BRNO UNIVERSITY OF TECHNOLOGY
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FACULTY OF INFORMATION TECHNOLOGY
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

AUTOMATIC SPEECH RECOGNITION SYSTEM CONTINUALLY IMPROVING BASED ON SUBTITLED SPEECH DATA

NEUSTÁLE SA ZDOKONALUJÚCI SYSTÉM ROZPOZNÁVANIA REČI ZALOŽENÝ NA OTITULKOVANÝCH REČOVÝCH DÁTACH.

MASTER'S THESIS
DIPLOMOVÁ PRÁCE

AUTHOR — **Bc. MARTIN KOCOUR**
AUTOR PRÁCE

TECHNICAL SUPERVISOR — **Dr. Ing. JORDI LUQUE SERRANO**
TECHNICKÝ VEDOUCÍ PRÁCE

SUPERVISOR — **Doc. Dr. Ing. JAN ČERNOCKÝ**
VEDOUCÍ PRÁCE

BRNO 2019

Brno University of Technology
Faculty of Information Technology

Department of Computer Graphics and Multimedia (DCGM) Academic year 2018/2019

Master's Thesis Specification

22041

Student: **Kocour Martin, Bc.**
Programme: Information Technology Field of study: Computer Graphics and Multimedia
Title: **Automatic Speech Recognition System Continually Improving Based on Subtitled Speech Data**
Category: Speech and Natural Language Processing

Assignment:

1. Get acquainted with the current techniques for automatic speech recognition (ASR) and with the KALDI toolkit.
2. Determine data sources for off-line training of an ASR system, source of gradually incoming subtitled speech data, and define a fixed test set.
3. Get acquainted with techniques for adaptation of an ASR system on loosely transcribed data.
4. Run several tests with simulated increasing data and select a suitable technique for adaptation.
5. Design and implement a KALDI-based system (recipe) for continuous improvements of an ASR system based on subtitled speech data in real-time.
6. Evaluate its functionality and discuss the results.
7. Prepare a 30 second video demonstrating your work.

Recommended literature:

- According to supervisor's and consultant's recommendations.

Requirements for the semestral defence:

- Items 1 to 4, significant advance in item 5.

Detailed formal requirements can be found at http://www.fit.vutbr.cz/info/szz/

Supervisor: **Černocký Jan, doc. Dr. Ing.**
Consultant: Luque Jordi, Telefónica
Head of Department: Černocký Jan, doc. Dr. Ing.
Beginning of work: November 1, 2018
Submission deadline: May 22, 2019
Approval date: November 1, 2018

Abstract

Today's large vocabulary speech recognition systems are very accurate. However, tens or hundreds of hours of manually transcribed speech are needed in order to train such system. This kind of data is often unavailable, or they even do not exist for the desired language. A possible solution is to use commonly available but lower quality audiovisual data. This thesis addresses the methods of processing such data for semi-supervised training of acoustic models. Afterwards, it demonstrates how to continually improve already trained acoustic models by using these practically unlimited data. In this work is proposed a novel approach for selecting data based on similarity with the target domain.

Abstrakt

V dnešnej dobe systémy rozpoznávania reči s veľkým slovníkom dosahujú pomerne vysoké presnosti. Za ich výsledkami však často stoja desiatky ba až stovky hodín manuálne oanotovaných trénovacích dát. Takéto dáta sú často bežne nedostupné alebo pre požadovaný jazyk vôbec neexistujú. Možným riešením je použitie bežne dostupných no menej kvalitných audiovizuálnych dát. Táto práca sa zaoberá technikou zpracovania práve takýchto dát a ich použitím pre trénovanie akustických modelov. Ďalej táto práca pojednáva o možnom využití týchto dát pre kontinuálne vylepšovanie modelov, kedže tieto dáta sú prakticky nevyčerpateľné. Pre tieto účely bol v rámci práce navrhnutý nový prístup pre výber dát.

Keywords

Large vocabulary continuous speech recognition, semi-supervised training, time delay neural network, subtitled speech data, acoustic modelling

Klíčová slova

Rozpoznávanie reči s veľkým slovníkom, trénovanie čiastočne s učiteľom, neuronové siete, otitulkovaná reč, akustické modelovanie

Reference

KOCOUR, Martin. *Automatic Speech Recognition System Continually Improving Based on Subtitled Speech Data.* Brno, 2019. Master's thesis. Brno University of Technology, Faculty of Information Technology. Technical Supervisor Dr. Ing. Jordi Luque Serrano. Supervisor Doc. Dr. Ing. Jan Černocký.

Automatic Speech Recognition System Continually Improving Based on Subtitled Speech Data

Declaration

Herby I declare that this master's thesis is my own work supervised by Dr. Ing. Jordi Luque Serrano and Doc. Dr. Ing. Jan Černocký. All the relevant information sources, which were used during preparation of this thesis, are properly cited and included in the list of references.

.....
Martin Kocour
May 22, 2019

Acknowledgements

I would like to thank my both supervisors Jordi Luque from Telefónica R&D and Honza Černocký for their guidance, positive attitude and lots of valuable suggestions throughout the work on this thesis. Many thanks belongs also to the researchers from Telefónica R&D department for sharing their speech resources as well as hardware power. This master thesis would not be done without their support. I would also like to thank all members of BUT Speech@FIT group for sharing their knowledge, namely Karel Veselý, Lucas Ondel, Lukáš Burget, Igor Szöke, Mireia Diez Sánchez, Ekaterina Egorova, Miroslav Skácel and Bhargav Pulugundla.

Contents

List of Acronyms

AM Acoustic Model

ASR Automatic Speech Recognition

CMLLR Constrained Maximum Likelihood Linear Regression

DNN Deep Neural Network

GMM Gaussian Mixture Model

HMM Hidden Markov Model

LM Language Model

LDA Linear Discriminant Analysis

LVCSR Large Vocabulary Continuous Speech Recogniser

NN Neural network

MFCC Mel Frequency Cepstral Coefficients

PLP Perceptual Linear Predictive Coefficients

RNN Recurrent Neural Network

TDNN Time-delay Neural Network

TID Telefónica I+D

VAD Voice Activity Detector

WFST Weighted Finite-state Transducer

Chapter 1

Introduction

Most of the recent state-of-the-art automatic speech recognition systems are based on Hidden Markov Models (HMM) to simulate the temporal variability of a speech signal. In the past, Gaussian Mixture Models (GMM) were used to evaluate how well each state of the HMM fits to the given frame, but relatively recent work by Mohamed et al. [23] showed that deep belief networks outperformed the HMM-GMM models. The deep learning approach has since became the new standard in various scientific areas such as computer vision where it reached state-of-the-art results in the ImageNet [12] competition or in natural language understanding via the word2vec [22] embedding model.

However, in deep learning and machine learning in general, the accuracy of the system highly depends on the amount and the quality of used data. The quality and variety of data is therefore also of crucial importance in automatic speech recognition (ASR). That is the reason why the data has to be manually created for specific ASR tasks. One possible solution to avoid manually transcribing speech segments is to use publicly available speech data from television broadcast (e.g,. TV news or TV shows) or from the Internet (e.g, YouTube). In case of television, the content often comes with closed-captions which can be used instead of manual transcriptions.

This thesis addresses the usage of such data for improving ASR systems. The main goal is to use these data to enhance the accuracy of already trained acoustic models as well as use them for model adaptation on different speech domains. A large portion of the work consists of data cleaning. The idea is to find speech segments, which best match the corresponding closed-captions. This work does not study the usage of these data for language models enhancement, all effort is put into improving acoustic models.

Several techniques for selecting the best utterances for training have been also explored since the amount of data is very large. This thesis proposes a novel idea for selecting utterances based on a covariance matrix estimated on raw DNN outputs. The intent is to avoid training a model on data samples which do not have a big impact on a model improvement. Therefore, the selection can also save the time and computational resources.

The thesis is organised as follows. Chapter 2 provides an overview to the theory behind essential parts of the automatic speech recognition. In Chapter 3 are discussed properties of closed captioned data and several data selection techniques. We used not only publicly available corpora but also a manually recorded data from speech, details are described in Chapter 4. The findings of both previous chapters are then used in building the baseline systems (Chapter 5). Chapter 6 describes the experiment with iterative training of DNN acoustic model using closed-caption data. Several adaptation techniques on different speech

domains were also tested on closed-caption data (Chapter 7). Finally, the conclusions are drawn in Chapter 8.

1.1 Related work

The idea of using publicly available data traces back to work done by Jang et al. [30], where they tried to improve acoustic models by accurate transcriptions from close-captioned data. They aligned the closed-captions with the speech hypotheses and selected sequences where numbers of the same words were greater than three. They selected more than 18% of data by this method and showed absolute improvement of 1.63% word error rate. Very similar idea was also presented in [14, 15], where the authors proposed a new algorithm for label alignment. The algorithm is based on dynamic time warping and can find not only well-transcribed segments but also imperfect transcripts.

A slightly different approach was explored in [13], where Lamel et al. simulated a semi-supervised approach. They trained a system on a small portion of transcribed data and used it to transcribe large amount of untranscribed data. Then they optionally removed the segments which did not correspond with closed-captions and re-estimated the models. The process was repeated until the system converged. This work also showed that the usage of filtered close-captioned data should improve a system in comparison with the unfiltered one, even if the amount of filtered data is significantly smaller.

Wessel et al. followed this idea in [37], where they introduced a confidence measure for recognized word hypothesis. The filtering process was controlled by word posteriors. Words with a posterior probability less then a given threshold were not included in a training corpus. The experiments resulted in a better WER performance on a dataset, where the word confidence was relatively high. They also tried to iteratively retrieve larger amount of data, while the seed system was trained on a smaller portion of data (1.2 h). The results were very close to the system trained on manual transcriptions. On the other hand, a similar technique was examined in [16] on YouTube corpora, but without a huge success. The problem was that a very large high quality training corpora was mixed with a lower quality large adaptation data. The experiments showed small improvement on general YouTube test set, but the accuracy on different news domain was degraded.

A more recent work done by Veselý et al. [35] examined the effect of the granularity of the confidences (per-sentence, per-word or per-frame). The conclusion is that data selection based on a word confidence is still a good practise, but the selection based on frames can easily compete with it. The authors also tried to use the confidences for weighing the data during training. This interesting idea turned out to lead to better results.

1.2 Terminology

Machine learning can be divided into the following tasks:

1. **Supervised learning** – Data are annotated with labels in supervised learning. The label defines a class of a corresponding sample. The group of samples with same class shares similar features. In case of speech recognition, a transcription of the corresponding speech is considered as a label. Common problem covered by supervised learning is *classification*.

2. **Unsupervised learning** – The unsupervised training algorithms do not use any supervision on how data should be handled. There is no teacher. Common tasks solvable by unsupervised learning are *data clustering*, *novelty detection* and *dimensionality reduction*, see [40].

3. **Semi-supervised learning** – A database consists of both labelled and unlabelled samples. The number of annotated samples is significantly less than the number of samples without annotation. Common approach is to use labelled data for training a small model. The model is then used for annotating unlabelled samples. This technique was used in [34].

The approach studied in this work could be considered as a semi-supervised learning, because the utterances are annotated with captions. However, these captions contain a lot of mistakes, so it is necessary to use some model to filter them out. Therefore, the studied approach should be recognized as something between semi-supervised and supervised learning according to the classification mentioned above.

Chapter 2

Automatic speech recognition based on DNN

The purpose of automatic speech recognition (ASR) is to correctly recognize a sequence of words corresponding to a speech signal. In a simple case, the goal is to recognize just a short sentence or a small set of words, i.e. „one“, „two“, „OK“. More often, the aim is to recognize whole sentences of natural language. However, the complexity of such a system is increasing exponentially. For instance, the spoken English vocabulary contains more than 6 000 words [26] and each word is pronounced differently by various speakers. Generally, the vocabulary used by a recognizer has a size of around 50 000 words and it has to cover all possible pronunciations of each word. That is why the system consists of an acoustic model (AM), which simulates each word as a sequence of smaller acoustic units called phonemes.

Moreover, the recognizer has to distinguish words with similar pronunciation but a different transcription. Fortunately, the word occurrence depends on the context. Some words never occur side by side in the same sentence. Furthermore, each language has some structure also known as a syntax, which is part of language's grammar. These basic principals are covered inside a language model (LM).

Figure 2.1 depicts the essential components of a large vocabulary continuous speech recognizer (LVCSR). At first, the input speech waveform is converted into a sequence of fixed size acoustic vectors in a feature extraction process (sec. 2.1). Secondly, the acoustic model generates posterior probabilities of all acoustic units for each vector (sec. 2.3). Eventually, the decoder finds the most probable sequence of words based on the posteriors and a decoding network (sec. 2.2 and sec. 2.5).

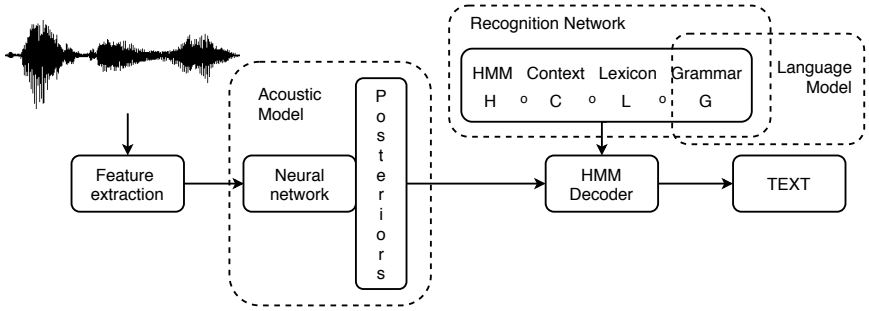


Figure 2.1: Scheme of large vocabulary continuous speech recognition system. Figure taken from ZRE lecture [33].

2.1 Feature extraction

The first step in the LVCSR pipeline is signal processing with feature extraction. The procedure seeks to provide suitable compact representation of a speech signal without losing information about spoken words. In ASR tasks, the process suppresses irrelevant information in order to reduce the dimensionality of highly dimensional speech signal. The most popular features are PLP [10], MFCC [4] and recently good results have been attained using bottle-neck features [9].

Since the Mel-Frequency Cepstral Coefficients were used in experiments, brief description of the extraction process is given. The speech signal is first segmented into 20 – 25 ms long frames at a rate of 10 ms by the windowing function. Discrete Fourier Transform is then computed on the frame. The mel-scale is then applied on the magnitudes by means a filter bank. The mel-scale is derived from better human sensitivity on lower frequencies. Finally, the discrete cosine transform is computed on log of filter bank energies. Whole process is shown on Figure 2.2. Furthermore, some discriminative transforms can be used like linear discriminant analysis (LDA) or feature-space maximum likelihood linear regression (fMLLR) to achieve more robust features, for more details see [20].

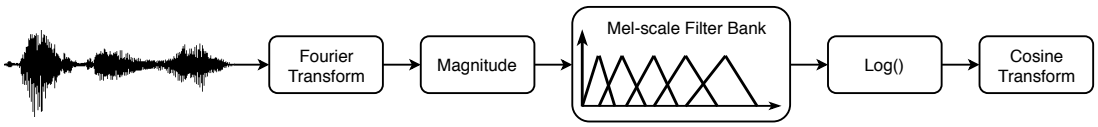


Figure 2.2: Feature extraction scheme of Mel-Frequency Cepstral Coefficients [18].

2.2 Hidden Markov Model

The goal of speech recognition is to find the word sequence $\hat{W}$ with a maximum posterior probability given the sequence of observed input acoustic vectors $\mathbf{O} = [\bar{\mathbf{o}}_1, \bar{\mathbf{o}}_2, \ldots, \bar{\mathbf{o}}_n]$

$$\hat{W} = \arg\max_W P(W|\mathbf{O}). \tag{2.1}$$

However, the probability $P(W|\mathbf{O})$ is hard to model directly. According to the Bayes Rule, equation 2.1 can be rewritten to

$$\hat{W} = \arg\max_W \frac{P(\mathbf{O}|W)P(W)}{P(\mathbf{O})}, \tag{2.2}$$

where the likelihood $P(\mathbf{O}|W)$ is a score for input vector $\mathbf{O}$ given the word sequence W generated by an acoustic model. The term $P(W)$ is the prior probability of word sequence W estimated by a language model, see Section 2.4. Assuming the probability for all input sequences $P(\mathbf{O})$ is the same, it is not necessary to consider it. So the aim of the decoder is to find the maximum likelihood $P(\mathbf{O}|W)$.

In LVCSR systems, the likelihood $P(\mathbf{O}|W)$ is modelled by Hidden Markov Models [6], where W marks sequence of context dependent phonemes (e.g. triphone) instead of words. The Hidden Markov Model, depicted in Figure 2.3, is a generative model, which has the features of finite state automaton. It consist of probabilistic transitions a_{ij} from state i to state j with left-to-right topology and output probability distributions associated with each non-terminal state. The distributions $b_j(\bar{\mathbf{o}})$ are used for scoring the decoding path for given sequence of feature vectors $\mathbf{O}$. The model representing a word sequence W is simply created by concatenating smaller individual HMMs for triphones.

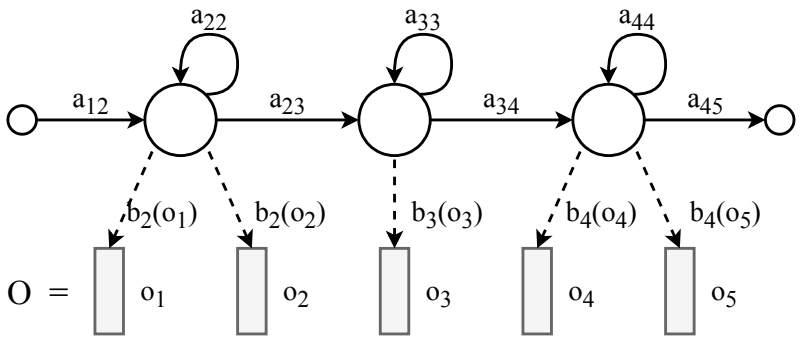


Figure 2.3: Example of a Hidden Markov Model of context-dependent phoneme with three states, i.e. senones or tied-states.

2.3 Feedforward deep neural network

Current state-of-the-art LVCSR systems use artificial neural networks for acoustic modelling. They replace the GMM models in estimating how well a feature vector fits the given HMM state. However, the GMM models are still used in force-alignment stage of the training for obtaining more precise phoneme-level transcriptions. Since the aim of this thesis is to improve the acoustic model, a brief introduction is provided.

A neural network is a structured function with trainable parameters. It maps a multi-dimensional input to an output. The network consist of many units also known as neurons. The function of the neuron is to transform its weighted input $\mathbf{x}^T\mathbf{w}$ to a scalar by applying some non-linear activation function h (e.g. sigmoid or a p-norm [39]).

These units are then grouped into larger layers. The feedforward deep neural networks (DNN) consist of an input layer, multiple hidden layers and the output layer. It is called feedforward because the neurons are connected only between neighboured layers. The output is usually normalised to become a true probability distribution. The most often used normalization function is the *softmax* [3]. A feedforward neural network with one hidden layer can be represented as:

$$\bar{y} = \text{softmax}(\mathbf{W}^{(2)}h(\mathbf{W}^{(1)}\bar{\mathbf{o}} + \bar{\mathbf{b}}^{(1)}) + \bar{\mathbf{b}}^{(2)}), \tag{2.3}$$

where $\bar{\mathbf{o}}$ is the observed feature vector, $\mathbf{W}$ is a weight matrix, where each row is the weight vector $\mathbf{w}$ for corresponding neuron and $\bar{\mathbf{b}}$ is a vector with bias terms for each neuron. More details can be found in [34, Section 2.4].

Stochastic gradient descent training

The trainable parameters are the weight matrices $\mathbf{W}$ and bias terms $\bar{\mathbf{b}}$. These variables are adjusted in a backpropagation training. The goal is to find the weight matrices and the bias terms which minimize the objective function. For the 1-of-K classification problem of n-th data point, the objective function is the multi-class cross-entropy (CE):

$$E_n = -\sum_{k=1}^{K} t_k^{(n)} \ln {y_k}^{(n)}, \tag{2.4}$$

where vector $\mathbf{t}$ has the 1-of-K encoding, so the sum always picks up only the k-th output. More recent works showed better performance by replacing the cross-entropy with maximum mutual information (MMI) or minimum Bayes risk (MBR) objective functions [11].

The idea of *stochastic gradient descent* training is to minimize the loss by doing small steps in the direction of the steepest descent of the loss function, i.e. opposite of its gradient $-\nabla E_n$. Due to the practical reasons the parameters are updated after processing M input features, where the M vary between 128 and 1024. These input vectors are selected randomly from training data and grouped into so called mini-batches. Each batch thus contains M data points. The updating formula has a form:

$$\mathbf{w}^{\tau+1} = \mathbf{w}^{\tau} - \eta \sum_{n=1}^{M} \nabla E_n(\mathbf{w}^{\tau}), \tag{2.5}$$

where η is a learning rate, $\mathbf{w}^{\tau}$ is a vector with all trainable parameters included the bias terms and $\nabla E_n(\mathbf{w}^{\tau})$ is the gradient of E_n w.r.t. weight vector $\mathbf{w}^{\tau}$. The training usually ends, when the update is very low or after given number of epochs (the number of times the whole training dataset has been observed).

Time-Delay Neural Network

Speech is a dynamic signal, that is the reason why we need to model long-range temporal dependencies between acoustic events. The architecture of Time-Delay Neural Network (TDNN), first introduced by Waibel et al. [36], is inspired by this idea. The output depends not only on the current frame but also on its context. In a standard DNN architecture, each neuron in hidden layer is fully-connected with all neurons from the previous layer. However, in TDNN, the activation of the neuron from the initial layer is estimated on a narrow context. The neurons in deeper layers are able to utilize a progressively wider context up until the last layer, which has the ability to learn from the whole context, see Figure 2.4.

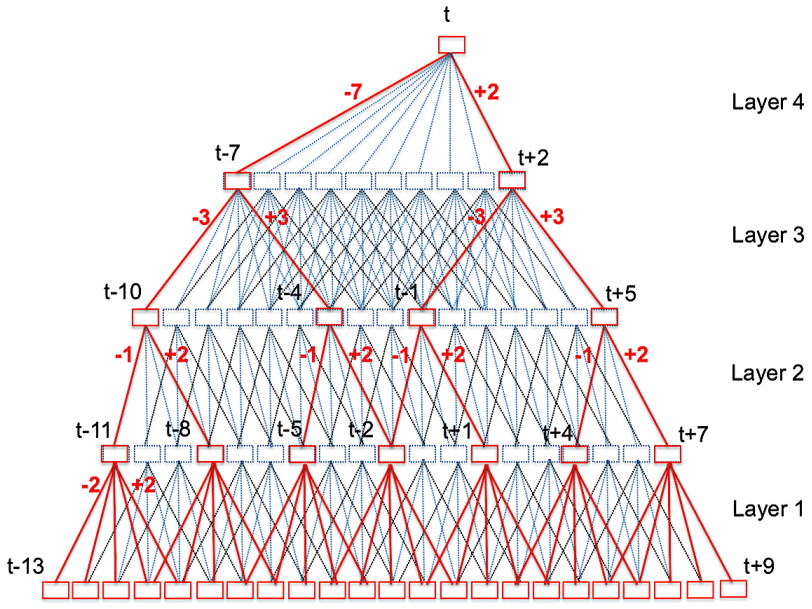


Figure 2.4: Time-Delay Neural Network with (red color) and without (blue color) sub-sampling. The figure was taken from [29].

The work done by Peddinti et. al [29] shows that not all connections are necessary. They came out with an idea of sub-sampling the neighbouring activations. The intent is to splice

only two frames from previous hidden layers and thus to save time during training. This can be done under assumption that the context overlaps between neighbouring activations in a typical TDNN architecture and thus their outputs are highly correlated.

2.4 Language model

The aim of language model is to estimate a probability of the sequence of words W, see equation 2.6. The most popular language model is the N-gram model, where $N-1$ marks the number of previous words being considered in the estimation, e.g. the trigram model considers two previous words. The model is trained on a large text corpus, where the frequency of each N-gram is divided by the frequency of all N-grams with same history, see equation 2.7. However, some N-grams might not be included in the training corpus. For such case, there is a mechanism like *Kneser-Ney smoothing* [7], which assigns them a non-zero probability.

$$P(W) = P(w_1, w_2, \dots w_k) \approx \prod_{i=1}^{k} P(w_i | w_{i-n+1}^{i-1}) \tag{2.6}$$

$$P(w_i | w_{i-n+1}, \dots, w_{i-1}) = \frac{C(w_{i-n+1}, \dots, w_{i-1}, w_i)}{\sum_{w_j} C(w_{i-n+1}, \dots, w_{i-1}, w_j)} \tag{2.7}$$

The problem of N-gram models is that they cannot represent patterns over more than a few words. In order to prevent this issue, it is possible to cover wider context of words by increasing the N-gram order, although the number of possible parameters increases exponentially. Another disadvantage is that the N-gram models assume exact match of history, they cannot model similar histories.

Therefore, current state-of-the-art language models are based on recurrent neural network (RNN LM), where the context is represented by a special hidden layer [21]. The hidden layer represents a state of the network also referred as a memory. Despite all disadvantages, we decided to use N-gram language models in this work because they can be easily and quickly trained.

2.5 Decoding network

All mentioned components are included in a single structure – the *decoding network*. The network is represented by a weighted finite state transducer, a special kind of finite state automaton, where each transition has an input label, output label and a weight (e.g. a probability). The network is a composition of four transducers [24]

$$HCLG = H \circ C \circ L \circ G. \tag{2.8}$$

The H transducer represents an HMM topology of all context-dependent phonemes. The inputs are senones from deep neural network. The context-dependency transducer C accepts context-dependent phonemes (e.g. triphones) and creates phonemes from them. The pronunciation lexicon L maps a sequence of phonemes to words. Finally, the word-level grammar G rates the probability of each possible word sequence.

2.6 Word lattice

Each frame of the audio recording is first processed by the neural network. The sequence of NN-posteriors are then processed by a decoding network. The outputs from the network are the possible word sequences. Each segment might be represented by several hypotheses. Modern decoders represent the hypothesis in a compact data structure called *word lattice*. The lattice encapsulates not only the hypothesis but also a cost (e.g. the probability) and a time information when a particular word was said as depicted in Figure 2.5. The sum of probabilities of all parallel words in a range $< t_i, t_j >$ is equal to 1.

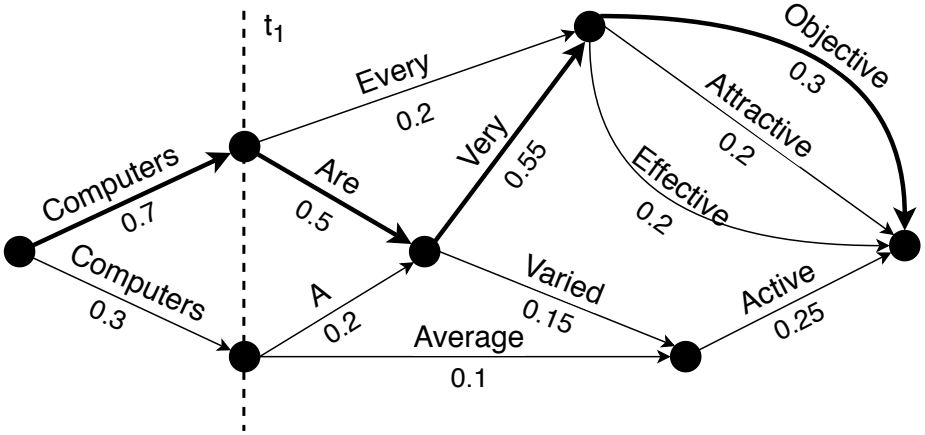


Figure 2.5: Word Lattice: a compact representation of the search space [2].

Chapter 3

Realistic data

In 1972, the National Institute of Standards and Technology first introduced the concept of closed-captioned television programs. Until then, deaf people could not watch properly the television broadcast without restrictions. The idea was to combine the audiovisual content with text files, which contained the spoken words and the corresponding playback times such that the texts can be displayed as subtitles to the video. Since then, the concept has spread massively and captions have become a legal obligation for television networks in many countries.

Another source of closed-captioned data is the Internet. For instance, YouTube, the biggest video platform, recently launched a function, which allows authors to add their own subtitles to their video. Thanks to this and to the mentioned law, there is a lot of labelled speech data, which can be used to train the acoustic models.

On one hand, there are many arguments why this data is not suitable for training robust ASR systems. One big issue is the quality of the subtitles. The training procedure is very error-prone in case of poor timing alignments of the labels. Moreover, subtitles often contains advertisement or other unrelated text. Other problems are discussed in Section 3.1.1.

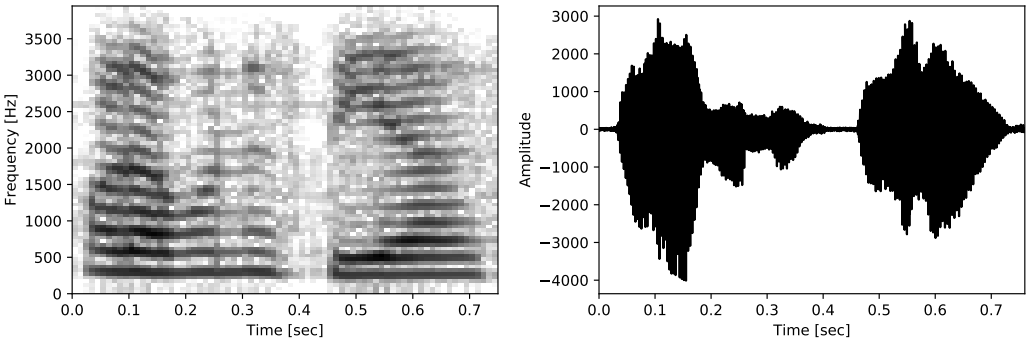
On the other hand, there are few reasons why we should consider them anyway. The first strong reason is the cost. Realistic data are publicly available and can usually be used for free. This is their main advantage when compared with paid corpora, where the prices are still very high. Another reason is their great variety. The data cover the speech in different conditions such as noisy street environment or conversations in a crowd of people, but also include clean speech from telecasters. The training process profits from such natural variability of speech. That is why we should try to extract as much data from closed-captions as possible. Section 3.2 discusses some selection techniques for obtaining high quality labels.

3.1 Data analysis

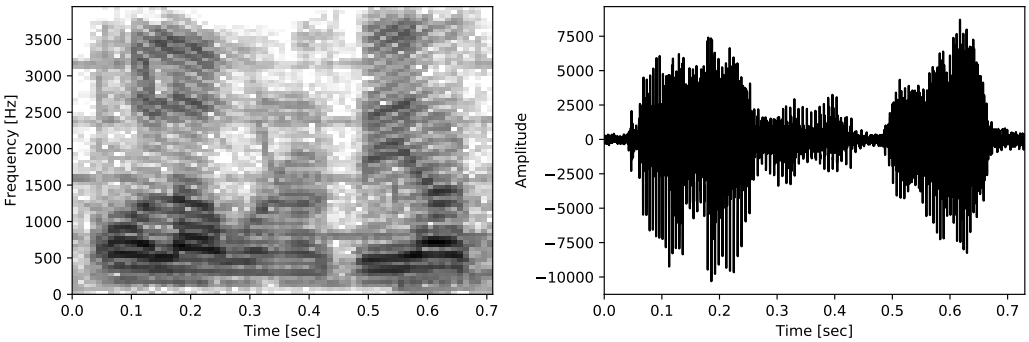
Figure 3.1 depicts a common Spanish sentence – „Hola, buenos días“ – in three different speech conditions. The first utterance was taken from call-center database, the others two are from RTVE2018 database (see Chapter 4). Spectrogram 3.1a clearly shows the word boundaries. The first 0.2 s refers to the word „hola“. In next 0.25 s, the word „buneos“ was said. Then there was a silence followed by the word „días“. However, it is much harder to find these word boundaries in the other spectrograms. Especially, the silence is not very

clear in Figure 3.1c. The reason can be found in the corresponding waveform. The signal contains a lot of noise in the background. That is also the reason why higher frequencies are not present in the signal.

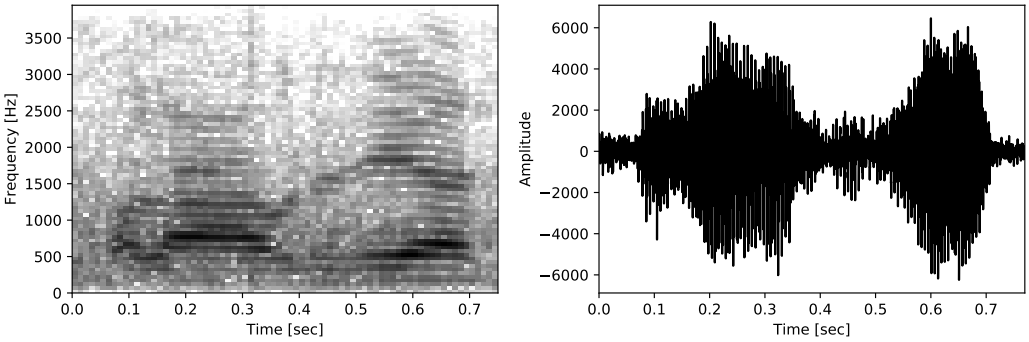
All three spectrograms are different at first sight, even if they represent the same sentence. But if we look closer, we might notice that the ending of each spectrogram is more or less similar. It differs only in the representation of individual frequencies. This positive property affects the training of the model. It can learn the acoustics of phonemes from different contexts and thus becomes more robust.



(a) The spectrogram and the waveform of the utterance in clear conditions.



(b) The spectrogram and the waveform of the utterance pronounced by TV moderator.



(c) The spectrogram and the waveform of the utterance pronounced by street reporter.

Figure 3.1: A comparison of same sentences said by three different speakers in various conditions.

3.1.1 Quality of captions

Despite the fact that closed-captions data are mostly created by professional captioners, they contain several types of errors. The following mistakes were encountered during analysis of TV shows in the RTVE2018 database.

Shifted captions. It was experimentally found that approximately 15 % of subtitles are shifted by more than 5 seconds. It is a serious issue because such captions cannot be properly aligned with the audio due to incorrect timing. In better case, the automatic aligner simply refuses such data while in worse case it tries to align them with wrong audio frames. Such data might then cause more damage than if they were not used. It is hard to say why they are shifted. In some recordings, the offset is even negative at the beginning and positive at the end. Figure 3.2 shows an example how shifted subtitles might look like.

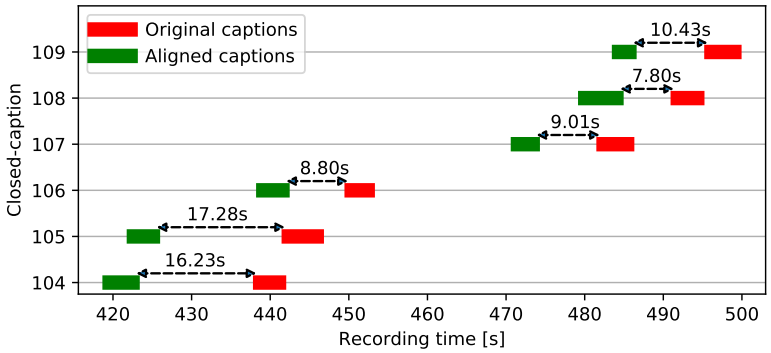


Figure 3.2: Example of shifted captions in TV news *20H*.

Partly said captions. Partly said captions are another found occurring feature. This issue involves misspelled words and not said parts. It happens mostly in dialogues, when two captions overlap. The first caption contains whole sentence said by one speaker, while the following caption contains sentence from another speaker. The problem is that the end of the first caption is before the start of the second one, while the speech related to the first caption ends after the second caption in real time. This type of error is caused mostly by the captioner. It is also a very negative phenomenon, but it happens rarely.

Not said captions. The closed-captions without corresponding speech might be found in this kind of data too. They mostly describe various non-speech sounds in the movies or in documentaries. This kind of captions might be used for training the model on non-speech sounds. In case of the YouTube video, we might come across the captions which contains the advertisement. Such data does not contain any relevant information for training. Therefore, they should be removed.

Missing captions. During advertising breaks, the closed-captions are often not broadcast. This is not an issue. The audio segments without labels are simply not added into training corpora.

3.2 Data selection techniques

In Section 3.1 was described why these data are desirable for acoustic model training despite containing several issues. The problems related to the timing can be solved by decoding the raw recordings with biased language models. The obtained transcripts are then compared with original closed-captions and best-matching sequences are retained for the training. This procedure is described in Section 3.2.1. We also explored how to retrieve the data with the highest positive impact on training acoustic models. We study two different approaches, the first based on confidences and the second based on similarity with domain data. This is described in Sections 3.2.2 and 3.2.3 respectively.

3.2.1 Transcript retrieval with biased language model

Following lines describe the technique developed by Manohar et. al. [19]. The method was used in the MGB challenge on Arabic YouTube videos, which has very similar properties as Spanish closed-captioned TV data used in this thesis. The method is implemented in a Kaldi [31] recipe[1].

The closed-captions data related to the same segment, i.e. 10 minutes or the whole show, are first spliced together according to the time when they occur. This will create a text corpus containing a few hundreds of words. The text corpus is used for training a biased N-gram language model with $N = 7$. We use term biased because the LM is adapted only on the currently processed captions. It does not incorporate the information of other segments. The purpose is to use the LM in decoding the hypothesis from the given range of time. The decoding is done by an acoustic model, which can be trained on different domain or on a small amount of transcribed speech. During the decoding, the weight of AM is significantly smaller than the weight of LM, because we believe that the captions should occur in hypotheses.

In the second stage, the best hypotheses are aligned with spliced captions using Smith-Waterman algorithm [32] to select the best matching sub-sequences. The advantage of the algorithm is that it does not treat misaligned sequences on edges as errors. Otherwise, it is very similar to the common Levenshtein distance algorithm, assuming the same cost for insertions and deletions.

Since the captions may occur in different order than the hypotheses due to wrong timing of the captions, they are split into several parts of 100 words also referred to as documents. The transcript retrieval is based on TF-IDF similarity score [1]. A high TF-IDF score is reached by a high term frequency of the given document and a low document frequency of the terms in all documents. The TF-IDF of a documents is compared with a TF-IDF score of the hypothesis. The sequence of words of the best retrieved documents are then aligned with corresponding hypotheses using the Smith-Waterman algorithm.

In order to ensure that the retrieved transcripts are correct, the whole process is repeated again using an oracle aligner[2]. In this case, the objective is to find the best paths in decoded lattices with minimum edit distance w.r.t the reference transcripts. Finally, the retrieved transcripts are segmented based on CTMs from oracle alignments. The abbrv. CTM stands for time-marked conversation file and contains a time-aligned phoneme transcriptions of the utterances. It is a standard file format for phoneme transcriptions used in Kaldi.

[1] Kaldi script segment_long_utterances.sh.
[2] Kaldi script clean_and_segment_data.sh.

3.2.2 Selection based on confidences

Data selection is very popular in semi-supervised training approach. The work usually starts with training the acoustic model on a small portion of transcribed data. This baseline model is used for transcribing unlabelled data. The resulting data are then used for training a new model. The process is repeated several times, until the achieved improvement is negligible or the maximum number of iterations is reached. In [37, 35, 38], the words in hypotheses are scored according to the reliability using word-level confidences. The words with a very low confidence are not included in the next training stage. The authors showed that this approach has an positive impact on the final accuracy.

The idea of scoring hypotheses is used also in this work. The goal is to find the best utterances, which maximize the model improvement during the training. There are at least two possible variants how to compute the word-level confidences. One variant is to use the statistics from the Minimum Bayes Risk decoding. The confidence is expressed as a quantity $\gamma(q, s)$ of the word s at the position q in all possible decoding hypotheses [35].

Another strategy is to use the NN-posteriors of senones. This strategy is based on an assumption that if a word is hypothesised correctly, then all the frames should be present in word's posteriorgram, hence the confidence should be high [38]. The NN-posterior confidence measure is calculated on the sequence of posterior estimations p_k, i.e. the softmax output of each frame. The word-level confidence score is expressed as an accumulation of the log posterior of each frame corresponding to the word w_i.

$$C(w_i, t_s, t_e) = \frac{1}{t_e - t_s + 1} \sum_{k=t_s}^{t_e} \log p_k, \tag{3.1}$$

where $< t_s, t_e >$ represents the time interval of hypothesised word. The utterance-level confidence is defined as the average of all word-level scores in the utterance u.

$$C(u) = \frac{1}{K} \sum_{i=1}^{K} C(w_i, t_s, t_e), \tag{3.2}$$

where K denotes total number of words in the utterance u.

3.2.3 Selection based on similarity with target domain

This work presents a novel method for extracting data. The method is based on selecting those data which share the same characteristics as the target domain. It is believed that this kind of data should maximally boost the acoustic model training towards the direction where the error rate on evaluation data has a decreasing trend. This idea was inspired by the work done by Pascanu et. al. [28]. The authors presented the Natural Gradient Descent algorithm for training deep neural networks. The main difference between the classical gradient and natural gradient is that the classical gradient points to the steepest direction of the loss-function, whereas the natural gradient points to the direction, where the loss-function has a local minimum.

A very similar approach was used in the proposed selection method, where the main goal is to score utterances according to the similarity with reference evaluation data. The data samples with high similarity rating are included into training corpora. It is done by comparing a test covariance matrix $\mathbf{\Sigma_r}$ with a covariance matrix $\mathbf{\Sigma_c}$ estimated on closed-caption data. Each covariance matrix $\mathbf{\Sigma}$ is calculated on raw DNN outputs $\mathbf{Y}$, i.e. the outputs from hidden layers before normalization and softmax function, see Figure 3.3.

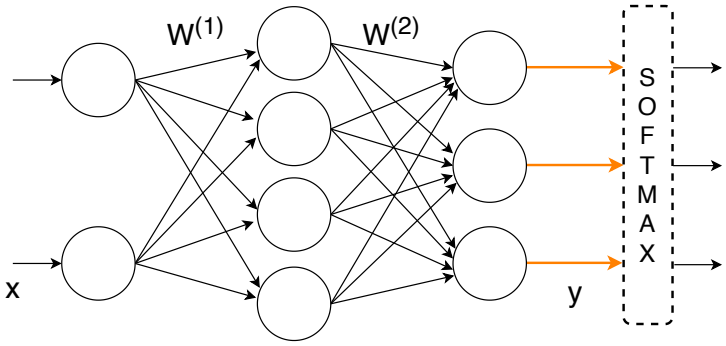


Figure 3.3: The activations of the simple DNN before the last softmax layer, marked with orange colour.

The output representing silence or other non-speech classes is not included, because it can greatly affect the result and obscures the speech classes. The auto-covariance matrix is computed as:

$$\mathbf{Y} = [\bar{\mathbf{y}}_1, \bar{\mathbf{y}}_2, \ldots, \bar{\mathbf{y}}_N] \tag{3.3}$$

$$\mathbf{A} = \frac{1}{N}\sum_{i=1}^{N} \bar{\mathbf{y}}_i \bar{\mathbf{y}}_i^T \qquad \bar{\mathbf{y}}_i \in Y \tag{3.4}$$

$$\mu_{\mathbf{Y}} = \frac{1}{N}\sum_{i=1}^{N} \bar{\mathbf{y}}_i \tag{3.5}$$

$$\mathbf{\Sigma} = \mathbf{A} - \mu_{\mathbf{Y}} {\mu_{\mathbf{Y}}}^T, \tag{3.6}$$

where $[\bar{\mathbf{y}}_1, \bar{\mathbf{y}}_2, \ldots, \bar{\mathbf{y}}_N]$ is the sequence of DNN outputs before softmax, $\mathbf{A}$ is the correlation matrix and $\mu_{\mathbf{Y}}$ is the mean vector.

The last problem is to determine the distance between two matrices by a scalar value. There is a plenty of matrix norms which might be suitable for transforming the matrix similarity into one value. The problem is that the most of them consist of a trace operator, which simply sums only the values on the main diagonal. However, the covariance matrix contains a lot of valuable information in other elements. After several experiments, we therefore decided to implement a custom norm for this purpose. The norm uses sum over all elements instead of the trace operator. The similarity metric is expressed by Equation 3.7 and it uses the *Frobenius norm*. The experiments with other norms are described in Section 7.2.

$$\mathbf{\Sigma_r} \approx \mathbf{\Sigma_c} \quad \Leftrightarrow \quad \|\mathbf{\Sigma_r} - \mathbf{\Sigma_c}\|_F \Rightarrow 0 \tag{3.7}$$

$$\|\mathbf{\Sigma_r} - \mathbf{\Sigma_c}\|_F = \left(\sum_{i=1}^{m}\sum_{j=1}^{n} (\mathbf{\Sigma}_{r_{ij}} - \mathbf{\Sigma}_{c_{ij}})^2\right)^{\frac{1}{2}}. \tag{3.8}$$

Chapter 4

Datasets

In this work, three Spanish datasets were used with different properties – TID, RTVE2018 and Custom TV. TID database was collected by Telefónica Investigación y Desarollo. It includes recordings from call centres and phone calls as well as licensed data corpora. It contains mostly clean speech without background noise. On the contrary, the RTVE database includes Spanish TV shows and TV news with a lot of noisy speech data. The database was used in the IberSPEECH-RTVE Speech to Text Transcription challenge, a part of the Iberspeech 2018 conference[1]. The Custom TV database consists of manually recorded TV channels. The database shares a similar content with the RTVE database. It contains also Spanish movies and documentaries. Table 4.1 shows an overview of the content of each database.

Database	Train	Dev	Eval	**Total**
TID	186	3	1.5	190.5
RTVE	433	72	41	546
Custom TV	2 160	-	-	2 160
Total	2 779	75	42.5	**2 896.5**

Table 4.1: Numbers of hours of speech in individual databases. Custom TV database had 2160 hours before cleaning.

4.1 TID – Telefónica in-house database

The TID database contains Spanish call centre data. The phone calls were recorded by Telefónica Móviles in 2016 with the users consent. The data collection was performed taking into account a broad coverage of speaker and dialect variability. It consists of 22 hours of speech in total. The test dataset comprises 1.5 hours of audio for measuring the ASR performance. In addition, the following licensed data corpora are included for improving acoustic models and lexicon:

a) *SALA* [25][2] – Phonetically annotated database of separate words in Latin American Spanish recorded over telephone network. The database comprises eight different dialectal areas in Latin America. The TID database contains only the Chilean dialect

[1] http://iberspeech2018.talp.cat

[2] http://universal.elra.info/product_info.php?products_id=199

part of SALA, with 1,024 Chilean speakers recorded over the Chilean fixed telephone network. It comprises a broad range of recordings, sequences of 10 isolated digits, questions, time phrases, spelled-out words, etc. as well as a lexicon and a pronunciation dictionary.

b) *Fisher Spanish Speech* [8][3] – The database developed by Linguistic Data Consortium (LDC), comprises recordings covering 163 hours of spontaneous telephone speech from 136 native Caribbean Spanish and non-Caribbean Spanish speakers with full orthographic transcripts. The recordings consists of 819 telephone conversations lasting around 10 to 12 minutes each.

4.2 RTVE2018

The RTVE database [17][4] comprises 15 different TV programs broadcast between 2015 and 2018 by the Spanish public TV station called Radiotelevisión Española (RTVE). The programs comprise a great variety of speech scenarios from read speech to spontaneous speech, live broadcast, studio debates, etc. They cover also different Spanish accents, including Latin-American ones. The database consists of 569 hours of audio data, from which 460 hours are provided with subtitles, and 109 hours have been human-revised. The database is provided in 4 different partitions:

- train, that includes 433 hours of data with closed-captions from 16 TV shows,
- dev1, that consists of 57 hours and human-revised labels,
- dev2, that contains 15 hours with human-revised labels,
- test, that comprises 41 hours audio without labels used for evaluation of participants' systems.

The train and dev1 partitions were used for training the acoustic models and dev2 dataset was kept for measuring the performance.

4.3 Custom TV – Manually recorded Spanish TV channels

Custom data were obtained by recording Spanish TV channels and corresponding closed-captions. We recorded two channels for 6 months and obtained 2160 hours of audio together with 1 770K subtitles (14M words). The content covers huge variability in speech signal. The database comprises not only Spanish TV Shows and TV news but also various movies and documentaries. On the other hand, some segments do not contain closed-captions, e.g. the advertisement. These data are simply not used.

The TV stream was recorded via a USB TV tuner. The stream consists of *SRT* files with closed-captions and audio encoded in *MP2* format. Due to legal rights, the content is not stored directly. The audio is instead subsampled using 8000 Hz and converted into MFCC+Pitch features. The database was first cleaned, because the subtitles were not properly aligned with the audio. We keep only 200 hours (10 %) of clean data for training acoustic models.

[3] https://catalog.ldc.upenn.edu/LDC2010S01
[4] http://catedrartve.unizar.es/reto2018.html

Chapter 5

Baseline system

This chapter provides details about the used baseline speech recognition systems.

5.1 Used tools

Kaldi toolkit

This thesis used the Kaldi speech recognition toolkit [31], which is an open-source toolkit for speech recognition research. Its core is implemented in C++, using mathematical library ATLAS and OpenFST library for working with finite-state transducers. Most of the functionality is wrapped into a bash script, which makes work with the toolkit easy. The main advantage of the toolkit is that it comes with a number of complete recipes for building ASR on popular databases.

Voice activity detector

As it was mentioned, the *dev2* partition from RTVE2019 database was used for measuring the performance on television speech data. However, the audio segments were not properly time-aligned with transcript. This issue might cause inaccurate measurement the performance. One possible solution was to clean dev2 data with Kaldi scripts, but some data would be lost and, moreover, the evaluation would be strongly biased by the model used for data cleaning.

Therefore, we decided to segment the data with voice activity detector developed by Veselý et. al. [27]. This detector is based on DNN acoustic model and is able to recognize 5 different classes: speech, non-speech, bad-speech, unknown, music. The frames with a logit posterior higher than a specified threshold are simply considered as a speech. The segments consist only of a few speech frames are not counted as a speech segments. The recommended value for the logit threshold is -0.5. However, it is desired to obtain as much speech segments as possible in this case, so the logit threshold is set to -1.0 (Figure 5.1). The final segmented dev2 partition comprises 14 hours of speech segments.

5.2 Language modelling

The baseline language model was trained on 90% of Fisher database which is part of TID data corpus. This model was introduced in the work done by Egorova [5]. The TID language model was used in the experiments, where we tried to improve the models on

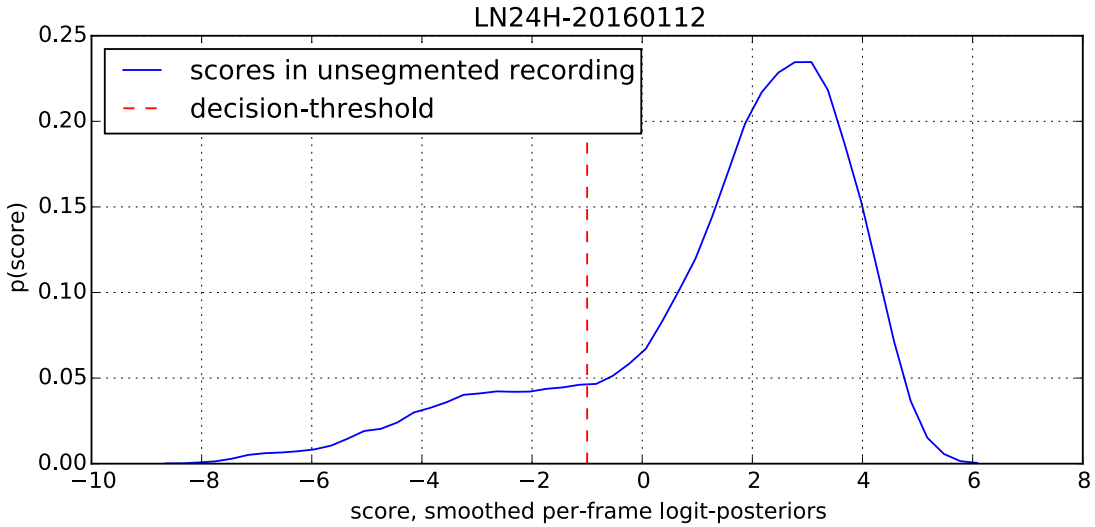


Figure 5.1: The plot of the score histogram of „La noche en 24 horas“ TV show. The frames with score value less than −1.0 are recognized as non-speech.

telephone speech domain. Several language models were also created on top of the combination of all datasets – TID, RTVE2018 and Custom TV. We decided to combine the text transcripts from *dev1* and *train* from RTVE2018 database together with all transcripts from TID database and Custom TV database. We did experiments with both 4-gram and 5-gram language models smoothed with *Kneser-Ney* algorithm. We achieved relatively good performance with 5-gram LM trained on all available data (see Table 5.1), but the $HCLG$ transducer requires a lot of RAM during decoding, hence we could not use parallel decoders to accelerate the decoding process. Therefore, we decided to use 4-gram LM in further experiments, which was trained on the same portion of transcripts and achieved very similar performance.

Language model properties			**WER** [%]	
Data	N-gram	Lexicon size	test	dev2
TID	3-gram	36k words	38.12	90.72
TID+RTVE+TV	4-gram	36k words	**36.05**	78.41
TID+RTVE+TV	4-gram	46k words	39.35	75.76
TID+RTVE+TV	5-gram	46k words	50.02	**75.34**

Table 5.1: Results for n-gram language modelling improvements. The test partition from TID database contains telephone speech data, while the dev2 from RTVE2018 contains television speech data. High WER on dev2 partition is caused by the decoding problem. We tried to decode unsegmented dev2 partition with online decoder from Kaldi, but it did not help much.

5.3 Acoustic modelling

All DNN acoustic models share the same setup configuration. The training script is based on Kaldi NNet2 recipe[1] for Switchboard database. We decided to use this script because it is based on English telephone speech data, a very similar database to Fisher Spanish telephone corpus we used. We preferred the NNet2 implementation because we needed to decode online the audio recordings from dev2 partition of RTVE database due to inaccurate timing of the segments. Unfortunately, the online decoder does not work properly, so we decided to segment the data with VAD described in Section 5.1.

The model is a feed forward neural network with 4 hidden layers and softmax output layer. More specifically, it is a TDNN network [29] with multiple levels of splicing. All 9 frames are spliced together $[t-4, t+4]$ at the input layer. The context of $t-5$, $t-1$ and $t+3$ frames is spliced in a second layer. Other layers are connected without splicing. The output of each hidden neuron is computed by a p-norm function with $p = 2$. The p-norm function is a standard Maxout non-linearity used in the Kaldi toolkit [39].

Each frame is represented by 13-dim MFCC and 3-dim pitch features. Therefore, the input layer consists of 144 feature vectors. The DNN also contains the LDA transformation layer to transform the correlated input in feature space. The output of the DNN represents the posteriors of roughly 3700 different senomes. We model the acoustics of 169 Spanish phonemes. The phonemes with stressed vowels are represented as independent classes.

TID models

The baseline GMM-HMM model was already built by Telefónica researchers using speaker-adaptive techniques. The model is trained on 40-dimensional MFCC + Pitch features, on which LDA, MLLT and fMLLR transformations are applied. This model was used to align the labels with corresponding audio frames for DNN training. We also use it for data cleaning. The name of the model is **tri2LDA_SAT**. The baseline DNN-HMM acoustic model was trained for 9 epochs on the same 190 hours of telephone speech used in the GMM-HMM training. The model is named as **nnet_ms_a_v2**. We have built also a DNN model, which is trained for 6 of epochs on TID database. This model was used in iterative training as a seed model, which was further improved in next iterations. In is called **nnet_ms_a_v1**. The results of all models are shown in Table 5.3.

RTVE2018 models

Creating the baseline system on RTVE2018 database was not as simple as in case of TID database. The data in *dev1* and *dev2* partitions contains a lot of serious issues even though they were manually annotated by professional captioners. The problem was with the timing of segments (see Section 3.1.1). So we decided to remove the segments and discard the speaker information.

Data cleaning

The baseline system development began by cleaning the data. The transcripts in dev1 partition were manually extended by non-speech labels (i.e. *noise* or *silence*) in order to improve the silence model. After the data augmentation, the Kaldi 'segment long utterance' and 'clean and segment' scripts were applied to automatically fix the timing of

[1] https://github.com/kaldi-asr/kaldi/blob/master/egs/swbd/s5b/local/online/run_nnet2_ms.sh

Data partition	Before cleaning [h]	1st cleaning [h]	2nd cleaning [h]	% Loss
dev1	57	39.25	41.16	-27.78
dev2	15	10.54	11.04	-26.40
train	433	-	82.24	-81.00

Table 5.2: Number of hours of data before cleaning and after cleaning.

transcripts (details were described in Section 3.2.1). The data cleaning was performed using the tri2LDA_SAT model. These scripts also removed utterances which were unlikely to occur in the recordings.

Using the clean dev1 data, a new SAT model **tri4a** was trained. Finally, the whole cleaning process was performed again, but with the new GMM model. This resulted in 41.16 hours of data in dev1 and 11.04 hours of data in dev2 (see Table 5.2). We lost more than 81.06% of train data. This is because the model was trained on a very small amount of data, but it is not a significant issue.

DNN models

Two DNN-HMM models were built. Both models share the same TDNN architecture, described above. The **nnet_ms_albayzin_v1** was trained on 41 hours of dev1 dataset with manually annotated non-speech segments. It was trained for 6 epochs, which corresponds to 24 iterations. The training data of the second version consist both the dev1 and train datasets, which resulted into 123 hours. The model was trained for 15 epochs. By incorporating the train set, the overall accuracy decreased by 8.2% relative, from 32.04% to 29.41%. Overall word error rate results are shown in Table 5.3. The best result on telephone speech domain was achieved with DNN-HMM model trained on TID database. On the other hand, the model trained on the combination of closed-captions and manual transcriptions achieved the best result on television speech domain.

Name	Data	Type	LM	Lexicon	% WER test	% WER dev2
tri2LDA_SAT	train^{+}	GMM	TID	36k words	45.13	68.46
nnet_ms_a_v1	train^{+}	DNN	TID	36k words	38.12	53.26
nnet_ms_a_v2	train^{+}	DNN	TID	36k words	**37.26**	53.16
tri4a	dev1	GMM	MIX	46k words	60.37	43.69
nnet_ms_albayzin_v1	dev1	DNN	MIX	46k words	58.31	32.04
nnet_ms_albayzin_v2	dev1 + train*	DNN	MIX	46k words	54.68	**29.41**

Table 5.3: Word error rate [%] results for acoustic modelling improvements. The *test* and *train*$^{+}$ data (telephone speech) are from TID database, while *dev1*, *train*$\star$ and *dev2* data (television speech) are from RTVE2018 database. The MIX language model refers to the combination of RTVE2018, TID and Custom TV databases.

Chapter 6

Experiments with iterative training

This chapter describes an experiment with acoustic model retraining using manually recorded television speech data (Custom TV database). The idea is to train iteratively several models per week of TV data (109 hours on average): in each iteration, we continue with training the model from previous iteration but with the data from the next week. The goal is to use these data for improving the performance of baseline model. The advantage of iterative training is a relative resistance to erroneous data. The inaccurate labels does not affect the performance as much as it is in the case of training from scratch because the learning rate at higher iterations is significantly lower. Another advantage is that this way can save a lot of computational time and resources in comparison with training from scratch.

This chapter contains the description of iterative training procedure in Section 6.1, the results from evaluation are provided in Section 6.2. The results for television speech domain are interpreted in Section 6.3. Other experiments are described in Section 6.5. The experiments follow the findings described in Chapter 3.

6.1 Description of the iterative training

The model was first trained on telephone speech for 6 epochs. It was then retrained in a cycle on training data from both cleaned TV database and telephone speech database. The retraining continued until the exhaustion of the Custom TV database. All iterations used the same language model trained exclusively on telephone speech data (see Section 5.2). The formal description of iterative training is demonstrated in Algorithm 6.1.

Data preparation

Manually recorded television data were collected every day and are organized into buckets, where each bucket comprises audiovisual content of one week of recording. The MPEG-2 audiovisual broadcast was first converted into WAV audio stream subsampled to 8000 Hz frequency. The original audio was compressed using linear pulse-code modulation quantization on 16 bits. The WAV stream was then converted into MFCC+Pitch features using Kaldi scripts. We did not store the original audiovisual content. The closed-captions were streamed in SubRip (SRT) format. Each caption consists of a time stamp and the corresponding text. This text stream was processed and stored in files in Kaldi-like format.

Algorithm 6.1: Formal description of iterative training.

Input: TID database – TID_DATA
Input: Clean TV database – TV_DATA
Result: Several DNN acoustic models

train the *model* for 6 epochs on TID_DATA
data ←TID_DATA
tvcursor ← createDbCursor(TV_DATA)
tvcursor.moveToFirst()
while tvcursor.hasNextWeek() **do**
    tvdata ← tvcursor.GetNextWeek()
    tvcursor.MoveToNextWeek()
    tvdata ← clean *tvdata*
    data ← combine *data* with *tvdata*
    i ← compute the last iteration of the *model* according to Equation 6.1
    retrain the *model* for 1 more epoch on *data*, start the training from iteration *i*
    evaluate the *model*
end

Data cleaning

As mentioned in Section 3.1, the data are full of inaccurate transcripts. Consequently, not all 109 hours were used in training. These data were first cleaned with 'segment long utterances' and 'clean and segment' Kaldi scripts using tri2LDA_SAT GMM model (see Section 3.2.1 for details). The implicit parameters were slightly modified because the objective was to retain as much data as possible.

The N-gram order of biased language model was set to $N = 7$ because we believed that most words should occur in utterances in the same order as it was written in corresponding transcripts. We have also changed the maximum number of words for TF-IDF source document. The implicit value was adjusted from 1000 to 500 because the captions might be rearranged in real time due to a mistake in timing of the caption. Even though, altering this parameter meant lowering the accuracy of TF-IDF searching. Finally, the last changed parameter was word mismatch error rate, which is used in comparing the transcripts with decoded hypothesis. This parameter was set to WMER = 70% from the implicit value of 50%. These changes resulted in 10 hours of clean data per week on average, whereas only 5 hours were obtained on average before parameter tuning. All results are shown in Table 6.1.

Week	1	2	3	4	5	...	18	Total
Original [h]	94.32	102.91	105.63	113.35	105.43	...	114.97	1967.05
Cleaned [h]	6.83	9.16	12.49	10.80	7.01	...	11.62	**181.30**

Table 6.1: Number of hours of original data per each week before and after cleaning. The last column shows the total number of hours of all 18 weeks.

Data combination

The clean TV data were split into 10-hour blocks and then they were combined with data from previous iterations. In the first iteration, the data were mixed with telephone speech data in order to prevent catastrophic forgetting during the retraining of the first model. The mixture in the last iteration comprised all 18 weeks of Custom TV and the training partition of TID dataset, i.e. 367.3 hours. The audio was also force-aligned with transcripts using the GMM model.

Model retraining

All Kaldi NNet2 models are trained in the same fashion. The training labels are divided into archives also called egs. Each archive contains approximately 400 000 frames. The models are trained either on CPU or on GPU in parallel. At the beginning, the training runs on few processes, e.g. 4 (parameter 'initial jobs') and the number of processes rises during the training until it reaches the maximum (parameter 'final jobs'). The training is divided into several iterations with each iteration processing only a subset of training labels. Training for one epoch comprises minimum number of iterations, where all labels were used. The final model consists of the averaged combination of weight matrices trained in previous iterations (e.g. 20).

In iterative training experiment, the first model was trained for a few epochs (e.g. 6) from scratch and the following models continued with training using the final iteration of the previous model (they are not trained from scratch again). In order to do this, we need to know the last iteration of the predecessor. The iteration can be set manually or it can be derived as:

$$\text{iters} = \frac{2 \times \text{epochs} \times \text{frames}}{400\,000 \times (\text{initial jobs} + \text{final jobs})}. \tag{6.1}$$

System evaluation

In the last step, the retrained model was evaluated using word error rate metric with zero insertion penalty. There was no need to recreate the WFST because the final output layer of DNN model remained the same during all iterations.

6.2 Telephone speech evaluation

In this experiment we first trained the model on train set of TID database for 6 epochs (see nnet_ms_a_v1 in Table 5.3) with WER = 38.12% on TID test set. This model was then retrained in the same way as it was described in Section 6.1.

The overall results of iterative training are demonstrated in Figure 6.1. As the results suggest, the impact of incorporating the data from out-of-the domain database – Custom TV is significant. The best word error rate result was achieved in 9^{th} iteration, in which the training data consisted of 180 hours of telephone speech and 90 hours of television speech (270 hours in total). In comparison with the baseline model trained on TID database for 9 epochs, the word error rate was decreased by 3.24% relative, from 37.26% to 36.05%. From 9^{th} iteration the word error rate varies between 36.22% and 36.54%. This phenomenon might be caused by the fact that the training data in higher iterations contains a lot of out-of-domain data. The model simply started to adapt on different domain and forgot some characteristics of telephone speech.

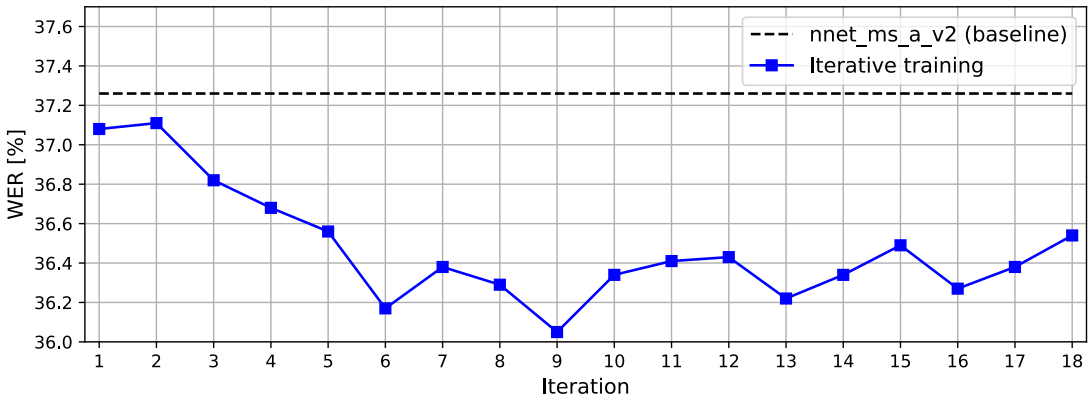


Figure 6.1: Overall WER results of iterative training (blue line) evaluated on the test set of TID database. The dashed line indicates the WER result of the baseline model trained for 9 epochs on TID database.

6.3 Television speech evaluation

Thanks to the assumption made in the previous section we were curious about how the models would perform on television speech domain. In this experiment we used the same systems (including the baseline) and evaluated them on dev2 dataset from RTVE2018. The word error rate was significantly decreased by incorporating Custom TV data as depicted in Figure 6.2. It was reduced by 16.23% relative, from 53.16% to 44.54%. This finding is not very surprising because the Custom TV database is very similar to the RTVE2018 database. The only difference is in the recorded TV channels. The green dashed line demonstrates the performance of the model trained on RTVE2018 database (see nnet_ms_albayzin_v2 in Table 5.3).

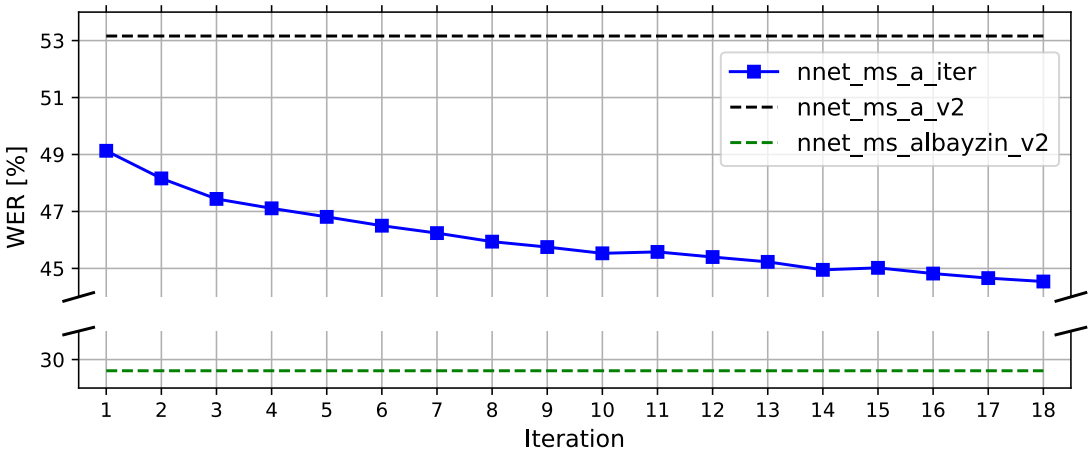


Figure 6.2: Overall WER results of iterative training evaluated on the dev2 set of RTVE2018 database. The black dashed line indicates the WER result of the baseline model trained on TID database, whereas the green dashed line indicates the WER result of model trained only on RTVE2018. The blue solid line presents the performance of trained models.

6.4 Comparison with training from scratch

In this experiment both training approaches were compared in terms of both the accuracy and the necessary resources. The older version of processing pipeline was used in the experiment, which differs only in the data cleaning step. In this version was extracted only 5 hours of data per week on average instead of the above mentioned 10 hours.

The training from scratch was done with a fixed number of epochs. The model in the first iteration was trained from scratch for 6 epochs (i.e. $1 + 5$), whereas the model in the last iteration was trained for 13 epochs (i.e. $8 + 5$). The procedure remained the same for iteratively trained models. The performance of both procedures is depicted in Figure 6.3. The accuracy of models trained from scratch varies between 37.61% and 37.3% except the 6^{th} iteration (drop to 36.76% WER). The overall performance of these models is worse than the accuracy of the baseline model. It is caused by mixing two different domain data and training first iterations with relatively high learning rates. On the other hand, the iteratively trained models outperformed not only other ones but also the baseline model. The best result was achieved during 4^{th} iteration with WER = 36.74%.

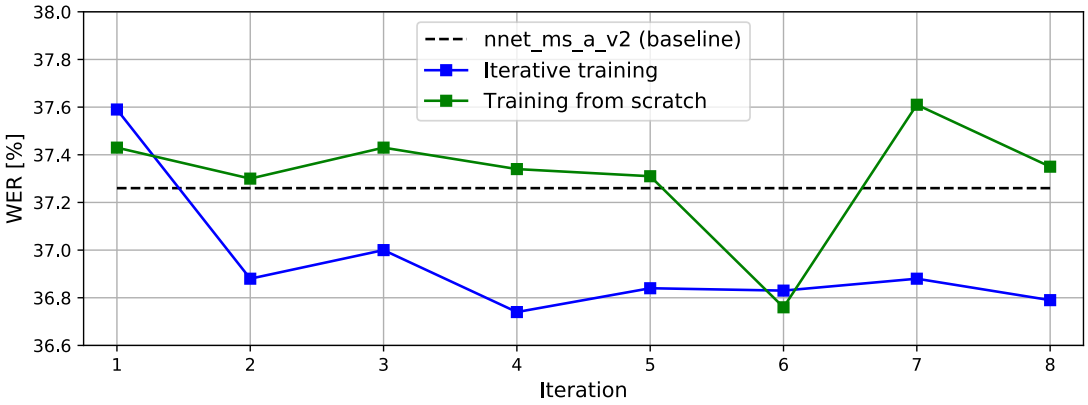


Figure 6.3: Comparison between iterative training and training from scratch.

The time consumption was also measured. It includes only the training time because both procedures differed only in this step, everything else remained the same. The training time of the first six models was measured in this experiment. Table 6.2 shows the measured values in hours. The last column presents the sum of all training times. As the experiment suggests, the iterative training is faster by 9 hours. In next iterations, the saving per training is expected to be even higher.

Week	1	2	3	4	5	6	**Total**
Scratch [h]	4.31	4.51	4.66	4.71	4.96	4.99	28.14
Iterative [h]	3.33	2.49	3.26	2.72	4.10	3.35	19.25
Saving [h]	0.98	2.02	1.40	1.99	0.86	1.64	**8.99**

Table 6.2: Difference between training from scratch and iterative training in terms of the time consumption. The first 6 models were trained for 28 hours in the case of training from scratch and for 19 hours in the case of iterative training. The total saving represents almost 9 hours.

6.5 Other experiments

The iterative training was used to test two other approaches. In the first experiment, we tried to filter words based on their frequency and retrained the model on the utterances, which contained less frequent words. In another experiment, we tried to use speaker-adaptive techniques in order to boost the performance of retrained models.

6.5.1 Word filtering

The idea of this experiment was to incorporate less frequent phonemes into next iterations of the training. The intent was to balance the amount of training data representing individual acoustic classes. We tried to do it by counting the word occurrences in the training set. Only those utterances were included in next iterations, which contained words with frequencies beneath the specified threshold. The results after 3$^{\text{rd}}$ iteration are shown in Table 6.3. The best WER was achieved with threshold equal to 10. The results were still worse than the WER of the iteratively trained model without word filtering. However, there is still a real chance to obtain better results by counting phonemes instead of words.

Threshold	10	25	50	100	∞	Baseline
% WER	**36.98**	37.15	37.06	37.03	**36.82**	37.26

Table 6.3: WER results of the word filtering experiment after 3$^{\text{rd}}$ iteration of iterative training. All systems outperformed the baseline model. The best WER was achieved with $tr = 10$, however it is still worse than the WER of the system without filtering. The WER result with $tr = \infty$ refers to the experiment where no filtering was performed.

6.5.2 Speaker adaptation

Another experiment was done with speaker adaptation. The objective was to annotate speaker turns in audio in order to transform the features with feature-space MMLR transformation. We believed that the adapted features should provide more information during the training, hence the accuracy should improve. However, the closed captions do not contain any speaker information. Therefore, we made an assumption that the speakers were changing every TV show without repetition, i.e. each segment of t minutes was said by one unique speaker. We experimented with 15 minute and 30 minute long segments according to the average duration of a TV show. However, this experiment simply did not work. After 20 iterations of iterative training the WER differed from 40.3% to 39.5% in both cases. The results were much worse than the WER of the baseline model trained on much smaller TID database. Such bad results were caused by the inaccurate speaker adaptation. The duration of one speaker turn was too long.

Chapter 7

Experiments with adaptation on television speech domain

In Section 6.3, we found that incorporating the Custom TV into training corpus had a positive effect on overall accuracy of the acoustic model. This fact motivated us to improve the acoustic model training process. The idea was to select the data from Custom TV database, which would enhance the most the baseline system trained on a telephone speech. All experiments used the clean 180 hours of data extracted from manually recorded TV database. The nnet_ms_a_v1 acoustic model was chosen as the baseline model. These experiments simulated the adaptation on television speech domain while we did not have proper in-domain data for training.

In the first experiment, the data were split into buckets according to the utterance-level confidences of the baseline model. Several acoustic models were then trained separately on each bucket. Details are described in Section 7.1. In the next experiment, we tried to incorporate the information about the target evaluation dataset. The goal was to include the utterances into the training corpora with similar characteristics to the target television speech domain. More information are provided in Section 7.2.

Finally, both methods were compared in the last experiment, where we built several models using different numbers of hours of training data. We also tried to combine the data from both techniques. Details are provided in Section 7.3.

7.1 Data selection based on utterance-level confidences

This experiment is based on findings in Section 3.2.2. All 180 hours of TV data were divided into buckets according to the confidence score. The score was estimated for each utterance using the word lattice from data cleaning step. We used the 'lattice-scale', 'lattice-to-nbest' and 'nbest-to-linear' commands from Kaldi to compute the score. The output of the last command consists of two floating point values. The first value is the graph cost, i.e. a sum of the LM cost for words in the utterance. The second value stands for the acoustic cost.

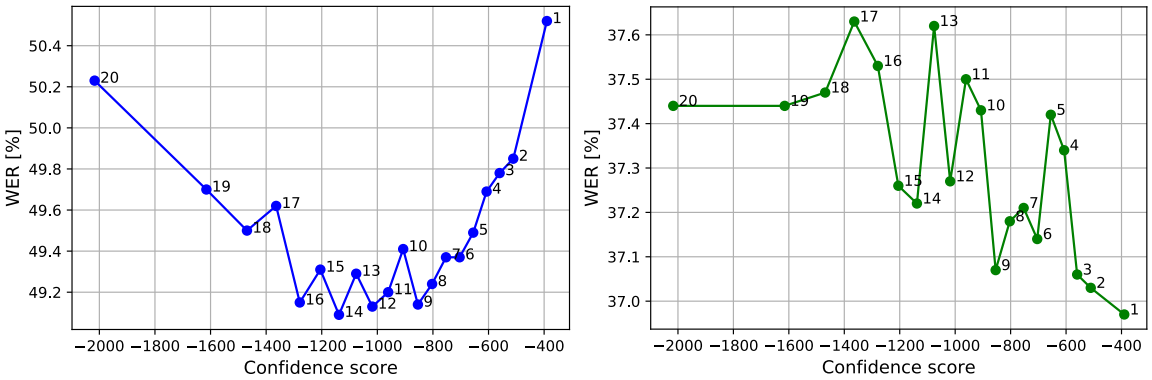
The score was represented as:

$$c_{utt} = -(w_{LM} + w_{AM}), \tag{7.1}$$

where c_{utt} is a utterance-level confidence score, w_{LM} is the cost of the language model and w_{AM} is a the cost of the acoustic model. The score improves along with the model's confidence in the utterance. The utterances were sorted according to the score and split

into 20 buckets of 9 hours each. Subsequently, each bucket was combined with training corpora from TID database. Finally, 20 acoustic models were trained for each training dataset.

7.1.1 Results



(a) WER on RTVE dev2 dataset (television speech).

(b) WER on TID test dataset (telephone speech).

Figure 7.1: Word error rates according to the confidence score. The x-coordinate represents the average confidence score per bucket. The labels mark the bucket number. The bucket with high confidence score is annotated with label „1“.

The models were trained iteratively like in the previous experiments. The training continued from the last iteration of nnet_ms_a_v1 for next 3 epochs. They were evaluated on both television and telephone speech domain. The word error rates are depicted in Figure 7.1.

The models trained on the buckets with averaged confidence score between −700 and −1300 obtained better accuracy on television speech domain (Figure 7.1a) than the other models. This discovery was quite surprising, so we decided to analyse the data. We found that these buckets contained longer utterances than the buckets with higher confidence score. The short double-word utterances with ordinary words like „el“, „a“, „para“, . . . did not improve the model as much as the utterances with wider context. On the other hand, the model trained using these utterances obtained the best result on telephone speech domain (Figure 7.1b). We found that the original model did a lot of substitutions on these words, so it makes sense that incorporating them into training dataset enhanced the performance.

7.2 Data selection based on similarity with target domain

In this experiment, the goal is to find similar data with target domain from Custom TV database. It is possible to search for them in whole database. However, since we are not sure about the transcripts, we decided to do this experiment on clean 180 hours of television speech. The goal is to find covariance matrices estimated on raw DNN posteriors, which are similar with the covariance matrix estimated on test dataset from target domain (e.g. RTVE2018).

At the beginning, the covariance matrix was estimated on the raw outputs of the baseline model for RTVE2018 dev2 dataset, i.e. the outputs from the last affine transform layer

before the normalization layer and the softmax function of the DNN model (see Figure 3.3). The rows and columns related to the silence classes were removed from the matrix. Afterwards, the covariance matrices were estimated for each bucket arranged according to the confidence score from the previous experiment. Eventually, each covariance matrix was compared with the dev2 matrix. The objective was to rank each bucket according to the similarity with dev2 dataset.

7.2.1 Comparison of matrix similarity measuring methods

We compared two different ways of measuring the similarity between two matrices. The idea was to represent the similarity by a scalar. In the first case, the similarity was measured according the equality between their eigenvalues. Mathematically, two similar matrices have similar eigenvalues. Therefore, the similarity was estimated as:

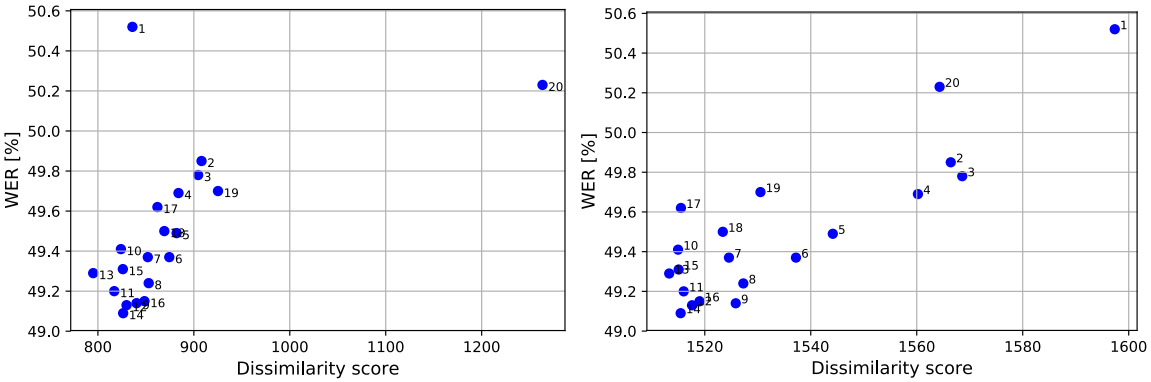
$$\mathbf{\Sigma_a} \approx \mathbf{\Sigma_b} \Leftrightarrow \sum_{i=1}^{n} \lambda_{a_i} - \sum_{i=1}^{n} \lambda_{b_i} \Rightarrow 0, \tag{7.2}$$

where λ_{a_i} and λ_{b_i} denotes the i-th eigenvalue of covariance matrix $\mathbf{\Sigma_a}$ and $\mathbf{\Sigma_b}$ respectively. Since the sum of eigenvalues is the same as the trace of original matrix, the equation can be rewritten as:

$$\mathbf{\Sigma_a} \approx \mathbf{\Sigma_b} \Leftrightarrow \mathrm{trace}(\mathbf{\Sigma_a}) - \mathrm{trace}(\mathbf{\Sigma_b}) \Rightarrow 0. \tag{7.3}$$

In the second case, the similarity was measured as a Frobenius norm of a matrix subtraction according to the Equation 3.7.

7.2.2 Results



(a) Trace

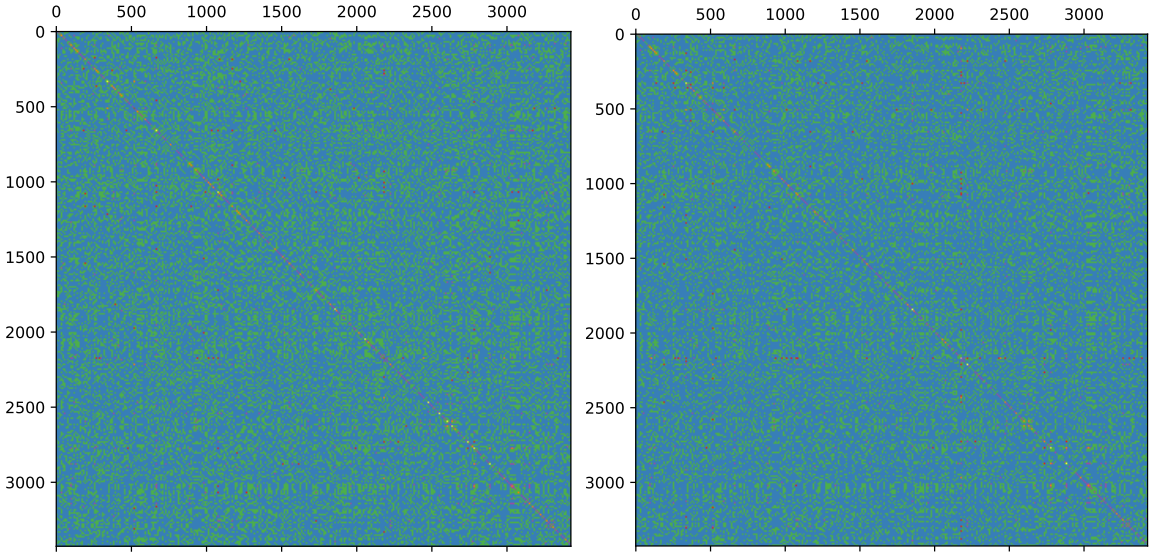
(b) Frobenius norm

Figure 7.2: The comparison between similarity based on the trace and the Frobenius norm. The labels indicate the buckets rated according to the confidence.

The results of both techniques are depicted in Figure 7.2. Since we did not change data in the buckets from previous experiment, both figures shows the same word error rates. The objective was to compare both matrix similarity metrics. Figure 7.2a refers to the similarity based on the trace, while Figure 7.2b refers to the similarity based on Frobenius norm. The bucket with the lowest confidence score (20) can be easily filtered out using both methods, while the bucket with the highest confidence score (1) can only be distinguished

using the Frobenius norm. It is desirable to filter both buckets out because both of them obtained very high word error rates. Consequently, we decided to use the Frobenius norm for measuring similarity.

Figure 7.3 depicts the difference between covariance matrices estimated on the best (Figure 7.3a) and the worst bucket (Figure 7.3b). The matrix estimated on 14^{th} bucket is denser than the other one. It means that the 14^{th} bucket covers greater variability in speech signal.



(a) Covariance matrix estimated on 14^{th} bucket. (b) Covariance matrix estimated on 1^{st} bucket.

Figure 7.3: The comparison between covariance matrices. Both matrices are plotted using the qualitative color mapping to highlight the differences.

7.2.3 Generic recipe

Finally, we present the generic recipe for finding the best utterances according to similarity with the target domain:

1. **Silence removal** Silence can be removed from closed-captioned data with VAD. This step is optional because the silence classes are not used in the covariance matrix estimation.

2. **Splitting data into buckets** The closed-captioned data are split into several buckets. The granularity of the buckets is constrained by the quality of the covariance matrix estimation. If the buckets contain more data, the covariance matrix will be estimated better. But there is a chance that better utterances will be mixed with some worse ones and thus the metric will not work properly.

3. **Covariance matrix estimation** The covariance matrix is estimated for each bucket as well as on the whole target domain dataset. The silence classes are removed from the DNN's raw output vector before the estimation.

4. **Computation of the similarity metric** The target domain covariance matrix is compared with each matrix estimated on the buckets using Frobenius norm similarity, i.e. according to Equation 3.7.

5. **Building the training corpus** The utterances from buckets with low dissimilarity are included in the training corpus.

6. **Acoustic model training** Eventually, the AM is trained on the created corpus.

7.3 Comparison of both selection metrics

In the final experiment, both techniques for data selection were compared. The closed-captioned data were split into 10, 25, 50, 75 and 100 hours long datasets according to the confidence score and the similarity with dev2 dataset from RTVE database. Random data selection was also used to provide a baseline for comparison.

In the case of the confidence splitting, the data from the best buckets were taken. That means that the first 10h dataset contained the data from 12^{th} bucket, the 25 hours dataset contained the data from 11^{th}, 12^{th} and 13^{th} bucket and the 100 hours dataset contained the date from buckets $[7 \dots 17]$, other datasets were created using the same procedure.

In case of datasets split based on the similarity, the covariance matrices were estimated on 1 minute long buckets and then compared with the matrix estimated on the dev2 dataset. The datasets contained the utterances from the most similar buckets.

In addition, we also experimented with combination of both approaches. For instance, the 10 hours long dataset contained the randomly picked utterances from both the 10 hours dataset based on the confidence and the 10 hours dataset based on the similarity. The overlapping between the datasets is displayed in Table 7.1. The *Conf* and *Frob* 10h datasets contain only 5.72% of common utterances, whereas the 100h dataset contains 55.32% shared utterances. The reason is that the 100 hours of data were selected from 180 hours long close-captioned database.

Dataset [h]	10	25	50	75	100
Conf vs. Sim	5.72	14.74	27.99	41.75	55.32
Conf vs. ConfSim	52.39	54.23	62.41	68.36	71.84
Frob vs. ConfSim	51.45	54.12	54.77	64.70	71.74

Table 7.1: Overlapping between the datasets. The numbers show the percentage of common utterances in both datasets. The abbrv. *Conf* stands for datasets arranged based on the confidence, the *Sim* stands for datasets arranged according to the Frobenius norm similarity and the *ConfSim* stands for datasets with the combination of both approaches.

7.3.1 Overall results

All datasets were combined with the original training partition from TID database. The acoustic models were trained on each training corpus. Moreover, we compare both training from scratch and the iterative training. The models were trained for 8 epochs in case of training from scratch. The transfer-learning was done by continuing from the last iteration of nnet_ms_a_v1 DNN model (5 epochs). The retraining was stopped after 3 additional epochs. The baselines were trained using all 180 hours of clean close-captioned data. All models were evaluated on both television speech domain and telephone speech domain.

As the results from Figure 7.4 suggest, the models trained from scratch outperformed the models trained iteratively. It turns out that the most efficient way is to select utterances

based on the confidences in the case of iterative training. However, the other models also performed well. The biggest difference between both metrics is illustrated on 70 hours long adaptation dataset, where the similarity metric outperformed other techniques with the model trained from scratch. The same trend continued until the last dataset except the one with 75h, where the model performed a slightly better using the confidence metric. Surprisingly, the combination of both metrics performed consistently worse in all scenarios. Detailed results are provided in Table B.5.

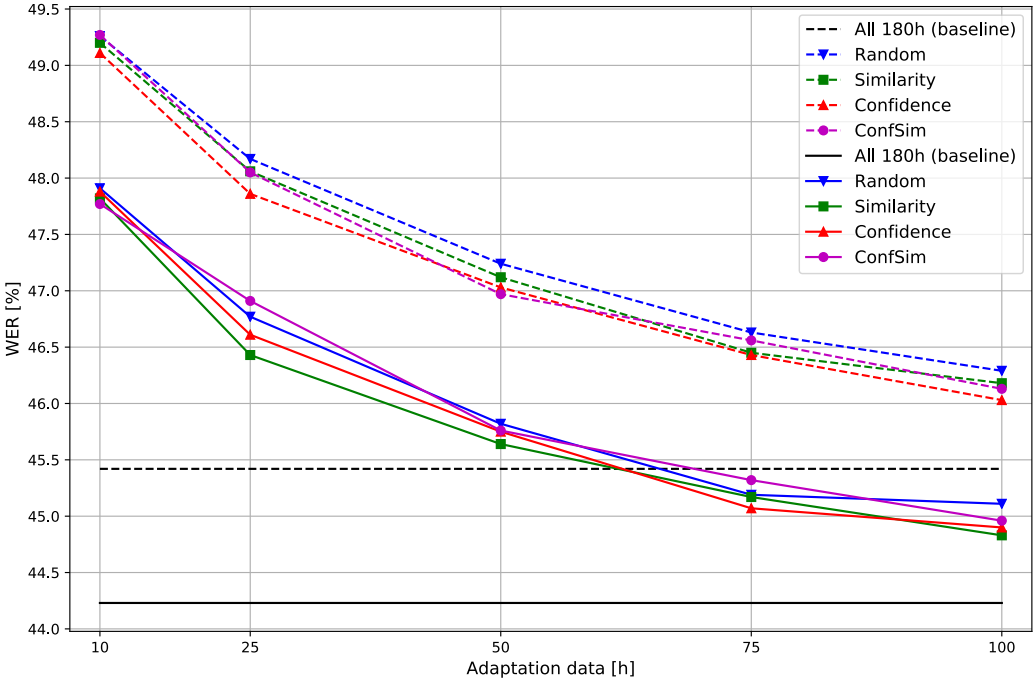


Figure 7.4: Overall results of both selection metrics evaluated on television speech domain (dev2 from RTVE2018). The dashed lines presents the WER results using the iterative training, while the solid lines presents the results using training from scratch. The x-axis shows the number of hours of each dataset not counting the hours from TID database.

Chapter 8

Conclusion

In this thesis, we have dealt with the application of subtitled data in an automatic speech recognition task. More precisely, we focused on audiovisual content with close-captions from a television broadcast. The main idea was to incorporate these data into a training corpora in order to enhance the original automatic speech recognizer. This was accomplished by improving the acoustic models. We analysed the closed-captioned data for commonly occurring problems. These findings were then used in the following research.

Great effort was dedicated to data cleaning because of imperfections in the transcripts. We managed to successfully extract 180 hours of high quality labels. The clean data were then used in two experiments. The goal of the first one was to enhance acoustic models using transfer learning techniques. The initial model was first trained on clean telephone speech data from the TID database. The model was then iteratively retrained using the clean close-captioned data. Based on the results, all models managed to achieve lower word error rate in comparison with the model trained only on telephone speech database. Moreover, the accuracy improved on the television speech evaluation dataset as well.

In the next experiment, we investigated these findings more closely. Our goal was to detect the close-captioned data samples which provide the greatest model improvement. We proposed a novel method for selecting data based on similarity with the target domain. The method was compared with selection based on sentence-level confidence. The results suggest that both selection methods show similar performance across the experiments. On one hand, the accuracy was higher in case of models iteratively trained on utterances selected based on confidence. On the other hand, the models achieved lower word error rates using similarity metric in case of training from scratch. It is noteworthy that both described methods achieved better results than the models trained on randomly selected data. Furthermore, incorporating the closed-captioned data had a positive effect on accuracy in all experiments.

In the thesis, we also managed to create very accurate ASR system trained on the RTVE2018 database with WER = 29.41%. To our best knowledge, this system can easily compete with the models presented in the Albayzin Challenge of the IberSpeech 2018 conference (provided that we improved the language model as well, which was not the intent of this work).

8.1 Future research

We did not cover some other aspects, which could affect the overall performance. The obtained results could be improved by tuning the parameters of the deep neural network.

We did not experiment with larger TDNN models consisting of 5 or more hidden layers. Using a wider context of input features could also increase the overall accuracy as well as the number of neurons in the hidden layers. However, the greatest improvement could be achieved by tuning the language model in all scenarios. We also did not research the impact of softmax activations on covariance matrix estimation as well as the impact of log-posteriors or other DNN outputs in the experiment with similarity selection. In the near future, I would like to continue working on this research.

Bibliography

[1] Beel, J.; Gipp, B.; Langer, S.; et al.: Research-paper recommender systems: A literature survey. *International Journal on Digital Libraries*. 07 2015: pp. 1–34. doi:10.1007/s00799-015-0156-0.

[2] Beneš, K.: Language Modeling in Automatic Speech Recognition. 4 2015. Retrieved from: https://www.fit.vutbr.cz/study/courses/ZRE/public/pred/10_wfst_lvcsr/lm_lvcsr_present.pdf

[3] Bishop, C. M.: *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Berlin, Heidelberg: Springer-Verlag. 2006. ISBN 0387310738.

[4] Davis, S.; Mermelstein, P.: Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. *IEEE Transactions on Acoustics, Speech, and Signal Processing*. vol. 28, no. 4. aug 1980: pp. 357–366. ISSN 0096-3518. doi:10.1109/TASSP.1980.1163420.
Retrieved from: http://ieeexplore.ieee.org/document/1163420/

[5] Egorova, E.; Serrano, L. J.: Semi-Supervised Training of Language Model on Spanish Conversational Telephone Speech Data. In *Procedia Computer Science*, vol. 2016. Elsevier Science. 2016. ISSN 1877-0509. pp. 114–120. doi:10.1016/j.procs.2016.04.038.
Retrieved from: http://www.fit.vutbr.cz/research/view_pub.php?id=11223

[6] Gales, M.; Young, S.: The Application of Hidden Markov Models in Speech Recognition. *Foundations and Trends in Signal Processing*. vol. 1. 01 2007: pp. 195–304. doi:10.1561/2000000004.

[7] Goodman, J. T.: A bit of progress in language modeling. *Computer Speech & Language*. vol. 15, no. 4. 2001: pp. 403 – 434. ISSN 0885-2308. doi:https://doi.org/10.1006/csla.2001.0174.
Retrieved from:
http://www.sciencedirect.com/science/article/pii/S0885230801901743

[8] Graff; David; et al.: Fisher Spanish Speech. *LDC2010S01. DVD. Philadelphia: Linguistic Data Consortium*. 2010.

[9] Grézl, F.; Karafiát, M.; Kontár, S.; et al.: Probabilistic and bottle-neck features for LVCSR of meetings. In *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2007)*. IEEE Signal Processing Society. 2007. ISBN 1-4244-0728-1. pp. 757–760.
Retrieved from: http://www.fit.vutbr.cz/research/view_pub.php?id=8249

[10] Hermansky, H.: Perceptual linear predictive (PLP) analysis of speech. *The Journal of the Acoustical Society of America*. vol. 87, no. 4. apr 1990: pp. 1738–1752. ISSN 0001-4966. doi:10.1121/1.399423.
Retrieved from: http://asa.scitation.org/doi/10.1121/1.399423

[11] Karel Veselý and Arnab Ghoshal and Lukáš Burget and Daniel Povey: Sequence-discriminative Training of Deep Neural Networks. In *Proceedings of Interspeech 2013*. 8. International Speech Communication Association. 2013. ISBN 978-1-62993-443-3. ISSN 2308-457X. pp. 2345–2349.
Retrieved from: http://www.fit.vutbr.cz/research/view_pub.php?id=10422

[12] Krizhevsky, A.; Sutskever, I.; Hinton, G. E.: ImageNet Classification with Deep Convolutional Neural Networks. In *Advances in Neural Information Processing Systems 25*, edited by F. Pereira; C. J. C. Burges; L. Bottou; K. Q. Weinberger. Curran Associates, Inc.. 2012. pp. 1097–1105.
Retrieved from: http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf

[13] Lamel, L.; Gauvain, J.-L.; Adda, G.: Lightly supervised and unsupervised acoustic model training. *Computer Speech & Language*. vol. 16, no. 1. jan 2002: pp. 115–129. ISSN 08852308. doi:10.1006/csla.2001.0186.
Retrieved from:
https://linkinghub.elsevier.com/retrieve/pii/S088523080190186X

[14] Lecouteux, B.; Linarès, G.; Nocera, P.; et al.: Imperfect transcript driven speech recognition. In *INTERSPEECH*. Pittsburgh, United States. September 2006. pp. 1626–1629.
Retrieved from: https://hal.archives-ouvertes.fr/hal-01318085

[15] Lecouteux, B.; Linarès, G.; Oger, S.: Integrating imperfect transcripts into speech recognition systems for building high-quality corpora. *Computer Speech & Language*. vol. 26, no. 2. apr 2012: pp. 67–89. ISSN 08852308. doi:10.1016/j.csl.2011.06.001.
Retrieved from:
https://linkinghub.elsevier.com/retrieve/pii/S0885230811000337

[16] Liao, H.; McDermott, E.; Senior, A.: Large scale deep neural network acoustic modeling with semi-supervised training data for YouTube video transcription. In *2013 IEEE Workshop on Automatic Speech Recognition and Understanding*. IEEE. dec 2013. ISBN 978-1-4799-2756-2. pp. 368–373. doi:10.1109/ASRU.2013.6707758.
Retrieved from: http://ieeexplore.ieee.org/document/6707758/

[17] Lleida, E.; Ortega, A.; Miguel, A.; et al.: RTVE2018 Database Description. 2018.
Retrieved from: http://catedrartve.unizar.es/reto2018/RTVE2018DB.pdf

[18] Luque Serrano, J.: *Speaker diarization and tracking in multiple-sensor environments*. PhD. Thesis. Universitat Politècnica de Catalunya. dec 2012.
Retrieved from: https://upcommons.upc.edu/handle/2117/94956

[19] Manohar, V.; Povey, D.; Khudanpur, S.: JHU Kaldi system for Arabic MGB-3 ASR challenge using diarization, audio-transcript alignment and transfer learning. In *2017 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, vol. 2018-. IEEE. 2017. ISBN 9781509047888. pp. 346–352.

[20] Martin Karafiát: *Study of Linear Transformations Applied to Training of Cross-Domain Adapted Large Vocabulary Continuous Speech Recognition Systems*. PhD. Thesis. Brno University of Technology, Faculty of Information Technology. 2009.
Retrieved from: http://www.fit.vutbr.cz/study/DP/PD.php?id=110

[21] Mikolov, T.; Karafiát, M.; Burget, L.; et al.: Recurrent Neural Network Based Language Model. In *INTERSPEECH 2010*. Makuhari, Chiba, Japan. 2010. pp. 1045–1048.
Retrieved from:
https://www.isca-speech.org/archive/interspeech_2010/i10_1045.html

[22] Mikolov, T.; Sutskever, I.; Chen, K.; et al.: Distributed Representations of Words and Phrases and their Compositionality. In *Advances in Neural Information Processing Systems 26*, edited by C. J. C. Burges; L. Bottou; M. Welling; Z. Ghahramani; K. Q. Weinberger. Curran Associates, Inc.. 2013. pp. 3111–3119.
Retrieved from: http://papers.nips.cc/paper/5021-distributed-representations-of-words-and-phrases-and-their-compositionality.pdf

[23] Mohamed, A.; Dahl, G. E.; Hinton, G. E.: Deep Belief Networks for phone recognition. In *NIPS Workshop on Deep Learning for Speech Recognition and Related Applications*. 2009.

[24] Mohri, M.; Pereira, F.; Riley, M.: *Speech Recognition with Weighted Finite-State Transducers*. Berlin, Heidelberg: Springer Berlin Heidelberg. 2008. ISBN 978-3-540-49127-9. pp. 559–584. doi:10.1007/978-3-540-49127-9_28.
Retrieved from: https://doi.org/10.1007/978-3-540-49127-9_28

[25] Moreno, A.: SALA: SpeechDat Across Latin America.

[26] Nation, I.: How Large a Vocabulary is Needed For Reading and Listening? *The Canadian Modern Language Review*. vol. 63, no. 1. 2006: pp. 59–82.
Retrieved from: https://doi.org/10.3138/cmlr.63.1.59

[27] Ng, T.; Zhang, B.; Nguyen, L.; et al.: Developing a Speech Activity Detection System for the DARPA RATS Program. In *Proceedings of Interspeech 2012*, vol. 2012. International Speech Communication Association. 2012. ISBN 978-1-62276-759-5. ISSN 1990-9772. pp. 1–4.
Retrieved from: http://www.fit.vutbr.cz/research/view_pub.php?id=10099

[28] Pascanu, R.; Bengio, Y.: Revisiting Natural Gradient for Deep Networks. *arXiv e-prints*. Jan 2013: arXiv:1301.3584. 1301.3584.

[29] Peddinti, V.; Povey, D.; Khudanpur, S.: A time delay neural network architecture for efficient modeling of long temporal contexts. In *INTERSPEECH*. 2015. pp. 3214–3218.
Retrieved from:
https://www.danielpovey.com/files/2015_interspeech_multisplice.pdf

[30] Photina Jaeyun Jang; Hauptmann, A. G.: Improving acoustic models with captioned multimedia speech. In *Proceedings IEEE International Conference on Multimedia*

Computing and Systems, vol. 2. 1999. ISBN 0-7695-0253-9. pp. 767–771.
Retrieved from: http://ieeexplore.ieee.org/document/778582/

[31] Povey, D.; Ghoshal, A.; Boulianne, G.; et al.: The Kaldi Speech Recognition Toolkit. In *IEEE 2011 Workshop on Automatic Speech Recognition and Understanding*. IEEE Signal Processing Society. December 2011. iEEE Catalog No.: CFP11SRW-USB.

[32] Smith, T.; Waterman, M.: Identification of common molecular subsequences. *Journal of Molecular Biology*. vol. 147, no. 1. 1981: pp. 195 – 197. ISSN 0022-2836. doi:https://doi.org/10.1016/0022-2836(81)90087-5.
Retrieved from:
http://www.sciencedirect.com/science/article/pii/0022283681900875

[33] Veselý, K.: The project Kaldi – Open source speech recognition. 4 2018.
Retrieved from: https://www.fit.vutbr.cz/study/courses/ZRE/public/pred/14_Kaldi/2018-kaldislidesforzre.pdf

[34] Veselý, K.: *Semi-supervised training of Deep Neural Networks for speech recognition*. PhD. Thesis. Vysoké učení technické v Brně, Fakulta informačních technologií. 2018.
Retrieved from: http://www.fit.vutbr.cz/study/DP/PD.php?id=568

[35] Veselý, K.; Burget, L.; Černocký, J.: Semi-supervised DNN training with word selection for ASR. In *Proceedings of Interspeech 2017*, vol. 2017. International Speech Communication Association. 2017. ISSN 1990-9772. pp. 3687–3691. doi:10.21437/Interspeech.2017-1385.
Retrieved from: http://www.fit.vutbr.cz/research/view_pub.php?id=11584

[36] Waibel, A.; Hanazawa, T.; Hinton, G.; et al.: Phoneme recognition using time-delay neural networks. *IEEE Transactions on Acoustics, Speech, and Signal Processing*. vol. 37, no. 3. March 1989: pp. 328–339. ISSN 0096-3518. doi:10.1109/29.21701.

[37] Wessel, F.; Ney, H.: Unsupervised training of acoustic models for large vocabulary continuous speech recognition. *IEEE Transactions on Speech and Audio Processing*. vol. 13, no. 1. Jan 2005: pp. 23–31. ISSN 1063-6676. doi:10.1109/TSA.2004.838537.

[38] Zhang, P.; Liu, Y.; Hain, T.: Semi-supervised DNN training in meeting recognition. In *2014 IEEE Spoken Language Technology Workshop (SLT)*. IEEE. dec 2014. ISBN 978-1-4799-7129-9. pp. 141–146. doi:10.1109/SLT.2014.7078564.
Retrieved from: http://ieeexplore.ieee.org/document/7078564/

[39] Zhang, X.; Trmal, J.; Povey, D.; et al.: Improving deep neural network acoustic models using generalized maxout networks. In *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. May 2014. ISSN 1520-6149. pp. 215–219. doi:10.1109/ICASSP.2014.6853589.

[40] Zhu, X.; Goldberg, A. B.: Introduction to Semi-Supervised Learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*. vol. 3, no. 1. jan 2009: pp. 1–130. ISSN 1939-4608. doi:10.2200/S00196ED1V01Y200906AIM006.
Retrieved from:
http://www.morganclaypool.com/doi/abs/10.2200/S00196ED1V01Y200906AIM006

Appendix A

Manual

The attached CD contains the source codes for each experiment. The codes are divided into following directories:

- **exp-albayzin** – This directory provides all necessary scripts for data cleaning and model training using RTVE2018 database. A reader may find here also the script for dev2 dataset segmentation. The script uses VAD developed by Karel Veselý [27]. The VAD itself is not provided. The directory also contains the script for downloading the data. However, the user have to know the credentials because the database is not publicly available.
- **exp-data-selection** – This is the main directory with the source code for the adaptation experiments. The directory contains also README file, which should help a reader to understand how the experiments were done.
- **exp-filtering** – This directory contains scripts to run the experiments using word selection. The older version of the experiment is in **exp-filtering-all**. The difference is that the new version uses datasets of equal size while the other one do not.
- **exp-iterative-training** – This directory contains the main script (i.e. `run_tv_recordings.sh`) for iterative training on top of Custom TV database.
- **internal** – This directory includes miscellaneous scripts for data preprocessing.

A.1 Guide for data selection based on similarity with target domain

This section provides detailed instructions how to obtain the results presented in Chapter 7. The process can be described as follows:

1. Clean data using `segment_long_utterances.sh` and `clean_and_segment_data.sh` (see script `exp-iterative-training/prepare_data.sh`).
2. Sort data by sentence-level confidences and train the models:
   (a) Tag each utterance with its confidence with `get_confidence.sh` script.
   (b) Split utterances into 20 buckets based on the confidence with `split_utt_conf.sh`.

(c) Combine each bucket with TID database.

(d) Train the model for each bucket with `train_iter_general.sh` and `train_scratch_general.sh`

3. Find similar data to target domain (dev2) with Frobenius norm similarity metric:

   (a) Use voice activity detector to remove the silence (e.g. by `vad_remove_silence.sh`).

   (b) Split data into 1 minute long chunks `divide_utt.py`.

   (c) Estimate covariance matrix on target domain data with `compute_covariance_mat.sh`.

   (d) Compute the similarity with the target domain for each data chunk with `compute_frobenius_metric.sh`.

   (e) Merge data into 20 buckets according to the similarity using `merge_frobenius_buckets.sh`.

   (f) Combine each bucket with TID database.

   (g) Train models with `train_iter_general.sh` and `train_scratch_general.sh` or train one model on top of N-best buckets with `train_klc_best.sh`.

Appendix B

Detailed experimental results

In this appendix are provided the detailed results for all experiments, which do not fit into the main text of this thesis.

Week	1	2	3	4	5	6	7	8	9
WER [%]	37.08	37.11	36.82	36.68	36.56	36.17	36.38	36.29	36.05
Week	10	11	12	13	14	15	16	17	18
WER [%]	36.34	36.41	36.43	36.22	36.34	36.49	36.27	36.38	36.54

Table B.1: The word error rates of the models presented in Section 6.2

Week	1	2	3	4	5	6	7	8	9
WER [%]	49.13	48.16	47.44	47.11	46.81	46.50	46.24	45.94	45.75
Week	10	11	12	13	14	15	16	17	18
WER [%]	45.53	45.58	45.40	45.23	44.95	45.02	44.82	44.66	44.54

Table B.2: The word error rates of the models presented in Section 6.3

Week	1	2	3	4	5	6	7	8
Iterative training	37.59	36.88	37.00	36.74	36.84	36.83	36.88	36.79
Training from scratch	37.43	37.30	37.43	37.34	37.31	36.76	37.61	37.35

Table B.3: The word error rates [%] of the models presented in Section 6.4

Bucket	1	2	3	4	5	6	7	8	9	10
dev2	50.52	49.85	49.78	49.69	49.49	49.37	49.37	49.24	49.14	49.41
test	36.97	37.03	37.06	37.34	37.42	37.14	37.21	37.18	37.07	37.43
Bucket	11	12	13	14	15	16	17	18	19	20
dev2	49.20	49.13	49.29	49.09	49.31	49.15	49.62	49.50	49.70	50.23
test	37.50	37.27	37.62	37.22	37.26	37.53	37.63	37.47	37.44	37.44

Table B.4: The word error rates [%] of the models presented in Section 7.1

Iterative training					
Data [h]	10	25	50	75	100
Random	49.26	48.17	47.24	46.63	46.29
Similarity	49.20	48.06	47.12	46.45	46.18
Confidence	49.11	47.86	47.03	46.43	46.03
ConfSim	49.27	48.05	46.97	46.56	46.13
Training from scratch					
Random	47.91	46.77	45.82	45.19	45.11
Similarity	47.82	46.43	45.64	45.17	44.83
Confidence	47.88	46.61	45.75	45.07	44.90
ConfSim	47.77	46.91	45.76	45.32	44.96

Table B.5: The word error rates [%] of the models presented in Section 7.3