to several setups. Experiments were performed for every language individually observing the influence of a language model and phone inventory. Then the performance of the DNN acoustic model was observed. Multilingual experiments were performed in the similar manner.

5.1 Experimental Setup

Let's summarize the setup for experiments, e.g the used database, acoustic analysis configuration, the setup of the acoustic models and evaluation criterion to measure the recognition performance.

5.1.1 Used Data

As it was mentioned in the chapter 3, the SpeechDat-E [21] database is used throughout this thesis. Let's revise the applied conventions to provide immediate information required for working with these databases without the need to study the official documentation. Since SpeechDat-E contains a telephone speech, the audio signals are stored as 8kHz, 8bit, A-law audio files. Each audio file name consists of a recording session(NNNN), corpus code(CC) and language code (LL) in the format

`A3NNNNCC.LLA.`

The language code represents the following utterances types given in Tab. 12

A1-6	application words	B1	one sequence of isolated digits
C1-4	numbers	D1-3	dates
E1	one word spotting phrase	I1	isolated digit
L1-3	spelled words	M1-2	currency
N1	natural number	O1-3,O5,O7-8	directory assistance names
Q1-2	yes/no questions	S0-9/Z0-1	phonetically rich sentences
T1-2	time phrases	W1-4	phonetically rich words
X1-4	spontaneous utterances		

Table 12 Utterances types in SpeechDat-E databases.

set/lang	CS	RU	SK	HU	PL
train	43090	91079	36525	37861	42506
test	1134	1827	1117	1085	1148
dev	1135	1829	1113	1085	1164

Table 13 The total number of used utterances per language.

Let's pinpoint the noise and mispronunciation markers:

- /∼/ - word truncation
- /*/ - mispronunciations
- /**/ - non-understandable speech

Non-speech acoustic events are:

- /[fil]/ filled pause
- /[spk]/ speaker noise
- /[int]/ intermittent noise
- /[sta]/ stationary noise

All the provided languages were used, namely Czech (CS), Russian (RU), Slovak (SK), Hungarian (HU) and Polish (PL). The utterances including some noise and mispronunciation markers, listed above, were completely discarded. Every database is distributed with the list of utterances suitable for testing located in the .../INDEX/ directory, so the test set was generated using these lists and halved into the dev and test set. If provided, the train set was generated with respect to the train list, otherwise the remaining signals were used for training. Since the LVCSR task is investigated, the test and dev set consist from whole sentence utterances only, thus the signals with S and Z in their names. The resulting number of utterances for individual languages is different and by applying described approach, the resulting data set size, noted in Tab. 13, was used.

5.1.2 Front-End Processing

MFCC features were extracted with the following setup. The pre-emphases coefficient was 0.97. The 25 ms long Povey window with 10 ms shift was used. The 24 bins of filter-bank designed in the range from 100 Hz to 3800 Hz were applied. The mean value of each frame was subtracted and the resulting 13 coefficients were kept ($c(0)$ included). A further feature processing included cepstral mean and variance normalization over the speaker and the standard delta and delta-delta computation. Finally, the high dimensional vector obtained from the context of 3 frames was processed with LDA and MLLT transformation into to the resulting dimension size of 40.

5.1.3 Acoustic Modeling Setup

The HMM-GMM model was trained in the following steps. The monophone model (mono) was further expanded to the context dependent crossword triphone system (tri1) with the use of MFCC dynamic parameters. Then, the system (tri2) was trained with LDA + MLLT features. In the next step, the fMLLR per each speaker was applied (tri3). Then, the SGMM system (sgmm) based on the UBM was trained. The resulting system was discriminatively retrained with bMMI (mmi). The DNN acoustic model was trained with fMLLR features estimated by *tri3* HMM-GMM system. DNN input layer consisted of 440 inputs, this was given by the context of 5 frames and 40 dimensional fMLLR features with cepstral mean and variance normalization. DNN had 6 hidden layers, each with 2048 neurons with the sigmoid activation function and was pre-trained and initialized with RBMs. DNN was then trained with the cross-entropy training and sMBR sequence-discriminative training process.

5.1.4 Language Modeling

Only the transcriptions of the whole sentences of the training sets were used to generate the text corpus for LM estimation. Based on a such filtered training transcription, the 0-gram and 3-gram model were computed. The resulting values of perplexity(PPL) for these models and out of vocabulary words counts (OOV) are given in Tab. 14

	model	CS	SK	HU	RU	PL
PPL	3-gram	13.22	7.53	6.06	4.75	4.98
	0-gram	14802	7885	7550	9652	7453
OOV	both	0	0	0	2	0

Table 14 Perplexity values of 3-gram models and OOV counts with respect to the test set.

Compared with the 0-gram LM, the perplexity of the 3-gram models is very low. Thus, the recognition results are expected to be noticeable better when the 3-gram model is used. To explain the difference between 3-gram and 0-gram model, the transcription overlap of the individual sets was examined. Tab. 15 shows the numbers for the individual languages.

set	CS	HU	SK	PL	RU
train all	9763	9711	8963	10425	14611
test all	1134	1085	1117	1148	1827
train unique	5526	2977	3442	3104	2903
test unique	1057	900	922	939	1360
overlap	793	833	808	833	1263

Table 15 Transcription overlap for the individual languages.

If the difference is enumerated, 75 % of utterances was the same in both test and train set for Czech, 93 % for Hungarian, 88 % for Slovak, 89 % for Polish and 93 % for Russian. The majority of the utterances was included in the training set from which the LM was estimated that resulted in the obtained perplexity values. Some general LM should be used to properly evaluate the recognition performance. For this purpose, the 2-gram LMs for Slovak were estimated with the use of the Slovak national corpus `http://korpus.juls.savba.sk/prim(2d)7(2e)0(2f)models.html`. Influence of the different LMs size was observed, namely 60 000, 180 000 and 340 000 words. The individual perplexity values and OOV counts for every size of LM are noted in Tab. 16.

	60k	180k	340k
PPL	2471.15	3053.97	3397.38
OOV	725	263	115

Table 16 Perplexity values of 2-gram models and OOV counts with respect to the test set for the Slovak national corpus LM.

Based on these values, the recognition accuracy can be expected significantly worse compared to the 3-gram model estimated from the train set and obviously better then in case of the 0-gram LM.

5.2 Results

At first, the performance of the separate LVCSR with the language specific AM were examined. Then the experiments with the multilingual AM were performed. To compare and evaluate the results, the standard Word Error Rate (*WER*) criterion was used, i.e.

$$WER = \frac{S + D + I}{N} \cdot 100\%$$

where S represents the number of substitutions, D is the number of deletions, I stands for insertions and N is the total count of words.

5.2.1 GMM-HMM LVCSR with the Language Specific AM

As the first step, the HMM-GMM system was tested for every language individually. For all the following experiments, the X-SAMPA was used.

3-gram LM

The *WER* values for decoding with 3-gram model are shown in Tab. 17. The best performance for all the languages was observed for the *mmi* AM with the lowest obtained *WER* 1.77 % for the Hungarian test set. As the opposite, the Czech system performed the worst with 12.22 % *WER*.

lang	set	mono	tri1	tri2	tri3	sgmm	mmi
CS	test	17.30	12.96	12.90	12.78	12.30	12.22
	dev	15.71	12.72	12.25	12.05	12.18	12.17
HU	test	3.00	2.00	2.13	1.98	1.80	1.77
	dev	1.91	0.86	1.13	0.83	0.81	0.69
RU	test	4.40	2.60	2.63	2.32	2.30	2.38
	dev	3.83	2.30	2.51	2.71	2.56	2.65
SK	test	9.07	6.94	7.04	6.82	6.35	6.32
	dev	7.44	5.67	5.68	5.11	4.79	4.87
PL	test	7.74	8.16	8.55	8.81	7.52	7.51
	dev	8.11	8.20	8.58	8.88	8.42	8.39

Table 17 The results for acoustic models trained for every language separately and for 3-gram language model estimated from whole sentences of a training set.

These results can be compared to the LM model analysis given above. The Czech LM has the highest perplexity significantly overcoming the remaining languages and it also has the lowest overlap between the train and test utterances that confirms the obtained results. A performance of the Hungarian system is probably a result of the several influences. 93 % of the Hungarian test utterances are included in the train set that together with the low LM perplexity and a quite large amount of training data resulted in a such low *WER* value.

0-gram LM

The main task was to investigate the performance of an acoustic model. So for further experiments, it was decided to create 0-gram language model to decrease LM influence. The detailed results for this setup are in Tab. 18. Again, the *mmi* system performed the best. These results are therefore taken as a baseline. Comparison of 0-gram and 3-gram results from the previous experiment is illustrated in Fig. 22.

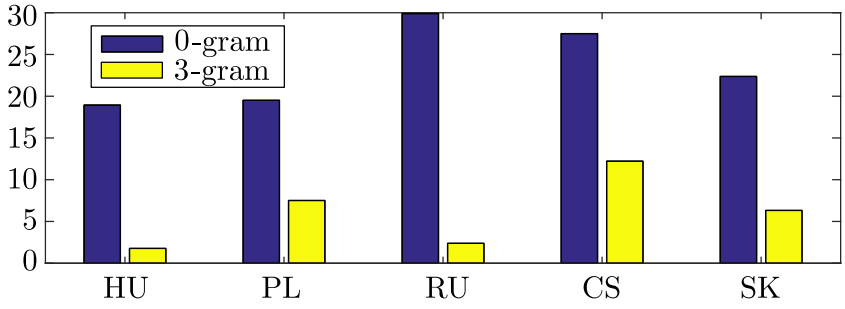


Figure 22 Comparison of *mmi* results for 3-gram and 0-gram LM.

By using 0-gram LM, the *WER* rapidly increased about 17 % on average. Hungarian is still the best performing despite the fact that it is, in our case, the language with the largest phonetic inventory. With the 0-gram language model, Russian system became

the worst one. In all the upcoming experiments, the 0-gram is supposed if it is not stated differently.

lang	set	mono	tri1	tri2	tri3	sgmm	mmi
HU	test	60.44	33.41	29.86	22.63	20.44	18.94
	dev	59.45	25.59	23.32	15.72	16.91	13.40
PL	test	52.75	34.37	31.52	23.03	20.66	19.51
	dev	53.08	28.36	25.67	18.44	18.52	14.96
RU	test	74.05	50.16	45.64	35.63	32.17	28.90
	dev	73.24	50.36	45.96	36.33	32.16	28.96
CS	test	66.39	42.57	38.32	30.76	29.57	27.50
	dev	66.08	42.71	38.01	30.19	28.47	26.55
SK	test	66.28	41.12	37.39	27.79	24.62	22.37
	dev	65.88	39.30	36.89	26.24	23.34	21.35

Table 18 Results for 0-gram language model estimated from the training set transcriptions.

Despite the fact, that individual databases should contain the signals of equal quality and phonetically rich utterances, this is not necessarily fulfilled. The differences in the utterance distributions in the train and test set across the languages led to the results presented above.

Hungarian Phone Set Reduction

The Hungarian language is the phonetically richest language with its 67 phones that was used in this thesis. As illustrated in Tab. 8, there are also the long variants for almost every consonant. With an aim to increase the coverage of Hungarian by the global phonetic set in a multilingual acoustic modeling, it implies to remove the long consonants and map them on the short ones, which reduces the Hungarian set down to 44 phones. Recognition accuracy results are in Tab. 19. When compared with results for

lang	set	mono	tri1	tri2	tri3	sgmm	mmi
HU	test	59.24	32.90	28.65	23.18	19.73	20.22
	dev	59.04	24.97	21.98	15.44	16.58	16.79

Table 19 Results for Hungarian reduced phone set.

the original set, the accuracy was about 1.28 % *WER* worse for *mmi*. But *mono*,*tri1*,*tri2* and*sgmm* systems behaved better. Although, the Hungarian phone set was remained unchanged for further experiments.

Russian Phone Set Reduction

The Russian Phone set also offers and opportunity for experiments. Its stressed vowels [o"] and [e"] are pronounced as [o] and [e], which are already defined. Also, similarly

as long consonants for Hungarian, the Russian consonants have palatalized variants. For this experiment, the palatalized variants were removed. The stressed vowel were mapped to the vowels without diacritics that are already present in the set. The results are shown in Tab. 20

lang	set	mono	tri1	tri2	tri3	sgmm	mmi
RU	test	78.46	55.23	49.94	39.03	34.68	30.78
	dev	78.27	54.20	49.34	39.00	34.71	30.98

Table 20 Results for the reduced Russian set.

The Russian phonetic set reduction increased the *WER* about 1.88 % for the best *mmi* system when compared with the original one in Tab. 18. Thus even for the Russian, whole it set was kept unchanged for the multilingual acoustic model.

Slovak National Corpus LM

To demonstrate the influence of the global language LM on the recognition performance, the several sizes of LM estimated from the Slovak national corpus were used for decoding in the Slovak system.

lang	set	mono	tri1	tri2	tri3	sgmm	mmi
60k	test	63.80	41.33	39.49	35.31	32.00	30.92
	dev	66.04	41.18	40.07	34.62	31.10	30.37
180k	test	56.99	30.82	29.71	24.30	20.48	19.72
	dev	59.82	31.76	30.62	24.35	21.23	20.37
340k	test	57.21	30.30	29.04	22.37	18.59	18.02
	dev	59.65	30.62	28.85	23.02	19.59	18.59

Table 21 Table with *WER* for a global language model used in the Slovak system.

As it was expected, the higher the vocabulary size is, the lower is the *WER* value. For 60k words of vocabulary size, the *WER* is even higher then in the of 0-gram model which is caused by the high number of OOV.

5.2.2 DNN-HMM LVCSR with the Language Specific AM

In the further experiments, the performance of DNN-HMM system was investigated. The results for the individual languages are in the Tab. 22.

The hybrid HMM-DNN system has a significant influence on the speech recognition accuracy. The improvement for the Russian test set was 5.1 % *WER*, 3.13 % for Slovak, 5.83 % for Czech, 6.78 % for Polish and 2.12% for Hungarian sMBR discriminatively trained DNN system. Thus, the DNN AM outperformed, in the case of sMBR system, the conventional GMM-HMM model about 5 % on average.

	HU		PL		RU		CS		SK	
	test	dev	test	dev	test	dev	test	dev	test	dev
dnn	19.13	15.34	18.48	16.11	26.74	27.09	25.80	25.30	22.16	21.39
smbr	16.82	13.17	12.73	15.48	23.80	23.34	21.67	22.00	19.24	18.78

Table 22 Table with the results for the DNN-HMM system.

5.2.3 GMM-HMM LVCSR with the Multilingual AM

In the following sections, the experiments with the multilingual acoustic modeling are described. Firstly, the HMM-GMM system was trained and the multilingual acoustic model was used to recognize all the languages separately.

Decoding Languages from the Train Set

All available data for all the languages were combined together to train the multilingual acoustic model. Then every language was decoded separately. Recognition results are in Tab. 23.

		mono	tri1	tri2	tri3	sgmm	mmi
CS	test	74.47	51.08	47.28	37.03	35.17	33.34
	dev	7321	50.66	46.80	36.02	34.98	32.60
SK	test	75.83	51.68	46.75	35.24	33.24	31.48
	dev	76.29	50.56	46.11	33.79	32.15	30.94
HU	test	72.96	46.71	43.29	30.26	28.78	27.53
	dev	74.75	46.16	42.40	28.29	27.46	25.94
RU	test	84.19	60.90	55.51	42.38	39.67	37.61
	dev	84.44	60.56	55.55	42.33	39.36	37.30
PL	test	65.99	43.59	39.65	28.56	27.08	25.49
	dev	66.14	41.82	37.45	27.25	26.06	24.47

Table 23 *WER* results for the multilingual AM used for the individual languages.

The lowest *WER* was obtained for Polish. This is the expected result, since the Polish language has the lowest number of the individual phones. The nasalized vowels eãnd oãre close enough to the remaining vowels in the set and the alveolo-palatal fricatives tz\and ts\are probably sufficiently modeled with fricatives in other languages. When compared with the language separate AM, there is expected accuracy deterioration on average about 4.8% caused by the increased information to be modeled.

Unseen Language Decoding

In this experiment, the multilingual acoustic model was trained without Polish, therefore the term unseen language. This was decided with respect to the result of the previous experiment. Let's revise, that the Polish phones out of the global phone set are [e∼],

[o~], [ts\] and [tz\]. For the decoding purposes, these phones were mapped onto the closest available ones in the following way:

[e~] → [e]
[o~] → [o]
[ts\] → [ts]
[tz\] → [dz]

The results are noted in Tab. 24.

		mono	tri1	tri2	tri3	sgmm	mmi
PL-unseen	test	80.40	69.03	65.23	57.66	51.82	51.11
	dev	81.15	68.74	65.76	60.07	54.20	53.32
PL-included	*test*	*65.99*	*43.59*	*39.65*	*28.56*	*27.08*	*25.49*
	dev	*66.14*	*41.82*	*37.45*	*27.25*	*26.06*	*24.47*

Table 24 *WER* results for multilingual AM used for unseen Polish language and it comparison with the AM trained with Polish

Even for the *mmi* system, the recognition accuracy is poor. Over the half of the words are recognized wrong. Despite the fact, that Polish did well in the the previous experiment, this result was unexpected and its explanation requires further examination.

5.2.4 DNN-HMM LVCSR with the Multilingual AM

In this experiment, the multilingual DNN acoustic model was trained and 0-grams of the individual languages were used for decoding. The results are noted in the Tab. 25

	HU		PL		RU		CS		SK	
	test	dev	test	dev	test	dev	test	dev	test	dev
dnn	20.95	18.39	21.41	19.50	29.73	29.49	28.71	27.29	22.76	22.50
smbr	18.79	15.98	18.16	16.15	26.45	26.83	24.58	23.63	19.81	19.73

Table 25 Table with the results for the ML DNN-HMM system.

As expected, the DNN-HMM system improved the recognition accuracy about The recognition accuracy significantly increased and even for the DNN model itself, the results are very close to the GMM-HMM systems trained for every language individually and the *smbr* system even outperformed the language dependent HMM-GMM system about 2.92 % *WER* for Czech, 2.45 % *WER* for Russian and 1,35 % WER for Polish. In the case of Slovak and Hungarian, these systems performed similarly when compared.

6 Conclusions

The multilingual acoustic modeling based on the global phones inventory was investigated in this thesis. The impact of the multilingual acoustic modeling was studied for 5 languages which are available in the SpeechDat-E databases. The analyses were performed for the large vocabulary continuous speech recognition task with the acoustic modeling based on GMM or DNN respectively as the most important task for the possible applications from the field of an automatic speech transcription. Phoneme recognition was not finally analyzed in details within this thesis. The designed systems were implemented using the Kaldi toolkit with a recommended setup and methods. The experiments were performed for the individual languages at first, that means the separate per language acoustic models were used and the influence of a language model and a phone inventory was observed. In the next step, the analyses were repeated with shared multilingual acoustic model which was created out of data from all the available languages. The most important results of this thesis are discussed in more details within the next paragraphs.

Phonetic Transcription Unification

Spechdat-E pronunciation lexicons distributed with every language use the SAMPA to represent the phonetic transcription. It was discovered, that these alphabets use a slightly different standards and every language defines its own conventions. Therefore the direct SAMPA usage was not suitable to create the global phone set. Thus, the first step was to unify the phonetic transcription. The X-SAMPA was used as the unifying phonetic alphabet because it represents the machine readable equivalent of the international IPA. Every element of the individual SAMPAs was investigated and mapped onto to the appropriate counterpart in X-SAMPA. This mapping is considered as one of the main outputs of this thesis, since it may be used in other experiments and databases as well.The current set of the shared phones across the five studied languages is ready to be extended when the data for some other languages are available.

Kaldi Recipe

One of this thesis outcome is the Kaldi recipe for the SpeechDat-E database. This recipe allows to train the GMM-HMM models of the different levels from the basic monophone model to the SGMM system. Also, the DNN model can be trained. The SpeechDat recipe was used to train the individual languages in the experimental part and it is written in a such way, that all the experiments for selected languages can be performed at once without the need of a script modification. With this recipe, the scripts performing the mapping from SAMPAs to X-SAMPA are included in the `local/` directory. This recipe was used as the base for the multilingual system, which is a part of the CD content together with the scripts for decoding of the languages separately.

Language Specific and Multilingual Experiments

In the first experiments, the 3-gram LM model estimated from the training transcription was used. The further analysis initiated based on these results indicated, that the overlap between the transcription of the train and test set was too significant and the impact of such LM led to the distorted results. Therefore the 0-gram LM model was chosen as the baseline and allowed to observe the behavior of the AM more directly. The AM model with the best performance was discriminatively trained with MMI for the Hungarian language with the best WER of 18.94 %. The language with the worst performance was Russian with 28.90 % WER. For these two language, the experiments with the reduction of phonetic set were performed with the aim of the reduction of the global phones set in the multilingual acoustic model. In the both cases, the WER increased so the the phonetic inventories of these languages were kept unchanged. The HMM-GMM multilingual acoustic model increased the WER about 8 % on the average for all the languages and the language with the best recognition accuracy was Polish with 25,49 % WER. The worst recognition accuracy of 37,61 % WER was obtain for the Russian language as well. The significant improvement was noticed for DNN-HMM acoustic model. These systems outperformed the conventional GMM about 4 % WER in the case of language separate AM.

In the case of the multilingual acoustic model, the HMM-GMM system increased the WER about 8 % on the average when compared with the language specific AMs. The DNN-HMM approach had noticeable impact on the results and it outperformed the multilingual HMM-GMM system about 9 % on average. In the case of *smbr* system, the multilingual AM results got below the WER of the Czech, Polish and Russian separate acoustic models. It achieved the similar results for the Slovak and Hungarian HMM-GMM systems.

Future work

Based on the obtained results, the following steps can extend the work that was done within this thesis. In the near future we suppose to add German and French using the data the from available Globalphone databases. For the practical usage of LVCSR it is necessary to use a more general LM for Hungiarian, Polish, and Russian and Czech. Also the further optimization of DNN-GMM multilingual LVCSR system will be investigated, as DNN-GMM LVCSR significantly overcomes base-line HMM-GMM systems.

Bibliography

[1] J.C.Wells. "Computer-coding the IPA: a proposed extension of SAMPA". In: (1995).

[2] International Phonetic Association. *Handbook of the International Phonetic Association: A Guide to the Use of the International Phonetic Alphabet*. Cambridge University Press, 1999.

[3] Josef Psutka et al. *Mluvíme s počítačem česky*. Academia, 2006.

[4] Czech SAMPA. *https://www.phon.ucl.ac.uk/home/sampa/czech-uni.htm*.

[5] Slovak SAMPA. *http://www.ui.sav.sk/speech/sampa_sk.htm*.

[6] Polish SAMPA. *https://www.phon.ucl.ac.uk/home/sampa/pol-uni.htm*.

[7] Hungarian SAMPA. *https://www.phon.ucl.ac.uk/home/sampa/hungaria.htm*.

[8] CMU Sphinx. *http://cmusphinx.sourceforge.net*.

[9] Julius. *http://julius.osdn.jp/en_index.php*.

[10] Radoslav Pavlík. "Slovenské hlásky a medzinárodná fonetická abeceda". In: *Jazykovedný časopis*. 2004.

[11] Daniel Povey et al. "The Kaldi Speech Recognition Toolkit". In: *IEEE 2011 Workshop on Automatic Speech Recognition and Understanding*. IEEE Catalog No.: CFP11SRW-USB. Hilton Waikoloa Village, Big Island, Hawaii, US: IEEE Signal Processing Society, 2011. URL: `http://publications.idiap.ch/index.php/publications/showcite/Povey_Idiap-RR-04-2012`.

[12] HTK Toolkit. *http://htk.eng.cam.ac.uk/*.

[13] Geoffrey Hinton et al. "Deep Neural Networks for Acoustic Modeling in Speech Recognition". In: *Signal Processing Magazine* (2012).

[14] Lawrence R. Rabiner and Ronald W. Schafer. "Introduction to Digital Speech Processing". In: *Foundations and Trends® in Signal Processing* 1.1–2 (2007), pp. 1–194. ISSN: 1932-8346. DOI: `10.1561/2000000001`. URL: `http://dx.doi.org/10.1561/2000000001`.

[15] Hynek Hermansky, Daniel P. W. Ellis, and Sangita Sharma. "Tandem connectionist feature extraction for conventional HMM systems". In: *PROC. ICASSP*. 2000, pp. 1635–1638.

[16] Hynek Hermansky. "Perceptual linear predictive (PLP) analysis of speech". In: *the Journal of the Acoustical Society of America* 87.4 (1990), pp. 1738–1752.

[17] R. Haeb-Umbach and H. Ney. "Linear discriminant analysis for improved large vocabulary continuous speech recognition". In: *Acoustics, Speech, and Signal Processing, 1992. ICASSP-92., 1992 IEEE International Conference on*. Vol. 1. Mar. 1992, 13–16 vol.1. DOI: 10.1109/ICASSP.1992.225984.

[18] J. V. Psutka and A. Prazak. "Multiple application of the MLLT based on clustering supported by phonetic knowledge". In: *Signal Processing, 2008. ICSP 2008. 9th International Conference on*. Oct. 2008, pp. 613–617. DOI: 10.1109/ICOSP.2008.4697207.

[19] M. J. F. Gales. "Maximum likelihood linear transformations for HMM-based speech recognition." In: *Computer Speech & Language* 12.2 (July 21, 2009), pp. 75–98. URL: http://dblp.uni-trier.de/db/journals/csl/csl12.html#Gales98.

[20] T. Anastasakos et al. "A compact model for speaker-adaptive training". In: *Spoken Language, 1996. ICSLP 96. Proceedings., Fourth International Conference on*. Vol. 2. Oct. 1996, 1137–1140 vol.2. DOI: 10.1109/ICSLP.1996.607807.

[21] P. Pollak et al. "SpeechDat(E) - Eastern European Telephone Speech Databases". In: May 2000, pp. 20–25.

[22] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2006. ISBN: 0387310738.

[23] Lawrence R. Rabiner. "A tutorial on hidden Markov models and selected applications in speech recognition". In: *PROCEEDINGS OF THE IEEE*. 1989, pp. 257–286.

[24] Lawrence Rabiner and Biing-Hwang Juang. *Fundamentals of Speech Recognition*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1993. ISBN: 0-13-015157-2.

[25] Keith Vertanen. *An overview of discriminative training for speech recognition*. Tech. rep.

[26] Jan Vanek. "DISCRIMINATIVE TRAINING OF ACOUSTIC MODELS". PhD thesis. University of West Bohemia, 2009.

[27] P. S. Gopalakrishnan et al. "An inequality for rational functions with applications to some statistical estimation problems". In: *IEEE Transactions on Information Theory* 37.1 (Jan. 1991), pp. 107–113. ISSN: 0018-9448. DOI: 10.1109/18.61108.

[28] Daniel Povey et al. "Boosted MMI for model and feature-space discriminative training." In: *ICASSP*. IEEE, Feb. 10, 2010, pp. 4057–4060. ISBN: 1-4244-1484-9. URL: http://dblp.uni-trier.de/db/conf/icassp/icassp2008.html#PoveyKKRSV08.

[29] Karel Veselý et al. "Sequence-discriminative training of deep neural networks". In: *INTERSPEECH 2013, 14th Annual Conference of the International Speech Communication Association, Lyon, France, August 25-29, 2013*. 2013, pp. 2345–2349. URL: http://www.isca-speech.org/archive/interspeech_2013/i13_2345.html.

[30] Mehryar Mohri, Fernando Pereira, and Michael Riley. "Weighted finite-state transducers in speech recognition". In: *Computer Speech & Language* 16.1 (2002), pp. 69–88. ISSN: 0885-2308. DOI: http://dx.doi.org/10.1006/csla.2001.0184. URL: http://www.sciencedirect.com/science/article/pii/S0885230801901846.

[31] Tanja Schultz and Katrin Kirchhoff. *Multilingual speech processing*. Amsterdam, Boston, Paris: Elsevier Academic Press, 2006. ISBN: 0-12-088501-8.

[32] Liang Lu. "Subspace Gaussian Mixture Models for Automatic Speech Recognition". PhD thesis. Institute for Language, Cognition and Computation School of Informatics University of Edinburgh, 2013.

[33] Zbynek Zajic. "Adaptace akustickeho modelu v uloze s malym mnozstvim adaptacnich dat". PhD thesis. The University of West Bohemia, 2012.

[34] Daniel Povey et al. "Subspace Gaussian Mixture Models for Speech Recognition". In: *ICASSP*. 2010. URL: http://research.microsoft.com/apps/pubs/default.aspx?id=118813.

[35] Petr Pollak. "KALDI Recipes for the Czech Speech Recognition Under Various Conditions". 2016.