Karlsruhe Institute of Technology

Unsupervised Acoustic Model Training for Simultaneous Lecture Translation in Incremental and Batch Mode

Diploma Thesis of

Michael Heck

At the Department of Informatics

Karlsruhe Institute of Technology (KIT), Germany
Nara Institute of Science and Technology (NAIST), Japan

Supervisors:
Dr. Sebastian Stüker
Dr. Sakriani Sakti
Prof. Dr. Alex Waibel
Prof. Satoshi Nakamura

Duration: 01. July 2012 – 31. December 2012

Hiermit erkläre ich, dass ich diese Diplomarbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Michael Heck

Abstract

In this work the theoretical concepts of unsupervised acoustic model training and the application and evaluation of unsupervised training schemes are described. Experiments aiming at speaker adaptation via unsupervised training are conducted on the KIT lecture translator system. Evaluation takes place with respect to training efficiency and overall system performance in dependency of the available training data. Domain adaptation experiments are conducted on a system trained for European parliament plenary session speeches with help of unsupervised iterative batch training. Major focus is on transcription pre-processing methods and confidence measure based weighting and thresholding on word level for data selection. The objective is to lay the foundation for an unsupervised adaptation framework based on acoustic model training for use in KIT's simultaneous speech-to-speech lecture translation system.

Experimental results show, that it is of advantage to let the Viterbi algorithm during training decide which pronunciations to use and where to insert which noise words, instead of fixating these informations in the transcriptions. With weighting and thresholding it is possible to improve unsupervised training in all test cases. Tests of iterative incremental approaches show that potential performance gains strongly correlate to the performance of the baseline systems. Considerable performance gains are observable after only one iteration of unsupervised batch training with applied transcription pre-processing, weighting and thresholding.

Acknowledgements

I would like to thank Prof. Dr. Alex Waibel and Prof. Satoshi Nakamura for giving me the opportunity to conduct the research for this thesis within the frame of the interACT program at the Nara Institute of Science and Technology in Japan. Heartfelt thanks go to my supervisors Sebastian Stüker and Sakriani Sakti for their constant support and guidance during this project.

Contents

1. Introduction

The scientific field of automatic speech recognition has it's origins in a time where personal computers were not even in the minds of the researchers working at the frontiers of information technology. Since more than fifty years, automatic speech recognition systems play a distinctive role in the field of human-machine-interaction. Moreover, automatic language processing technologies have seen large improvements in terms of performance, use and acceptance in recent years. Speech recognition and speech-to-speech translation systems manifest themselves in a large variety of applications used in daily life scenarios, be they of private nature or part of the business environment. In a globalizing world and growing multi-cultural societies one of the most important requirements to spoken language technology is the ability to cope with language in a robust and natural fashion. Inherent to a human being, this poses a complex task for machines, demanding the development of technologies that enable artificial systems to process, interpret and synthesize speech signals in way which makes this high-level human-machine interaction acceptable by the vast majority of the audience. Today's smart systems are capable of multi-lingual and simultaneous speech processing and translation, but usually high-performance systems are tailored to a specific field of application. Usually, high-quality training data resembling the target domain is required to build systems for accuracy-critical scenarios such as the automatic transcription of parliament speeches or scientific lectures. The latter domain is addressed by the simultaneous lecture translation system developed at KIT and started its operation in a real life scenario recently. In the summer of 2012 the KIT lecture translator went on duty recording and simultaneously translating lectures of selected courses [CFH+12].

In the past decades a vivid interest grew in improving the acoustic model training of such systems with help of well-established speech processing and machine learning technologies. The bottleneck of those training techniques generally is the lack of high-quality transcriptions of potential training data. Whereas the amount of freely available audio recordings at least for the major languages of the world grew beyond countability especially due to the rapid growth and extensive use of multimedia web platforms for informational and scientific purposes as well as commercial and pop-cultural usage, most data lacks the respective transcriptions needed for a supervised training. Additional textual information may give some insight into the content of the respective recordings in general, but do not suffice for the common methods of model training. The scientific field of machine learning knows techniques for training models without transcriptions at hand, known as unsupervised learning. The associated field of research within the scope of training the acoustic

models of speech processing systems is referred to as *unsupervised acoustic model training*. Moreover, techniques for *lightly supervised training* are capable of utilizing associated textual data such as annotations, closed captions or textual summaries for establishing certain degrees of supervision during model training. The main idea of those techniques is to exploit the vast amount of unannotated and partly annotated audio media that is publicly available and potentially utilizable for training and improving speech processing systems with the help of automatically generated transcriptions for this data, and making use of these erroneous data sets instead of relying on fully supervised material only. The advantages are clearly visible: With the ability to benefit from a merely unlimited source of audio recordings in form of the multimedia contents found in the world wide web, building new speech processing systems and improving existing applications may be rendered a constant process, not bound to the need of detailed transcriptions, which are expensive in terms of production costs and time. The challenge in developing an efficient way of unsupervised training is the exploration of methods for filtering and processing the generically obtained and thus erroneous transcriptions and maximizing the gains of utilizing possibly available, yet inaccurate and coarse textual information.

1.1 Automatic Speech Recognition

The task of *automatic speech recognition (ASR)* is the machine made transformation of a spoken utterance, embodied by sound waves transmitted through air, with previously unknown content into it's textual representation. The acoustic speech signal needs to be transformed into a parametric representation for further processing. The digitalization results in a representation of the time domain based continuous wave form as time discrete, quantized digital signal. Further pre-processing results in a stream of multi-dimensional feature vectors over time. Today's state-of-the-art systems almost exclusively follow the principle of statistical pattern recognition, modelling and decoding speech by means of statistics [ST95, You96]. The statistical approach describes automatic speech recognition as decoding process which aims at transferring an encoded message stream, i.e., a sequence W of words $w_1, \cdots, w_n$ into a respective stream X of real valued feature vectors $x_1, \cdots, x_m$ following the maximum-likelihood criterion [ST95]. It is the task of the decoder to find the most likely sequence of words W^*, given the representation X of the original sequence of words W. With help of mathematical formulation it is possible to decompose this task into several sub-problems. Identifying a sequence of words W^* upon a pool $\mathcal{W}$ of all possible sequences can be formulated and transformed by the *Bayes formula* as follows:

$$W^* = \underset{W \in \mathcal{W}}{\operatorname{argmax}} P(W|X) = \underset{W \in \mathcal{W}}{\operatorname{argmax}} \frac{P(X|W) \cdot P(W)}{P(X)} = \underset{W \in \mathcal{W}}{\operatorname{argmax}} P(X|W) \tag{1.1}$$

which models the probability of X being observed when W is the voiced sequence of words. X is the acoustic observation according to the processed signal, $P(W|X)$ is the probability of W being observed, given X. $P(X)$ is the a priori probability of observing X. As the decoder varies W trying to maximize it, $P(X)$ is constant for the classification decision and thus negligible [Jel76]. The probability $P(W)$ and the probability density function $P(X|W)$ are known as language model and acoustic model, respectively. The former models the probability of observing W, independently of the sequence of observations X, the latter is the probability that a stream of feature vectors X is observed, given the input sequence W of voiced words. This formulation is commonly known as the *fundamental equation of speech recognition*. Provided that the acoustic model and language model along with the respective dictionary are known, the Bayes formula delivers the optimal decoding principle according to [Nie90]. However, it is crucial to find the probability distributions occurring in Equation 1.1 beforehand, rendering the computation of approximations,

which are preferably as accurately as possible, a major task in the development process of automatic speech recognition systems.

1.2 Acoustic Modeling

One of the sub-tasks mentioned above is the acoustic modelling, described as $P(X|W)$ in Equation 1.1. In fact, we do not have the exact knowledge of the underlying parameters. Instead, we model them by estimating emission probabilities $P(X|\lambda)$ of Markov models, likely to give a good approximation of the real articulatory event. Today, almost exclusively hidden Markov models (HMMs) are the concept of choice for estimating the defined elementary sound units, utilizing annotated training samples of voiced utterances. HMMs are especially useful for modelling dynamic processes that are structured in discrete states and respective probabilities of state switches. In principal it is sufficient to define a feature space of observable events and establishing an assignment of HMM states to specific units of sound in order to define an HMM for modelling speech [Rog05].

The basic principle of statistical speech recognition using HMMs is to approximate $P(X|W)$ by the concatenation of word models $\lambda(w_1), \cdots, \lambda(w_n)$ for $W = w_1, \cdots, w_n$ following the maximum-likelihood criterion. The training algorithms of choice, Viterbi and Baum-Welch, demand representative, exact utterance samples of all elements w_l in the search dictionary W_dict for iteratively optimizing the word models $\lambda(w_l)$, which themselves are compounds of phonemes. The phoneme based modelling approach, compared to a higher-level modelling scheme, has several crucial advantages:

Precision: The sound unit is specific to it's articulation, i.e each element of the sound inventory is clearly distinguishable of every other, given appropriate approximations.

Robustness: Crucial to the above criterion is the quality as well as quantity of applied training samples. Further, the application of appropriate approximation algorithms and interpolation of models aiming at enhanced robustness is a factor.

Modularity: Representing words by means of smaller sub-units implicits a finite inventory of models. Ideally, all acts of speech are derivable by proper concatenation of selected units [ST95]. This representation implicits scalability.

Transferability: It is possible to synthesize new high-level models by falling back to elemental units such as phonemes.

In order to establish a sound inventory fulfilling the above criteria, some conceptual design is demanded regarding it's definition phase.

The sum of all structural and parametric knowledge regarding the sound units we want to model is known as the *acoustic model (AM)* of a speech recognition system. Word models are usually a compound of smaller sound units, e.g., phonemes, which themselves are further decomposable into sub-phonemes. The ideal elementary sub-unit should be defined in a way that it is estimable acoustically precise and statistically robust [ST95]. In order to approximate the variabilities of voiced sound units such as phonemes in form of co-articulatory effects, acoustic model training makes use of context-dependent model training of allophonic sound units, commonly known as polyphones.

Sample recordings ordinarily contain not only the relevant acoustic representation of a word, but also silence or various noises and co-articulatory distortions especially at word boundaries. To compensate for those effects, the HMM corresponding to the word of interest will be altered, instead of the sample data [ST95]. By following this approach of acoustic modelling, besides the textual representation of each recorded training sample no further annotation of the data is necessary [ST95].

1.2.1 Unsupervised Acoustic Model Training

In the previous section it was stated that acoustic model training is in need of textual representations of audio training samples. The field of machine learning is aware of unsupervised training techniques. Training schemes belonging to this class of algorithms can utilize material without the knowledge of a ground truth. For acoustic modelling that implies the possibility of performing training without a priori available transcriptions. In other words, by making use of appropriate training techniques it is possible to incorporate huge amounts of audio data into acoustic model training, without manual transcriptions at hand. The core idea of all unsupervised acoustic model training schemes is to run an existing, presumably mediocre automatic text-to-speech (TTS) system on audio data to automatically generate transcripts. Countering the significant amount of errors kept in these transcriptions, various efforts are indispensable. In general, two approaches are distinguishable, namely adaptation to the domain and acoustics of the training data, and utilising confidence annotations for training, the latter being computed during automatic transcription generation [Rog05]. Confidence scores depict a certain probability that the recognizer is correct or wrong with producing a particular hypothesis or parts of. Automatic scores can be applied as weighting factors c_t, multiplied with $\gamma_t(i)$ for all time steps t before performing the Baum-Welch training steps. A second way of employing confidence scores is by thresholding. Particular sectors within the training data, whose automatic confidence is below a pre-defined or automatically calibrated threshold will be skipped, and thus excluded from training. The general assumption is, that the repetition of iterative transcription runs followed by the training of an expectably improved text-to-speech system using that very data converges to a system being capable of producing competitive recognition results. As a consequence of the necessity of multiple iterations, and given the fact that confidences merely correlate with veritable probabilities, suggesting a certain wariness of the errors in the data, a significantly larger amount of training material is needed compared to a training on supervised data [Rog05]. [KW99] reports, that approximately twice the amount of initially untranscribed data is needed for training in order to achieve a comparable performance as with supervised training on manually transcribed data. It is worth mentioning that this is but a scarce estimate, as the effectiveness of unsupervised AM training heavily depends on the baseline system used for automatic transcription generation, and the target training data. Exemplarily, [LGA02] demonstrates the effectiveness of unsupervised training: A system trained system on 140 hours of unsupervised data resulted in a system performance of 23.4% WER, compared to a system supervisedly trained on 50 hours of manually annotated data yielding a performance of 20.7%, thus verifying the assertion of [KW99].

Besides training acoustic models in a supervised or unsupervised manner, one can think of a training scheme in between. Any textual information related to the recorded training samples may be utilised in place of eventually missing manual transcriptions. Automatically generated annotations may be filtered based on available textual information of a certain degree of detail and accuracy, e.g., closed captions, utilising confidence measures or skipping non-matching parts in both annotations. Closed captions may also be used for training directly, with the constraint that missing information such as non-annotated noise, unknown speaker identities or non-speech segments have to be produced automatically. Moreover, the alignment of text and audio must allow for transcription errors such as insertions, deletions or substitutions [LGA02]. It is also conceivable to use related textual information for dictionary adaptation and language model training, which introduces the option to generate the most likely strings of words given the presumably more suitable models. The latter approaches are known as *lightly supervised* acoustic model training [LGA02].

1.3 The JANUS Recognition Toolkit

The speech decoding modules of the systems used and described in this work are realized with the JANUS Recognition Toolkit (JRTk), which has been developed at the Karlsruhe Institute of Technology and Carnegie Mellon University as a part of the JANUS speech-to-speech translations system [FGH+97, LWL+97]. The toolkit provides an easy-to-use Tcl/Tk script based programming environment which gives researchers the possibility to implement state-of-the-art speech processing systems, especially allowing them to develop new methods and easily perform new experiments. JANUS follows an object oriented approach, forming a programmable shell. For this thesis, JRTk Version 5 was applied, which features the IBIS decoder. IBIS is a one-pass decoder, thus being advantageous with respect to real-time requirements of today's ASR and other language processing applications [SMFW01].

1.4 The KIT Lecture Translator

Lectures at universities around the world are often given in the official language of the respective university's location. At the Karlsruhe Institute of Technology (KIT), for instance, most lectures are held in German language. Often, this poses a significant obstacle for students from abroad that wish to study at KIT, as they need to learn German first. In order to be able to truly follow the often complex academic lectures, the level of proficiency in German that the foreign students need to reach is quite high.

While in principal simultaneous translations by human interpreters might be a solution to bridge language barriers in such a case, this approach is too expensive in practice. Instead, technology in the form of spoken language translation (SLT) systems can provide a solution, making translations of lectures available in many languages at affordable costs. Therefore, one of KIT's current research focuses is the automatic translation of university lectures [FWK07, FÖ8], with the aim to aid foreign students by bringing simultaneous speech translation technology into KIT's lecture halls.

The simultaneous lecture translation system that is used for this purpose is a combination of an automatic speech recognition (ASR) and a statistical machine translation (SMT) system. For the performance of such an SLT system the word error rate of the ASR system is critical, as it has an approx. linear influence on the overall translation performance [SPK+07].

Automatic speech recognition for university lectures is rather challenging. In order to obtain the best possible ASR performance, the recognition system's models, including acoustic model and language model, need to be tailored as closely as possible to the lecturer's speech and the topic of the lecture.

The speaker independent system that is used in the experiments described in Chapter 4 of this study was taken from the inauguration of the lecture translation system at KIT on June 11th 2012 [CFH+12]. For the inauguration, first a speaker-independent acoustic model system was trained on all available training data from the KIT lecture corpus for Speech Translation [SKM+12], and then adapted to the individual lecturers.

1.5 Objective of This Work

This thesis addresses the theoretical concepts of unsupervised acoustic model training and describes the application and evaluation of unsupervised training schemes. Starting with a speaker independent version of the KIT lecture translator system, experiments aiming at speaker adaptation via unsupervised training are conducted. Iterative as well as incremental training approaches are evaluated and compared with respect to the training

efficiency in terms of minimal amount of training data needed to observe improvements, and overall recognition performance after training. Having a large amount of unsupervised out-of-domain data at hand, a system trained for appliance to European Parliament Plenary Session (EPPS) speeches is intended to be re-trained to a new domain by an iterative batch training approach. Given these two experimental scenarios, it is a major objective to investigate the impact of various transcription pre-processing methods, as well as the effectiveness of confidence measure based data filtering methods applied during acoustic model training, in the form of confidence measure based weighting and thresholding on word level. The objective is to lay the foundation for an unsupervised adaptation framework based on acoustic model training for use in KIT's simultaneous speech-to-speech lecture translation system [FÖ8].

This thesis is organized as follows: Chapter 2 outlines the basic principles of acoustic model training. An insight into the standard training procedure along with a probabilistic formulation will be given, as well as an overview of the various levels of supervision that are applicable during model training. Chapter 3 provides a detailed insight into unsupervised acoustic model training approaches. A major focus is on various design decisions that have to be made when establishing a training scheme given the available resources. The chapter concludes with a view on related work. The designs of the training frameworks for the KIT lecture translator system is explicated in chapter 4. Chapter 5 elaborates the applied strategies given the EPPS system as starting point. Both chapters begin with an introduction of the respective dataset being worked on, followed by a detailed account of the baseline system. Following the explanation of the strategies for decoding, training and testing is a detailed presentation of the experimental results, which comprises the evaluation of various applied transcription pre-processing and data filtering techniques, as well as variations of iterative training schemes. Each of the chapters is concluded by an Analysis of the results. Chapter 6 summarizes this study and gives an outlook on future work.

2. Acoustic Model Training

In speech recognition as well as for pattern classification tasks in general, main principles are fragmentation of large problems into smaller problems, whose solutions are optimally separately realizable [Rog05]. ASR systems most commonly model acoustics and linguistics separately in the form of *acoustic model* and *language model*. Training of the acoustic models is the main topic of this chapter.

The purpose of the acoustic model is to provide a method of computing the likelihood of any sequence of feature vectors, given a specific sequence of words [You96]. As it is impractical for large vocabulary speech recognition systems to model words as a single entity, the actually modelled sound units are further split into single phones, where each phone is represented by a particular hidden Markov model (HMM). The core concepts used during training of HMM-based acoustic models are the Baum-Welch rules and the Expectation-Maximization algorithm (EM algoritm). The general training process can be divided into three steps, the *initialization* step, the *iterative optimization* and the *evaluation* step [Rog05].

2.1 Probabilistic Formulation

A hidden Markov model is a five-tuple *(S,A,B,π,V)*, where

- $S = s_1, \cdots, s_n$ is the set of all states of the HMM
- $A = (a_{i,j})$ is the state transition matrix, $a_{i,j}$ being the probability of a transition from s_i to s_j
- $B = b_1, \cdots, b_n$ is the set of emission probabilities for a discrete V, or emission densities for a continuous V, where $b_i(x)$ is the probability of observing x when being in state s_i
- π is the probability distribution of the start states, where $\pi(i)$ is the probability of s_i being the initial state
- V is the feature space of b_i, where in the discrete case $V = v_1, v_2, \cdots \Rightarrow b_i$ is a probability, and in the continuous case $V = (R)^n \Rightarrow b_i$ is a density

For mathematical correctness the following stochastic constraints must be satisfied:

Start probabilities

It must be $\sum_{i=1}^{n} \pi(i) = 1$. A common set-up in practice is $\pi(0) = 1$ and $\pi(i) = 0 \forall i > 0$

Transition probabilities

It must be $a_{i,j} \geq 0 \ \forall i, j$ and $\sum_{j=1}^{n} a_{i,j} = 1$, i.e., all outgoing transitions of a state s_i have to be 1.

Furthermore, for the special case of a discrete first order Markov chain as it is used for the purpose of acoustic modelling, it is

$$P(q_t = s_i | q_{t-1} = s_j, q_{t-2} = s_k, \cdots) = P(q_t = s_i | q_{t-1} = s_j) \tag{2.1}$$

and

$$a_{i,j} = P(q_t = s_j | q_{t-1} = s_i) \quad , 1 \leq i, j \leq N \tag{2.2}$$

because only these processes are considered where the right hand side of Equation 2.1 is independent of time [You96].

An HMM can be interpreted as a finite state machine that serves as a generator of vector sequences, where a state $q_t = s_i$ is changed to $q_{t+1} = s_j$ once for a particular point t in time, and a feature vector v_t is output with an emission probability $b_j(v_t)$ [You96]. Thus, the joint probability of a produced sequence of feature vectors X and the sequence of visited states S given the HMM λ is calculated as

$$p(X, S|\lambda) = a_{0,1} \prod_{t=1}^{T} b_t(x_t) a_{t,t+1} \tag{2.3}$$

The three fundamental problems of HMMs are known as the evaluation problem, the decoding problem and the optimization problem [Rab89]. Given an existing HMM and an observation, the evaluation problem addresses the computation of the probability of how likely the HMM emits the observation. The decoding problem describes how to compute the most probable sequence of visited states for generating the observation. The optimization problem is also known as learning problem and addresses the task of re-computing a new HMM that emits the given observation with a higher probability than the initial HMM. Consequently, the core of acoustic model training for HMM-based models is the optimization problem of HMMs.

2.2 Optimization Problem

The optimization problem raises the question, how to adjust the HMM model parameters S, A, B, π, V so that $P(O|\lambda)$ will be maximized [Rab89]. hidden Markov models are optimized iteratively in a way that for every point i in time $Q(\lambda_{i+1}) > Q(\lambda_i)$, where Q is a pre-defined optimization function. The predominant training scheme in the field is following the maximum-likelihood criterion by trying to maximize the observation probability of the training data, which corresponds with the evaluation problem for HMMs [Rog05]. Thus, after running through a training sequence a model should be capable of describing a given observation better than before.

Formally, the optimization problem is to find a λ' with

$$p(X|\lambda') > p(X|\lambda) \quad \text{, with given } \lambda, X = x_1, \cdots, x_T \tag{2.4}$$

There is no known way to analytically solve this training problem of maximizing the probability of outputting a given observation [Rab89]. Given any finite observation sequence as training data, there is no optimal way of estimating the model parameters. However, it is possible to choose model parameters so as to locally maximize the probabilities. With the Baum-Welch rules and the EM algorithm at hand there exist methods of iteratively optimizing all relevant model parameters.

The primary task of the training algorithm is to optimize all parameters of a state s_i. For that, it has to have knowledge about the probability of being in a particular state s_i at time t when making the observation $x_1, \cdots, x_T$. This probability is defined as

$$\gamma_t(i) = P(q_t = i|X, \lambda) \tag{2.5}$$

By applying the Bayes rule and subsequent decomposition $\gamma_t(i)$ can be described as

$$\gamma_t(i) = \frac{P(q_t = i, X|\lambda)}{P(X|\lambda)} \tag{2.6}$$

The numerator of this term is computed by the Forward-Backward algorithm, which is used to solve the evaluation problem. The probability of being in state s_i at time t and making the full observation X can be described as

$$P(q_t = i, X|\lambda) = P(q_t = i, x_1, \cdots, x_t|\lambda) \cdot P(x_{t+1}, \cdots, x_T|q_t = i, \lambda) = \alpha_t(i) \cdot \beta_t(i) \tag{2.7}$$

where $\alpha_t(i)$ is the probability of being in state s_i after having seen the partial observation $x_1, \cdots, x_t$, and $\beta_t(i)$ is the probability of being in state s_i and making the future partial observation $x_{t+1}, \cdots, x_T$ [Rab89]. That implies that

$$\gamma_t(i) = \frac{P(q_t = i, X|\lambda)}{P(X|\lambda)} = \frac{\alpha_t(i) \cdot \beta_t(i)}{\sum_j \alpha_t(i) \cdot \beta_t(i)} \tag{2.8}$$

Given this formulation it is sufficient for the training algorithm to know the observation $X = x_1, \cdots, x_T$ and the corresponding $\gamma_t(i)$ for optimizing the emission probabilities of an HMM [Rog05].

The probability of a transition from s_i to s_j when observing X is defined as

$$\xi_t(i, j) = P(q_t = i, q_{t+1} = j|X, \lambda) \tag{2.9}$$

By applying the Bayes rule and decomposition by utilization of the α and β terms, this probability can be expressed as

$$\xi_t(i, j) = \frac{P(q_t = i, q_{t+1} = j, X|\lambda)}{P(X|\lambda)} = \frac{\alpha_t(i) a_{ij} b_j(x_t + 1) \beta_{t+1}(j)}{\sum_l \alpha_t(l) \beta_t(l)} \tag{2.10}$$

By having α, β, γ and ξ at hand, the Baum-Welch rules can be applied for HMM parameter optimization:

$$a'_{i,j} = \frac{\sum_{t=1}^{T} \xi_t(i, j)}{\gamma_t(i)} \tag{2.11}$$

is the updated probability for a transition from s_i to s_j, and

$$\pi'(i) = \gamma_1(i) \tag{2.12}$$

is the updated probability of s_i being the initial state of the HMM.

The update step of the emission probabilities for each state depend on the nature of the emission probability models. In the continuous case, i.e., when using Gaussian mixture models as models for emission probabilities, the EM algorithm is applied for parameter updating. In the discrete case, the Baum-Welch rule

$$b'_i(v_k) = \frac{\sum_{t=1}^{T} \gamma_t(i)\delta(x_t, v_k)}{\sum_{t=1}^{T} \gamma_t(i)} \quad \text{, with } \delta(x_t, v_k) = \begin{cases} 0 & \text{for } x_t \neq v_k \\ 1 & \text{for } x_t = v_k \end{cases} \tag{2.13}$$

is applicable. In the case of emission probabilities modelled by neural nets one might utilize the Back-Propagation algorithm for training.

2.3 Initialization

Several strategies exist for initializing acoustic model training, depending on the available resources. The three common basic approaches are random initialization, initialization by utilizing labelled data, and initialization by parameter transfer.

2.3.1 Random Initialization

Following the theoretical formulation of the Baum-Welch rules and the EM algorithm there is no demand of an initialization of training parameters with a particular set of values. By definition, HMM training converges to a local optimum with every optimization step, in strict accordance with mathematical correctness [Rog05]. Nevertheless it is recommended to choose start values that represent an advantageous starting point for parameter optimization. There are mainly two reasons for the potential benefit by doing so: Firstly, applying the Baum-Welch update rules only guarantee the convergence to a local optimum. Secondly, an unfavourable parameter initialization may lead to very long optimization cycles. Thus, a pre-defined starting point may lead to a better local optimum than a mere random initialization, as well as sped-up training runs.

2.3.2 Utilization of labelled data

Labels are assignments of feature vectors to sound models. There exists a variety of options for gathering labels, beginning with the almost entirely manual production of observation-to-model assignments to fully automatic label generation techniques. Usually, the most reliable labels are labels based on man-made assignments of single sounds to audio segments, but naturally this is the most expensive way of obtaining labels, in both time and cost. Today, automatic label generation is commonly achieved by utilizing word based transcriptions that match the audio data intended for use as training data. These transcriptions usually hold a certain level of detail by covering not only the audible words, but also perceptible noises of articulatory (smacking, breathing, etc.) as well as linguistic (incomplete words, repetitions, etc.) and environmental (background noise, etc.) nature. With the help of this type of data, labels are generated by applying the Forward-Backward or Viterbi algorithm on the transcribed training data. For this, however, an already existent recognizer is indispensable. The resulting labels are usually significantly flawed, but still usable for initializing a new recognizer. Initialization of the HMM parameters is

done straightforwardly with help of the Baum-Welch rules. Initialization of the Gaussian mixture models for modelling emission probabilities is commonly done by using the k-means algorithm. Here, the labels determine which feature vector belongs to which sound model. Initial "codebooks", i.e., models for distinct sound units are then computed by the k-means algorithm on a full vector-to-model assignment.

2.3.3 Initialization by parameter transfer

Another applicable method for parameter initialization is a parameter transfer from an existing system to the new ASR framework. The complexity of a transfer depends on the divergence between the source and target system. If the architectures are similar or equal, a simple transfer by copying can be conducted. If both systems significantly differ, certain parameters have to be discarded, or modified to fit to the new models, if possible.

2.4 Iterative Optimization

Training schemes that follow the approach of iterative optimization have in common that one of the core principles is repeated, subsequent training and testing. The training step may either be another iteration of Baum-Welch or EM based model updating, or changing to a higher level of system complexity, e.g., by increasing the amount or size of GMMs or introducing more a more fine-grained parameter-typing [Rog05]. The test phases are tools for monitoring process and verifying the correctness of the training pipeline. Decisions regarding the finalization of training or modification of training steps can be made by reference to regular feedback through evaluation.

With the help of the Forward-Backward algorithm the probabilities $\gamma_t(i) = P(q_t = i|X, \lambda)$ used during training can be computed. Conducting training this way allows for a training sample, i.e., a particular feature vector to be assigned to various models at the same time, but with differing probabilities. As a consequence, single samples extracted from the training data contribute to the parameter update of multiple models. One drawback of using the Forward-Backward algorithm therefore is the increased complexity of the parameter update step, which usually leads to considerable run-times when training on large amounts of data.

Thus, it is a common practice in the field to use the Viterbi algorithm instead. As opposed to the Forward-Backward algorithm, Viterbi computes the most probable sequence of visited states:

$$Q = q_1, \cdots, q_T = \underset{Q}{\operatorname{argmax}}\, P(Q|X, \lambda) \tag{2.14}$$

Consequently, the probabilities $\gamma_t(i)$ used for training are approximated by

$$\gamma_t(i) = \begin{cases} 0 & \text{for } i \neq q_t \\ 1 & \text{for } i = q_t \end{cases} \tag{2.15}$$

The derivation of EM training for HMM parameter optimization is known as Viterbi training and utilizes the Baum-Welch rules with the constraints [ST95]:

$$\gamma_t(i) = \delta(q_t, s_i) \text{ and } \xi_t(i, j) = \delta(q_t, s_i)\delta(q_{t+1}, s_j) \tag{2.16}$$

With increasing T both algorithms result in an almost equally effective training set-up [Rog05]. One major advantage of Viterbi training is a significantly decreased training run-time due to the lower amount and complexity of computations, as well as easier application

of search space restrictions. An even higher speed-up is attainable by training along labels. Similar to parameter initialization by labels, the Baum-Welch rules can be applied on pre-computed alignments for parameter updating. In order to achieve a training effect, multiple training steps along labels are followed by a re-computation of labels, so that assignments of sample vectors to models may change. This training scheme is iterated multiple times.

2.5 Evaluation

The quality of an automatic speech recognition system can be measured by means of a recognition error. Usually, a recognition error is computed on word level, which leads to a word error rate, given a set of test utterances and their reference transcriptions. The word error rate on a test set $REF = ref_1, \cdots, ref_n$ and hypotheses $HYP = hyp_1, \cdots, hyp_n$ is defined as

$$WER(HYP, REF) = \sum_{i=1}^{n} \frac{N_i^{sub} + N_i^{ins} + N_i^{del}}{N_i} \tag{2.17}$$

where N_i is the total amount of words in reference ref_i. N_i^{sub}, N_i^{ins} and N_i^{del} count the substitutions, insertions and deletions of words in the hypothesis in comparison to the respective reference ref_i.

Computation of the WER may be done during system development for progress monitoring, or as decision aid for modifications on the training framework. Ultimately, the WER may be used as basis of assessment during final evaluation runs. Usually, prior to an evaluation on a separate data set, parameter tuning by minimizing the WER on a development set is conducted. JANUS, which is used for all experiments during this project, is equipped with a hypothesis scoring, whose parameters have a direct impact on the structure of generated hypotheses. Derived from the following formula:

$$P(W|X) = \frac{p(X|W) \cdot P(W)^{lz} \cdot lp^{|W|}}{p(X)} \tag{2.18}$$

the IBIS decoder used by JANUS scores the hypothesis related to an input utterance as follows:

$$score(W|X) = logP(X|W) + logP(W) \cdot lz + lp \cdot |W| \tag{2.19}$$

The *lz* parameter constitutes a language model weight, i.e., it determines the impact of the language model on the decoding process relative to the acoustic model. The parameter *lp* is a hypothesis length penalty or more precisely a word transition penalty, whose proper adjustment helps to normalize the length of sequences of words [SMFW01]. Fine-tuning the *lz,lp* value pair aims at minimizing the word error rate of the development set so that the final system is optimized to the previously unseen target evaluation data.

2.6 Levels of Supervision

As is the case for training of classifiers in general, it is particularly common for acoustic model training to utilize data of various levels of supervision, depending on the available amount of training data, as well as the objective target of system development. The following sections attempt to give an overview of the common levels of supervision in acoustic model training. It is noteworthy, however, that in practice terms have been used with a

certain inconsistency over time so that one might eventually encounter overlapping definitions when reading about unsupervised, semi-supervised and lightly supervised acoustic model training. In fact, the transitions between the approaches are fluent, and not uncommonly it might be difficult to strictly assign a particular approach a specific category of supervision.

2.6.1 Supervised training

Model training is performed on labelled data, i.e., audio data that comes with textual references of what was said serves as training data. In other words, the assignment of training samples to models is fully known and is intended to be learned by the system for generalization on previously unseen data. A training data set is comprised of training examples, where each example is a pair of audio recording and the desired ASR output, or ground truth. The goal of supervised training is to maximize the probability that the system's models hypothesize the a priori known reference.

2.6.2 Semi-supervised training

In a semi-supervised training framework, references are only available for a subset of the full set of training data, and the remainder of the data is without references. Often, the portion of unsupervised data is many times larger than the supervised subset. The process of gathering references for training samples is usually expensive, whereas unlabelled data may be available in much higher quantities. In the context of acoustic models, semi-supervised learning may be considered inductive learning: First, models that were trained on the supervised training subset are used to infer transcriptions of previously untranscribed data in order to include the latter into system development. Then, the objective is to produce an optimal prediction of what was voiced in one or more test utterances. This particular approach, which is also known as *self-training* [CSZ10].

2.6.3 Lightly-supervised training

In general, any kind of related linguistic information to the audio data intended for training can be used for supervision. Various ways of utilization are conceivable, e.g., by substituting missing detailed transcriptions, with application of proper matching strategies such as flexible transcription alignment [FW97]. Another way of exploiting textual data that is loosely coupled to the audio material is the use as training corpus for a language model, along with dictionary adaptation, which both can subsequently be applied for automatically generating more accurate transcriptions for model training. The advantage is that related textual data is commonly available on a comparatively larger scale than detailed transcriptions. Moreover, loose transcriptions such as closed captions as they are used for television broadcasting are producible with significantly less effort [LGA02]. A third way one can think of utilizing available textual data is as reference text, which for instance enables data filtering by comparison, e.g., with the help of distance measures or majority votes.

2.6.4 Unsupervised training

Unsupervised training is performed without any labelled data at hand. The core principle is to find the hidden structure in the labelled data so that it might become utilizable for training classifiers or models. Within the frame of acoustic model training the main task is to automatically find transcriptions for the unsupervised data in a way that they resemble the optimal solution as good as possible. The main issue is that there exist no intuitive measures of error or correctness that can be used to evaluate the proposed transcriptions, since no reference data is available. However, there exist several techniques

based on automatic confidence measures to pre-process and filter data. Similar to the semi-supervised approach, an existing system is commonly used for automatic transcription. The applied system, however, may show only poor performance on the target data. Thus, it has to be ensured that erroneous data is exempt from training. Again, this can be achieved by confidence based pre-processing and filtering. Another applicable strategy is adapting the transcription system to the target data in order to reduce the amount of emerging errors [Rog05]. With the now transcribed data, a full acoustic model training can be performed.

3. Unsupervised Acoustic Model Training

One of the major challenges in training of ASR systems, in particular the acoustic model training is the reduction of development costs. Here, a major cost factor is the production of detailed transcriptions or labels for acoustic model training data. Estimations of efforts to produce high-quality transcriptions for audio data are in double figures of real-time [LGA02]. Thus, usually a huge quantity of working hours, as well as high costs of personnel expenses is needed. Moreover, there is need of professional, trained transcriptors, and the search of experts may pose another issue in system development plannings. Further on, not only for full system training, but also for the task of adaptation there is need of accurate transcriptions, depending on the applied method.

On the other hand, the amount of available audio data that is untranscribed, but freely accessible is nearly unlimited. May it be web services such as youtube[1], with a very broad – if not to say boundless – spectrum of topics, TED[2] with multiple pre-defined thematic priorities, broadcasting services or specialized podcasts, all of them embody valuable data resources which are potentially utilizable for automatic speech processing in general. Today several approved unsupervised acoustic model training techniques are capable to efficiently use such untranscribed data for model training and model adaptation. The basic idea of these techniques is to use a speech recognizer system, which may have been into existence before, or that has been trained for this specific purpose, to transcribe this raw audio material. The resulting transcriptions, that usually are approximate and only partially correct, are then used for the ultimate acoustic model training. A key role plays the pre-processing and filtering of this error-prone data, as only this allows for efficient training after all.

3.1 Unsupervised Training

In the following a standard scheme for unsupervised training shall be elaborated. The minimal requirements for conducting unsupervised training is the availability of certain amounts of audio material that is in a condition to serve as training data. Also, one needs at least a minimal system to start with. This system may either be an existent ASR framework or a bootstrapped variant, or it may be a system that was just trained on a minimal set of data. In the former case the system may be an outdated or an intermediate version of a former development process. Typically the models used by these systems

[1] http://www.youtube.com
[2] http://www.ted.com

are less complex, and there might be a considerable mismatch between the source and target domains as well as significant differences in the channel properties. However, the utilized system might perform well enough to produce acceptable transcriptions for further processing. As opposed to this, the system used for transcription could also be optimized to the target data already, and possibly even be a baseline system with the objective to get further adapted and fine-tuned to this type of data.

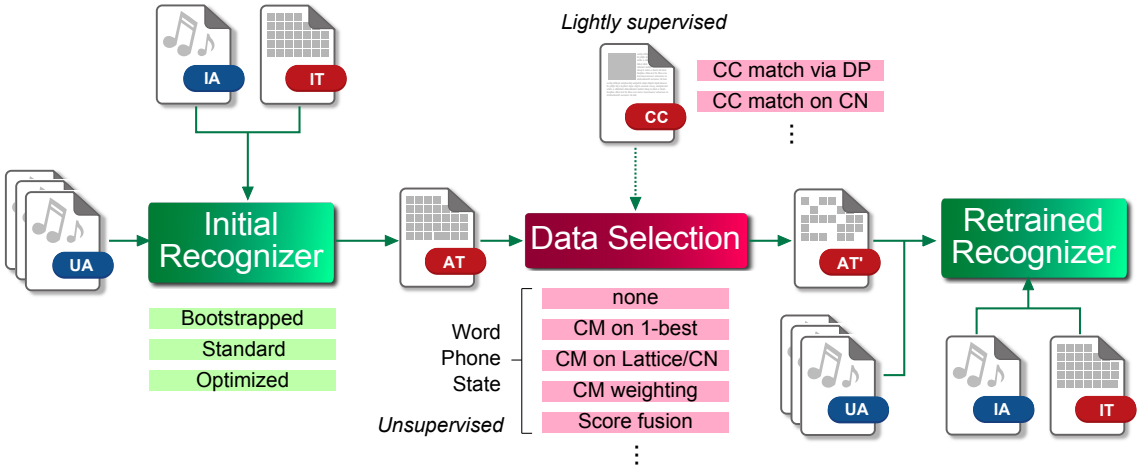


Figure 3.1: General unsupervised acoustic model training set-up. Unannotated audio data (UA) is transcribed by a system that was trained on initial audio (IA, IT). The automatic transcriptions (AT) are pre-processed for training in a data selection step. The re-trained recognizer may also be trained on the initial supervised data.

If there is no ASR system available for a straightforward application as automatic transcription system, one might derive a new system from old models by bootstrapping. Experiments have shown that already very small amounts of manually transcribed data can be used for training a minimal system that can be used for automatic transcription of an untranscribed portion of the training data [LlGA02]. Thus, in practice it became popular to manually transcribe a small portion of the large amounts of available training data and using this subset for supervised training of a minimal ASR, that subsequently serves as transcription engine. Here, the initial system blends seamlessly in the whole development process as a mismatch between channels and/or domains can be avoided.

Following the acquisition of an initial system that can serve as generator for automatic transcriptions, the actual transcription of the unsupervised training data takes place. The transcription system decodes the target data and stores the textual representations in an appropriate way. There might be differences in the decoding strategy, depending on the steps that will follow, or the kind of training that shall be applied. If rapid gain of additional training data is the goal, decoding might be performed with a one-pass decoder and without lattice re-scoring, whereas for the acquisition of higher-quality transcriptions the latter may be applied, along with other multi-pass strategies, or even system combination approaches.

The automatic transcription is followed by a data selection phase. In principal, this phase is borne by two actions, transcription pre-processing and transcription filtering. Transcription pre-processing – the term relates to a processing step prior to an actual acoustic model training – comprises textual processing methods and does not necessarily include any active rejection of data in larger quantities, e.g., the dismissal of whole sentences, although that might be the case under certain circumstances. In general, the pre-processing that is applied aims at filtering the textual data. Decoder outputs may still include non-word

tokens tagging occurrences of noise or articulatory artifacts or the voicing of filler words. It can be desirable to free the automatic transcriptions of such additional annotations, which for instance may depend on the expectation regarding the accuracy of the utilized decoder. If it is expected that the decoder's hints regarding occurrences of non-speech events are error-prone it may be of advantage to let the actual training decide where to suspect non-speech. The data filtering step actively exempts parts of the data from the subsequent training. This is commonly done by applying confidence measure techniques. Upon the most prominent methods is the utilization of lattice-based posterior probabilities for confidence measurement [KW99]. Many variants are conceivable, regarding the granularity of sound units to observe, ranging from (sub)phone level to sentence or even document level. Common is the operation on word level. The possibility of obtaining confidence scores is manifold. Confidence measures may be taken into consideration on a 1-best hypothesis only. It may also be operated on a word lattice or confusion networks so as to include more informations into the decision process.

Data can effectively be exempt from training by defining a confidence threshold. If a word or whole parts of the transcribed data fall below a particular confidence score, the respective data will be ignored during training. Another possibility of filtering is the application of weighting according to confidences. This way, no data is completely thrown away, but it slips into the training with weights adjusted to the confidence of the respective parts, thus limiting the damage potentially erroneous data may evoke.

The data selection phase results in a cleaned and filtered set of automatically transcribed training data. However, there is no guarantee that the data set is free of errors. On this pre-processed set the actual acoustic model training is performed. Depending on the availability of supervised data that has been used for training the transcription system, as well as the nature of that data – it may be that there is a mismatch between the supervised and unsupervised data in terms of domain or channel characteristics – it is to decide whether or not to include this data into the subsequent system training. In case of inclusion one might also speak of a semi-supervised training framework [LZM12].

Figure 3.1 schematizes a typical framework for unsupervised acoustic model training, which follows the approach of automatic transcription with the help of an initial system followed by model parameter updating. What has not yet been taken into consideration in the above outlines is the option of light supervision by utilizing additional textual data such as closed captions. With the automatically generated transcriptions and the partial references at hand, data selection may also be conducted by several techniques such as dynamic programming, application of confusion networks or simple word based majority votes. This approach is known as lightly supervised training [LGA02]. The optional light supervision via language modelling has no graphical representation in the figure, however it should be mentioned herewith for the sake of completeness. The graphic also illustrates the previously described decision whether or not a priori available supervised data is included in the actual system training.

3.2 Design decisions

Depending on the availability and condition of potential training data, as well as the objective target of the system training there are various design decisions to make, regarding:

- the pre-processing of data
- the filtering for training
- the training paradigm
- the application of additional supervision

Although any additional supervision leads to a training scheme that is not fully unsupervised anymore, it shall be included in the list of design decisions, as in practice there usually is no clear and absolute distinction between fully unsupervised training or essentially unsupervised training with a certain degree of supervision.

3.2.1 Amount of Acoustic Training Data

The amount of data that will be actually used for model training has a direct impact on the maximal complexity of the acoustic models. With more data being usable for training, models can represent the feature space in a more fine-grained way, allowing for smaller sound units such as context-dependent phonemes instead of context-independent ones, or generalized polyphones instead of tri-phones, for instance. More data also allows for a less generalized parameter coupling. Another important factor remains the quality of the data, as a more fine-grained model set is more prone to erroneous data and outliers. Obviously, with more data at hand, systems can also potentially improve more from training as with less data, as long as the applied techniques allow for further improvement.

3.2.2 Pre-processing of Acoustic Training Data

The results from automatic transcription runs are raw decoder outputs. As such, the textual representation of what was said – according to the used decoder – is still annotated with non-word tokens that may mark occurrences of noise of various kinds, as well as silence or other artifacts that are irrelevant for a final hypothesis if the decoding were a standard ASR task. For unsupervised training, both detailed decoder outputs as well as cleaned transcriptions are applicable. It is primarily the quality of the decoder which decides whether it is useful to use the estimates of noise and filler occurrences or not. The presence of additional information about the audio data provides a higher flexibility during training and may also lead to a more efficient training due to the increased detail about the data. On the other hand, the same circumstance poses a risk of leading the models into a wrong direction, if the data is heavily error-prone or simply not reliable enough in general. In the latter case, it might be beneficial to let the training algorithm decide how to handle non-speech parts in the training data.

3.2.3 Filtering of Acoustic Training Data

The most common methods for processing unreliable, erroneous transcriptions in unsupervised acoustic model training are based on lattice confidence measures at word or state level [FSGL11]. [Kea97] has shown that the word lattice output of a ASR system contains useful informations that can be used to estimate the likelihood of each word that has been hypothesized by the decoder. The general idea is that the existence of a high number of word hypotheses with a similar likelihood in a specific time span implies a higher risk of confusion, or error in general. In contrast to that, if for a specific word hypothesis in a specific time segment the probability is significantly higher than the probability of other words, this implies a rather low risk of error.

Confidences can be applied in several ways, where the most prominent are:

Weighting factor During training, the data is weighted by posteriori probabilities that have been computed on the lattice informations. For instance, the weighting may be conducted by adjusting the γ-probabilities during Viterbi training according to the confidence score of the respective part of the training data.

Threshold By setting a threshold value, e.g., a lower bound for confidence scores, parts of the training data that fall below this specific confidence limit will be exempt from

training. The strictness of the threshold value is essential for a positive training effect: If the threshold is too lax, more incorrect data tends to slip in the training, whereas a too strict threshold leads to the increased dismissal of actually reliable word hypotheses.

These attempts of error filtering and error avoidance are not free of making errors either, as the confidences are but estimates with limited reliability. Thus, further fine-tuning during development might be necessary in order to benefit the most from these techniques.

3.2.4 Training Paradigms

Basically three major training paradigms are applicable for unsupervised training, namely *batch training*, *iterative training* and *incremental training*. One can also think of combinations of these individual set-ups. The term batch training describes a single iteration of unsupervised training. This approach is the most simple solution for the training problem given unannotated data. Batch training is applicable for all amounts of training data. If only a very limited amount of audio material is available, a single iteration of automatic transcription followed by model training may already suffice for adapting an existent system to new conditions.

There exists a large variety of works describing the iterative and incremental approaches. The latter scheme is comprised by multiple iterations of automatically transcribing data and subsequently training a new system, where for each iteration the by this point best system is used for decoding. The incremental approach also performs training in an iterative fashion, but operates on a growing set of training data. In practical terms this means that each iteration a certain portion of previously unseen unannotated data is added to the training set. A common approach is to double the data for each subsequent iteration. Also, the decision can be made whether or not to use the data from previous iterations during model training.

3.2.5 Additional Supervision

Although any form of additional supervision leads to a training that is not fully "unsupervised" anymore, one common practice may be mentioned briefly for the sake of completeness and in order to do justice to in-the-field practices. The application of language models that are tailored to the domain of the data intended to use for training is quite common. The reason is the relatively simple generation of such models. With a sufficient amount of related textual data at hand, it is straightforward to adapt an existing background language model to the target domain, thus enhancing the decoding process easily by long approved methods. In fact, many studies regarding unsupervised acoustic model training apply this method of very light supervision without refraining from the terminology "unsupervised" [LlGA02, GHSN07, FSGL11].

3.3 Related Work

Using untranscribed audio data for unsupervised acoustic model training is in the focus of research since more than 15 years, beginning with studies like [ZaTCB98] and [KW99], both already including the utilization of confidence measures. In recent years, also large-scale experiments have been conducted by several research groups, e.g., [LGA02], [FSGL11] and [MMKS06], the latter utilizing more than 1900 hours of unnanotated data for unsupervised training.

[ZaTCB98] raised the question, what amount of unannotated training data would suffice, and how adding automatically transcribed data to the training set compares with adding

manually transcribed material. Their bootstrapped initial system was trained on 3 hours of manually transcribed data and subsequently used for automatically transcribing 25 hours of unannotated data, using confidence estimation via score fusion and thresholding for data selection. The newly transcribed data was then added to the manually annotated portion and subsequently used for re-training the acoustic models. The observed improvements were bigger for new data from speakers already seen during training, than for data from new speakers. The discussion already proposed an iterative approach to unsupervised training.

The lattice-based confidence measure introduced in [Kea97] was applied in [KW99]. For automatic transcription, a bootstrapped version of the view4you [KGS+98] speech recognizer was trained on 30 minutes of manually transcribed data. The training followed an incremental approach by training intermediate systems, which in turn were used for subsequent decoding runs on previously unseen training data. System combination was further applied via ROVER [Fis97]. The study showed that comparable performance can be achieved by using at least twice as much untranscribed data as transcribed data.

Also following an incremental approach was [LlGA02], but the focus was on minimizing the efforts of bootstrapping a system for automatic transcription. A system trained on 10 minutes of manually annotated data served as basis to start from. A major focus was also on data pre-processing by deleting and rejecting non-speech parts via iterative maximum-likelihood segmentation and clustering on speech segments. The study observed three start conditions for unsupervised training: Training with removed story boundary filtering, training on a very small amount of manually annotated data, and decoding with language models that have been trained on substantially less data. Training was performed with six iterations, each time doubling the amount of data. The study was able to show that manual transcriptions are no necessary requirement for successful acoustic model training.

Carrying on with the latter studies, the focus of [LGA02] was on light supervision via utilization of closed captions (CCs). Automatic transcripts that have been generated by a bootstrapped system were matched against CCs with the help of dynamic programming. Furthermore, CCs were used for training a language model that has then been interpolated with a background model. Training was performed incrementally on data that added up to 550 hours, split into chunks that have been only used once in one single iteration each. The study concluded in addition to previous results that filtering for erroneous data is helpful but not required.

[CLG04] used a strongly biased language model that has been trained on solely on CCs. Also, hypothesis lattices of the decoded unannotated data were aligned with CCs via a consensus network. All segments for which the words in the CC were found on a path through the network were kept. According to the conclusions, this approach led to a higher yield of training data, when compared to [LGA02].

An alternative to word based confidence measures was considered in [WN05]. The observation that the phoneme error rate on the training data was only roughly half as high as the word error rate backed the hypothesis that the recognition results from automatic transcription runs may be rather similar to what was spoken, despite a high word error rate. [GHSN07] further observed data selection methods on state level using word based confidence scores and allophone state based confidence scores.

A lattice-based approach working on word level was examined in [FSGL11]. The hypothesis selection took place in the word lattice by rejecting segments for which the posterior probabilities are below a certain threshold. Paths through the lattice were pruned away, if they fell below a limit for the likelihood of a path.

[LZM12] introduced a confirmation based self learning algorithm for an incremental training framework. Data selection was conducted with a two-stage approach resembling a voting method. Filtering by confidence measures was conducted on sentence level. Further, data selection was performed on word level by a confirmation criterion. The study further showed that adding data with lower-scored transcriptions to the training data fosters the generalization capabilities of the resulting system.

3.4 Conclusion

Similar to [LlGA02] the first part of this study working with the KIT lecture translator system intends to evaluate the possible improvements of a system by unsupervised acoustic model training in dependency of the amount of training data. The same same basic conditions, that no closely related texts are available for any kind of supervision, are shared. The overall training conditions can be compared to [ZaTCB98] where new data for retraining come from the same speaker, channel and related conversation topics.

Like [GHSN07], the second part of this study utilizes a system trained on EPPS data as starting point and investigates the impact of different confidence score thresholds. Further, the effects of multiple iterations of batch training are observed, similar to [WN05].

Similar to [GHSN07, KW99], use of state confidence scores on word level is made. As a pre-processing step to unsupervised training, automatic transcriptions are filtered by using word posterior confidence scores for thresholding. Following the implications of [LZM12], data scored with low confidences is added to the training, but unlike in other work, word-based weighting is applied in order to compensate for errors, as it was done by [GB08] for acoustic model adaptation. The assumption is that erroneous data is helpful to improve system generalization. Unlike other work, e.g., [FSGL11], it has been refrained from a lattice-based approach.

4. Iterative Incremental Training for Speaker Adaptation

The iterative unsupervised training runs as described in the following experimental set-up were done in an incremental fashion, that means during each training iteration step, new and previously unseen data was added to the pool of training samples. The purpose of these experiments was to find out, in which way an incremental unsupervised training approach could improve the performance of a given system. The baseline system used for the initial transcription runs and re-training is trained speaker-independently. The unsupervised training data is in-domain, i.e., the existing system already performs reasonably well, yet too poor for generating reliable hypotheses for potential end-users demanding a certain accuracy given the fields of application such as real-time subtitling. The initially speaker-independent system will be pushed towards a speaker dependent recognizer by re-training on unsupervised training data of a single speaker, thus a final system ought to perform best for data of the same source. In order to adapt to the previously unseen, speaker dependent data, we adapted the models by a single Viterbi training iteration using the data of a training samples chunk of pre-defined size. It was the goal in particular to evaluate the minimal amount of training samples necessary for observing an increase in performance after unsupervised training. Further, it was an objective to investigate the effectiveness of re-transcribing the training data for a subsequent training iteration by utilizing the currently most actual system, instead of using the un-adapted baseline for all transcription runs.

4.1 Databases

The experiments described in this chapter were conducted with the help of the KIT Lecture Corpus for Speech Translation [SKM+12]. The corpus consists of recorded scientific lectures that were held at the Karlsruhe Institute of Technology (KIT) in German language. All available data, that means the data for the supervised training of a baseline system, the data chunks for adaptive Viterbi re-training, as well as the test data for the evaluation of all experiments belong to the same general domain. The utilized lecture recordings almost exclusively cover topics from the information technology domain, with categories such as “cognitive systems”, “speech processing”, “machine translation”, “systems architecture” or “formal systems”. They are completed by a small amount of off-topic lectures and some recordings of ceremonial talks. All audio recordings were conducted in a six-year period from 2006 to 2012.

The recordings were conducted by trained student part timers [SKM+12]. After storing each channel at 48 kHz and a resolution of 24 bit, each lecture was post-processed by normalizing and down-sampling the signal to 16 kHz with a resolution of 16 bit, which conforms the standard input for all utilized speech recognition systems. Additional information such as speaker identity, time stamps and characteristics of the lecture were documented and stored.

All of the data has been transcribed manually. The transcriptions were again created by students which received linguistic training. The transcribers followed the transcription guidelines as explicated in [Bur97, SKM+12]. The transcription process as a whole is performed in three stages: Stage one delivers a segmentation and initial transcriptions. Stage two, performed by a second transcriber, enhances the transcriptions. Stage three is a spell and sanity check of the output of stage two. Each transcription of a recording is organized in turns, whereas a new turn begins at the beginning of a new sentence or after a position in the audio signal which encompasses more than 300 ms of silence.

4.1.1 Training Data

Table 4.1 lists the details of the training sample database. From the full amount of data, about 7.6% remained unused during training by applying a skip list of utterances not to be used due to overlap with the evaluation set or too poor quality of audio or transcription material. Each database entry basically consists of an utterance ID, a generic speaker ID, the time span in the original audio signal, an audio file path and file ID as well as the manually created transcription of what was spoken.

Data	#files	#spk (real)	#spk (gen.)	#utt	dur	dur/utt
Scientific	123	23	788	109165	87.5 h	2.89 s
Off-topic	16	5	81	11008	10 h	3.27 s
Ceremonial	8	1	31	3950	2.5 h	2.31 s
Sum	147	29	900	124123	100 h	2.91 s
Sum w/o skips	-	-	-	115842	94 h	2.92 s

Table 4.1: Statistics of the speech data used for the baseline acoustic model training, including the total number of recordings (*#files*), the amount of distinctive real speakers (*#spk (real)*) and generic speakers (*#spk (gen.)*), the total number of utterances (*#utt*), the overall speech duration (*dur*), and average speech duration per utterance (*dur/utt*).

The training data was constrained to two distinct speakers, in order to compare the experimental results given two pre-defined sets of available unsupervised training recordings. The total amount of data is 23 hours, corresponding to 26991 utterances. Both data sets are stored in a separate database for reason of convenience, being of the same structure as explicated above. Major differences concern the transcriptions provided to each training utterance: Here, the transcriptions are generated automatically, and additionally comprise word-based confidence scores, i.e., word posterior probabilities being generated by the decoder that was utilized for computing the initial transcriptions.

Due to the limited amount of available material for the adaptive Viterbi training runs it has been decided to split up both sets of recordings into smaller data chunks, resulting in five chunks for $speaker_A$ and seven chunks for $speaker_W$.

It may be noteworthy that data of both speakers is already contained in the training set for the speaker-independent baseline system. However, the data disjointedness is maintained.

Data	#files	#utt	dur	dur/utt
$speaker_A$	11	5193	7 h	4.76 s
$speaker_W$	19	21798	16 h	2.64 s

Table 4.2: Statistics of the speech data used for re-training the speaker-independent baseline system, including the total number of recordings (*#files*), the total number of utterances (*#utt*), the overall speech duration (*dur*) and average speech duration per utterance (*dur/utt*).

4.1.2 Test Data

For the intended experiments two distinct test sets were generated by using held-out data of both speakers, whose data is used for the adaptive Viterbi re-training. Where for $speaker_W$ a sufficient amount of data was available, it was only a minimal amount of no more than 0.5 hours of test material accessible for $speaker_A$.

Data	#files	#utt	dur	dur/utt
$speaker_A$	1	78	28 min	22.19 s
$speaker_W$	4	324	125 min	23.25 s

Table 4.3: Statistics of the speech data used for testing the recognizer performances, including the total number of recordings (*#files*), the total number of utterances (*#utt*), the overall speech duration (*dur*) and average speech duration per utterance (*dur/utt*).

4.2 KIT Lecture Translator Baseline System

The baseline system used for the following experiments is derived from an intermediate lecture translator system, developed at KIT. The speaker-independent system was taken from the inauguration of the lecture translation system at KIT on June 11th 2012 [CFH+12]. It was trained on all available training data from the KIT lectures corpus and has been adapted to the individual lecturers. Acoustic models are fully continuous and were trained in a supervised fashion. These models are combined with a language model specifically tailored to the lecture domain.

4.2.1 Feature Extraction

The pre-processor is a variation of the ones used in [SKN11] by the KIT systems participating in the Quaero[1] 2010 speech-to-text evaluation campaign.

The feature extraction is based on the warped minimum variance distortion-less response (MVDR). It has been shown that Mel frequency cepstral coefficients (MFCC) or perceptual linear prediction (PLP) coefficients are outperformed by warped MVDR cepstral coefficients [WM05a] in noisy conditions, which replaces the traditional computation of Fourier transformation by a warped MVDR spectral envelope [WM05b]. The latter is a time domain technique to estimate an all-pole model using a warped short time frequency axis such as the Mel scale. The use of the MVDR eliminates the overemphasis of harmonic peaks typically seen in medium and high pitched voiced speech when spectral estimation

[1] http://www.quaero.org

is based on linear prediction. During training and decoding, spectral features are obtained every 10 ms. A model order of 22 without any filter-bank is used, as the warped MVDR already provides the properties of the Mel-filter-bank, in detail warping to the Mel-frequency and smoothing. The advantage of this approach over the use of a higher model order and a linear-filter-bank for dimensionality reduction is an increase in resolution in low frequency regions which cannot be attained with traditionally used Mel-filter-banks. Furthermore, with the MVDR we apply an unequal modelling of spectral peaks and valleys that improves noise robustness, due to the fact that noise is mainly present in low energy regions. Further, vocal tract length normalization (VTLN) [ZW97] is applied, in the MVDR case this is done in the warped frequency domain.

The front-end selects the 20 lowest cepstral coefficients as features. The mean and variance of the cepstral coefficients are normalized on a per-utterance basis [SKN11]. For the incorporation of temporal information seven adjacent frames are combined into one single 300 dimensional feature vector via frame stacking. A linear discriminant analysis (LDA) reduces the vectors to 40 dimensions and maximizes the discrimination of the classes.

4.2.2 Acoustic Modelling

In order to simplify the training process and moreover to speed up, it has been made use of already existing labels – or fixed state alignments – that were written with an earlier system. The context-dependent models were generated with the help of an entropy-based clustering technique. First, mixture weights for all polyphone models were trained. Afterwards, a classification-and-regression-tree (CART) based top-down clustering was applied by deploying phoneme based questions regarding context and position. The polyphone models were split up gradually until the pre-defined amount of 4000 context-dependent models was reached. All models are context-dependent quinphones with a standard three-state left-to-right HMM topology with self loops, but without skip states. The model for silence poses an exception, as it is comprised of four states. All transition probabilities remain fixed throughout the whole training process.

After clustering the models use 4000 distributions over 4000 codebooks. This fully continuous system was further trained by using an incremental splitting of Gaussians training (MAS) [KFN98], followed by optimal feature space training (OFS) which is a variant of *semi-tied covariance* (STC) [Gal99] training using one global transformation matrix. Each model then has a variable amount of Gaussians, up to 128. The models are further refined by 2 iterations of Viterbi training. The Viterbi training is performed in order to compensate for eventually misaligned labels. The phoneme set consists of 39 phonemes, completed by 9 additional tags for noises such as fillers, breath, laughter, general human and non-human noise, and silence. All noise phonemes are modelled context independently, in contrast to the genuine phonemes.

The full training cycle looks as follows:

1. Label writing
2. Linear discriminant analysis (LDA)
3. Sample extraction
4. Merge and split (MAS) training (incremental growing of Gaussians)
5. Optimal feature space training (OFS)
6. Viterbi training (2 Iterations)

4.2.3 Dictionary & Language Model

Depending on the purpose of decoding – testing or automatic transcription of training data – different dictionaries were applied. For automatic transcription a slightly modified version of the training dictionary was used, where noise tags were added to the vocabulary as pronunciation variants of the optional word $. This has been done due to the use of several transcription pre-processing techniques, which are described in the following sections. This speaker non-specific dictionary contains 283197 unique words, and an overall inventory of 452438 elements by taking into account all pronunciation variants. For running the evaluation runs on the test sets, two distinct dictionaries, each oriented to one of the test speaker's expected lecture topics were used.

Word counts	Dictionaries		
	$dict_{SI}$	$dict_A$	$dict_W$
unique	283197	293494	194774
+ variants	452438	388571	271738

Table 4.4: Statistics of the dictionaries used for all decoding tasks, including the total number of unique words (*#words (unique)*) and total number of all words (*#words (+ variants)*). $dict_{SI}$ is used for the automatic transcription runs, $dict_A$ and $dict_W$ are utilized for the test runs.

The same distinctions have to be made for the language models. For automatic transcription, a speaker non-specific 4-gram language model, which was combined with the baseline system in previous experiments was used. It has been trained on texts from various sources like web dumps, newspapers and acoustic transcripts of sources such as broadcast news, talks and speeches. In total, 28 text corpora were used, which range in size from about 5 MByte to over 6 GByte [CFH+12].

For evaluation purposes with regard to real-life applications of the trained systems using the experimental approach, test runs were performed by making use of two specific language models, each particularly tailored to one of the two target speaker's expected lecture topics. The idea is to simulate an in-the-field usage of these techniques and thus to obtain approximative estimates for feasible performance gains. In order to achieve these language models, a background LM was mixed with exactly one other model, trained on specific text sources that were crawled from the web and whose contents are close to the expected topics of the lectures the respective speaker holds. For $speaker_A$, the main language model was mixed with a model trained on a text sources comprising topics such as computer engineering, whereas for $speaker_W$ the additional model for mixing was built on a corpus comprising subjects such as "cognitive systems" and "anthropomatics"[2].

All applied language models are N-gram LMs with highest $N = 4$ and were built using the SRI Language Modelling Toolkit [Sto02]. Besides the application of Good-Turing discounting, for $N < 4$ Chen and Goodman's modified Kneser-Ney discounting is applied, and for $N = 4$ Witten-Bell discounting is used.

[2]The term "anthropomatics" was coined by a Karlsruhe informatics professor ten years ago as the science of symbiosis between human and humanoid and refers to a research field, which focuses on human-centred environments, with an aim to researching and developing people-friendly systems using informatics [KIT].

N-grams	Language Models		
	LM_{SI}	LM_A	LM_W
1-grams	293181	296382	296382
2-grams	75823147	85846441	81549098
3-grams	121886615	137806193	132400531
4-grams	190019036	216648814	206998851

Table 4.5: Statistics of the language models applied during decoding, including the n-gram counts for n-grams 1 to 4 (*n-grams*). LM_{SI} is used for the automatic transcription runs, LM_A and LM_W are utilized for the test runs.

4.3 Decoding

Classification, i.e., recognition is done on scores produced by the decoder, which likewise incorporate an acoustic score and a language model score, given an utterance to get decoded. The standard procedure to achieve the most likely word sequence as hypothesis to a test utterance during the decoding step is to tune the parameters of the decoder that regulate the impact of the language model on the tokenization process (see 2.5).

For reason of comparability of the experimental systems built during the complete development phase the decision was made to refrain from decoder parameter tuning. Thus, the application of the baseline recognizer for automatic transcription runs on the unsupervised training data listed in Table 4.1 is done by straightforwardly using given parameters, which were used by the speaker-independent baseline acoustic models in combination with the general language model LM_{SI} during another project.

4.4 Training

The standard unsupervised training cycle applied during these experiments consists of four concrete steps: First, the unsupervised training data must be transcribed by the baseline system. The transcriptions consist of a 1-best hypothesis per training utterance, and corresponding word-based confidence scores which are – to be more precise – posterior word probabilities: With the help of word lattices, very word in a transcription is annotated with a posterior probability, which serves as a measure of confidence. Next, a training database is built, combining the a priori available informations about the audio material such as the segmentation, and the newly produced transcriptions along with the confidence scores. Each word of the transcription is followed by it's corresponding score, so that a single database entry would look as follows:

{*TEXT* {*Der 0.367924*} {*Multiplexer 0.753735*} {*hatte 0.822427*} {*einen 0.44854*} {*Ausgang 0.867014*} {*und 0.913126*} {*und 0.681813*} {*mehrere 0.901334*} {*Eingänge 0.954381*} {*hier 0.828877*} {*haben 0.979821*} {*wir 1*} {*nur 0.984496*} {*einen 0.963042*} {*Eingang 0.999559*} {*und 0.937915*} {*mehrere 1*} {*Ausgänge 1.00002*}}

Table 4.6: Exemplary excerpt of a database entry without the utterance details. The *TEXT* field features word based and confidence annotated transcriptions.

As can be seen, posterior probabilities $score(w_i)$ from time to time can be $score(w_i) > 1$. However, before processing the scores during training, they are capped at a maximum

of 1. Third step of the training procedure is to compute a Viterbi alignment for every training utterance, and store them as labels. This is done because of the fact, that upon one training data set a multitude of systems is trained with modifications such as varying amounts of data. This way, a Viterbi alignment has to be undergone only once for each database and as a consequence can be used for multiple system trainings, leading to a saving of time by doing a training along labels, instead of computing Viterbi alignments on-the-fly for every training run. The last step is the training itself, which is performed along the previously stored labels.

4.5 Testing

Performance measurement is done on the two speaker specific development sets as explicated in Table 4.3. Given this data and already existing language models, each adapted to one of the speakers, decoding for evaluation purposes of the experimental systems is performed in a speaker-dependent fashion. For performance measurement, a simple word error rate (WER) was calculated for each utterance hypothesis and reference pair (see Section 2.5 of Chapter 2). For the scoring of the automatically generated hypotheses a number of normalization steps are undergone in order to match the decoder output to the references. The decoding of the test set was done with an off-line set-up that is similar to the way decoding is performed in the lecture translation system, i.e., without any lattice re-scoring, in real time, and with incremental VTLN and feature space constrained MLLR [Gal97].

4.6 Experimental Results

The goal of the experiments is to simulate, how an ASR system would work when being used in the simultaneous lecture translation system deployed in KIT's lecture halls. When the system is initially applied to a new lecturer, only a generic, speaker-independent acoustic model will be available. With every new lecture, new audio data will become available, but without any manual transcriptions. Utilizing the new audio material by an unsupervised training framework aims at incrementally transforming the initially speaker-independent acoustic model into a speaker-dependent model tailored to a specific lecturer.

For comparison, the baseline system was adapted to both target speakers by applying a Viterbi training iteration each, given the respective speaker-dependent training data and the manually produced transcriptions. These results serve as expected upper bound for the proposed approach of unsupervised training. Besides the performance of the system adapted this way, Table 4.7 shows the WER of the unmodified baseline, applied on both test sets, giving A lower limit for the performance of the speaker-dependent models that were trained on unsupervised data. As can be seen, there is a significant gap between the performance of both speakers. The system is able to produce much better hypotheses on the data of $speaker_W$. This seems natural, as the share of data of this particular speaker adds up to approximately 17% in terms of overall duration and 19% of all utterances respectively, whereas data of $speaker_A$ sums up to about 7% in length and 4% of the set of utterances.

In order to adapt the speaker-independent acoustic models AM_{SI} to one of the test speakers each, iterative incremental training was applied on the baseline system. Unsupervised training data was automatically transcribed with help of the baseline recognizer using the aforementioned acoustic models combined with a general, speaker non-specific language model LM_{SI} and pre-set, non-optimized decoder parameters. All available speaker-specific data was transcribed at once, that means the training iterations were run on data solely

System	WER in %			
	Transcription		Test	
	$speaker_A$	$speaker_W$	$speaker_A$	$speaker_W$
baseline	23.2	19.6	19.7	16.4
+ sup	-	-	17.3	15.6

Table 4.7: Performance of the baseline system (*baseline*) and two supervisedly re-trained reference systems (*+ sup*) in word error rate (WER), evaluated on test sets for $speaker_A$ and $speaker_W$.

Training	Duration	WER	Training	Duration	WER
$speaker_A$			$speaker_W$		
speaker independent	94h	19.7%	speaker independent	94h	16.4%
	2.29h	22.1%		2.28h	18.5%
	3.05h	21.3%		4.57h	18.2%
	3.81h	20.1%		6.86h	17.8%
+ unsup	4.57h	18.9%	+ unsup	9.15h	17.5%
	6.87h	18.7%		11.44h	17.4%
				13.73h	17.4%
				16.02h	17.3%
+ sup	6.87h	17.3%	+ sup	16.02h	15.6%

Table 4.8: Performance of the re-trained intermediate systems (*+ unsup*), plotted in dependency of the amount of unsupervised training data and measured in terms of word error rate (WER). The development systems are compared to the *speaker independent* baseline and two supervisedly re-trained reference systems (*+ sup*). All systems were evaluated on separate test sets for $speaker_A$ and $speaker_W$, respectively.

transcribed using AM_{SI}. No other acoustic models from intermediate states after various iterations of unsupervised training were used for re-transcribing the training material.

The intention was to evaluate the amount of training data to be needed for getting an actual improvement in recognizer performance, as well as a direct comparison of supervised and unsupervised training on the exact same training data. We therefore divided our training data for $speaker_A$ into five chunks (2.29h, 3.05h, 3.81h, 4.57h, 6.87h) and for $speaker_W$ into seven chunks (2.28h, 4.57h, 6.86h, 9.15h, 11.44h, 13.73h, 16.02h), and trained models for increasing amounts of training data. The intermediate systems display the improvements in dependency of the amount of data used. Table 4.8 displays the results of developing both speaker adapted systems by straightforward incremental training.

It is noteworthy that with using only little data for re-training the recognition accuracy decreases so that the resulting system performs worse than the unmodified baseline. Moreover, although there is improvement observable when using an increasing amount of additional training data, $speaker_W$ is not able to recover from this deterioration, whereas for $speaker_A$ an amount of approximately 4 hours of newly transcribed data is already sufficient to beat the baseline. If using all available data of this speaker, a 1% absolute improvement to the starting point is achievable, albeit the increase obtainable by supervised training with 2.4% absolute remains unmatched. One reason for the harmfulness of unsupervised training for the second speaker might be the fact that the baseline did already see a significant amount of speaker specific in-domain data, which narrows the beneficial outcomes of further adaptive training. Especially imperfect or even erroneous training data is potentially malicious to the resulting system. This assumption is underpinned by the observations made for this particular experiment on $speaker_W$: Where it is still possible to fairly improve the system performance by supervised training with manually and carefully annotated training data, the automatically transcribed material apparently holds too few informations that could lead to a beneficial training and on the contrary seem to be defective in a manner that leads to deteriorated systems.

4.6.1 Transcription pre-processing

A first set of experiments aimed at examining how to treat pronunciation variants and noise models in training in order to potentially improve the performance of the training process.

The automatic transcriptions used for the training described above are filtered decoder outputs. Filtering is done in a way that the resulting transcriptions have the characteristics of plain text, i.e., they are cleaned of noise tags and filler tags and informations about occurring silence. The intention of further experiments was to elaborate, whether additional information carried by the transcriptions is beneficial for the training process. The training pipeline for unsupervised training in general looks as follows:

1. Automatic transcription
2. Training database generation
3. Label writing
4. Viterbi re-training (1 iteration)

JANUS allows modifications on the generated hypothesis during label writing. Within the process of writing labels, the decoder chooses the most probable variant of a recognized word and autonomously inserts optional words – or "optWords" – which are marked as $, into the hypothesis for the currently processed utterance. It must be admitted, that between two neighbouring words there can not be placed more than one optWord at a

time. This obviously diminishes the dynamics of label writing if working on plain text as transcriptions. One option to circumvent these restrictions is to define noise and filler words and even silence tags which are comprised of pronunciations, i.e., strings of phonemes with a length $l_{pron} > 1$. Another option is to use hypotheses which contain noise, filler and silence annotations instead of mere plain text. In this way it may happen during label writing that an optWord $\$_{ins}$ is inserted between two words w_i and w_j, where

$$w_i \vee w_j \in \mathfrak{N} \tag{4.1}$$

with $\mathfrak{N}$ being the quantity of all noise, filler and silence tags. By Default, inserting optWords and allowing for the automatic selection of the most probable pronunciation variant during label writing is enabled. However, both may also be disabled, which would result in not allowing for later insertions of optWords for *-optWords 0*, and for *-variants 0* the pronunciation selection during the build-up of the most probable HMM each tree for label writing would be disabled. The latter leads to the effect that the decoder which is used for automatic transcription has sole power of decision regarding the pronunciation of each word in the hypothesis to build. This may be a desired feature if is intended to trust the transcription system to a greater extent. For further investigation of the efficiency of the unsupervised training via adaptive Viterbi re-training various degrees of transcription preprocessing were to evaluate in regard to their influence on the overall system performance after training. Labels were written for four different types of transcriptions, where the types differ in the rigidity of filtering. It is to differentiate between the categories *filtered*, *baseAll*, *baseWords* and *recognition*.

recognition The annotation of noise words and pronunciation variants are taken as is from the recognition output and is not altered by the Viterbi training.

baseAll Pronunciation variants in the recognizer output are mapped to their base form, and the pronunciation variant used during training are picked by the Viterbi alignment in training. Wherever a noised word was hypothesized, all other noised words are inserted as alternative paths, and the actual noise word used for training is again picked by the Viterbi alignment

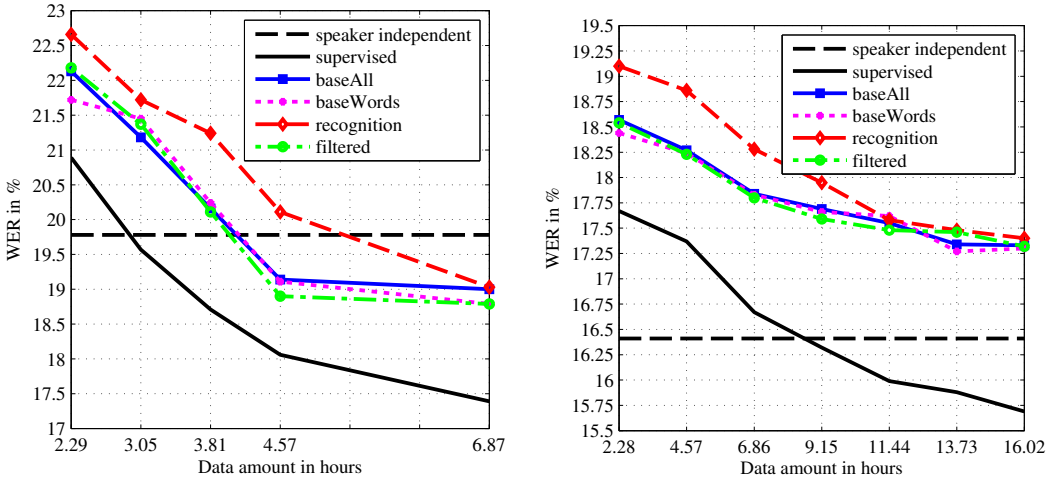
baseWords Only regular words are mapped to their base form and their pronunciation variants are inserted as alternative paths. The hypothesized noise words are left as recognized.

filtered All regular words are mapped to their base form, their pronunciation variants are inserted as alternative paths; all recognized noise words are removed, and instead inserted as alternative paths between regular words.

Filtering	Example
filtered	*also wenn wir hier*
baseAll	*$ also wenn wir $ hier*
baseWords	*$(<noise>) also wenn wir $(<breath>) hier*
recognition	*$(<noise>) also(1) wenn(1) wir(6) $(<breath>) hier*

Table 4.9: Illustration of the different filtering categories applied as automatic transcription pre-processing. *filtered* corresponds to plain text, *baseAll* contains general noise tags, *baseWords* is enhanced by annotations of pronunciation variants, and *recognition* resembles the unprocessed decoder output.

Transcriptions of the category filtered were used for training as described in Table 4.9. The same training schemes have been applied using the three new types of transcriptions: First, a new database for each transcription type was generated. Then, labels were written, allowing for the insertion of optWords and selection of the most probable pronunciation variants of each word. For *baseAll* the effect is, that multiple noise, filler and silence tags may be neighbouring. This is also the case for *baseWords*, whereas now the word pronunciations remain fixed, i.e., the decoders' choice will be used during label writing, without renewed decision-making during the construction of the search tree. For the detailed, unmodified *recognition* outputs, variants and optWords are deactivated: The decoder solely decides how the final transcriptions will look like. No further insertion of non-word tags takes place, nor is there a possible re-selection of pronunciation variants. The general assumption is: With more detail in the transcriptions, the label writing process may react more dynamically to difficult or noisy parts. Figure 4.1 shows the resulting performance of the four configurations on the test speakers for increasing amounts of training data.



(a) $speaker_A$ (b) $speaker_W$

Figure 4.1: Performance of speaker-dependent system re-training after applying different filtering methods for transcription pre-processing. Performance is plotted in dependency of the amount of unsupervised training data and is measured in terms of word error rate (WER). The development systems are compared to the *speaker independent* baseline and two supervisedly re-trained reference systems (*supervised*). All systems were evaluated on separate test sets for $speaker_A$ and $speaker_W$, respectively.

It can be seen that the training on the exact transcriptions from the recognition run (*recognition*) improves slower with increasingly available training data than the other three methods. It also performs worse when taking all available training material. Thus, using transcriptions as provided by the recognizer is not necessarily the best procedure for unsupervised training. Instead it turns out that it is of advantage to let the Viterbi algorithm during training decide which pronunciations to use for the words during training, as well as, where to insert which noise words. The decision is done by inserting pronunciation variants as alternative paths, in addition to the base forms of the words. Noise words are inserted as alternative paths between regular words. Even though *baseWords* and *baseAll* perform about equally well, the latter seems to be more robust in most cases, as the speed of improvement is slightly higher, being more stable at the same time, i.e., resulting in a smoother performance curve as a function of the amount of training data. Comparing

baseAll and *filtered*, it is interesting to see that the former seems to be beneficial on less data, whereas the latter is better when more data becomes available.

For $speaker_A$, all speaker-dependent models perform better than the speaker-independent models when at least 3.8 hours of training data are available, with the exception of the models that were trained on the exact transcriptions from the recognition run. As was expected, training on exact, i.e., manual transcriptions of the training data outperforms the unsupervised training.

Keeping the decoder-selected pronunciation variants fixed seems to harm the generation of labels for further training, thus contributing to a degradation of recognition performance. For the following sections results are unrolled for systems trained on the transcription mode *baseAll* only, as this mode turned out to work most reliably for all subsequent experiments.

4.6.2 Confidence Weighting & Thresholding

The most common methods for processing unreliable, erroneous transcriptions in unsupervised acoustic model training are based on lattice confidence measures at word or state level [FSGL11]. The idea of thresholding is to use only segments for training if the corresponding confidence is higher than a pre-set threshold. This error filtering again is not free of making errors, as the confidences are estimates with limited reliability. First and foremost, the strictness of thresholding is essential: If a too small threshold is chosen, potentially incorrect words may slip in the training, whereas for a too high threshold potentially reliable word hypotheses are dismissed, thus wasted. Moreover, in the latter case the system does not learn anything new. Both extremes render the training process harmful. In these experiments the word level posterior probabilities obtained during decoding were utilized. The confidences were applied in three ways:

weighted Sets the gamma probabilities of the states of a word during Viterbi training to the posteriori probability of the respective word

thresh Removes words with a confidence below a certain threshold from training

weighted+thresh Combines both methods

Section 2.5 of Chapter 2 gave an insight into the scoring mechanism of JANUS and its decoder. The output of the automatic transcription is lattice-based, representing alternative hypothesis, which is used to estimate word-level confidences. [Kea97] has shown that the information stored in lattices is sufficient for building high-accuracy word confidence taggers.

A word lattice as produced by JANUS is a directed graph, where words are represented by nodes, and links represent possible neighbouring words, according to the different hypotheses. The word based acoustic scores are stored in the links between nodes, rather than in the nodes themselves. If the nodes are viewed at as HMM states and the links as transitions between the states, one can basically apply the forward-backward algorithm to estimate the probability of each link, given the word lattice. The probabilities can be interpreted as word based a-posteriori probabilities [Kea97].

Training by JANUS is applied utterance-wise. For each utterance in the training set, a Viterbi training along labels is performed. The labels were previously computed and stored, and reloaded for the actual training. Prior to the update step for the codebook and distribution weights, the previously computed and stored confidence measures are applied for filtering the training data. Given a Viterbi path through a built up HMM, a weighting factor *gamma* can be assigned to every frame. By default, this weighting factor is set to 1. If set to 0, parts of a path are excluded from training. By setting *gamma*

to 0, a frame will effectively be discarded during training, or more precisely during the update step for the codebook and distribution weights. A $gamma \neq 0$ results in a weighted contribution of this particular frame to the training, where a weighting factor in general may also be considerably higher than 1 or a negative value. For this study, the *gamma* value corresponds to the posterior probability of the word w to which a frame fr_i^w belongs. Thus, the weighting factors of all frames are restricted to $gamma \in [0, 1]$.

First, the word based confidences are loaded from the confidence annotated training database. Then, the HMM for training is built, but instead of computing the path for training via forced alignment, the pre-computed and reloaded label informations are used. The subsequent step is to match the loaded confidence scores to the training HMM: The path used for training may look different from the one built for label writing in a way that there might be inserted more optional words $. The confidence measures are word based. During the update step, however, gamma weighting is done frame-wise, i.e., thresholding as well as weighting training data is processed on frame level. This makes it necessary to apply the word-based confidence measures to a sub-word level. For weighting, each fr_i^w belonging to w is assigned the confidence score $conf(w) \in [0, 1]$ by setting the parameter $gamma(fr_i^w)$. If for a certain word $\tilde{w}$, e.g., a newly inserted optional word $ there is no confidence $conf(\tilde{w})$, the gamma value of the respective frames is set to 1. For thresholding with a threshold t, if for a $\hat{w}$ the $conf(\hat{w}) < t$, the gamma value of the respective frames is set to 0:

$$gamma(fr_i^w) = \begin{cases} 0 & \text{if } \exists t \wedge \exists conf(w) \wedge conf(w) < t \\ 1 & \text{if } \nexists conf(w) \\ conf(w) & \text{else} \end{cases} \tag{4.2}$$

After accumulating all training statistics in a so called senone set using the state probabilities [FMS], the accumulators are saved for joining parallel processes and a potential reuse, then reloaded and used for parameter updating. The default configuration is to do a maximum-likelihood update [FMS].

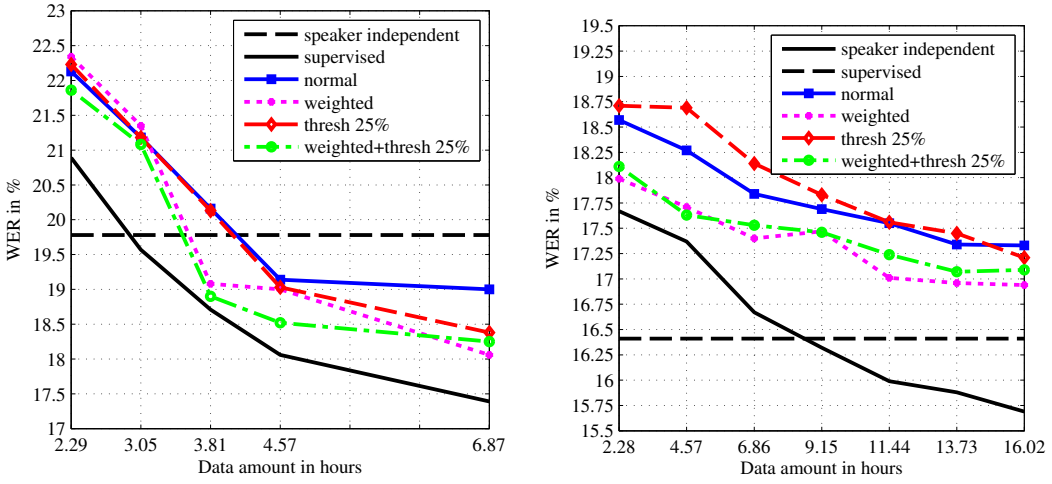
Figure 4.2 shows the result of weighting with confidences as well as applying a threshold for both test speakers, either as exclusive techniques or in combination. According to the depicted curves it can be assumed that using word-based weighting harms the reliability of the training data, as the speed of WER reduction with respect to the amount of training data is becoming slower, compared to unweighted training. On the other hand, when all available training data is used, weighting with the confidences gives the best performance. Using this method leads to up to 1% absolute error reduction, compared to systems not utilizing any confidence measures. Whereas weighting leads to less smooth performance curves, the application of an experimentally set threshold $t = 25$ instead leads to a much smoother curve resembling the one achieved without using confidence measures, with the difference of a better convergence and final performance. The combination of weighting and thresholding leads to a significant gain in terms of sped up convergence while maintaining a smooth performance curve. This leads to the assumption that parts of a training utterance with low confidence are potentially harmful even with lowered weights during training, as thresholding helps to smoothen the gain in performance again after applying confidence based weight factors, whereas thresholding alone seems to a generally less erroneous, but not optimal training.

As can be seen, thresholding only gives advantages over not using confidences at all, in cases where sufficient amounts of training data are available. However, the threshold should be chosen with care, as using too high a threshold may discard too much data, and as a result the performance degrades. On the other hand, setting the threshold too low, less erroneous

Algorithm 1 Single training iteration using confidence measures

Require: AM
Ensure: AM'

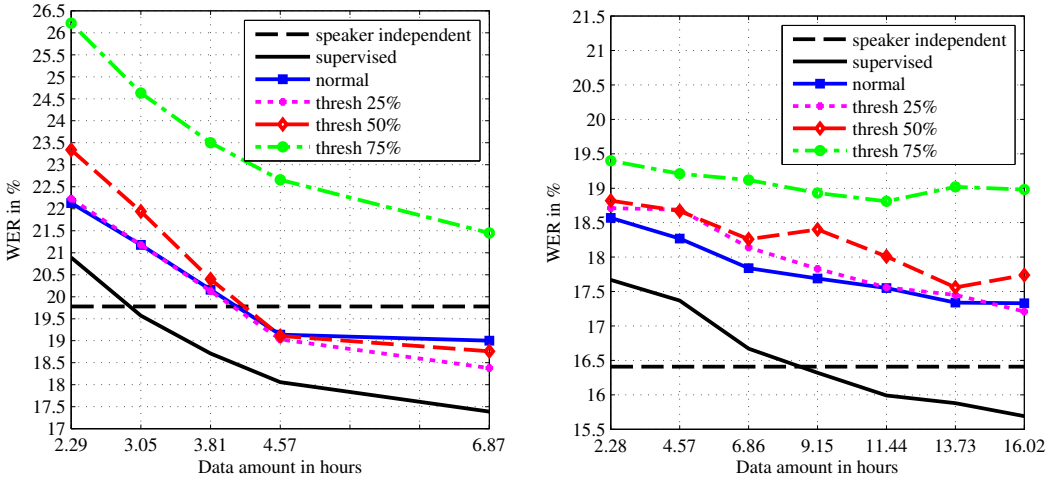
for all i with $utterance_i \in DB$ **do**
  $T_i \leftarrow$ transcript loaded from DB
  $C_i \leftarrow$ word based confidences loaded from DB
  $L_i \leftarrow$ labels loaded from disc
  Compute $path_i$ through training HMM with L_i, T_i
  for all k with $frame_k \in path_i$ **do**
    Get word identity $word_k$ of frame k
    if $\exists threshold \wedge \exists C_i(word_k) \wedge C_i(word_k) < threshold$ **then**
      Set gamma of $path_i^k$ to 0
    else if $\nexists C_i(word_k)$ **then**
      Set gamma of $path_i^k$ to 1
    else
      Set gamma of $path_i^k$ to $C_i(word_k)$
    end if
  end for
  Accumulate training data
end for
Update models $AM \rightarrow AM'$



(a) $speaker_A$ (b) $speaker_W$

Figure 4.2: Performance of speaker-dependent system re-training when applying confidence measures for weighting and/or thresholding. Performance is plotted in dependency of the amount of unsupervised training data and is measured in terms of word error rate (WER). The development systems are compared to the *speaker independent* baseline and two supervisedly re-trained reference systems (*supervised*). All systems were evaluated on separate test sets for $speaker_A$ and $speaker_W$, respectively.

data is exempt from training, rendering the training potentially harmful to the models. Figure 4.3 shows the impact of different threshold values exemplarily for $speaker_A$. As can be seen, a threshold of 25% still does not lead to a significant degradation of performance after training on even a very small amount of data. Moreover, with increasing amounts of training data a threshold of 25% still can be beneficial for filtering out supposedly erroneous parts. However, with application of a stricter threshold above this limit, considerably less data remains for training, thus leading to significantly worse systems: A stricter cut-off is of no use and starts to break the training process.



(a) $speaker_A$ (b) $speaker_W$

Figure 4.3: Performance of speaker-dependent system re-training after applying different thresholds for confidence measures. Performance is plotted in dependency of the amount of unsupervised training data and is measured in terms of word error rate (WER). The development systems are compared to the *speaker independent* baseline and two supervisedly re-trained reference systems (*supervised*). All systems were evaluated on separate test sets for $speaker_A$ and $speaker_W$, respectively.

The impact of various thresholds has been analysed by counting the words that were dismissed from training due to a too low confidence score. Additionally, a sorted list of the words that fell below the threshold most often was computed. As expected, given the method of confidence score computation explained in this section, especially short words with high risk of confusion were discarded. In the case of German this particularly affects conjunctions (“und”, “dass”), pronouns (“wir”, “das”, “es”), prepositions (“in”, “im”) and other short words like auxiliary verbs (“ist”, “sind”) and interjections (“ja”). Another considerable proportion contribute the filtered noise and filler tags. Where the distinction was made between different noises, which was only given for the unfiltered *recognition* results used as transcriptions, the tags for breath noises and filled pauses, especially vowel-nasal combinations (“ähm”) were filtered by their frequently low confidences. Table 4.10 shows the impact of the thresholds that have been experimented with, on consideration of the full training set comprising both speakers. Regarding the transcriptions, the distinction has to be made whether noise tags – either mapped to $ or as is – are contained or excluded, as this has an impact on the overall accumulated word count and consequently the relation of accepted and dismissed words given a particular threshold.

Threshold	Rejection rate in %	
	uncleaned transcriptions	cleaned transcriptions
25%	13.5%	13.8%
50%	34.3%	36.3%
75%	60.1%	64.3%

Table 4.10: Rejection rate of words when applying different confidence measure thresholds. For instance, a threshold of 25% discards all words with a confidence score lower than 0.25. Uncleaned transcriptions correspond to the raw *recognition* results, i.e., the unmodified decoder output. Cleaned transcriptions correspond to the transcription pre-processing category *filtered*.

4.6.3 Light Supervision by Language Modelling

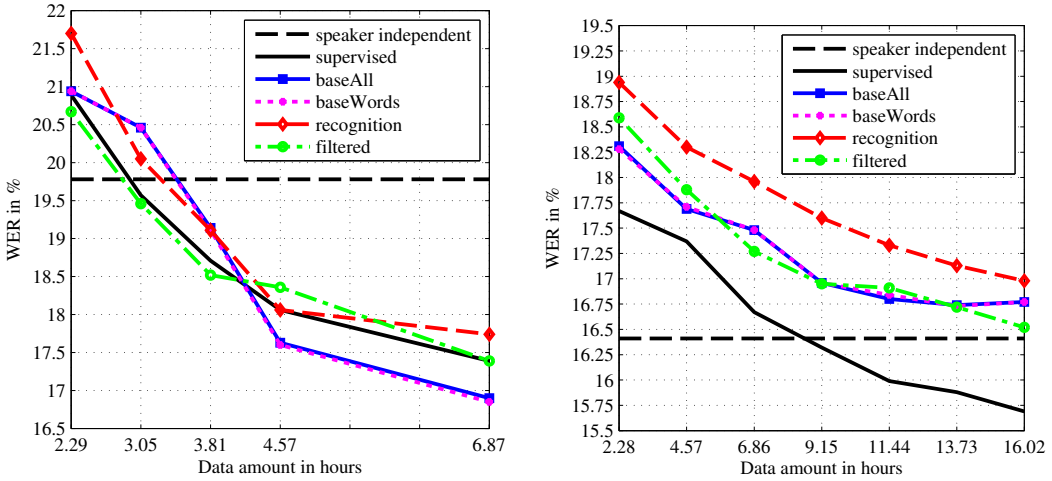
With the topic specific language models LM_A and LM_W at hand, to use for $speaker_A$ and $speaker_W$ respectively, the decision was made to experiment on light supervision by language modelling. Therefor, automatic transcription of the training data was undergone by using these language models instead of the general language model previously used. As brief reminder, both language models were tailored to the respective speaker by mixing a main language model with specific language models that were trained on text corpora whose contents are close to the expected topics of the speaker's held lectures.

After system training and testing, in general similar observations as explicated above can be made, concerning the application of transcription filtering methods: Transcriptions preprocessed by the policies *baseWords* and *baseAll* respectively show almost identical results. *Filtered* transcriptions lead to smoother performance curves, and using the unmodified *recognition* results leads to systems easily beaten by the other systems using different transcription types.

There are some differences to observe: Despite the fact that the most potent systems again are the ones using transcriptions mapped via the *baseAll* and *baseWords* rules, for $speaker_W$ a system trained on *filtered* transcriptions is capable of beating all other systems when using all available data. For $speaker_A$, the same set-up manages to outperform systems trained on smaller amounts of supervised data. On the other hand, the best system for $speaker_A$ outperformed the supervisedly re-trained reference system by 3.1% relative, using the transcriptions filtered with the *baseWords* modality. These observations lead to the assumption that the initial and manually generated transcripts for this speaker are partly unreliable due to a less careful transcription process, as missing informations concerning fillers, noise or incomplete words indicate. This way, the automatic transcriptions seem to hold an advantage. The improvements of the training results on very small data indicate transcription errors in the original transcriptions, that potentially may be corrected by automatic transcription. The increasing effectiveness of using automatic transcriptions also including noise informations, may they be mapped to base forms or not, give a hint to missing informations in the manual transcriptions which are implicitly included in the decoder output of automatic transcription runs.

Most importantly, performance of all re-trained systems increased significantly by up to 11% relative. Moreover, the best system for $speaker_A$ outperformed the supervisedly re-trained reference system by 3.1% relative, using transcriptions with words and/or non-word tokens mapped to their base forms.

The application of confidence measures for further data filtering and weighting during training brought partial improvements in terms of recognition performance when using



(a) $speaker_A$ (b) $speaker_W$

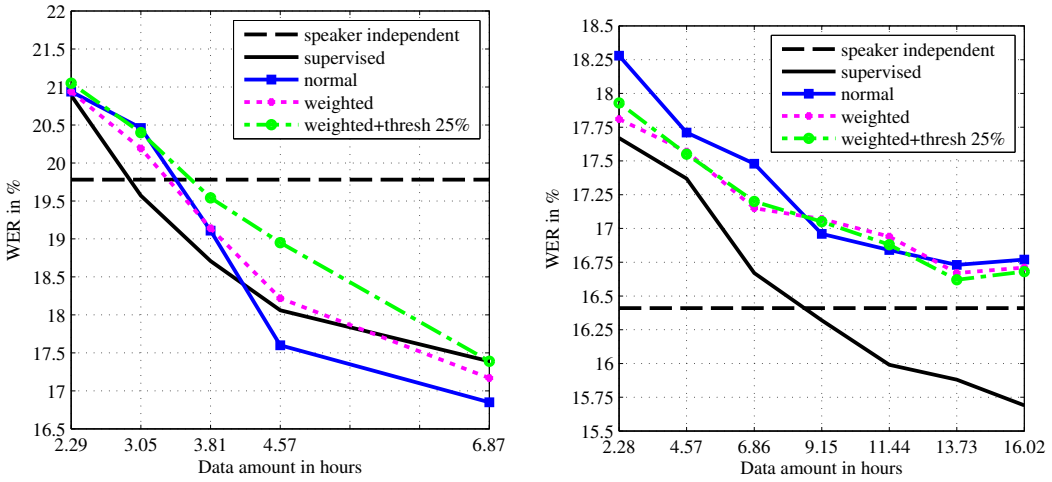
Figure 4.4: Performance of speaker-dependent system re-training when applying light supervision during automatic transcription runs. Performance is plotted in dependency of the amount of unsupervised training data and is measured in terms of word error rate (WER). The development systems are compared to the *speaker independent* baseline and two supervisedly re-trained reference systems (*supervised*). All systems were evaluated on separate test sets for $speaker_A$ and $speaker_W$, respectively.

small amounts of training data for both speakers. Regardless of the kind of transcriptions, whether filtered, mapped to base forms or raw decoder output, using the confidences for weighting resulted in the best performing systems if trained on few data. If using more, or even all available data for training, however, weighting as well as thresholding caused a drop in recognition performance for $speaker_A$, with $speaker_W$ at the same time gaining an improvement.

Compared to the introduction of confidence measures to the fully unsupervised training approach as explicated in Subsection 4.6.2 the possible improvements by application of these techniques to a framework for light supervision by language modelling are less clearly recognizable. One reason might be an increased precision of the decoding for automatic transcription, as by using language models that provide light supervision the recognition accuracy significantly increases. In terms of WER a relative performance gain of more than 17.5% for $speaker_A$ and 19.5% for $speaker_W$ on the in-domain test set listed in Subsection 4.1.2 were obtainable just by changing language models. Table 4.11 lists the improvements of the baseline system performance in terms of WER when applied on the aforementioned test set, using the topic specific language models LM_A and LM_W.

If compared to Table 4.7 it is noteworthy that with using these topic specific language models, the resulting systems outperform the set-up that serves as baseline for the experiments of this work, although for evaluating the baseline performance – as well as testing all other systems developed during this thesis – the exact same language models were utilized. The reason is, that for the latter no lattice re-scoring is applied, as this resembles the real-life application of the deployed lecture translator system. For transcription runs, however, there exist no run-time restrictions as limitation, whereby lattice re-scoring is becoming applicable, which potentially contributes to better recognition results and thus more reliable transcriptions than without any re-scoring.

Table 4.12 shows that – if compared to the numbers of Table 4.10 – the rejection rate



(a) $speaker_A$ (b) $speaker_W$

Figure 4.5: Performance of speaker-dependent system re-training when applying light supervision during automatic transcription runs, and utilizing confidence measures for weighting and/or thresholding. Performance is plotted in dependency of the amount of unsupervised training data and is measured in terms of word error rate (WER). The development systems are compared to the *speaker independent* baseline and two supervisedly re-trained reference systems (*supervised*). All systems were evaluated on separate test sets for $speaker_A$ and $speaker_W$, respectively.

		WER [%]	
Supervision	Language model	$speaker_A$	$speaker_W$
no supervision	LM_{SI}	23.2	19.6
light supervision	LM_A/LM_W	18.7	16.1

Table 4.11: Performance of the baseline system on the test sets for $speaker_A$ and $speaker_W$ for *no supervision* during transcription runs when using language model LM_{SI} and for *light supervision* by utilizing the topic specific models LM_A and LM_W, respectively.

Threshold	Rejection rate in %	
	uncleaned transcriptions	cleaned transcriptions
25%	8.6%	8.8%
50%	24.8%	26.1%
75%	48.4%	51.5%

Table 4.12: Rejection rate of words when applying different confidence measure thresholds and using topic specific language models for light supervision. For instance, a threshold of 25% discards all words with a confidence score lower than 0.25. Uncleaned transcriptions correspond to the raw *recognition* results, i.e., the unmodified decoder output. Cleaned transcriptions correspond to the transcription pre-processing category *filtered*.

of words given a particular threshold significantly decreases when using language models that comprise additional information regarding the target domain, indicating an increased confidence in the decoder's decisions.

For all remaining experiments explicated until the end of this chapter, light supervision by language modelling has been applied during the automatic transcription runs.

4.6.4 Iterative Viterbi Training

The intention of the following experiment was to always re-train the currently most up to date system S_{i-1} with a set of transcribed training data T_i^{new} for a subsequent iteration i, where

$$T_i^{new} \cap T_{i-x}^{new} = \emptyset \quad \forall i, x \in [1, i_{final}] \tag{4.3}$$

thus T_i^{new} being a new portion of previously unseen and automatically transcribed training data. Consequently, each new set of training data should optimally be transcribed by the most recent system available, as this particular system is supposed to be the by then most potent. However, for reasons of practicality this procedure has been simulated by using the baseline system S_0 for each iteration, instead of always using the most recent system. Algorithm 2 schematizes a single training iteration of the simulated procedure: As it was the case for all previous experiments, the automatic transcriptions of the full set of available unsupervised training data were all written by the very same system. Instead of the by then most recent system S_{i-1}, the transcription of T_i^{new} in line 4 is executed by the very same system S_0 for all i.

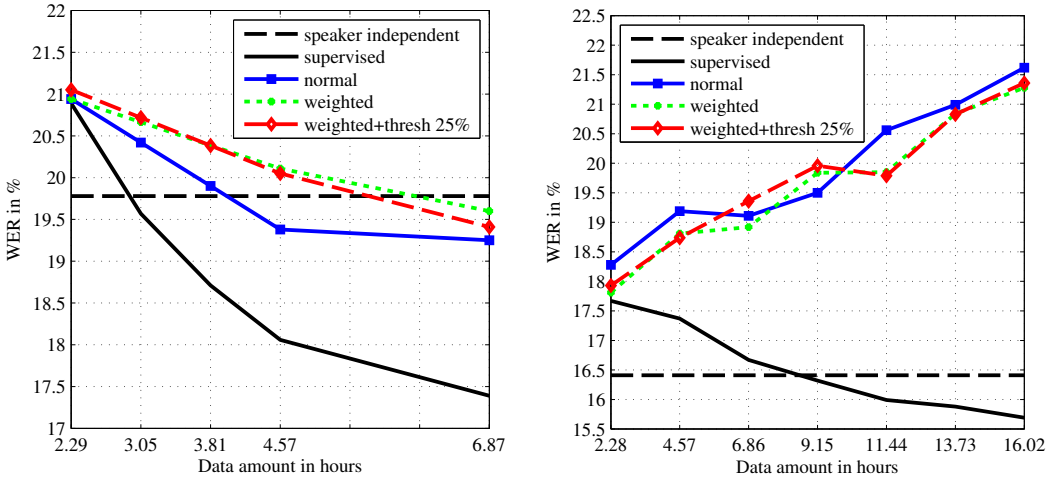
Algorithm 2 Iterative Incremental Viterbi Training

Require: $\exists S_0 \wedge i_{final} > 0$
Ensure: $S_{i_{final}}$
1: $i_{final} \leftarrow$ max. iteration
2: **for** $i = 1$ **to** $i_{final} + 1$ **do**
3: $T_i^{new} \leftarrow$ previously unseen untranscribed data
4: $T_i^{trans} \leftarrow transcribe(S_0, T_i^{new})$
5: generate training database $DB_i \leftarrow T_i^{trans}$
6: write labels L_i
7: $S_i \leftarrow trainingAlongLabels(S_{i-1}, L_i, DB_i)$
8: **end for**

The training step for system S_i is performed in line 7 as a Viterbi training along labels L_i, using training data stored in a database DB_i, and by re-training a preceding system S_{i-1}. Practically this means that, starting from a baseline system, its acoustic models are re-trained with each iteration by another additional Viterbi training. I.e., multiple consecutively executed re-trainings of the same models lead to a final system $S_{i_{final}}$.

Without loss of generality, the transcription pre-processing configuration *baseWords* has been utilized throughout these experiments. Figure 4.6 shows the results of testing this approach.

As can be seen, there is only little benefit for $speaker_A$ compared to the baseline performance, and only when used all data in the end. There is a discernible convergence, however the curve already indicates a deterioration likely to occur with increasing amounts of training data. For $speaker_W$ this is clearly seen, as the whole training process is spoiled even with very little training data in use. There exists a variety of probable causes for the observed phenomenon. One reason might be the dimension of the gap between baseline



(a) $speaker_A$ (b) $speaker_W$

Figure 4.6: Performance of speaker-dependent, iterative and incremental Viterbi re-training. Each iteration, the most recent system is re-trained. Performance is plotted in dependency of the amount of unsupervised training data and is measured in terms of word error rate (WER). The development systems are compared to the *speaker independent* baseline and two supervisedly re-trained reference systems (*supervised*). All systems were evaluated on separate test sets for $speaker_A$ and $speaker_W$, respectively.

performance and the theoretical upper bound given by the supervisedly trained system. Training for $speaker_A$ proved to be clearly more effective during previous experiments than for $speaker_W$, presumably due to the lower amount of related data already seen for the baseline system training. Under these circumstances, it seems possible to obtain higher gains with less data, whereas with a system that is already "aware of" the target speaker, further improvement becomes rather challenging. In this case, another reason might be that such a system is in higher danger of early over-fitting, which leads to a quick deterioration of performance and results in unhelpful systems. The obtained results lead to the conclusion that – besides marginal gains in Viterbi training runtime due to lower amounts of data per iteration and visible convergence for one of both speakers – this approach is not competitive and previously introduced frameworks are preferable to the former.

4.6.5 Incremental Training

Following a similar approach, the goal of further experiments was to iteratively train a system S_i in an unsupervised fashion by using data T_i^{trans} with automatic transcriptions written with an old system S_{i-1} that resulted from a preceding training iteration. T_i is an incrementally growing set of training data for iterations $i \rightarrow i_{final}$, where

$$T_i = \begin{cases} T_i^{new} & \text{if } i = 1 \\ T_{i-1} \cup T_i^{new} & \text{if } i > 1 \end{cases} \quad \text{with} \quad T_i^{new} \cap T_{i-x}^{new} = \emptyset \quad \forall i, x \in [1, i_{final}] \tag{4.4}$$

with T_i^{new} being a new portion of previously unseen and automatically transcribed training data. In contrast to Subsection 4.6.4, where with each iteration the most recent system S_{i-1} was re-trained by an iteration of Viterbi training atop of the updated acoustic models, this set-up again solely re-trains the models of S_0, as it was done initially. Again, the transcription pre-processing category *baseWords* has been used. Without loss of generality,

confidence measures have been utilized in order to apply word based weighting throughout the model update step.

The operating procedure of training differs to Algorithm 2, especially regarding the usage of new data and the training step:

Algorithm 3 Incremental Training

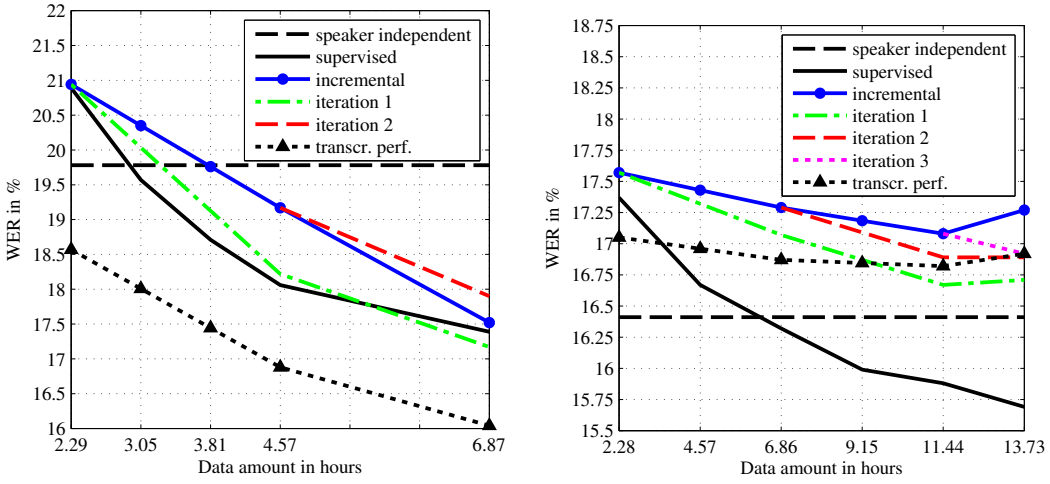
Require: $\exists S_0 \wedge i_{final} > 0$
Ensure: $S_{i_{final}}$
1: $i_{final} \leftarrow$ max. iteration
2: **for** $i = 1$ **to** $i_{final} + 1$ **do**
3: $T_i^{new} \leftarrow$ previously unseen untranscribed data
4: **if** $i = 1$ **then**
5: $T_i \leftarrow T_i^{new}$
6: **else**
7: $T_i \leftarrow T_{i-1} \cup T_i^{new}$
8: **end if**
9: $T_i^{trans} \leftarrow transcribe(S_{i-1}, T_i)$
10: generate training database $DB_i \leftarrow T_i^{trans}$
11: write labels L_i
12: $S_i \leftarrow trainingAlongLabels(S_0, L_i, DB_i)$
13: **end for**

The training process is as follows: First, initial transcriptions of a subset of all available training data are generated. Then, S_0 is re-trained using this data. A subsequent transcription run on a growing set of data is performed using the re-trained models. This particularly means, that all previously transcribed data from earlier iterations will be transcribed again, now using the hopefully more efficient system. The latter is repeated until iteration i_{final}, where at each time the baseline models of S_0 will be re-trained.

For both speakers, different step sizes regarding the growth of unsupervised data were applied. For $speaker_A$ the step size remained the same than for all previous experiments. For $speaker_W$, however, the interval was doubled, purposely leading to less iterations given the amount of available data due to feasibility reasons.

As can be seen by the Figures 4.7, the final systems trained in this way are not able to outperform the systems of the previous experiments, although for $speaker_A$ the supervisedly re-trained system's performance is touched again. However, incremental training on data of $speaker_W$ leads to a non-competitive system. After some stable gain, the performance curve even shows a degradation in terms of WER with increasing amount of utilized data. Supposedly, the reason for these different manifestations is the dependency of the performance of the transcribing system. The figures depict the respective transcription performance for each speaker. The difference arises from the use of topic specific language models, the baseline performance of the initial acoustic models, and the application of lattice re-scoring for automatic transcription, since there exist no run-time restrictions for the latter. The assumption is, the higher the gap between transcription performance and actual recognition performance, the more efficient the incremental training becomes. For $speaker_A$ transcription performance is well beyond the baseline system's WER as well as the supervisedly trained reference system, whereas the performance curve for $speaker_W$ during transcriptions can neither reach the scope of the baseline, nor the reference, except when only very small amounts of data are used.

However, by observing the outcomes for both speakers one might realize that with increasing recognition power, i.e., improved capacities for automatic transcription, the performance curves for the test sets not only catch up, but notably approach the respective



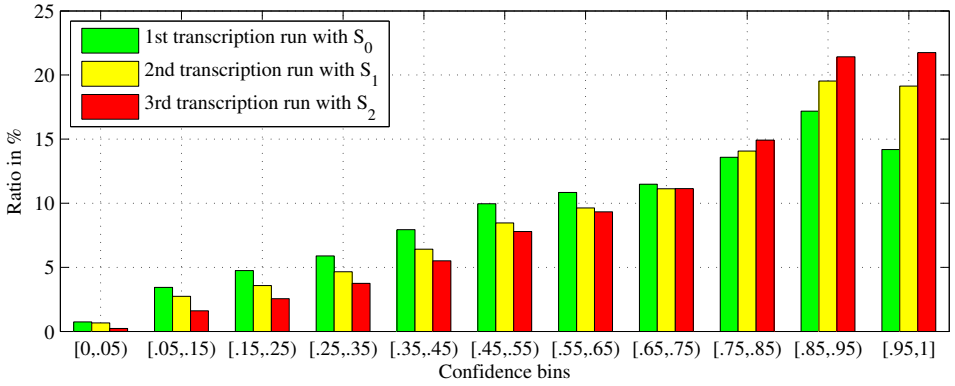
(a) $speaker_A$ (b) $speaker_W$

Figure 4.7: Performance of speaker-dependent system re-training. Each iteration, an incrementally growing training set is re-transcribed by the most recent system. Performance is plotted in dependency of the amount of unsupervised training data and is measured in terms of word error rate (WER). The development systems are compared to the *speaker independent* baseline and two supervisedly re-trained reference systems (*supervised*). All systems were evaluated on separate test sets for $speaker_A$ and $speaker_W$, respectively.

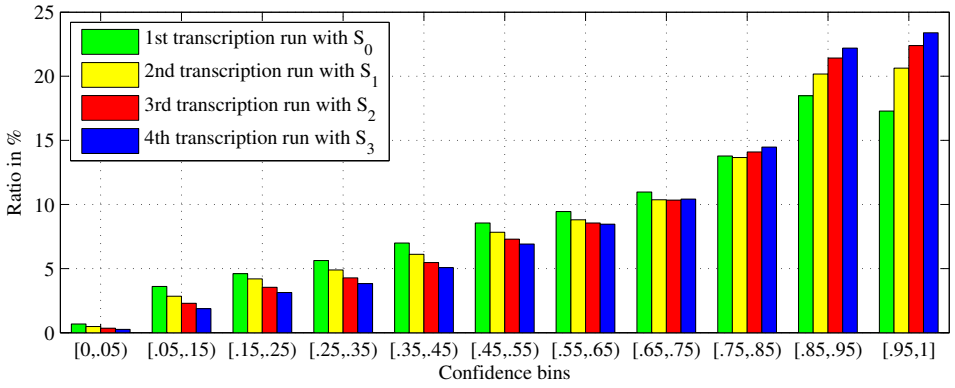
transcription performance curve. This means, that even with marginal improvements or stagnation in terms of WER regarding the automatic transcription efficiency, the resulting text output appears to increase in accuracy, given the corresponding audio data. A decreasing transcription performance however is likely to cause a performance drop of the system resulting from training on this data, as can be seen by the example of the last iteration for $speaker_W$. If the latter is not taken into consideration when analysing the results for both speakers, one can observe that, although transcription performance converges up to 53% less strongly than the performance on test sets, the discrepancy between both performance curves decreases by up to 50%. These observations show that the tested approach of incremental training has the potential to increase the quality of automatically generated transcriptions over time. Due to lack of sufficient amounts of data it was not possible to verify an ongoing positive development regarding the conceivable outperforming of a supervisedly trained system in case of $speaker_A$.

The Figures 4.7 present further details: Curves entitled *iteration* i reflect the theoretical development of the performance curve over the available data, when from then on solely transcriptions of the particular iteration i are used. It can be seen for $speaker_A$ that the transcriptions resulting from the first transcription run performed by S_0 initially are superior for training of S_2, i.e., on the data chunk with 4.57 hours in size, when compared to the ones written with S_1, which is the system that results from training with an additional 2.29 hours of automatically transcribed data, i.e., the system after iteration 1. With growing data, however, the performance curve resulting from the usage of the initial transcriptions flattens, whereas the curve representing the incremental approach shows a steady improvement in WER. That means for the training of S_3 on the largest amount of data of 6.87 hours, that the transcriptions delivered by S_0 turn out to be inferior to the ones generated by the most recent system, S_2. Moreover, if writing all transcriptions with help of S_1, performance also decreases, compared to the strictly iterative procedure. Even if in this case the data was very limited, both phenomena indicate, that with more untran-

scribed training data becoming available, the iterative course of action, i.e., subsequently re-training the baseline models and re-transcribing a cumulative set of training material leads to an overall steeper and more steady decrease of recognition errors. On the other hand, the outcomes for $speaker_W$ reveal the dangerous nature of too low initial automatic transcription capacities: Uniformly, all systems S_i with $i < i_{final}$ apparently deliver more reliable transcriptions for the full set of data than every respective subsequent system.



(a) $speaker_A$



(b) $speaker_W$

Figure 4.8: Distribution of the word based confidence measures on confidence bins. The distribution is depicted in dependency of the amount of performed re-transcription runs followed by re-training. Clearly visible is the gradual shift of rates for confidence score spans to higher bins.

Figures 4.8 depict the gradual shifts of the rates for single confidence bins with respect to the undergone training iterations. Only the confidence measures for word tokens were taken into account. Non-word tokens, i.e., noise and filler tags were ignored in order to determine the impact of the iterative training process to the changing of recognition capabilities solely for words. For both speakers initial transcriptions that were written with the baseline system feature generally moderate confidence scores, which manifests in a median of 0.7 for $speaker_A$ and 0.74 for $speaker_W$. With every iteration, the confidences shift to higher regions, ending with a median of 0.8 after 3 iterations for the former, and a median of 0.82 after 4 iterations for the latter speaker. Moreover, even though the highest bin $[.95, 1]$ has a smaller range, it holds most of the words in terms of ratio for both speakers after several iterations. The re-distribution of increasing amounts of words to higher confidence bins and at the same time decreasing WER in tests prove that the systems get potentially more reliable given the respective speaker they are trained to.

4.6.6 Analysis

In this chapter work on unsupervised adaptation of the acoustic model by training has been described. The goal was to evaluate new techniques for integration in the simultaneous lecture translation of KIT. For this, automatic transcriptions of new lectures with a speaker-independently trained baseline system were produced in order to improve the same system for a specific lecturer in an incremental as well as iterative fashion.

Evaluating four different ways of processing of the decoder outputs led to the conclusion that it is of advantage to let the Viterbi algorithm during training decide which pronunciations to use and where to insert which noise words, instead of fixating these informations in the transcriptions as they are produced by the decoder during automatic transcription runs. Mapping all words and non-word tokens to base forms (*baseAll*) seems to be beneficial on less data, whereas *filtered* transcriptions work better when more data becomes available. For $speaker_A$, speaker-dependent models perform better than the speaker-independent models when at least 3.8 hours of training data are available.

Further, the word level posterior probabilities obtained during decoding were utilized by weighting and thresholding the words of a transcription. Combining word-based weighting and a threshold of 25% led to the fastest drop in word error rate with increasing amount of training data. The best systems in terms of WER reach an error rate of 18% for $speaker_A$ and 16.9% for $speaker_W$, being trained on *baseAll* processed transcriptions and *weighted* training. In terms of convergence time, an additional *threshold* of 25% led to an increased performance, yielding a competitive WER of 18.2% and 17.0%, respectively.

Experiments on light supervision by language modelling led to similar results than previous tests with respect to the impact of transcription pre-processing policies. Performance of all re-trained systems increased by up to 11% relative. Moreover, the best system for $speaker_A$ outperformed the supervisedly re-trained reference system by 3.1% relative, which possibly indicates transcription errors in the manual transcriptions, that potentially may be corrected by automatic transcription.

Based on the latter set-up, an iterative incremental approach was evaluated. A growing set of training data was always re-transcribed by the most recent system and then used to re-train the baseline again. The final systems trained this way were not able to outperform the systems of previous experiments. It has been shown that potential performance gains strongly correlate to the performance of the systems for automatic transcription. The redistribution of word confidences to higher confidence bins and likewise decreasing WER in tests prove that with each iteration systems get more reliable given the respective speaker they are trained to.

5. Iterative Batch Training for Domain Adaptation

Having a large amount of potential, yet unannotated training data at hand, iterative batch training can easily be applied for enhancing an existing system. During these experiments, a mediocre baseline system was utilized for transcribing additional audio data intended to add to the acoustic model training. The purpose of this set-up was to overcome a domain mismatch between the given baseline framework and the target domain for the actual decoding process. Large quantities of annotated in-domain audio data was available, that made it possible to train a reference system delivering an upper bound for the expected performance gains by applying unsupervised training. For the unsupervised iterative batch training the existing transcriptions have simply been ignored. Starting from a system trained on European Parliament Plenary Session (EPPS) recordings, the objective was to push the system towards the general domain of TED talks with help of domain adaptation by unsupervised acoustic model re-training. The re-training took place in an iterative fashion, i.e., at each point in time, the by then most recent system was used to re-transcribe a fixed set of unannotated training data which was then used to re-train the baseline system by adding this very data, including the new transcriptions to the supervised data that was used for the baseline training. Further, it was an objective to investigate the impact of the transcription pre-processing methods that were explained in detail in Subsection 4.6.1 of Chapter 4, as well as the effectiveness of confidence measure based data filtering methods applied during acoustic model training, i.e., weighting with confidence scores and thresholding by a bounding confidence value, as elucidated in Section 4.6.2 of Chapter 4.

5.1 Databases

The experiments described in this chapter were conducted with the help of two distinct corpora derived from EPPS speeches and TED talks, respectively. The former was used for the baseline acoustic model training, the latter for unsupervised acoustic model training experiments on the baseline produced in this way. The EPPS data comprises approximately 80 hours of manually transcribed speeches, provided by RWTH Aachen within the TC-STAR project [GBK+05]. Around 157 hours of TED talks have been downloaded from the TED websites[1]. In contrast to the former speaker adaptive training experiments conducted with the KIT lecture translation system, that very choice of training sets for baseline and adaptive training leads to an implicit domain adaptation experiment.

[1] http://www.ted.com/talks

5.1.1 Training Data

The speakers of the EPPS corpus can be categorized in different ways: Besides the distinction between original speakers and interpreters, who simultaneously translate speeches in other official languages of the European Parliament, it can be distinguished between native speakers and non-native speakers. Most of the speeches are planned. The interpreters' contributions show effects such as pauses following by dense speech intervals, that arise from the process of simultaneous translation [GBK+05]. The TED talks collection is a freely accessible web repository of recordings of public speeches and talks with varying length of 5 to 25 minutes each. They are held by people from various fields of expertise covering repetitive topics related to technology, entertainment and design (TED).

Each recording was processed by normalizing and down-sampling the signal to 16 kHz with a resolution of 16 bit, which conforms the standard input for all utilized speech recognition systems. The EPPS data was manually segmented and transcribed, and thus already contained speaker labels ready to make use of during supervised training. The TED data was automatically segmented with the help of a decoding pass on the input data in order to discriminate speech and non-speech regions and doing a forced alignment given the subtitles that accompany each recording for simultaneous video closed captioning. The time stamps of those closed captions are not precise enough for training purposes, which finally led to the necessity of generating more accurate segment boundaries. However, they were also utilizable for exposing the relevant speech part of each downloaded video soundtrack by cutting away intro and outro sequences given the annotations. The final segmentation was then done by splitting at non-speech regions of notable length. For the TED data the simplified assumption that each talk is spoken by exactly one speaker has been made. Table 5.1 lists the details of the resulting training sets.

Data	#talks	#spk	#utt	dur	dur/utt
EPPS	63	1894	52464	79.6 h	5.46 s
TED	699	699	204502	146.6 h	2.58 s
Sum	762	2593	256966	226.2 h	3.16 s

Table 5.1: Statistics of the speech data used for the baseline acoustic model training (*EPPS*) and subsequent unsupervised training (*TED*), including the total number of recordings (*#talks*), the amount of speakers (*#spk*), the total number of utterances (*#utt*), the overall speech duration (*dur*), and average speech duration per utterance (*dur/utt*).

5.1.2 Test Data

For the described experiments a development set for system development and parameter optimization ("dev2010"), and a test set for evaluation ("test2010") was used, both containing held-out TED talks that were also used as development and test sets for the Automatic Speech Recognition (ASR) task of the International Workshop for Spoken Language Translation (IWSLT) 2010 evaluation campaign[2]. Both sets were used with the original pre-segmentation provided by the IWSLT organizers.

5.2 EPPS-based Baseline system

The baseline system is derived from the ISL 2007 English speech transcription system for European parliament speeches, developed at the University of Karlsruhe, now KIT

[2] http://iwslt2010.fbk.eu

Set	#talks	#utt	dur	dur/utt
dev2010	8	934	1.52 h	5.87 s
test2010	11	1664	2.47 h	5.36 s

Table 5.2: Statistics of the speech data used as development set ("dev2010") and test set ("test2010"), including the total number of talks (*#talks*), the total number of utterances (*#utt*), the overall speech duration (*dur*) and average speech duration per utterance (*dur/utt*).

[SFKW07]. Unlike the original system which has been trained on the 80 hours of unsupervised training data mentioned in Section 5.1 plus an additional amount of 167 hours of unsupervised data with automatically generated transcriptions, both provided by RWTH Aachen, the system used as baseline for the recent experiments was solely trained with the supervised data. Also, acoustic models are fully continuous and remain it throughout the development process for maintaining comparability with respect to the impact of unsupervised training methods. The system uses a language model biased to EPPS and general parliament speeches for all decoding tasks.

5.2.1 Feature Extraction

As for the experimental systems described in Chapter 4 the acoustic front-ends of the proposed systems have been trained and tested with the help of the JANUS Recognition Toolkit (JRTk) [SMFW01].

The feature extraction is based on the traditional and widely used *mel-frequency cepstral coefficients* (MFCC). For training the front-end provides features every 10 milliseconds. During decoding this was changed to 8 milliseconds after the first stage. During both, training and decoding, the features were obtained by a discrete Fourier transform followed by a Mel-filter-bank. *Vocal tract length normalization* (VTLN) is applied in the linear domain [ZW97]. The utilized MFCC front-end uses 13 cepstral coefficients, where mean and variance are normalized on a per-utterance basis. As a last step, 15 adjacent frames are stacked into one single 195 dimensional feature vector which is finally reduced to 42 dimensions by applying a *linear discriminant analysis* (LDA).

5.2.2 Acoustic Modelling

All models are context-dependent quinphones with a standard three-state left-to-right HMM topology with self loops and without skip states. The model for silence poses an exception, as only the last state is equipped with a self loop. All transition probabilities remain fixed throughout the whole training process. After a clustering step the models use 4000 distributions over 4000 codebooks. This system was trained by using incremental splitting of Gaussians training (MAS) [KFN98], followed by optimal feature space training (OFS) which is a variant of *semi-tied covariance* (STC) [Gal99] training using one global transformation matrix. Compared with the original system, no further Viterbi training for refinement was applied, because this additional training step resulted in first signs of over-fitting.

The phoneme set in use is modified version of the phoneme set used by the CMU dictionary and consists of 45 phonemes and allophones [CMU]. The phonemes are completed by 9 additional tags for noises such as fillers, breath, laughter, general human and non-human noise, and silence. All noise phonemes are modelled context independently, in contrast to the genuine phonemes. Each model has a fixed amount of Gaussians, where the standard phone models are comprised 60 Gaussians, some specific noise models have 64 Gaussians, and silence is modelled by 24 Gaussians.

The full training cycle for the baseline system looks as follows:

1. Linear discriminant analysis (LDA)
2. Sample extraction
3. Merge and split (MAS) training (incremental growing of Gaussians)
4. Optimal feature space training (OFS)

5.2.3 Dictionary & Language Model

For decoding during automatic transcription runs and testing a slightly modified version of an already existing dictionary tailored to the EPPS domain and general speeches was used [SKK12]. It was build by utilizing textual data that was primarily used for language model training (see Table 5.3). Pronunciations of known words were extracted from a large background dictionary. Pronunciation for unseen words were generated with Festival [BT97]. For the recent experiments, noise tags were added to the vocabulary as pronunciation variants of the optional word $. As for the previous experiments this has been done due to the use of the several transcription pre-processing techniques described in the previous chapter. The test and transcription dictionary is comprised of 141313 pronunciations for 127806 unique, case-sensitive vocabularies.

Corpus	#words	mixing parameter
UK parliament	29.478.882	0.009
EPPS	33.793.627	0.001
UN	40.991.279	0.051
Gigaword	593.472.957	0.41
Broadcast News	130.768.508	0.273
Web dumps	144.062.839	0.253

Table 5.3: Statistics of the utilized training corpora for language modelling, including the corpus name, the overall word count per corpus (#words) and the mixing parameter used as interpolation weighting factor.

Also derived from previous experiments, a 4-gram language model that was build on the data listed in Table 5.3 was used for transcription and testing. The textual data was extracted from various sources like EPPS transcripts, other parliamentary transcripts, broadcast news data, web dumps and the Gigaword corpus. The applied language model was built using the SRI Language Modelling Toolkit [Sto02]. Besides the application of Good-Turing discounting*good1953population*, Chen and Goodman's modified Kneser-Ney discounting is applied*modified-kneser-ney*. Between the different sources interpolation weights have been computed on a development set, where

$$P(w|h) = \lambda_1 P_1(w|h) + \lambda_2 P_2(w|h) + \cdots + \lambda_k P_k(w|h). \tag{5.1}$$

The interpolation weights $\lambda_1, \ldots, \lambda_k$ were selected so as to maximize the likelihood of the held-out data. As can be seen Table 5.3, the contribution of web data, Gigaword and broadcast news data sum up to over 93%, whereas the EPPS data takes only a nominal part in interpolation. In order to reduce memory demands the language model was finally pruned [SKK12].

N-grams	count
1-grams	130625
2-grams	30264678
3-grams	36809561
4-grams	34072904

Table 5.4: N-gram count statistics of the language model applied during decoding.

5.3 Decoding

Decoding for automatic transcription and for system evaluations is performed the exact same way as explicated in Section 4.3 of the last Chapter. Again, for reason of comparability of the experimental systems the decision was made to refrain from decoder parameter tuning. Moreover, the language model remains fixed for all fields of application. That means a performance change of the development systems is directly deducible from the quality of the automatically generated textual references.

5.4 Training

The framework for re-training the original EPPS system in an unsupervised fashion includes several steps: First, unannotated TED training data gets transcribed by the baseline system that has been trained solely on EPPS data before, and by using the general language model introduced above. The resulting transcriptions consist of confidence annotated 1-best hypotheses for all generic utterances having been produced by the automatic segmentation that was undergone before training. Posteriori probabilities were computed with the help of word lattices in order to use them as a measure of confidence.

One iteration of re-training follows the same steps than the baseline training described in Subsection 5.2.2, with two additional steps: First, training is performed along labels, where the labels were computed and stored in a preceding step so that training can be performed along the same Viterbi paths, but with modifications such as varying amounts of data. Second, after OFS training, two iterations of Viterbi training is applied due to the significantly larger amount of training data being available for unsupervised training. The results elaborated in the following sections will show that with more data at hand, the additional Viterbi training is beneficial for the overall system performance.

With these expansions of the training cycle, one step of unsupervised acoustic model training is performed as follows:

1. Label writing
2. Linear discriminant analysis (LDA)
3. Sample extraction
4. Merge and split (MAS) training (incremental growing of Gaussians)
5. Optimal feature space training (OFS)
6. Viterbi training (2 iterations)

5.5 Testing

System performance for all development steps is measured with help of the data listed in Table 5.2. The unit of choice for describing performance is again a simple word error rate (WER). For each intermediate system, the WER was determined after each training step, i.e., after MAS training, OFS training and each iteration of Viterbi training in order to keep track of the system improvement in dependency of the applied training routines. However, the main focus is on analysing the impact of several strategies for unsupervised training, in particular filtering methods for potentially unclean and error-prone automatically generated transcriptions. Thus, systems were trained multiple times, using different approaches of data pre-processing as well as several methods for weighting and filtering training data based on automatic confidences. Decoding of the test data was undergone with lattice re-scoring and with incremental VTLN and feature space constrained MLLR [Gal97].

5.6 Experimental Results

The intention of all following experiments was to evaluate how well a system can be adapted to a new domain by the application of unsupervised iterative batch training, given a fixed training set. That means, in contrast to the experimental set-up described in Chapter 4 the main focus now is on appropriate ways of pre-processing erroneous training data and methods for avoiding the use of malicious data during training, which are potentially harmful to the correctness of the acoustic models intended to get improved.

Initially, the unmodified baseline system performs reasonably well on data of the domain it was optimized for, i.e., recordings of EPPS talks, whereas the base performance of data belonging to the new target domain, i.e., recordings of TED talks, is significantly worse. Table 5.5 lists the WER of the baseline system, given test data of both, the old as well as new target domain.

Test set	Data	WER in %		
		Training		
		MAS	OFS	VIT2
eval2007	EPPS	17.5	16.7	16.5
dev2010		39.3	38.0	38.6
test2010		39.6	37.3	37.9
dev2010	+ TED	24.9	24.3	24.2
test2010		23.1	22.4	22.3

Table 5.5: Performance of the baseline system trained on *EPPS* data and a supervisedly re-trained reference system trained on *EPPS + TED* data in word error rate (WER), evaluated on test sets *eval2007* (EPPS specific test set, comprised of 53 utterances summing up to 2.7 hours of data), *dev2010* and *test2010* (both TED specific test sets).

As can be seen, the baseline performance is compared to a supervisedly re-trained system using the TED data. The subtitles that came along with the downloaded and converted TED talks were re-aligned to the audio material via a forced alignment on the full talks, as was described in Section 5.1. As a reminder, the reason for this procedure is the low accuracy of the given segmentation, which was particularly unusable for a correctly

conducted supervised acoustic model training. The system trained in this way served as reference for the following experiments, whose achieved WER is representing an upper limit for the potential improvements via the unsupervised training approach.

The table also reflects the impact of Viterbi training, given different amounts of training data. Where for the systems solely trained on EPPS data the additional Viterbi training iterations clearly harm the overall performance, the opposite is the case for the systems trained on both corpora, EPPS as well as TED. In this case, Viterbi training leads to a slight improvement in terms of WER, compared to systems finalized with OFS training only.

The intention of the following experiments was to firstly determine a promising set-up for pre-processing the training data resulting from automatic transcription runs, as well as the most useful training scheme, that optimally utilizes pre-computed confidence scores for the automatically generated textual annotations and excludes erroneous and potentially harmful parts from system training. Secondly, after finding the optimal pre-processing and training scheme, this framework is applied for iterative batch training in order to evaluate the potential system improvements by successive re-transcription and re-training runs given a single, fixed set of data.

5.6.1 Transcription Pre-processing

Based on the developments made in Chapter 4, it has been decided to follow the elaborated work flow in a similar way by adaptation and transfer to the new training set-up and task. This decision has also been made in order to back up made conclusions and reproduce previous observations, thus ascertaining consistency in the usefulness of particular methods and techniques.

Consequently, the first set of experiments aimed at evaluating the effect of various transcription pre-processing styles. However, for the sake of simplicity, only the two winning pre-processing strategies were selected for further experiments. Without loss of generality, experiments have been conducted with *filtered* transcriptions, that are cleaned from noise tags and informations about pronunciation variants, as well as the variant *baseWords* (all noise tags are used as they were output by the decoder, whereas all words are mapped to their base forms), that embodies one of the two previously evaluated middle ways between fully filtered and raw decoding outputs (see Section 4.6.1 of the last Chapter), were examinated.

The training pipeline for unsupervised training in general looks as follows:

1. Automatic transcription
2. Training database generation
3. Label writing
4. MAS, OFS, and Viterbi training (2 iterations)

Section 4.6.1 of Chapter 4 elaborates the details of all four particular pre-processing methods that were applied, and how JANUS handles textual annotations during label writing.

Table 5.6 shows the resulting performance for the two selected configurations on both, the development and test set after one single iteration of unsupervised batch training.

Compared to the baseline system, an improvement of up to 22.1% relative on the development set and 26.6% relative on the test set is observable after only one training iteration. It can be seen that both pre-processing methods perform about equally well, with *filtered* transcriptions having a slight advantage on the development set. Performance after MAS

Training	WER in %			
	dev2010		test2010	
	baseWords	filtered	baseWords	filtered
MAS	30.3	30.0	29.4	29.1
OFS	29.6	29.6	28.3	28.3
VIT1	29.7	29.6	28.3	28.6
VIT2	29.7	29.5	28.4	28.5

Table 5.6: Performance of the re-trained system using additional unsupervised data with transcriptions pre-processed in different ways. Performance is measured in word error rate (WER), evaluated on data sets *dev2010* and *test2010*.

and OFS has additionally been taken into consideration in order to examine the tendency of decreasing generalization after Viterbi training. Presumably, using transcriptions with words mapped to their base forms (*baseWords*) is more robust to over-fitting when applying further model training. However, performance does not increase either, rendering Viterbi training on top of OFS training ineffective.

5.6.2 Confidence Weighting & Thresholding

The main focus of these domain adaptation experiments was on evaluating the potential improvements of unsupervised training via weighting and thresholding the erroneous training data. Where weighting ensures a differentiated procession of partly unreliable data, thresholding actively excludes potentially harmful data from training. Section 4.6.2 of the last Chapter describes in detail how JANUS computes and treats confidence scores internally and how these confidence measures can be utilized to affect the actual acoustic model training.

In these experiments the confidence scores obtained during decoding were utilized in the same fashion than in the experiments elaborated in the previous chapter:

weighted Sets the gamma probabilities of the states of a word during Viterbi training to the posteriori probability of the respective word

thresh Removes words with a confidence below a certain threshold from training

weighted+thresh Combines both methods

For this set-up, prior to Viterbi training, incremental splitting of Gaussians training (MAS), followed by optimal feature space training (OFS) is performed. This leads to a major difference in handling confidence scores during training: When applying the Viterbi algorithm for updating the models, statistics for all speakers in the training set are accumulated in a first phase. In a second phase, these accumulated statistics are used to update the model parameters according to the maximum-likelihood criterion [FMS].

Quite different from this scheme, MAS and OFS training is conducted on sample feature vectors that were extracted from the training data beforehand. Sample extraction is done on frame level, given pre-computed labels, i.e., fixed state alignments that were stored to disc before the actual training. Prior to sample extraction, a linear discriminant analysis (LDA) is performed, which – besides an LDA matrix that is used during pre-processing in the ASR module – outputs the number of occurrences for every codebook. These statistics are then used during the sample extraction to extract an evenly distributed number of

example feature vectors. The sample data collection is stored in separate files for each codebook.

In order to be able to train the acoustic models on confidence annotated training material, several modifications were to undergo. First, sample extraction was extended so as to write out sample vectors into bins according to their respective confidence. The confidence score enabled sample extraction proceeds as follows: During initialization, the word based confidences are loaded from the confidence annotated training database. Then, the HMM for training is built by loading the previously computed fixed state alignments. The word based confidence measures are assigned to each frame fr_i^w that belongs to a particular word w by setting a parameter $gamma(fr_i^w)$ (see Section 4.6.2 of the last Chapter). For words without a confidence, e.g., optional words that were added to the training path by the Viterbi algorithm, the gamma value of the respective frames is set to 1, i.e., the default value:

$$gamma(fr_i^w) = \begin{cases} 1 & \text{if } \nexists conf(w) \\ conf(w) & \text{else} \end{cases} \quad , conf(w) \in [0,1] \tag{5.2}$$

Because there is no mechanism available for internally passing confidence scores on to the actual model training, a circumvention has to be accepted. By writing out samples to separate folders, where one particular folder contains the extracted samples sharing confidences of a particular range, and subsequent folder-wise sample re-loading during training the confidences become communicable into the training pipeline. This approach, however, has a major drawback: The described way of transducing confidence scores has a discretizing effect and hence leads to a loss of information. Yet this effect may be counteracted to a certain degree by selecting an appropriate bin size for discretizing the confidences. For the following experiments, the range of word confidence scores reaching from $c_{min} = 0.0$ to $c_{max} = 1.0$ was split into b_n bins with equidistant centres $\hat{c}_i$, $i = 1, 2, \ldots, b_n$ and uniform width $\Delta c = \frac{c_{max}}{b_n}$:

$$bin_i := \begin{cases} \big[\Delta c(i-1), \Delta ci\big) & \text{if } i = 1, 2, \ldots, b_n - 1 \\ \big[\Delta c(i-1), \Delta ci\big] & \text{if } i = b_n \end{cases} \tag{5.3}$$

This binning leads to an error err_k for a sample k with confidence c_k, where

$$err_k = |\hat{c}_i - c_k| \tag{5.4}$$

Besides sample extraction, the training framework for MAS training as well as OFS training had to be adapted so that either became capable of incorporating confidence measures into the update step for the acoustic model parameters. The training data is accumulated frame-wise during MAS training. Every frame that gets added to the training accumulators with a default weighting factor *-factor* of 1. By loading the training samples folder-wise, where a folder with designation Δcf solely contains samples belonging to the particular confidence bin bin_f, frame-based weighting gets enabled by setting *-factor* $\frac{\Delta cf}{100}$.

The same approach is applicable for OFS training, with the exception that data is accumulated from matrices. A scaling vector v_s can be defined for re-scaling the data prior to the accumulation. With this tool at hand, weighting of an accumulation matrix $ACC_{m,n}$ can be enabled by explicitly setting a parameter *-scaleVect* v_s, where

$$v_s = \frac{\Delta cf}{100} \cdot \vec{1} \quad , \text{with } \vec{1} = (1, 1, \ldots, 1)^T, dim(\vec{1}) = m \tag{5.5}$$

Thresholding becomes trivial for both training steps and gets realized by not loading folders that contain samples that were assigned a confidence score below a pre-defined limit.

Weighting an thresholding during the final Viterbi training step is realized the exactly same way as explicated in Section 4.6.2 of the last Chapter.

Algorithm 4 Single training iteration using confidence measures

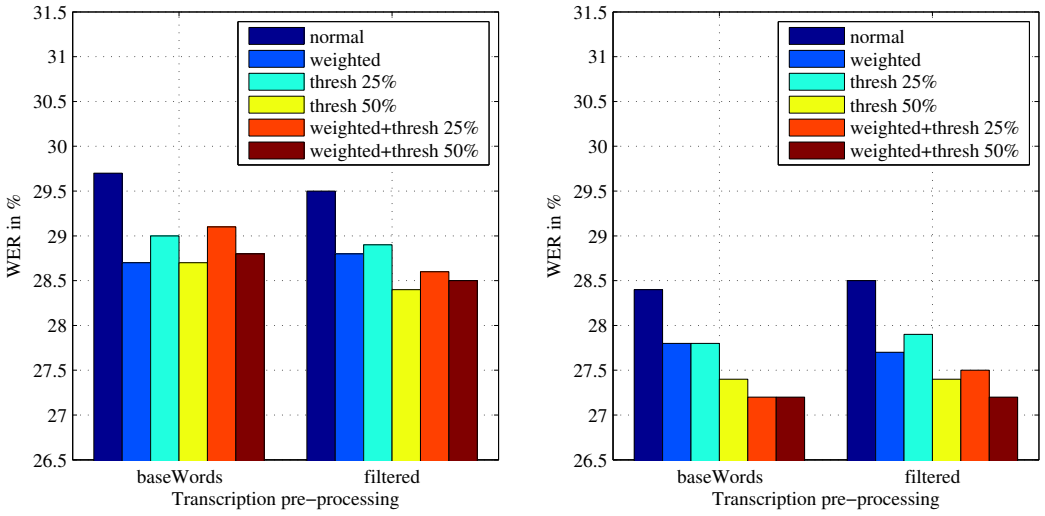
Require: AM
Ensure: AM'
1: $BINDIR \leftarrow$ binned data directory path
2: **for all** i with $bin_i \in BINDIR$ **do**
3: $C_i \leftarrow \frac{i}{100}$
4: **if** $\exists threshold \wedge C_i < threshold$ **then**
5: continue
6: **end if**
7: $S_i \leftarrow$ samples loaded from bin_i
8: **for all** j with $sample_j \in S_i$ **do**
9: Set factor of $sample_j$ to C_i
10: **end for**
11: Accumulate training data
12: **end for**
13: Update models $AM \rightarrow AM'$

Algorithm 4 schematizes the confidence score enabled MAS training update step. OFS training follows the same program flow, but instead of iterating over sample vectors $sample_j$, accumulation is done on matrices $matrix_k$, where one matrix comprises only samples assigned to one particular model each. As a result, instead of setting a single weighting factor in line 9, a scaling vector of the form like Equation 5.5 has to be defined and subsequently set as parameter for the accumulation step.

Figure 5.1 shows the results of applying the different confidence based weighting and thresholding techniques after one full training iteration, i.e., the graphic displays the system performances after two iterations of the final Viterbi training.

By reference of the word error rates on dev2010 it can be seen that the transcription pre-processing configuration *filtered* proves to be superior also when applying weighting or thresholding of the training data. Compared to not using any confidence measure annotations for data filtering, all techniques of weighting and thresholding lead to a gain in performance by up to 3.7% relative. Weighting the training data according to the confidences of the decoder already results in a competitive performance without losing any data in terms of quantity. If using the configuration *baseWords*, weighting alone already results in the best performance of all tested approaches. However, this is not the case for *filtered* transcriptions. Here, thresholding – whether exclusively or in combination with weighting – can further improve the system performance after training, compared to weighting alone. The graphic also reveals that it is of advantage to set the threshold to a higher value: Best results were obtained by skipping parts of the training data that were below a confidence threshold of 50%. A threshold of 50%, at the same time, was the strictest value tested during these experiments. Thus, the influence of an even stricter threshold may possibly lead to further improvements. A very rigorous thresholding, however, would also lead to a considerably higher loss of potentially useful training data.

It can also be seen, that the combination of both, weighting and thresholding leads to a competitive overall performance, if compared to using thresholding or weighting alone. Where for transcription configuration *baseWords* the combination affects the system only



(a) dev2010 (b) test2010

Figure 5.1: Performance of a full training iteration of MAS, OFS and Viterbi training (2 iterations). Performance is plotted for each configuration of transcription pre-processing and in dependency of the applied confidence based filtering. Filtering is applied via weighting the training data with confidences (*weighted*), removing words from the training data with a posterior probability below a certain *threshold* (25% and 50%), and a combination of both. Performance is measured in terms of word error rate (WER). All systems were evaluated on the TED-talk specific development set *dev2010* and test set *test2010*, respectively.

marginally, combined weighting and confidence based data filtering is beneficial if compared to weighting alone, and results in a better or comparable performance if compared to thresholding alone, where the additional weighting seems to be of higher advantage, if the threshold is set to a moderate value, that allows for more potentially erroneous data to slip into model training. By reference to the results a trade-off between the strictness of a threshold value for actively excluding data from training, and weighting the training data so that potentially erroneous parts will be cushioned is observable.

If compared to the system performances on the test2010 set, the observations made are consistent with the performance progression on the development set. For the *baseWords* configuration it is noteworthy that the combination of weighting and thresholding has a clear advantage over applying the techniques in isolation. When using *filtered* automatic transcriptions for training, the behaviour remains the same as for the dev2010 test, demonstrating that the combined approach again leads to the system with lowest word error rate.

According to the observed results it can be assumed that letting the Viterbi algorithm decide where to insert noise words and also what pronunciation variants to choose for each word in the training path triumphs over the decisions made by the decoder during automatic transcription. Thus, accepting the decoder decisions regarding noise and filler parts in the training utterances tends to harm the reliability of the training data. Given the considerably higher amount of utilizable unsupervised training data, it is recommendable to set a confidence score threshold to a higher value, so that a good share of potentially harmful data can be avoided to slip into model training.

The impact of thresholds with increasing strictness has been investigated by computing statistics of the word rejection rate given a particular threshold value. Furthermore, a

sorted list of the words that were excluded from training most frequently in dependency of a particular cut-off confidence score was computed, giving a more detailed insight into the effect of transcription filtering by thresholding. As expected, especially very short words that bear a high risk of confusion were discarded from training, even when setting the cut-off confidence to a moderate value of 25%. In particular this affects frequently used words such as articles ("the", "a"), auxiliary verbs ("are", "is"), prepositions ("of", "to", "in"), pronouns ("I", "we", "you"), and so forth. Another strongly affected word class in the case of the transcriptions conforming the *baseWords* configuration is the set of noise and filler tags, where especially the tags for breath noises were filtered by their frequently low confidences. Table 5.7 shows the impact of the thresholds that have been experimented with (25% and 50%), as well as the theoretical threshold of 75%, whose impact on the overall system performance has not been tested by any actually conducted experimental training runs. Regarding the transcriptions, the distinction has to be made whether noise tags are contained (as is the case for the *baseWords* configuration) or excluded (which applies for the *filtered* transcriptions), as this has an impact on the overall accumulated word count and consequently the relation of accepted and dismissed words given a particular threshold.

Threshold	Rejection rate in %	
	baseWords	filtered
25%	8.8%	9.5%
50%	27.3%	29.0%
75%	48.7%	51.1%

Table 5.7: Rejection rate of words when applying different confidence measure thresholds. For instance, a threshold of 25% discards all words with a confidence score lower than 0.25. *baseWords* corresponds to transcriptions that still include noise and filler tags, *filtered* correspond to transcriptions that have been cleaned from all annotations of noise parts and filler words.

5.6.3 Iterative Training

Given a considerably large amount of unsupervised training data adequate for full training runs, one major focus was on the evaluation of an iterative batch training approach. The core idea is to iteratively generate new, and hopefully improved automatic transcriptions with each most recent system at a given point in time. By always re-training the ASR-system with the updated transcriptions, system performance is supposed to gradually improve with each iteration.

Formally, the goal was to always perform a full system re-training with a data set

$$T_i = EPPS \cup TED_i^{trans} \quad \forall i \in [1, i_{final}] \tag{5.6}$$

where T_i is the data set for training S_i, $EPPS$ is the training data used for building the baseline system, TED_i^{trans} is a domain specific training set fixed in size, annotated by automatically generated text resulting from a decoding performed by S_{i-1}. Algorithm 5 outlines an iteration of unsupervised training, where the baseline system is denoted as S_0.

The individual training steps for system S_i are performed in lines 10 to 12. MAS training on previously extracted samples SMP_i is followed by OFS training. The final step is a Viterbi training along labels L_i, using training data stored in database DB_i. Practically

Algorithm 5 Iterative Batch Training

Require: $\exists S_0 \wedge i_{final} > 0$
Ensure: $S_{i_{final}}$
1: $i_{final} \leftarrow$ max. iteration
2: $EPPS \leftarrow$ supervised EPPS data
3: $TED \leftarrow$ unsupervised TED data
4: **for** $i = 1$ **to** $i_{final} + 1$ **do**
5: $TED_i^{trans} \leftarrow transcribe(S_{i-1}, TED)$
6: $T_i \leftarrow EPPS \cup TED_i^{trans}$
7: generate training database $DB_i \leftarrow T_i$
8: write labels L_i
9: extract samples SMP_i
10: $S_i^{'} \leftarrow masTrain(SMP_i)$
11: $S_i^{''} \leftarrow ofsTrain(S_i^{'}, SMP_i)$
12: $S_i \leftarrow trainingAlongLabels(S_i^{''}, L_i, DB_i)$
13: **end for**

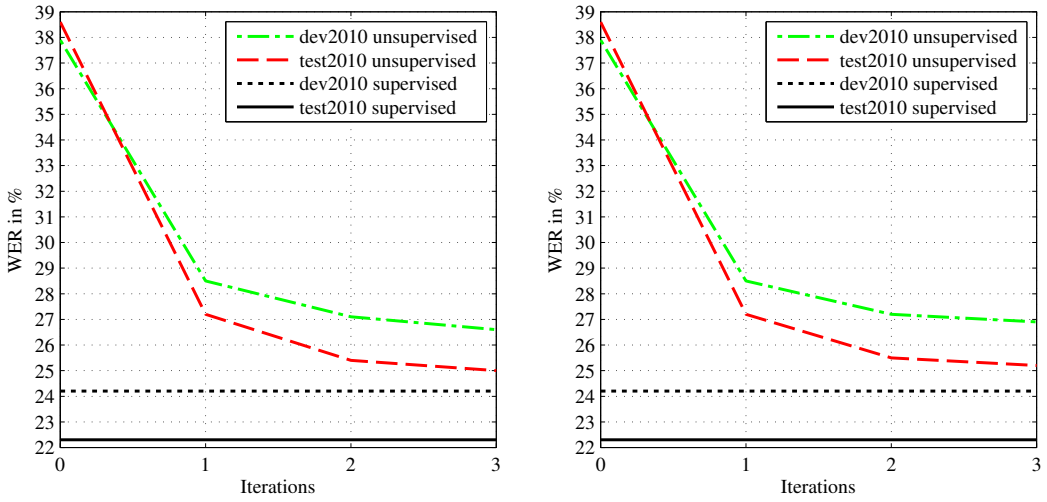
this means that – starting from a baseline system – its acoustic models are completely re-trained with each iteration. I.e., multiple consecutively executed re-trainings of lead to a final system $S_{i_{final}}$.

Without loss of generality, the transcription pre-processing configuration *filtered* has been utilized throughout the iterative batch training experiments. Transcriptions that are cleaned from any noise and filler tags as well as informations of pronunciation variations turned out to result in the most reliable systems, as can be seen in the previous sections.

Figure 5.2 shows the results of testing this approach. Experiments have been conducted by *weighting* the data during training, as well as *thresholding* the data. For the latter, two use cases have been hypothesized: Figure 5.2(a) depicts the progression of the performance curve, a fixed threshold of 50% is applied for each iteration, so that for all intermediate systems every word below this confidence score is exempt from training. Figure 5.2(b) depicts the performance curves, when using a degrading threshold over time. I.e., with every new iteration, the confidence threshold is set to a lower value, so that gradually more data will be effectively utilized for training. The assumption is, that with increasing transcription performance, training data becomes more reliable and less error-prone, thus filtering may be applied in a less strictly fashion. Furthermore, [LZM12] has shown that adding low confidence scored data to training can be beneficial for the recognition correctness rate of a system, following the assumption that this procedure is helpful to improve the system's generalization capabilities. In lowering the threshold, previously unused data with the tendency to be incorrect or not correct to a certain degree becomes available for system training.

As can be seen by reference to both graphics, the recognition performance of the re-trained systems increase considerably even after just one iteration. Particularly, this iteration gives the highest boost of all re-trainings, resulting in relative improvements of more than 29.5% as observed on the test2010 evaluation set. Subsequent iterations still give significant improvements in terms of recognition capabilities, but the gain remains much smaller compared to the initial iteration of unsupervised training. Re-training was halted after the completion of three full iterations due to the emerging convergence of the system's recognition performances.

The iterations plotted in Figure 5.2(a) consistently use a confidence threshold of 50%. As opposed to this, system training during the experiments that yield the performance curves of Figure 5.2(b) makes use of a gradually decreasing threshold: Where for iteration 1 the



(a) weighted, 50%

(b) weighted, decreasing

Figure 5.2: Performance of iterative system re-training in batch mode. Each iteration, the full training set is re-transcribed by the by then most recent system. Each system re-training comprises MAS, OFS and Viterbi training (2 iterations). Performance is plotted in dependency of the iterations and is measured in terms of word error rate (WER). The development systems are compared to *supervisedly* trained reference systems using the *dev2010* and *test2010* data sets, respectively. Figure (a) depicts system performance when using *weighting* and *thresholding* with a threshold of 50%. Figure (b) depicts system performance when using decreasing threshold limits for each iteration: 50% for iteration 1, 25% for iteration 2, 0% for iteration 3.

threshold likewise is set to 50%, it is reduced to 25% for iteration 2, and set to 0% for iteration 3, which practically means that for the final iteration it has been refrained from thresholding. The depicted curves clearly show that the lowering of the applied confidence score threshold for re-training system *(b)* is of no disadvantage for iteration 2, and results in a slightly worse, yet still competitive system after iteration 3. The latter phenomenon may be an indication of a too rapid reduction of the threshold value. With weighting alone, lower-scored training data with erroneously labels still has an impact on model training, where thresholding with even a low value can effectively reduce the risk of adversely biasing the models. Likewise, no observations can be made by reference to the depicted curves that attest an advantage of gradually lowering the applied threshold over keeping a fixed threshold for all iterations of re-training.

Figure 5.3 depicts the gradual shift of word based confidences. With every iteration, higher-valued bins proportionally comprise more data, whereas the lower-valued bins clearly contribute with considerably smaller fractions. It is noteworthy that only the confidence measures for word tokens were taken into account. Non-word tokens, i.e., noise and filler tags were ignored in order to determine the impact of the iterative re-training process on the recognition qualities of the ASR system targeted to words only.

The initial transcriptions that were generated with iteration 1 feature scores that are almost equally distributed in the middle ranges of the confidence bins, with a drop on very low confidences ($[, .05)$) and a peak for very high confidences ($[.95, 1]$), which is reflected in a median of 0.73. With each subsequent iteration, the relative portions of the total mass of confidences shifts to higher bins, which reflects in a median of 0.93 after the 3rd and final iteration. Where for lower confidence ranges the share gradually decreases, the occurrence

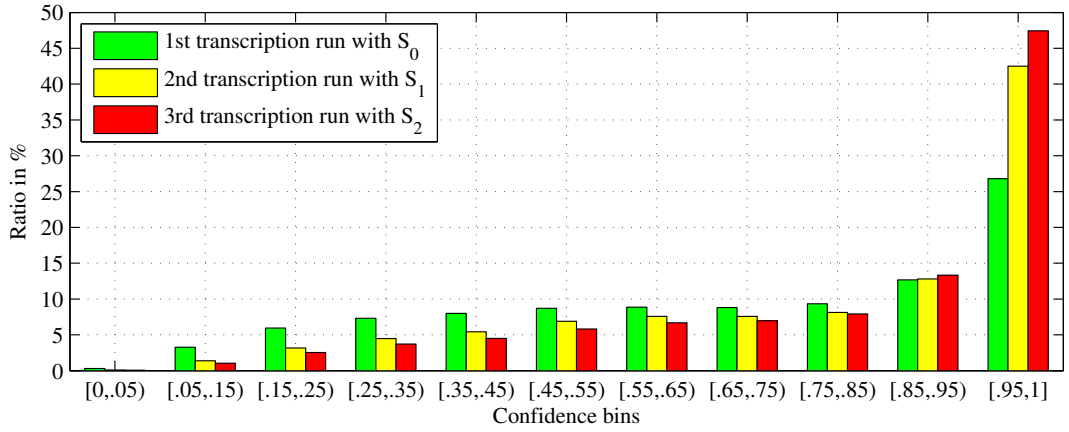


Figure 5.3: Distribution of the word based confidence measures on confidence bins. The distribution is depicted in dependency of the amount of performed re-transcription runs followed by re-training. Clearly visible is the gradual shift of rates for confidence score spans to higher bins.

of very high confidences increases rapidly. It is remarkable that, even though the range of the highest bin has half the regular bin size (as does the lowest bin), it holds more than 47% of all words after iteration 3. This re-distribution of words with initially lower confidence scores to bins covering higher confidence values, accompanied by a decreasing WER indicate that the system learns to produce more reliable recognition hypotheses given the new domain and channel it is adapted to. Even with a fixed threshold value, more data is used during subsequent iterations, as the amount of higher-scored words increases due to the improved confidence of the decoder. Moreover, by lowering the confidence threshold, this effect is further intensified. on the supposition that the decoder's ratings get more reliable with each training iteration, the risk of including potentially harmful data into model re-training increases by lowering the cut-off value. Thus, adjusting the threshold may be helpful to maximize the yield of the training data, but this step should be conducted with care.

5.6.4 Analysis

In this chapter work on unsupervised adaptation of the acoustic model by iterative batch training has been described. The goal was to evaluate how well a baseline system can be adapted to a new domain by utilizing unsupervised training data, fixed in size. For this, automatic transcriptions of the unsupervised data have been generated with the baseline system, which was then used for fully re-training a new system. This training scheme was iterated several times.

Two different ways of processing the decoder output were applied, that were selected according to the results of Chapter 4. It has been shown that both pre-processing methods lead to systems that perform about equally well, where *filtered* transcriptions performed slightly better on the development set. Although the final Viterbi training already showed signs of imminent over-fitting, the results indicate that transcriptions following the *base-Words* configuration have a slight advantage over *filtered* transcriptions in terms of robustness towards over-fitting when taking the results on both test sets into consideration. An improvement of up to 22.1% relative on the development set and 26.6% relative on the test set was observable after only one iteration of unsupervised training with applied transcription pre-processing.

During further experiments, word level posterior probabilities that were obtained during

automatic recognition runs on the unsupervised training data were utilized for word based weighting and thresholding. Transcriptions that were *filtered* proved to be superior on the development set in all tested combinations of weighting and thresholding. All applied techniques led to a significant gain in performance by up to 3.7% relative, compared to training without application of confidence measure based filtering. The positive effects of thresholding the data during training were higher than weighting the data. A threshold of 50%, which correlated with the highest value that has been tested, resulted in the system with the lowest WER on the development set, being 28.4%. The combination of weighting and thresholding lead to a competitive system on the development set, and to the winning system on the test set, yielding a WER of 27.2%.

By reference to the observations made it is of advantage to let the Viterbi algorithm decide which pronunciations to use and where to insert which noise words, establishing conformity with the analysis of Chapter 4. Given a considerable amount of unsupervised data it is beneficial to set a confidence score threshold to a value not lower than 50%.

For iterative batch training, two use cases have been hypothesized: Using a fixed threshold of 50% for each iteration, and using a threshold that degrades over time. It has been shown that a fixed threshold is of advantage. However, the alternative approach led to a competitive system and is potentially improvable by vernier adjustment of the threshold degradation speed. Recognition performance of the re-trained systems increased considerably even after a single iteration. With each subsequent iteration, higher-valued confidence score bins proportionally comprise an increasing amount of data, accompanied by a decreasing WER in tests prove that with each iteration the system get more reliable given the respective speaker they are trained to.

The iterative re-distribution of word confidences to higher confidence bins and likewise decreasing WER in tests indicate that the system qualifies to generate more reliable recognition hypotheses given the new domain and channel it is adapted to. Even with a fixed threshold value, more data is used during subsequent training iterations, which implicitly leads to an increased yield of the available data. The final system with highest performance on the test set used word *weighting* and a threshold of 50% and yielded a WER of 25.0% after 3 iterations of unsupervised training.

6. Summary

The goal of this thesis was to application and evaluation of unsupervised acoustic model training schemes, that lay the foundation for an unsupervised adaptation framework based on acoustic model training for use in KIT's simultaneous speech-to-speech lecture translation system. The focus during experimental test runs and evaluation on the KIT lecture translator system was on speaker adaptation via unsupervised training. Iterative and incremental training principles were compared with respect to the training efficiency and overall recognition performance after training, given certain minimal amounts of data. The focus of the experiments on a baseline system trained on EPPS data was the application of unsupervised iterative batch training for domain adaptation.

Given these two experimental scenarios, the impact of various transcription pre-processing methods and confidence measure based data filtering methods during acoustic model training was evaluated. Experiments showed that it is of advantage to let the Viterbi algorithm during training decide which pronunciations to use and where to insert which noise words, instead of fixating these informations in the transcriptions. With weighting and thresholding it was possible to improve unsupervised training in all test cases. The best system based on the KIT lecture translator reached a word error rate of 16.9% with processed transcriptions and weighted training. Tests with an iterative incremental approach showed that potential performance gains strongly correlate to the performance of the systems for automatic transcription. For the EPPS-based system, significant performance gains of up to 29.5% relative was observable after only one iteration of unsupervised training with applied transcription pre-processing, weighting and thresholding. Given a considerable amount of unsupervised data it is beneficial to set a confidence score threshold to a value not lower than 50%. For iterative batch training it has been shown that a fixed threshold is of advantage.

6.1 Future Work

The experimental results of this study showed, that unsupervised acoustic model training can effectively be applied to the tasks of speaker an domain adaptation. Already small amounts of new data were sufficient for obtaining significant improvements of the baseline systems. Future work will comprise the implementation and integration of an unsupervised acoustic model training framework into the KIT lecture translator system for automatic model adaptation. Further research will focus on comparisons of well approved adaptation techniques and adaptation via re-training, with a special focus on the dependency of adaptation effectiveness and amount of available unannotated training data.

Bibliography

[BT97] A. W. Black and P. A. Taylor, "The Festival Speech Synthesis System: System documentation," Human Communciation Research Centre, University of Edinburgh, Scotland, UK, Tech. Rep., 1997, avaliable at http://www.cstr.ed.ac.uk/projects/festival.html.

[Bur97] S. Burger, "Transliteration spontanprachlicher daten - lexikon der transliterationskonventionen - verbmobil ii," 1997. [Online]. Available: http://www.phonetik.unimuenchen.de/Forschung/Verbmobil/VMtrlex2d.html

[CFH+12] E. Cho, C. Fügen, T. Hermann, K. Kilgour, M. Mediani, C. Mohr, J. Niehues, K. Rottmann, C. Saam, S. Stüker, and A. Waibel, "A real-world system for simultaneous translation of german lectures," interACT, Karlsruhe Institute of Technology, Tech. Rep., December 2012.

[CLG04] L. Chen, L. Lamel, and J. Gauvain, "Lightly supervised acoustic model training using consensus networks," in *Acoustics, Speech, and Signal Processing, 2004. Proceedings. (ICASSP '04). IEEE International Conference on*, vol. 1, 2004, pp. I–189–92 vol.1.

[CMU] "The carnegie mellon university pronouncing dictionary," http://www.speech.cs.cmu.edu/cgi-bin/cmudict, CMU.

[CSZ10] O. Chapelle, B. Schölkopf, and A. Zien, *Semi-supervised learning*, ser. Adaptive Computation and Machine Learning Series. Mit Press, 2010.

[FÖ8] C. Fügen, "A system for simultaneous translation of lectures and speeches," Ph.D. dissertation, Universiät Karlsruhe (TH), November 2008.

[FGH+97] M. Finke, P. Geutner, H. Hild, T. Kemp, K. Ries, and M. Westphal, "The Karlsruhe-Verbmobil speech recognition engine," 1997.

[Fis97] J. G. Fiscus, "A post-processing system to yield reduced word error rates: Recognizer output voting error reduction (ROVER)," jul 1997.

[FMS] C. Fügen, F. Metze, and H. Soltau, "Jrtk and janus."

[FSGL11] T. Fraga-Silva, J.-L. Gauvain, and L. Lamel, "Lattice-based unsupervised acoustic model training," in *Proceedings of the ICASSP 2011*, Prague, Czech Republic, May 2011.

[FW97] M. Finke and A. Waibel, "Flexible transcription alignment," in *IEEE Workshop on Speech Recognition and Understanding*. IEEE, 1997, pp. 34–40.

[FWK07] C. Fügen, A. Waibel, and M. Kolss, "Simultaneous translation of lectures and speeches," *Machine Translation*, vol. 21, pp. 209–252, 2007.

[Gal97] M. Gales, "Maximum likelihood linear transformations for hmm-based speech recognition," Cambridge University, Engineering Department, Tech. Rep., May 1997.

[Gal99] ——, "Semi-tied covariance matrices for hidden markov models," *IEEE Transactions on Speech and Audio Processing*, vol. 7, no. 3, pp. 272–281, 1999.

[GB08] C. Gollan and M. Bacchiani, "Unsupervised acoustic model training," in *Proceedings of the ICASSP 2008*, Las Vegas, NV, USA, March 2008.

[GBK+05] C. Gollan, M. Bisani, S. Kanthak, R. Schluter, and H. Ney, "Cross domain automatic transcription on the TC-STAR EPPS corpus," in *Proc. of ICASSP*, Philadelphia, USA, 2005, pp. 825–828.

[GHSN07] C. Gollan, S. Hahn, R. Schlüter, and H. Ney, "An improved method for unsupervised training of lvcsr systems," in *Proceedings of the INTERSPEECH 2007*, Antwerp, Belgium, August 2007.

[Jel76] F. Jelinek, "Continuous speech recognition by statistical methods," *Proceedings of the IEEE*, vol. 64, no. 4, pp. 532–556, 1976.

[Kea97] T. Kemp and et al., "Estimating confidence using word lattices," 1997.

[KFN98] T. Kaukoranta, P. Fränti, and O. Nevalainen, "Iterative split-and-merge algorithm for VQ codebook generation," *Optical Engineering*, vol. 37, no. 10, pp. 2726–2732, 1998.

[KGS+98] T. Kemp, P. Geutner, M. Schmidt, B. Tomaz, M. Weber, M. Westphal, and A. Waibel, "The interactive systems labs view4you video indexing system," in *ICSLP*. ISCA, 1998.

[KIT] "Ifa - institute for anthropomatics," KIT. [Online]. Available: http://www.informatik.kit.edu/english/1323.php

[KW99] T. Kemp and A. Waibel, "Unsupervised training of a speech recognizer using tv broadcasts," in *Proceedings of the EUROSPEECH 1999*, Budapest, Hungary, September 1999.

[LGA02] L. Lamel, J.-L. Gauvain, and G. Adda, "Lightly supervised and unsupervised acoustic model training," *Computer Speech & Language*, vol. 16, no. 1, pp. 115–129, 2002.

[LlGA02] L. Lamel, J. luc Gauvain, and G. Adda, "Unsupervised acoustic model training," in *Proceedings of the ICASSP 2002*, Orlando, Florida, USA, May 2002.

[LWL+97] A. Lavie, A. Waibel, L. Levin, M. Finke, D. Gates, M. Gavalda, and P. Zhan, "JANUS-III: Speech-to-speech translation in multiple languages," 1997.

[LZM12] H. Li, T. Zhang, and L. Ma, "Confirmation based self-learning algorithm in lvcsr's semi-supervised incremental learning," *Procedia Engineering*, vol. 29, pp. 754–759, 2012.

[MMKS06] J. Ma, S. Matsoukas, O. Kimball, and R. Schwartz, "Unsupervised training on large amounts of broadcast news data," in *Acoustics, Speech and Signal Processing, 2006. ICASSP 2006 Proceedings. 2006 IEEE International Conference on*, vol. 3, 2006.

[Nie90] H. Niemann, *Pattern analysis and understanding*, ser. Springer series in information sciences. Springer-Verlag, 1990.

[Rab89] L. R. Rabiner, "A tutorial on hidden Markov models and selected applications in speech recognition," *Proceedings of the IEEE*, vol. 77, no. 2, pp. 257–286, 1989.

[Rog05] I. Rogina, "Sprachliche mensch-maschine-kommunikation," 2005.

[SFKW07] S. Stüker, C. Fügen, F. Kraft, and M. Wölfel, "The isl 2007 english speech transcription system for european parlimentary speeches," in *In Proc. Interspeech*, 2007.

[SKK12] S. Stüker, K. Kilgour, and F. Kraft, "Quaero 2010 speech-to-text evaluation systems," in *High Performance Computing in Science and Engineering '11*, W. E. Nagel, D. B. Kröner, and M. M. Resch, Eds. Springer Berlin Heidelberg, 2012, pp. 607–618.

[SKM+12] S. Stüker, F. Kraft, C. Mohr, T. Herrmann, E. Cho, and A. Waibel, "The kit lecture corpus for speech translation," in *LREC 2012*, Istanbul, Turkey, May 2012.

[SKN11] S. Stüker, K. Kilgour, and J. Niehues, "Quaero speech-to-text and text translation evaluation systems," in *High Performance Computing in Science and Engineering '10*, W. E. Nagel, D. B. Kröner, and M. M. Resch, Eds. Springer Berlin Heidelberg, 2011, pp. 529–542.

[SMFW01] H. Soltau, F. Metze, C. Fügen, and A. Waibel, "A one-pass decoder based on polymorphic linguistic context assignment," 2001.

[SPK+07] S. Stüker, M. Paulik, M. Kolss, C. Fügen, and A. Waibel, "Speech translation enhanced asr for european parliament speeches - the influence of asr performance on speech translation," in *Proceedings of the 2007 IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 4. Honolulu, HI, USA: IEEE, April 2007, pp. 1293–1296.

[ST95] E. G. Schukat-Talamazzini, *Automatische Spracherkennung - Grundlagen, statistische Modelle und effiziente Algorithmen*, ser. Künstliche Intelligenz. Braunschweig: Vieweg, 1995.

[Sto02] A. Stolcke, "SRILM-an extensible language modeling toolkit," in *In Proceedings of the 7th International Conference on Spoken Language Processing (ICSLP 2002)*, 2002, pp. 901–904.

[WM05a] M. Wölfel and J. McDonough, "Minimum variance distortionless response spectral estimation," *Signal Processing Magazine, IEEE*, vol. 22, no. 5, pp. 117 – 126, sept. 2005.

[WM05b] ——, "Minimum variance distortionless response spectral estimation, review and refinements," *IEEE Signal Processing Magazine*, vol. 22, no. 5, pp. 117–126, September 2005.

[WN05] F. Wessel and H. Ney, "Unsupervised training of acoustic models for large vocabulary continuous speech recognition," *IEEE Transactions on Speech and Audio Processing*, vol. 13, no. 1, pp. 23–31, 2005.

[You96] S. Young, "Large vocabulary continuous speech recognition: a review," Cambridge University Engineering Department, Tech. Rep., aug. 1996.

[ZaTCB98] G. Zavaliagkos, M.-H. S. abd Thomas Colthurst, and J. Billa, "Unsupervised acoustic model training," in *Proceedings of the ICSLP 1998*, Sydney, Australia, November 1998.

[ZW97] P. Zhan and M. Westphal, "Speaker normalization based on frequency warping," in *Proc. of ICASSP*, Munich, Germany, April 1997, pp. 1039–1042.