Rheinisch-Westfälische Technische Hochschule Aachen
Lehrstuhl für Informatik VI
Prof. Dr.-Ing. Hermann Ney

Diplomarbeit

Lattice Rescoring and Log-Linear System Combination

vorgelegt von:

Cand. Inform. André Altmann

Matrikelnummer 227 851

September 1, 2005

Gutachter:

Prof. Dr.-Ing. H. Ney
Prof. Dr. G. Lakemeyer

Betreuer:

Dipl.-Inform. Björn Hoffmeister

Hiermit versichere ich, daß ich die Arbeit selbständig verfaßt und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt, sowie Zitate kenntlich gemacht habe.

Aachen, den September 1, 2005

(André Altmann)

Abstract

This work describes a method to reduce the recognition errors of large vocabulary speech recognition systems. To achieve a reduction in the *word error rate* a combination of different recognition systems is performed. The combination will be carried out at the level of so-called word lattices. An initial recognizer is used to generate these lattices. The other systems will be applied in the form of a *lattice rescoring*. This results in a set of lattices with the same structure but with different scores at the arcs.

In the standard approach the recognized word sequence is represented by the best scoring path in the word lattice. Within the approach presented in this work, the recognized word sequence is represented by a the best scoring path in a combination of the word lattices for every system. Therefor the generated lattices are accumulated with respect to the underlying system.

Since the influence of the applied systems is crucial for the success of this method, discriminant methods are utilized to estimate weights for each system. These weights are used to handle the effect of the individual systems on the final result.

Zusammenfassung

Der in dieser Arbeit verfolgte Ansatz beschäftigt sich mit der Minimierung der Fehlerrate von Spracherkennern mit großem Vokabular. Um eine Reduktion der *Wortfehlerrate* zu erreichen, wird eine Kombination aus verschiedenen Erkennersystemen angestrebt. Die Kombination wird auf der Ebene von so genannten Wortgraphen ausgeführt. Ein einfacher Spracherkenner wird für die Generierung dieser Wortgraphen verwendet. Die Graphen der anderen Modelle werden durch eine Neubewertung der auf diese Weise erhaltenen Wordgraphen erzeugt. Das Resultat sind Wordgraphen mit gleicher Struktur aber unterschiedlichen Kantengewichten.

Bei Standardverfahren wird die erkannte Wortfolge durch den Pfad mit den geringsten Kosten in einem Wortgraphen dargestellt. Bei dem in dieser Arbeit verfolgten Ansatz repräsentiert der Pfad mit den geringsten Kosten in einer Kombination der beteiligten Wortgraphen die gesuchte Wortfolge. Hierzu werden die verwendeten Graphen nach der Erzeugung in abhängigkeit des zu Grunde liegenden Systems für jedes Sprachsegment akkumuliert.

Da der Einfluß der verwendeten Systeme entscheidend für das Gelingen dieser Methode ist, werden mit Hilfe diskriminativer Methoden Gewichte für jedes teilnehmende System ermittelt, deren Auswirkung auf diese Art geregelt wird.

Ich ging runter zu Monsieur Ibrahim, der lächelnd ein paar Erdnüsse kaute.
“Wie schaffen Sie es, Monsieur Ibrahim, glücklich zu sein?”
“Ich weiß, was in meinem Koran steht.”
“Vielleicht sollte ich Ihnen eines Tages Ihren Koran klauen. Aber als Jude darf man das ja nicht.”
“Hhm. Was bedeutet es denn für dich, Momo, ein Jude zu sein?”
“Tja, ich weiß nicht. Für meinen Vater bedeutet das, den ganzen Tag lang deprimiert zu sein. Für mich ... nur etwas, das mich daran hindert, etwas anderers zu sein.”
Monsieur Ibrahim gab mir eine Erdnuß.
“Deine Schuhe sind hinüber, Momo. Wir gehen dir morgen ein Paar neue kaufen.”
“Ja aber ...”
“Ein Mensch verbringt sein Leben in nur zwei Stätten: Entweder in seinem Bett oder in seinen Schuhen.”
“Ich habe kein Geld, Monsieur Ibrahim.”
“Ich werde sie bezahlen. Ein Geschenk von mir. Momo, du hast nur ein Paar Füße, man muß auf sie aufpassen. Drücken dich die Schuhe, wechsel sie. Füße kannn man nicht wechseln.”

– Ein Auszug aus “Monsieur Ibrahim und die Blumen des Koran”

Eric-Emmanuel Schmitt

Contents

List of Tables

List of Figures

// Danksagungen

An dieser Stelle möchte ich allen Menschen danken, die mich bei dieser Arbeit unterstützt und begleitet haben.

Zuallererst geht mein Dank an Prof. Dr.-Ing. H. Ney für die Einführung in die automatische Spracherkennung und für die Möglichkeite diese Arbeit am *Lehrstuhl für Informatik VI* durchzuführen.

Weiterer Dank geht an Prof. Dr. Lakemeyer für die Übernahme der Zweitkorrektur.

Björn Hoffmeister möchte ich für die hervorragende Betreuung während dieser Arbeit danken.

Georg Heigold gehört mein Dank für die unermüdliche Beantwortung meiner Fragen.

Danke an alle Mitarbeiter des *Lehrstuhls für Informatik VI* für das Angenehme Arbeitsklima.

Den anderen Diplomanden am *Lehrstuhl für Informatik VI* – Stefan Hahn, Andre Hegerath, Gregor Leusch, Arne Mauser, Thomas Scharrenbach, Thomas Schönemann und Daniel Stein – danke ich für die vielen Tipps, unzählige Runden “Bomberclone” und dafür, die Arbeitstage kürzer und lustiger gemacht zu haben.

Meinen aachener Studienkollegen danke ich für die schöne Zeit und die immer willkommene Ablenkung.

Ich danke meinen langjährigen Freunden in Leverkusen – für ihre Freundschaft.

Very special thanks go to Mr. Wataru Shoji.

Mein besonderer Dank gilt meinen Eltern, Bärbel Behr und Günter Göbel, die mir das Studium ermöglicht und mich die ganzen Jahre unterstützt haben.

1. Introduction

Speech recognition is the task of converting spoken language into sequences of written words that are readable for humans and/or machines. Automatic speech recognition systems slowly merge into daily life, as they are used to control devices or retrieve informations via expert systems. For example speech recognition is applied in some cars to change settings of i.e. the radio, or to provide "hands-free" dialing in mobile phones. But so-called command-and-control tasks are not the only application for speech recognition. It builds the core of automatic dictation systems, which could be seen as verbal typewriters. In fact, this is the first application people think of, when hearing about speech recognition. Computers will become a larger part in every ones life and so does speech recognition, because speaking is a very convenient and intuitive method to pose commands to a computer or a device connected to one, such as voice controlled ticket vending machines. Combining speech recognition with methods from machine translation and speech synthesis, allows it to translate a spoken sentence into a foreign language, the speaker himself neither speaks nor understands.

Purpose of speech recognition systems is to transform an acoustic utterance into a sequence of words, this is done by assigning all parts of the utterance to the correct class. Such classes are formed by words or subword units like phonemes. This process is called classification, and can be in general be performed in two different ways. One follows the methods from statistical pattern recognition, the other one is the application of knowledge based techniques, which are widely used in the field of artificial intelligence. Due to better performance, the statistical approach still offers the basics for state-of-the-art speech recognition systems. Research on automatic transcription of spoken words began in the 1950s [Davis & Biddulph$^+$ 52, Fry 59], but in the last three decades the largest progress was achieved, beginning with the introduction of hidden Markov models [Baker 75, Jelinek 76]. Hidden Markov models are stochastic finite automata that are used to model the acoustic of spoken language. An automatic speech recognition system has to be able to recognize every voice respectively the spoken words of every speaker, despite the different characteristics like speech rate and sound. To remove speaker individual features of an utterance, a number of techniques were introduced. Application of these techniques actually do improve the average performance of a recognizer but on some utterances or parts of utterances a recognizer with less sophisticated techniques may perform better. Due to that reason, recent developments dealt with combining different outputs from recognition systems.

1.1. Related Work

The first method, that was used to combine recognized word sequences, has been introduced in 1997, and is called ROVER. The abbreviation ROVER stands for *Recognition Output Voting Error Reduction* [Fiscus 97], where a reduction of the word error rate from 44.9% to 39.7 % on the 1997 HUB5 English task was reported. ROVER is only capable of combining outputs given as a sequence of words. Another method was proposed in 2000 and also takes alternative hypotheses of recognizers into account. Alternative hypotheses are word sequences that are not chosen by the recognizer as the best output, but are very close to be selected. The hypotheses are provided in the form of *confusion networks* [Mangu & Brill+ 99], therefor the approach is called *confusion network combination* [Evermann & Woodland 00], improvements of the word error rate are reported [Soltau & Yu+ 04, Evermann & Chan+ 04]. Chapter 3 provides a detailed description of theses techniques. The optimal algorithm to combine systems would act like an oracle, telling for every proposed word, which one should be used to maximize the performance of the overall system. A further approach tries to combine different models that are used as knowledge bases to improve the classification accuracy of the recognizer and is called *discriminant model combination* [Beyerlein 00], a reduction of the word error rate is reported. The theory of discriminant model combination is introduced in chapter 4.

1.2. Scientific Goals

This work will investigate a new method for combining speech recognition systems. Outputs of different recognition systems are stored in the form of *word lattices*. These are combined using the discriminant model combination. Word lattices contain all hypotheses that are considered by the recognizer during the decision process, therefor alternatives are considered while carrying out the combination step. Since the applied discriminant model combination is crucial for the success of that approach it has to work reliably to ensure good recognition results. Two different criteria used within discriminant model combination will be tested in this work: (1) the *maximum mutual information* criterion, which is funded in information theory, (2) the *minimum word error* criterion, which directly tries to minimize the number of incorrectly recognized words.

As it is very time consuming to generate the word lattices for every system by a full recognition process, this is only done for an initial system, and the word lattices for the other systems are generated using a technique called *lattice rescoring*. Rescoring uses the results of a prior recognition and calculates for the stored hypotheses new scores using other knowledge sources, thus a recognition process is simulated.

The lattice-based system combination will be tested on three different corpora: HUB4 English, EPPS English and Spanish. The audio data of the HUB4 task is taken from American broad cast news. The EPPS data consists of recorded debates in the European Parliament.

The scientific goals are summarized as follows:

- **Model selection**
  A tool shall be created, that provides information about the usability of a pair-wise combination of system, without actually carrying out the corresponding experiment. An overestimated performance is given by calculating an optimized error rate: it suffices if one of the systems decides for the correct word. That tool can be used for the selection of the models, that are used in the lattice-based system combination, because if the generated outputs of the systems are very similar, the optimized error rate will not deviate much from the error rate for the single outputs, thus no great benefit can be expected from that combination.

- **System Training**
  Some of the systems, that shall be combined, have to be trained, since they are not available at the beginning of this work. Creation of all HUB4 systems has to be carried out, and some of the acoustic models for the EPPS task have to be generated. The resulting acoustic models are used to build up individual speech recognition systems, that will be combined at a later stage of this work.

- **Acoustic rescoring**
  Acoustic across-word rescoring has to be implemented and introduced into the speech recognition system SPRINT of the "Lehrstuhl für Informatik VI", to provide an efficient method of generating the word lattices needed for the combination method.

- **Generation of an environment for lattice-based system combination**
  An environment that offers an convenient way to perform the steps that are involved in the investigated approach, such as rescoring, training of system weights, and the final evaluation has to be implemented.

- **Investigation on different training criteria**
  The comparative study of the performance of two training criteria, *maximum mutual information* and *minimum word error*, used to train the weights for the system combination.

- **Performance of lattice-based system combination**
  The available standard methods used to create different recognition systems are combined using the lattice-based system and the performance will be examined on different corpora: HUB4 English, EPPS English and Spanish.

2. Automatic Speech Recognition

The general setup for automatic speech recognition starts from training data consisting of an audio speech sequence together with the corresponding transcription. In the first step acoustic features are extracted from the audio data used to train one or more acoustic models. A language model is estimated on large running texts including but not restricted to the given transcription. Additional parameters are adjusted on a so-called development set consisting of audio and transcription, like the training data. Evaluation of the recognizer is then carried out on a test set, where no transcription is available during the recognition. The recognizer searches for the best transcription according to the probabilistic models. Using the reference transcription of the test set, the quality of the recognizer's output can be measured.

In the statistical approach of automatic speech recognition the search for the sequence of spoken words is based on *Bayes' decision rule* [Duda & Hart 73]. According to this rule, the word sequence, $w_1^N (:= w_1 \ldots, w_N)$, with the highest posterior probability with respect to the speech signal, minimizes the recognition errors. Though this is only true, if the true probability distributions are used, automatic speech recognition systems are still based on Bayes' decision rule. As a measure for the recognition errors the *edit distance* or *Levenshtein distance* [Levenshtein 65] is used. These express how many *edit operations* (*insertions, deletions, substitutions*) are needed, to transform the recognized word sequence, $\hat{w}_1^N$, into the correct word sequence, $\tilde{w}_1^{\tilde{N}}$. The number of edit operations is counted and divided by the number of words in the correct sentence $\tilde{N}$, the resulting measure is called *word error rate (WER)*.

A sequence of features, denoted with x_1^T, is extracted from the digitalized audio signal, using this features, the Bayes' decision rule can be expressed as follows:

$$\begin{aligned}
\hat{w}_1^N &= \operatorname*{argmax}_{w_1^N} \left\{ Pr(w_1^N | x_1^T) \right\} \qquad (2.1)\\
&= \operatorname*{argmax}_{w_1^N} \left\{ Pr(x_1^T | w_1^N) Pr(w_1^N) / Pr(x_1^T) \right\}\\
&= \operatorname*{argmax}_{w_1^N} \left\{ Pr(x_1^T | w_1^N) Pr(w_1^N) / \sum_{{w'}_1^{N'} \in \mathcal{M}} Pr(x_1^T | {w'}_1^{N'}) Pr({w'}_1^{N'}) \right\}\\
&= \operatorname*{argmax}_{w_1^N} \left\{ Pr(x_1^T | w_1^N) Pr(w_1^N) \right\},
\end{aligned}$$

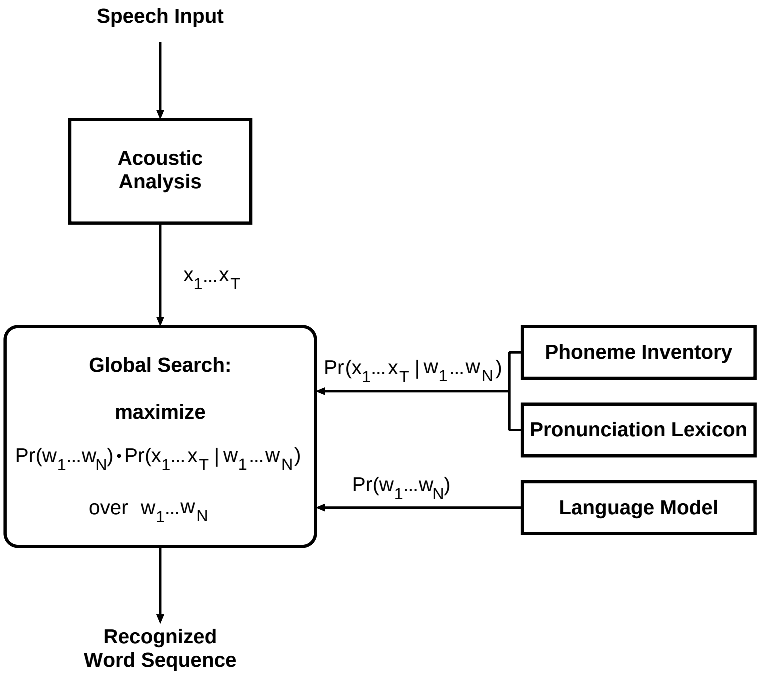

Figure 2.1.: Architecture of an automatic speech recognition system

where $Pr(w_1^N|x_1^T)$ denotes the true posterior distribution. The *Bayesian identity* is used to factorize it by an a-priori-distribution for the word sequence the language model $Pr(w_1^N)$ and a posterior distribution for the sequence of features x_1^T given a sequence of words w_1^N the acoustic model $Pr(x_1^T|w_1^N)$. The a-priori-distribution for the sequence of features can be expressed, as the sum over the probability of all possible word sequences ${w'}_1^{N'}$. Because the formula includes the argument maximum $\text{argmax}_{w_1^N}$, $Pr(x_1^T)$ can be omitted, as it remains constant with respect to w_1^N and does not influence the decision.

Due to the fact that the true distributions are not known, the probability of observing x_1^T given w_1^N $Pr(x_1^T|w_1^N)$ is modeled by an *hidden Markov model (HMM)* [Jelinek 76, Levinson & Rabiner$^+$ 83, Rabiner 89, van Kampen 92], where ϑ is the set of HMM-parameters, so that $Pr(x_1^T|w_1^N) \approx p_\vartheta(x_1^T|w_1^N)$. The Bayes' decision rule which is applied in an automatic speech recognition system has following form:

$$\hat{w}_1^N = \underset{w_1^N}{\text{argmax}} \left\{ p_\vartheta(x_1^T|w_1^N) Pr(w_1^N) \right\} \tag{2.2}$$

The parameters of the HMM have to be trained to approximate the true distribution (chapter 2.5) and subsequently to raise the performance of the speech recognizer which bases its decisions on the emission probabilities of the HMM.

2.1. Acoustic Analysis

The *acoustic analysis*, also called *feature extraction process*, is the step where the raw, digitized audio sample represented by a sequence of 32-bit values is transformed into a sequence of *feature vectors* x_1^T:

$$x_1^T = x_1, x_2, \ldots, x_T \quad \text{with} \quad x_t \in \mathbb{R}^D,$$

where D is the dimension of the *feature space*. Typically values for D range from 16 to 50. These features are the actual input for the recognition process.

Aim of this step is the extraction of discretized features that are transformed to optimize the classification, as well as to suppress portions of the signal which are irrelevant or disturbing to the recognition process, background noises for example. The feature extraction step can be used to extract/remove speaker dependent characteristics which is done during the so called *Vocal Tract Length Normalization* [Wakita 77, Pitz & Ney 03]. Feature vectors of low dimensionality allowing for a good distinction between the actually spoken phonemes should be generated. Typically a feature vector is extracted every ten milliseconds.

The feature extraction process begins with the generation of the signal's preemphasis, that is done by calculating the difference of consecutive samples s_n:

$$d_n = s_n - s_{n-1}$$

On this differentiated signal a 25 millisecond wide *Hamming window* is applied every ten milliseconds,$t = 1, \ldots, T$, in order to gain overlapping blocks, which are called *time frames*. The windowing is utilized because speech signals remain almost stationary within such a period:

$$x_{nt} = s_{nt} \cdot h_n = s_{nt} \left[0.54 - 0.46 \cos\left(\frac{2n\pi}{N_s - 1} \right) \right],$$

where N_s is the number of the samples, and $n \in \{1, \ldots, N_s\}$.

The application of a *fast Fourier transformation (FFT)* [Cooley & Tukey 65] on every time frame results in it's magnitude spectrum.

$$X_{kt} = \sum_{n=0}^{N_s} x_{nt} \cdot \exp\left\{ -j \frac{2\pi}{N_s} \cdot k \cdot n \right\}$$

To model the effect, that the frequency resolution of the human ear decreases towards higher frequencies, the axis f is warped by applying a logarithmic function:

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

$$X_{kt} \rightarrow X_{\tilde{k}t}$$

This step is called *Mel frequency warping*.

A bandpass filter bank is applied after the Mel frequency warping. The filter bank consists of overlapping, equally spaced triangular filters ($a_{\tilde{k}i}$). Output of the filter bank are logarithmized, to model the nonlinearity dependency between the sensed loudness and the intensity of a signal.

$$Y_{it} = \log \sum_{\tilde{k}} |X_{\tilde{k}t}| \cdot a_{\tilde{k}i}$$

The discrete cosine transformation is applied to the outputs of the logarithmized filter banks, resulting in M Mel-frequency cepstral coefficients:

$$c_{mt} = \sum_{i=0}^{I-1} \cos\left(\frac{\pi m(i + 0.5)}{I}\right) Y_{it}$$

To remove influences from an unknown constant transmission channel a simple and effective method is the subtraction of the mean over all cepstrum coefficients, $\overline{c_m}$:

$$c'_{mt} = c_{mt} - \overline{c_m} = c_{mt} - \frac{1}{T} \sum_{\tau=1}^{T} c_{m\tau}$$

Energy normalization is done by normalizing the 0th cepstrum coefficient, which represents the logarithmic energy:

$$c''_{mt} = \begin{cases} c'_{mt} - \max_\tau c'_{m\tau} & m = 0 \\ c'_{mt} & else \end{cases}$$

Typically only the first 16 cepstral coefficients are used as features, $c''_t = (c''_{1t}, \ldots, c''_{16t})$. In addition dynamic features are added, by appending the first and second order derivatives of c''_t to the feature vector:

$$\Delta c''_t = c''_t - c''_{t-\delta}$$

$$\Delta\Delta c''_t = c''_{t-\delta} - 2\dot{c}''_t + c''_{t+\delta},$$

where $\delta \in \{2, 3\}$ if a *time shift* of ten milliseconds is used. Finally, the features for recognition have following form:

$$x_t := \begin{bmatrix} c_t^{''} \\ \Delta c_t^{''} \\ \Delta\Delta c_t^{''} \end{bmatrix}$$

After these steps the resulting features are called *Mel-frequency cepstral coefficient (MFCC)* features, which are used in the baseline setup of the RWTH speech recognition system.

Linear Discriminant Analysis (LDA) [Fukunaga 90] represents the last step of the feature extraction process in the RWTH system, the features space is transformed using a matrix that reduces the dimension of the feature space and maximizes the separability of the acoustic classes. LDA is applied on a window of nine feature vectors, therefor the derivatives of the cepstral coefficients can be learned during the estimation of the LDA. Due to this reason, the derivatives are not part of the feature extraction process carried out during recognition.

2.2. Acoustic Model

The probability $Pr(x_1^T|w_1^N)$ of observing the features of an utterance with length T (x_1^T), given a sequence of N words (w_1^N), which is used by Bayes' decision rule (equation 2.1) during the search process. On tasks with a small vocabulary such as command-and-control tasks or the recognition of single digits so-called whole word models can be used. On larger tasks, with i.e. 10K words another method for modelling should be chosen. For tasks with a large vocabulary subword units, such as syllables or phonemes, are used due to training data sparseness. As phonemes are the smallest units of sounds that can change the meaning of a word, they are employed in modelling the emission probability. Depending on the language the phoneme inventory consists of 40 to 50 phonemes. Pronunciation lexica describe, which phoneme sequences correspond to a word. An entry in a pronunciation lexicon has following appearance: "n eh t w er k s" for networks. Because words in are modeled as sequences of phonemes, a recognition of words that were not seen during the training process is possible, since corresponding phonemes were seen. This allows a convenient way to extend the vocabulary, the set of recognizable words, of a speech recognition system.

Although phonemes represent the smallest units of spoken language, the actual pronunciation of a phoneme heavily depends on its context, i.e. the preceding and succeeding phonemes. Due to this fact, state-of-the-art Large Vocabulary Recognition Systems model phonemes within a *triphone* instead of a *monophone* context, to In large vocabulary recognition tasks the triphone context is usually modeled across-word boundaries

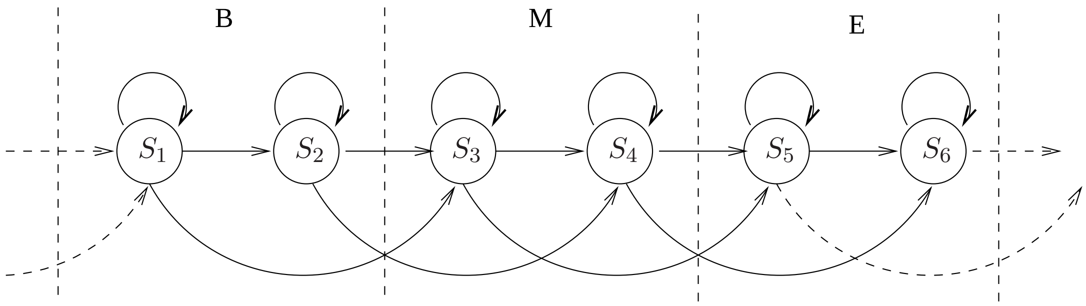

Figure 2.2.: 6-state model

(*across-word-models*), opposite to *within-word-models* used in command-and-control tasks. Triphones are denoted with the central phone and the two adjacent phonemes as subscripts before and after the central phoneme, i.e: the phoneme “er” in the above example with it’s left and right neighbors “w” and “k” is denoted as $_wer_k$.

Triphones are modeled as *hidden Markov models*, each triphone consists of three phases, a beginning, middle, and end phase. Typically every phase is represented by one or two states, therefore these models are called 3-state or 6-state models. All states of a phase share the same emission probability distribution. According to the Bakis topology [Bakis 76] from each state there are three possible transitions: *loop, forward, skip.*

Hidden Markov models are small stochastic finite state automata, defining a set of states and transitions. Transitions are used to model variations in speaking rate, since the HMM is a stochastic automaton, transitions have a certain probability. Additional to the *transition probabilities*, every state s is able to emit a feature vector x with a certain probability *(emission probabilities*). HMMs for whole words are built by concatenating the triphone HMMs of the word’s phoneme sequence. HMMs for whole words are then concatenated to model the emission probability for a given word sequence $p_\vartheta(x_1^T|w_1^N)$, where ϑ denotes the set of parameters, which have to be set to approximate the true distribution: $p_\vartheta(x_1^T|w_1^N) \approx Pr(x_1^T|w_1^N)$. This is done in the acoustic training (chapter 2.5). The probability $p_\vartheta(x_1^T|w_1^N)$ is calculated by summing up the probability of all paths s_1^T, emitting the features x_1^T, in the concatenated network of HMMs:

$$p_\vartheta(x_1^T|w_1^N) = \sum_{s_1^T} p_\vartheta(s_1^T, x_1^T|, w_1^N) \tag{2.3}$$

$$= \sum_{s_1^T} \prod_{t=1}^{T} p_\vartheta(x_t|x_1^{t-1}, s_1^t, w_1^N) p_\vartheta(s_t|x_1^{t-1}, s_1^{t-1}, w_1^N) \tag{2.4}$$

This equation can be simplified by assuming that emitted features do not depend on features emitted before. Another assumption is that a state is only depending to the previous state. This assumptions are summarized under the term *first order hidden Markov model*.

$$p_\vartheta(x_1^T|w_1^N) \cong \sum_{s_1^T} \prod_{t=1}^{T} p_\vartheta(x_t|s_t, w_1^N) p_\vartheta(s_t|s_{t-1}, w_1^N) \tag{2.5}$$

For an efficient evaluation of the sum in equation 2.5 a forward-backward algorithm [Baum & Petrie 66] is used. More efficient is an approximation, where only the most likely sequence is considered. With this so-called *maximum approximation* or *Viterbi approximation*, the evaluation can be done time synchronous by using dynamic programming [Viterbi 67, Ney 84, Ney 90]:

$$p_\vartheta(x_1^T|w_1^N) \cong \max_{s_1^T} \left\{ \prod_{t=1}^{T} p_\vartheta(x_t|s_t, w_1^N) p_\vartheta(s_t|s_{t-1}, w_1^N) \right\} \tag{2.6}$$

$$\stackrel{simplification}{=} \max_{s_1^T} \left\{ \prod_{t=1}^{T} p_\vartheta(x_t|s_t, w_1^N) q(s_t - s_{t-1}) \right\} \tag{2.7}$$

It has proved useful to make the transition probabilities of the HMM word and position independent, thus the probability distribution $p_\vartheta(s_t|s_{t-1}, w_1^N)$ is replaced by a function that depends only on the jump distance between two succeeding states $q(s_t - s_{t-1})$. This function calculates the *time distortion penalty (TDP)* and is used in equation 2.7 as substitute for the true transition probabilities. In the RWTH system continuous distributions are used to model the emission probabilities $p_\vartheta(x_t|s_t)$ [Ney & Noll 88], multimodal Gaussian distributions (Gaussian mixture densities) are applied as representation. The emission probabilities are calculated as the weighted sum over the individual densities for each state s:

$$
\begin{aligned}
p_\vartheta(x|s) &= \sum_{l=1}^{L} c_{sl}\mathcal{N}(x|\mu_{sl}, \Sigma_{sl}) && (2.8)\\
&\approx \max_{l} \{c_{sl}\mathcal{N}(x|\mu_{sl}, \Sigma)\} && (2.9)
\end{aligned}
$$

Within the practical implementation of the RWTH system two simplifications have been made. Firstly the weighted sum is approximated by the maximum. Secondly the covariance matrices Σ_{sl} are replaced by a state and density independent diagonal covariance matrix (equation 2.9), which can be estimated more reliably. A pooled diagonal covariance matrix has further advantages, since a closed form solution for the linear discriminant analysis exists, and optimizations can be applied in the distance calculation.

Assuming that the components of the feature vectors are stochastically independent, a single Gaussian distribution is defined as

$$
p_\vartheta(x|s,w,l) = \frac{1}{\prod_{d=1}^{D} \sqrt{2\pi\sigma_{swld}^2}} \exp\left\{ -\frac{1}{2}\sum_{d=1}^{D} \left(\frac{x_d - \mu_{swld}}{\sigma_{swld}}\right)^2 \right\}.
$$

Taking the negative logarithm from the maximum approximation of the sum in equation 2.9:

$$
-\log p_\vartheta(x|s,w) = \min_{l} \left\{ \frac{1}{2}\sum_{d=1}^{D} \left(\frac{x_d - \mu_{swld}}{\sigma_{swld}}\right)^2 - \log p(l|s,w) + \frac{1}{2}\sum_{d=1}^{D} \log(2\pi\sigma_{swld}^2) \right\},
$$

represents the distance between the feature vector x and the reference vector μ_{swl}. This distance is denoted with $d(x;s,w)$ and is also called the *score* of a feature vector.

The covariance matrix, the means and the distribution weights form the acoustic parameters ϑ. They have to be estimated from training data (see chapter 2.5), assuming that the training data resembles the data to be recognized.

2.3. Language Model

Purpose of a language model is to reflect the semantics, syntax and pragmatics of a language. Word sequences w_1^N have a certain probability, independent from the speech signal, i.e. "he has" is more likely (and correct) than "he have". The language model influences the decision as an a-priori distribution in Bayes' decision rule (equation 2.1). Usually the language is modeled as a Markov process: the word w_n depends only on the $n-1$ predecessor words w_1^{n-1}:

$$Pr(w_1^N) = \prod_{n=1}^{N} Pr(w_n|w_1^{n-1})$$

Estimation of the conditional probability for a word w_n given an unknown number of words is not possible, so instead of using all predecessor words as context for the word w_n, only the last $m-1$ words are taken into account during calculation of the probability. This corresponds to an $(m-1)$th order Markov process. Such language models are called m-gram language models [Bahl & Jelinek+ 83]:

$$\prod_{n=1}^{N} Pr(w_n|w_1^{n-1}) \approx \prod_{n=1}^{N} p(w_n|w_{n-m+1}^{n-1}), \tag{2.10}$$

where the sequence w_{n-m+1}^{n-1} is called the *history* h of w_n. The parameters for the language model are estimated on training data with the maximum likelihood training criterion. The domain on which the model is estimated should correspond to the domain where the recognition takes place. A measure for the quality of a language model is the *perplexity (PP)* or *log perplexity (logPP)*. The perplexity is defined as the inverse of the geometric mean of the language model probability of all words w_n in a sequence w_1^N [Bahl & Jelinek+ 83]:

$$PP = \left[\prod_{n=1}^{N} p(w_n|h)\right]^{-\frac{1}{N}}$$

The perplexity can be interpreted as the average number of concurring words at each word position the recognizer can choose from. If minimizing the perplexity is chosen as training criterion, a closed form solution for the estimation of the language model parameters can be derived. To estimate the language model probabilities $p(w|h)$ the Log-Likelihood function from equation 2.11 is used and it's maximum has to be determined.

$$L = \sum_{w,h} N(w,h) \log p(w|h) \tag{2.11}$$

$N(w, h)$ denotes the number of occurrences of the word w with history h in the training data. With the additional constraint $\sum_w p(w|h) = 1, \forall h$, the following maximum likelihood estimator is gained:

$$p(w|h) = \frac{N(h, w)}{\sum_{w'} N(h, w')} = \frac{N(h, w)}{N(h)}, \tag{2.12}$$

where $N(h)$ denotes the number of occurrences of the history h. So the probabilities are estimated by determining the relative frequency in the training data. The main problem of the relative frequency approach is the handling of unseen m-grams. This can not be solved completely by increasing the amount of training data. To provide a probability different from zero, so-called *smoothing* techniques have to be applied. Usually probability mass is taken (*discounting*) from the seen m-grams and redistributed among either the unseen (*backing-off*) or all (*interpolated*) m-grams [Ney & Essen$^+$ 94].

2.4. Search

Aim of the search process is finding, the most probable word sequence $\hat{w}_1^N$, for a given sequence of features x_1^T. Two knowledge sources: the acoustic model (section 2.2) and the language model (section 2.3). Combining equation (2.1) with equations (2.6) and (2.10) results in following expression:

$$\hat{w}_1^N = \operatorname*{argmax}_{w_1^N} \left\{ \max_{s_1^T} \left\{ \prod_{t=i}^{T} p_\vartheta(x_t|s_t, w_1^N) p_\vartheta(s_t|s_{t-1}, w_1^N) \right\} \cdot \prod_{n=1}^{N} p(w_n|w_{n-m+1}^{n-1}) \right\} \tag{2.13}$$

According to equation 2.13 the search process has to hypothesize all possible word sequences w_1^N and find the most likely one. It is easy to see that a straight-forward implementation of the search leads to an infeasible computational complexity, if the number of words that can be recognized is V then

$$\sum_{n=1}^{N} V^n = \frac{V^{N+1} - 1}{V - 1}$$

different word sequences with a length up to N have to be hypothesized. The computational complexity can be reduced by optimization and approximation. One optimization is the use of dynamic programming to decompose the search for a global maximum into a sequence of local optimizations [Bellman 57].

Two different search algorithms are used in most recognition systems, both based on dynamic programming. A time-synchronous approach, which takes advantage of the Viterbi approximation. And an approach based on A^* *search*. In A^* *search* or *stack decoding*, the hidden Markov model states of the hypothesized word sequences are expanded and searched in a time-asynchronous way. The probability of the already evaluated states is combined with an over-optimistic estimate of the remaining state sequence [Jelinek 69, Paul 91]. The algorithm converges to the global optimum, if the estimated probability is admissible. Performance of A^* *search* is heavily based on the quality of the heuristic used to estimate the unknown sequence.

Viterbi based time-synchronous search expands the states of a hidden Markov model at each time-frame [Vintsyuk 71, Ney 84], at this point the concurring probabilities can be compared directly, which offers the possibilities of using of efficient *pruning techniques*. Unlikely hypotheses can be omitted early in the search process, thus the search space is reduced significantly.

For further optimization the pronunciation lexicon is organized as a prefix-tree to exploit the redundancies in the lexicon. Common phoneme sequences at the beginning of different words have to be evaluated only once. Despite of this optimization, it is still not possible to explore the whole search space. That is the reason for using beam search [Ney & Mergel$^+$ 87, Ortmanns & Ney 95]. In beam search only the most promising hypotheses are pursued. If the unlikely hypotheses are pruned, reaching the global optimum can not be guaranteed any more. The optimal word sequence may be less probable in an early phase of the search, thus removed from the search space. But these search errors can be reduced by adjusting the pruning parameters.

So-called *look-ahead techniques* make pruning more efficient. If the pronunciation lexicon is implemented as a prefix tree, the actual words are not known, until the end of the word has been hypothesized. Every node in the tree is associated with a certain set of words ending in the subtree. By taking the language model probability into account at this early stage, the most probable word in a subtree can be determined. This probability can be seen as an upper bound for all the words in the set. If this probability is below a threshold, the whole subtree can be pruned. This method is called *language model look-ahead* [Steinbiss & Tran$^+$ 94]. Additional improvement of pruning is obtained by executing a *phoneme look-ahead* [Ney & Haeb-Umbach$^+$ 92].

2.4.1. Multi-Pass Search

For further improving the search process, so-called *multi-pass search* can be applied. In contrast to *integrated* or *single-pass search* described in chapter 2.4 more than one acoustic model and/or language model is used to generate the recognizer's output. There are two different variants of multi-pass search: sequential and parallel multi-pass search.

In *sequential* multi-pass search, an initial model (acoustic and language) is used to generate a reduced representation of the search space in a first recognition pass. In a second pass a more sophisticated model is applied to the remaining search space. This step can be repeated several times to make use of models that require more calculation time, as the search space is reduced. An approximation for the searched space are *N-best lists* [Schwartz & Chow 90] which contain the N most likely hypotheses. Usually, entries in the N-best lists differ only in some positions, hence there is a high redundancy in that representation. A more compact representation are *word graphs* [Schwartz & Austin 91], which are directed acyclic graphs storing intermediate word sequences as arcs.

Parallel multi-pass search combines different acoustic and language models or even different recognition systems. In the case of *ROVER (Recognition Output Voting Error Reduction)* [Fiscus 97], the results of several recognizers are combined by applying one alignment and one voting step, resulting in a merged output. That is taken as the finally recognized word sequence (for details see: chapter 3.1). Sequential and parallel multi-pass search can be utilized within the same recognition system. But obviously the multi-pass approach is not useful for building systems with a low *real-time factor*. The real-time factor denotes how much time is needed to transcribe the acoustic utterance compared to its length.

2.5. Acoustic Training

To be able to calculate $p(x|s,l)$ the model parameters $\vartheta_{s,l}$ for the hidden Markov models are estimated during the *acoustic training*. The RWTH system uses *Gaussian mixture models (GMM)*, those parameters are the mixture weight $c_{s,l}$ the mean $\mu_{s,l}$ and the pooled covariance matrix Σ. The training data consists of a set of acoustic signals or their feature vectors and the corresponding transcriptions. Domain and characteristics of training and test data should resemble, to optimize the models for the recognition task [Ney 02].

2.5.1. ML-Training

In the most systems, a *maximum likelihood (ML)* framework together with the *expectation maximization (EM)* algorithm is used for training. The training data set consists of R pairs (X_r, W_r), where X_r is a sequence of feature vectors, $x_{r,1}, \ldots, x_{r,T_r}$ and W_r is the corresponding word sequence $w_{r,1}, \ldots, w_{r,N_r}$ In the maximum likelihood approach ϑ is estimated to maximize

$$p_\vartheta(X|W) = \prod_{r=1}^{R} p_\vartheta(X_r|W_r).$$

Which means that the mean probability over the training data is maximized. In order to estimate an acoustic model which does not lead to errors in the recognition process following constraint should hold:

$$p_\vartheta(X_r|W_r) > p_\vartheta(X_r|W), \forall W \neq W_r,$$

that is, the correct transcription is more probable than any other word sequence. The ML-approach would provide optimal parameters, if the amount of training data was infinite and the model assumptions made for the acoustic model were correct.

Assuming that the phoneme inventory consists of 50 different context dependent phonemes, the number of all triphones is $50^3 = 125000$. Since the amount of training material will not suffice to reliably estimate every triphone, equally sounding triphones are clustered by linguistic questions with a *classification and regression tree (CART)*. Gaussian mixture models are estimated for every leaf of the CART instead for every triphone, thus more training data is available for every mixture. Due to this fact the resulting acoustic model will be more robust.

In the first step of the training a time alignment between the features of an utterance and the states of the hidden Markov model for the corresponding word sequence is generated by either the application of previous model or an initial model. That is generated by a linear segmentation of the audio data and ten iteratively improved time alignments. A CART is trained on the basis of this initial alignment. In the next step the resulting CART classes are used to estimate an LDA matrix. A second CART is trained on the LDA transformed features, to reduce the effect of CART and LDA belonging to the model used for the initial time alignment. A second LDA matrix has to be estimated on the new CART classes and the untransformed features. With the second CART and LDA matrix the parameters of the acoustic model are estimated, starting with one probability distribution for every class of the CART. To increase the accuracy, the Gaussian densities are split several times. Acoustic training is a most time demanding part during the setup of a complete recognition system, for 40h of acoustic training material used in the EPPS task approximately one week is needed to carry out one iteration of the acoustic training.

3. Methods for system combination

3.1. ROVER

In 1997 Fiscus [Fiscus 97] developed the first method to combine the output of different speech recognition systems. The algorithm, called *ROVER* for *Recognition Output Voting Error Reduction*, takes the generated word sequences of the individual recognizers as input. In addition to the words, word posterior probabilities have to be provided by the recognizers. In the first step of the combination process the individual outputs are aligned (figure 3.1). In a second step, a voting procedure is applied for every position in the alignment. This voting takes the relative frequency of a word w at position n and the *confidence measure* for that word into account:

$$\text{score}(w,n) = \alpha \frac{N(w,n)}{\sum\limits_{w'} N(w',n)} \cdot (1-\alpha) C(w,n).$$

$C(w,n)$ denotes the confidence of word w at position n, usually the posterior probability $p(w|x)$. α is used to balance the influence of the confidence measure and relative frequency. The best scoring word at position n is taken as output. If two or more words have the same score one is chosen randomly, the utilization of confidence measures reduces the risk of these situations, because it is unlikely that two words have

system 1:	the dog bakes loud
system 2:	a frog barks loud
system 3:	the dog barks a loud

$$\Downarrow$$

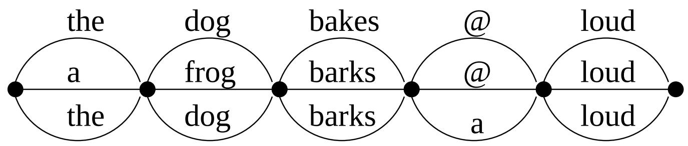

Figure 3.1.: ROVER alignment of three individual recognizer outputs

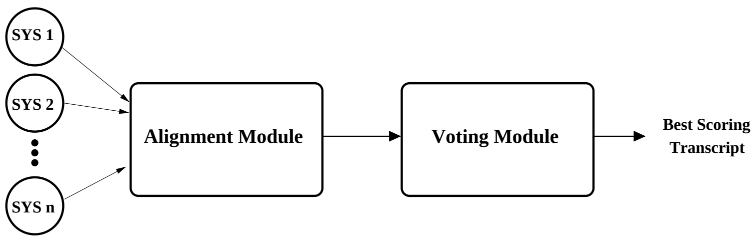

Figure 3.2.: Structure of the ROVER mechanism

the same relative frequency and the confidence measure. If $\alpha = 1$ ROVER performs a majority vote for every position. The final output of ROVER is computed as

$$\{w_1^N\}_{rover} = (\underset{w}{\text{argmax}}\{\text{score}(w, 1)\}, \ldots, \underset{w}{\text{argmax}}\{\text{score}(w, N)\}).$$

The ROVER system has two parameters, α and the confidence for *empty words*, denoted with @ in figure 3.1, that have to compete with words at the same position. Optimal values for these parameters have to be estimated on training data.

A clear disadvantage of this method is that only the single-best outputs of a recognition system can be taken into account, and if a majority vote is performed for the system combination, three systems are needed at minimum to yield useful results.

3.2. Confusion Networks

The standard performance metric for speech recognition systems is the *word error rate (WER)*. WER stands in contrast to the standard *maximum a-posteriori probability (MAP)* search, which combines acoustic and language model probabilities and takes the most probable sentence as result for the recognition process. This search process is designed to minimize the *sentence error rate*. An WER optimized search method was developed by Mangu et al. by using *confusion networks* [Mangu & Brill$^+$ 99]. Confusion networks are generated from word graphs and are in fact a very simple form of word graphs with one initial and one final node. Every other node has exactly one predecessor and one successor. Because of this structure, confusion networks tend to be called *word sausages*. The set of words labeling the arcs between two adjacent nodes in the confusion network is called confusion set. These words are the candidates for that position in the output sequence. During the construction process, every word in the confusion set is annotated with it's posterior probability. In the search from each confusion set the word with the highest posterior probability is selected as final output.

Hypothesis(w_1^N)			MAP	word posterior probabilities		
w_1	w_2	w_3	$p(w_1^N \mid x_1^T)$	$p(w_1 \mid x_1^T)$	$p(w_2 \mid x_1^T)$	$p(w_3 \mid x_1^T)$
I	do	inside	0.16	0.34	0.29	0.16
I	do	fine	0.13	0.34	0.29	0.28
by	doing	fine	0.11	0.45	0.49	0.28
by	doing	well	0.11	0.45	0.49	0.11
by	doing	sight	0.10	0.45	0.49	0.10
by	doing	bye	0.07	0.45	0.49	0.07
by	doing	thought	0.05	0.45	0.49	0.07
I	doing	fine	0.04	0.34	0.49	0.28
I	don't	buy	0.01	0.34	0.01	0.01
by	doing	fun	0.01	0.45	0.49	0.01

Table 3.1.: 10-best list for a sequence of features x_1^T, with MAP and word posterior probabilities (taken from [Mangu & Brill$^+$ 99])

Table 3.1 shows a 10-best list with the corresponding maximum a-posteriori probabilities and the word posterior probabilities. The word sequence "I do inside" has the highest MAP and is selected in the standard search process. Figure 3.3 shows a confusion network corresponding to the 10-best list, the words with the highest word posterior probability are marked red, so is the optimal path through the word sausage. The *confusion network decoding (CND)* procedure produces the sequence "by doing fine". Comparing both hypotheses with the reference sentence "I'm doing fine" the MAP decoding produces three errors, the CND only one.

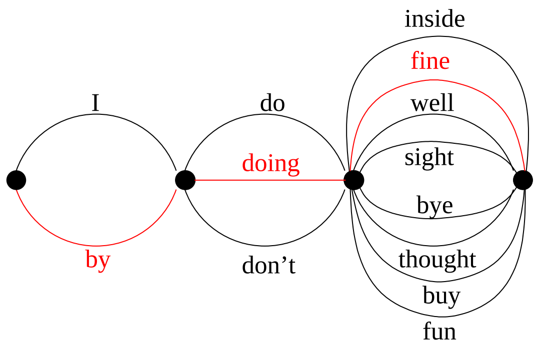

Figure 3.3.: Confusion network corresponding to the N-best list from table 3.1

3.3. Confusion Network Combination

Another method used for system combination is based on confusion networks. *Confusion network combination (CNC)* was introduced by G. Evermann and P.C. Woodland in 2000 [Evermann & Woodland 00] and is an extension of the ROVER algorithm described in chapter 3.1. The CNC takes confusion networks as input instead of only the single-best output of a recognizer. Due to this modification, the aligning and voting procedure is adapted. Advantage of this method is, that the process can take several alternatives from each recognizer into account. The construction of the confusion networks may be seen as a disadvantage of the system combination, as during the construction the word boundaries are approximated to form the confusion sets.

Architecture of the CU-HTK 2003 System

Evermann et al. built a multi-pass speech recognition system for conversational telephone speech, utilizing different acoustic models [Evermann & Chan$^+$ 04]. The various models are combined using CNC. Figure 3.4 shows a diagram of the system's architecture: with an initial system, lattices are generated, which are rescored using other acoustic models. The resulting set of lattices is transformed into confusion networks, which are combined to generate the final output of the system.

The different acoustic models were built with (or without): *heteroscedastic linear discriminant analysis (HLDA)* (a variation of the LDA), *speaker adaptive training (SAT)*, a single pronunciation dictionary and a triphone or quinphone context. Model A used SAT and HLDA, model B used only HLDA, model C a single pronunciation dictionary and HLDA. Model D was built omitting the HLDA. These four different types were trained using a triphone or a quinphone context.

At the beginning the eight resulting models (A-D with tri- and quninphone context) were combined on the development corpus, to discover which combinations yield the lowest error rate. Combining the triphone and quinphone models of variations A, C, D yielded to the lowest error rate. This combination was realized in the final system architecture as seen in figure 3.4. Combining the different systems results in a reduction of the word error rate by 5.7% relative on the development data, and 5.5% relative on the evaluation set.

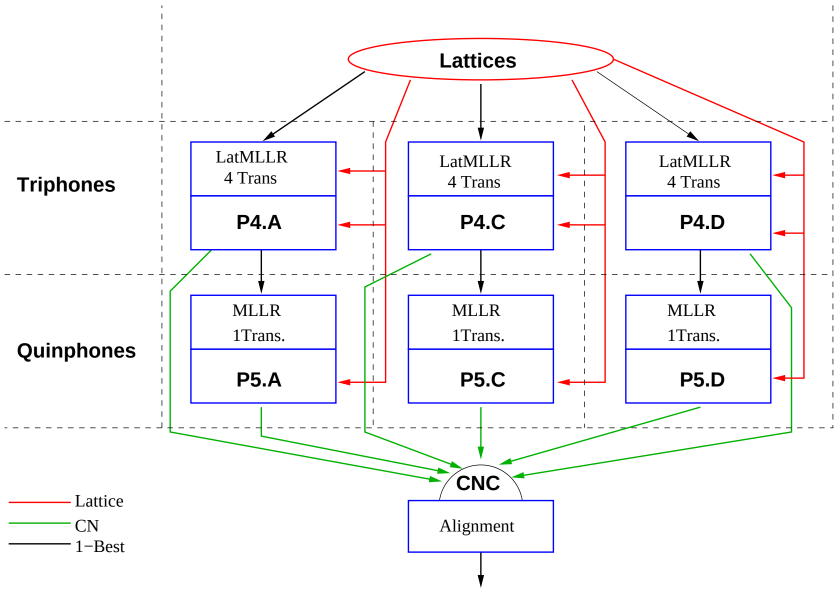

Figure 3.4.: Branched multi-pass architecture of the CU-HTK 2003 system

pass	method	WER on eval02	WER on eval03
P1-3	lattice gen.	24.8	23.5
P4.A	SAT HLDA,3phones	23.0	21.9
P4.B	HLDA, 3phones	23.6	-
P4.C	SPron HLDA, 3phones	23.4	22.2
P4.D	non-HLDA, 3phones	24.8	23.7
P5.A	SAT HLDA, 5phones	23.0	22.1
P5.B	HLDA, 5phones	23.5	-
P5.C	SPron HLDA, 5phones	23.3	22.3
P5.D	non-HLDA, 5phones	24.6	23.7
CNC P4[ACD] + P5[ACD]		21.7	20.7

Table 3.2.: Error rates of the single stages achieved by the CU-HTK 2003 system (figure 3.4) on the evaluation (eval03) and testing (eval02) set of the HUB-5 English task

4. Discriminative Model Combination

Given the true posterior distribution $Pr(w_1^N|x_1^T)$ the sequence of spoken words will be the one maximizing this probability, that is the one satisfying:

$$\forall {w'}_1^{N'}, {w'}_1^{N'} \neq w_1^N : \log Pr(w_1^N|x_1^T) - \log Pr({w'}_1^{N'}|x_1^T) \geq 0$$

The function $g(x_1^T, w_1^T, {w'}_1^{N'}) = \log(Pr(w_1^N|x_1^T)/Pr({w'}_1^{N'}|x_1^T))$ is called discriminant function, as it describes the decision boundaries.

The real distribution of human speech is not known and thus there is no exact model $Pr(w_1^N|x_1^T)$. To circumvent this problem, an approximate distribution $p(w_1^N|x_1^T)$ is trained, as described in chapter 2.2. Strictly speaking the discriminant function has to be rewritten:

$$\begin{aligned} g(x_1^T, w_1^N, {w'}_1^{N'}) &= \log p(w_1^N|x_1^T) - \log p({w'}_1^{N'}|x_1^T) && (4.1) \\ &= \log\left[p(w_1^N)p(x_1^T|w_1^N)\right] - \log\left[p({w'}_1^{N'})p(x_1^T|{w'}_1^{N'})\right] && (4.2) \end{aligned}$$

In equation 4.2, the estimated probability function p is decomposed into a language model and an acoustic model. Since p is an approximation for the real distribution, it deviates from Pr. Thus, the discriminant function will lead to a suboptimal classifier and subsequently to a suboptimal speech recognizer.

A possible solution for this problem is a different training procedure. The standard maximum likelihood training tries to estimate the true deviation of data. Another method understanding the parameter estimation of the acoustic model as an error minimization process could be utilized. In this approach, so-called discriminative methods are applied to optimize the parameters of the acoustic model, in order to improve the distinction of the observations.

An initial example for usage of the discriminative method is the *language model factor* λ, which is used in the RWTH system. Instead of using $p(w_1^N)$, $p(w_1^N)^\lambda$ is used in the discriminant function. If $\lambda = 1$ is chosen, the function is unchanged. Experiments [Lee 89] showed that for values of $\lambda \neq 1$ minimum word error can be acquired. This fact is a result of the approximation of the true probability distribution Pr, as the language model and the acoustic model are both approximations, the decision rule is not optimal anymore. Again the discriminant function has to be modified:

$$g(x_1^T, w_1^N, {w'}_1^{N'}) \;=\; \log\left[p(w_1^N)^\lambda p(x_1^T|w_1^N)\right] - \log\left[p({w'}_1^{N'})^\lambda p(x_1^T|{w'}_1^{N'})\right] \quad (4.3)$$

Since the language model factor is only one value, it can be optimized manually on a development set. With increasing number of factors for other components e.g. the TDP scale or the pronunciation scale, the computational effort raises quickly. For that reason automatic methods should be applied to determine the optimal settings.

Going one step further, combinations of more than one language model and one acoustic model are possible. These different models can be weighted by an arbitrary λ and used in the decision rule. Assuming M different acoustic and language models the true distribution Pr can be approximated by combining these models $p_m(w_1^N|x_1^T), m = 1, \ldots, M$. As in the case of the language model the combination is done log-linearly to compensate for certain derivations from the true distribution:

$$\begin{aligned} p_{\{\Lambda\}}(w_1^N|x_1^T) &= C(\Lambda) \prod_{m=1}^{M} p_m(w_1^N|x_1^T)^{\lambda_m} & (4.4)\\ &= \exp\left\{\log C(\Lambda) + \sum_{m=1}^{M} \lambda_m \log p_m(w_1^N|x_1^T)\right\} & (4.5) \end{aligned}$$

$\Lambda = (\lambda_1, \ldots, \lambda_m)$ are the weights of the models in the combination and $C(\Lambda)$ is a normalization factor. The improved discriminant function is given as

$$g(x_1^T, w_1^N, {w'}_1^{N'}) \;=\; \sum_{m=1}^{M} \lambda_m \left(\log p_m(w_1^N|x_1^T) - \log p_m({w'}_1^{N'}|x_1^T)\right). \quad (4.6)$$

The λ_m are optimized with respect to the decision error rate of the new discriminant function (equation 4.6). The approach based on this new decision rule is called *discriminative model combination (DMC)*. This kind of combination is a good idea in general, since the weight λ_m for each model p_m depends solely on the ability to provide information for a correct classification.

Since the performance measure for the speech recognizer is the edit-distance $\mathcal{L}(\cdot, \cdot)$ between the correct transcription and the recognized word sequence, it is applied as the optimization criterion. The error rate over the whole training set, depending on the free parameters Λ can be expressed by:

$$E(\Lambda) = \frac{1}{\sum_{i=1}^{I} N_i} \cdot \sum_{i=1}^{I} \mathcal{L}\left({w_i}_1^{N_i}, \operatorname*{argmax}_{\hat{w}_1^{\hat{N}}} \left(p_\Lambda(\hat{w}_1^{\hat{N}} | {x_i}_1^{T_i}) \right) \right) \tag{4.7}$$

$$= \frac{1}{\sum_{i=1}^{I} N_i} \cdot \sum_{i=1}^{I} \mathcal{L}\left({w_i}_1^{N_i}, \operatorname*{argmax}_{\hat{w}_1^{\hat{N}}} \left(\log \frac{p_\Lambda(\hat{w}_1^{\hat{N}} | {x_i}_1^{T_i})}{p_\Lambda({w_i}_1^{N_i} | {x_i}_1^{T_i})} \right) \right), \tag{4.8}$$

where I is the number of training utterances ${x_i}_1^{T_i}$ and ${w_i}_1^{N_i}$ is the corresponding transcription. Equation 4.8 will be the favored description of the recognizer's error rate in this chapter, since some optimizations rely on the log-ratio of the probabilities between the recognized word sequence and the correct transcript $\log p_\Lambda(\hat{w}_1^{\hat{N}} | {x_i}_1^{T_i}) / p_\Lambda({w_i}_1^{N_i} | {x_i}_1^{T_i})$.

As mentioned in the beginning of this chapter, the values for each model can be set empirically. This is a suboptimal solution because the range of the λ_m has to be discretized in order to run all experiments. A better way is to apply *stochastic search* algorithms.

In stochastic search n different assignments for Λ are generated, using a random generator: $\Lambda_1, \ldots, \Lambda_n$. With each of these assignments one experiment is run, to evaluate the combination. The results are ordered with respect to their word error rate. In the environment of the m best assignments, again n new mappings are generated. Resulting in $m(n+1)$ Λs. Now the algorithm is rerun several times until the gain does not justify further iterations. It is easy to see, that this approach as well has a high computational effort.

4.1. Training Of The DMC Coefficients

A method for training that needs less computational effort is the *Generalized Probabilistic Decent* introduced in [Juang & Chou$^+$ 95]. They minimize the smoothed empirical error rate of p_Λ (equation 4.8). Assuming that the training data consists of I utterances $X_i := {x_i}_1^{T_i}$ for which the correct transcription $W_i := {w_i}_1^{N_i}$ is available. For every segment there are K competing word sequences $\hat{W}_{i,k} := {\hat{w}_{i,k}}_1^{N_{i,k}}$, which are stored as a K-best list or a word lattice. The function $\mathcal{L}(\hat{W}_{i,k}, W_i)$ denotes the Levenshtein distance between one of the competing word sequences and the correct transcription.

In contrast to equation 4.8 another function $E_{MCE}(\Lambda)$ expresses the *minimum classification error (MCE)* on a given training set:

$$E_{MCE}(\Lambda) = \frac{1}{I}\sum_{i=1}^{I} l(X_i, W_i, \Lambda) \tag{4.9}$$

$l(X_i, W_i, \Lambda)$ is a loss function, depending on the smoothed misclassification function $d(X_i, W_i, \Lambda)$ of the observation X_i, and is defined as

$$l(X_i, W_i, \Lambda) = \frac{1}{1 + \exp\left\{-a(b + d(X_i, W_i, \Lambda))\right\}} \tag{4.10}$$

with

$$d(X_i, W_i, \Lambda) = \log\left(\frac{1}{K}\sum_{k=1}^{K}\exp\left\{\eta\log\frac{p_\Lambda(\hat{W}_{i,k}|X_i)}{p_\Lambda(W_i|X_i)}\right\}\right)^{\frac{1}{\eta}}. \tag{4.11}$$

A combination of these two terms leads to

$$l(X_i, W_i, \Lambda) = \left(1 + A\cdot\left(\frac{1}{K}\sum_{k=1}^{K}\exp\left\{-\eta\log\frac{p_\Lambda(W_i|X_i)}{p_\Lambda(\hat{W}_{i,k}|X_i)}\right\}\right)^{-\frac{B}{\eta}}\right)^{-1},$$

where $A = \exp\{-ab\}$, $B = a$, and $A, B, \eta > 0$ have to be adjusted properly. The value of l is within the range of $[0, 1]$, depending on whether the classifier decides for the correct class or not. Using the smoothed misclassification function and the Generalized Probabilistic Decent Theorem, an iterative formula for the computation of Λ can be derived:

$$\Lambda^{(J+1)} = \Lambda^{(J)} - \varepsilon U \nabla_\Lambda E_{MCE}(\Lambda)_{|\Lambda=\Lambda^{(J)}} \tag{4.12}$$

The matrix U has to be positive definite and ε determines the optimization step width.

4.1.1. The MCE Criterion

To yield a useful training criterion, the MCE formula has to be slightly modified by adding the Levenshtein distance. The edit distance between the correct transcript and the hypothesis $\mathcal{L}(\hat{W}_{i,k}, W_i)$ is included in $d(X_i, W_i, \Lambda)$:

$$d'(X_i, W_i, \Lambda) = \log\left(\frac{1}{K}\sum_{k=1}^{K}\exp\left\{\eta\mathcal{L}(\hat{W}_{i,k}, W_i)\log\frac{p_\Lambda(\hat{W}_{i,k}|X_i)}{p_\Lambda(W_i|X_i)}\right\}\right)^{\frac{1}{\eta}} \tag{4.13}$$

Using the new misclassification function (equation 4.13) in the loss function (equation 4.10) leads to:

$$l'(X_i, W_i, \Lambda) = \left(1 + A \cdot \left(\frac{1}{K}\sum_{k=1}^{K} \exp\left\{-\eta \mathcal{L}(\hat{W}_{i,k}, W_i) \log \frac{p_\Lambda(W_i|X_i)}{p_\Lambda(\hat{W}_{i,k}|X_i)}\right\}\right)^{-\frac{B}{\eta}}\right)^{-1}$$

The steps needed to build the gradient of the new target function

$$E_{MCE}(\Lambda) = \frac{1}{I}\sum_{i=1}^{I} l'(X_i, W_i, \Lambda)$$

are explained in detail in appendix C.1, and lead to the following iterative optimization of Λ, where J denotes the iteration step.

$$\begin{aligned}
\lambda_m^{(0)} &= 0 \\
\lambda_m^{(J+1)} &= \lambda_m^{(J)} - \varepsilon \sum_{i=1}^{I} l'(X_i, W_i, \Lambda^{(J)})(1 - l'(X_i, W_i, \Lambda^{(J)})) \cdot \\
&\quad \frac{\sum_{k=1}^{K} \mathcal{L}(\hat{W}_{i,k}, W_i) \log\left(\frac{p_m(\hat{W}_{i,k}|X_i)}{p_m(W_i|X_i)}\right)\left[\frac{p_{\Lambda^{(J)}}(\hat{W}_{i,k}|X_i)}{p_{\Lambda^{(J)}}(W_i|X_i)}\right]^{\eta\mathcal{L}(\hat{W}_{i,k}, W_i)}}{\sum_{k=1}^{K}\left[\frac{p_{\Lambda^{(J)}}(\hat{W}_{i,k}|X_i)}{p_{\Lambda^{(J)}}(W_i|X_i)}\right]^{\eta\mathcal{L}(\hat{W}_{i,k}, W_i)}} \\
\Lambda^{(J)} &= (\Lambda_1^{(J)}, \ldots, \Lambda_M^{(J)})^T
\end{aligned} \tag{4.14}$$

Equation 4.14 shows that the λ_m are changed from iteration to iteration by a weighted sum of the discriminant function. Unfortunately, the modified MCE criterion still does not describe or minimize the empirical error rate. An appropriate criterion was developed by [Beyerlein 00] that directly minimizes the WER.

4.1.2. The MWE Criterion

In order use an criterion that is based on the true empirical error rate, the Levenshtein distance between the true word sequence and the hypotheses is summed over the whole training data. An additional function S is used to smooth the error rate and therefor gain a differentiable criterion.

$$E_{MWE}(\Lambda) = \frac{1}{\sum_{i=1}^{I} N_i}\sum_{i=1}^{I}\sum_{k=1}^{K} \mathcal{L}(\hat{W}_{i,k}, W_i) S(\hat{W}_{i,k}, i, \Lambda)$$

If the function S was not used to smooth the error rate, that is, $S(\hat{W}_{i,k}, i, \Lambda) = \delta_{\hat{W}_{i,k},\hat{W}}$, where $\hat{W}$ is the word sequence selected by the classifier, the word errors are exactly summed. And $E_{MWE}(\Lambda)$ expresses equation 4.8. In order to work with an differentiable error criterion the indicator function S is not set to the Kronecker delta, but to a smoothed indicator function:

$$S(\hat{W}_{i,k}, i, \Lambda) = \frac{p_\Lambda(\hat{W}_{i,k}|X_i)^\eta}{\sum_{W'} p_\Lambda(W'|X_i)^\eta} \tag{4.15}$$

$$= \frac{\exp\left\{\eta \log \frac{p_\Lambda(\hat{W}_{i,k}|X_i)}{p_\Lambda(W_i|X_i)}\right\}}{\sum_{W'} \exp\left\{\eta \log \frac{p_\Lambda(W'|X_i)}{p_\Lambda(W_i|X_i)}\right\}} \tag{4.16}$$

A similar function was used in [Ney 02] to smooth the classification error. To make the new training criterion comparable to the other criteria introduced in this chapter, S has to be converted to equation 4.16. For large η the function S approximates the decision of the classifier better, whereas a η supports "weak" hypotheses. In [Beyerlein 00] η is set to 3, in order to balance both effects. This choice of η has practical issues as well, since now only a finite number of alternative hypotheses have to be considered. The derivative of $E_{MWE}(\Lambda)$ with respect to Λ has to be calculated (see: appendix C.2). Using this derivative it is possible, to build an iterative formula that describes the stepwise optimization of the empirical word error rate by adaptation of Λ, as in the case of the MCE criterion. Again, J denotes the iteration depth.

$$\begin{aligned}
\lambda_m^{(0)} &= 0 \\
\lambda_m^{(J+1)} &= \lambda_m^{(J)} - \frac{\varepsilon \cdot \eta}{\sum_{i=1}^{I} N_i} \sum_{i=1}^{I} \sum_{k=1}^{K} S(\hat{W}_{i,k}, i, \Lambda^{(J)}) \tilde{\mathcal{L}}(\hat{W}_{i,k}, i, \Lambda^{(J)}) \log \frac{p_m(\hat{W}_{i,k}|X_i)}{p_m(W_i|X_i)} \\
\Lambda^{(J)} &= (\lambda_1^{(J)}, \ldots, \lambda_M^{(J)})^T
\end{aligned}$$

and

$$\tilde{\mathcal{L}}(\hat{W}_{i,k}, i, \Lambda) = \mathcal{L}(\hat{W}_{i,k}, W_i) - \sum_{W' \neq W_i} S(W', i, \Lambda) \mathcal{L}(W', W_i) \tag{4.17}$$

The coefficient λ_m is the result of a weighted correlation between the Levenshtein distance and the discriminant function. A closed form solution can be acquired for the MWE criterion by applying the least squares method which is presented in [Beyerlein 98]. The use of an alternative smoothing function

$$S(\hat{W}_{i,k}, i, \Lambda) = \left(\frac{\log \frac{p_\Lambda(\hat{W}_{i,k}|X_i)}{p_\Lambda(W_i,X_i)} + B}{A + B} \right)^2$$

with:

$$-B < \log \frac{p_\Lambda(\hat{W}_{i,k}|X_i)}{p_\Lambda(W_i, X_i)} < A$$

and the λs have an additional constraint to normalize the values of the discriminant function:

$$\sum_{m=1}^{M} \lambda_m = 1$$

since the used smoothing function is part of a parabola, its derivative is linear in $\log \frac{p_\Lambda(\hat{W}_{i,k}|X_i)}{p_\Lambda(W_i,X_i)}$, which leads to another closed form solution.

4.2. Maximum Likelihood And MMI Criterion

The maximum likelihood approach can be used to estimate the weights for the combination, leading to the following criterion:

$$L(\Lambda) = \sum_{i=1}^{I} \log p_\Lambda(W_i, X_i)$$

As mentioned earlier in this chapter, the ML criterion tries to estimate the deviation of the data, and does not try to minimize the word error rate. Furthermore the use of the combined probability $p_\Lambda(W_i, X_i) = p_\Lambda(W_i|X_i)p_\Lambda(W_i)$ leads to a complicated target function:

$$L(\Lambda) = \sum_{i=1}^{I} \log \frac{\exp\left\{\sum_{m=1}^{M} \lambda_m \log p_m(W_i, X_i)\right\}}{\sum_{W'} \int_Y \exp\left\{\sum_{m=1}^{M} \lambda_m \log p_m(W', Y)\right\} dy} \tag{4.18}$$

The normalization factor is a the disadvantage of this criterion, because it anticipates the derivation of a closed form solution. The computational effort for using equation 4.18 as target function is infeasible. Summation over all possible hypotheses W' has to be approximated by the best hypothesis, which leads to a larger approximation error than caused by using the MWE criterion.

Another candidate for a target function can be derived by using the MMI criterion:

$$E_{MMI}(\Lambda) = \sum_{i=1}^{I} \log p_\Lambda(W_i|X_i) \tag{4.19}$$

$$= \sum_{i=1}^{I} \log \frac{\exp\left\{\sum_{m=1}^{M} \lambda_m \log p_m(W_i|X_i)\right\}}{\sum_{W'} \exp\left\{\sum_{m=1}^{M} \lambda_m \log p_m(W'|X_i)\right\}} \tag{4.20}$$

where equation 4.19 does not denote the exact mutual information, but the equivocation between the sequence of words and the observed features. The computational advantage of E_{MMI} over the ML approach is the normalization term. The integral is not needed in MMI target function. Furthermore, the maximum approximation needed before can be omitted But the summation over all hypotheses is as well infeasible, only a finite number of word sequences can be considered. The approximation will be worse than in the MWE case, because every probability for a particular hypotheses is raised to the power of η. Subsequently the influence of the neglected probability mass decreases by summing over a finite number of hypotheses. This implies that theoretically the MWE criterion is the most promising criterion for minimizing the word error rate.

5. Lattice Based System Combination

Inspired by the idea of confusion network combination and discriminative model combination, an approach was created to combine multiple outputs of automatic speech recognition systems at the level of lattices using discriminative model combination.

The weights for each systems were trained on a development set, using the maximum mutual information or the minimum word error criterion. According to the explanations in chapter 4 the lattices are combined in a log-linear way: the scores at every arc in a lattice are multiplied with the acquired coefficient λ_m. The resulting lattices are combined arc-wise by summing the scores at the corresponding arcs. The resulting word lattice is evaluated by executing the standard maximum a-posteriori decoding.

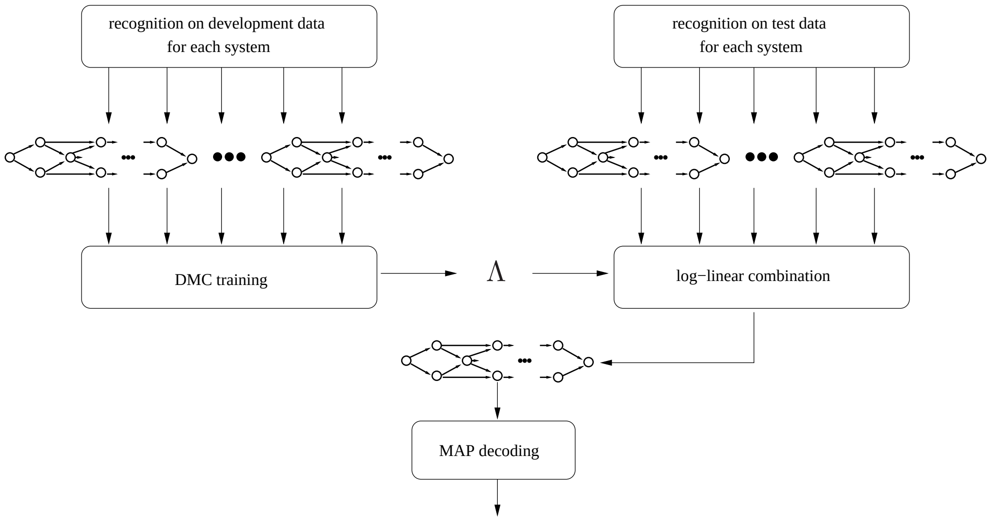

Figure 5.1.: Scheme of the steps involved in lattice-based system combination; the coefficients Λ for every systems are trained on a word lattices for the development set; the lattices for the evaluation data are extended by these coefficients and combined; the standard decoding is applied on the resulting lattice

5.1. Log-Linear System Combination On Lattices

Before the combination as proposed in equation 4.5 can be performed, it has to be checked if the combination can be carried out arc-wise on word lattices. This might by a problem, since the language model needs the context of previous arcs e.g. words. Starting with the initial equation for $p_\Lambda(w_1^N|x_1^T)$:

$$\begin{aligned} p_\Lambda(w_1^N|x_1^T) &= C(\Lambda, x_1^T) \prod_{m=1}^{M} p_m(w_1^N, x_1^T)^{\lambda_m} \\ &= C(\Lambda, x_1^T) \prod_{m=1}^{M} p_m(w_1^N)^{\lambda_m} p_m(x_1^T|w_1^N)^{\lambda_m} \end{aligned}$$

$C(\Lambda, x_1^T)$ is a normalization factor that remains constant with respect to w_1^N. The chain rule for probabilities is applied, since arcs represent single words rather than whole word sequences:

$$\begin{aligned} p_\Lambda(w_1^N|x_1^T) &= C(\Lambda, x_1^T) \prod_{m=1}^{M} \left(\prod_{n=1}^{N} p_m(w_n|w_1^{n-1})^{\lambda_m} \right) \cdot p_m(x_1^T|w_1^N)^{\lambda_m} \\ &= C(\Lambda, x_1^T) \prod_{m=1}^{M} \left(\prod_{n=1}^{N} p_m(w_n|w_1^{n-1})^{\lambda_m} \right) \cdot \left(\prod_{n=1}^{N} p_m(x_{|w_{n,m}}|w_n)^{\lambda_m} \right) \end{aligned}$$

$x_{|w_{n,m}} := x_{t_{mn}}^{t_{mn+1}-1}$ denotes the sequence of features vectors determined by model m belonging to word w_n. Assuming a function $t(m, w_1^N) = {t_m}_1^N$ which assigns word boundaries to a sequence of words depending on the model: w_n is spoken between the time frames t_{mn} and $t_{mn+1} - 1$.

$$p_\Lambda(w_1^N|x_1^T) = C(\Lambda, x_1^T) \prod_{m=1}^{M} \prod_{n=1}^{N} \left(p_m(w_n|w_1^{n-1}) p_m(x_{|w_{n,m}}|w_n) \right)^{\lambda_m} \quad (5.1)$$

$$= C(\Lambda, x_1^T) \prod_{n=1}^{N} \prod_{m=1}^{M} \left(p_m(w_n|w_1^{n-1}) p_m(x_{|w_{n,m}}|w_n) \right)^{\lambda_m} \quad (5.2)$$

Interchanging the order of the product symbols in equation 5.1 leads to equation 5.2 and makes it possible to use discriminative model combination on lattices. Equation 5.2 expresses that it is possible to combine the probabilities at each arc of the lattices before taking the language model dependencies into account.

5.2. Log-Linear Model Combination On Lattices

The combination of whole "system-lattices" turned out to be not as successful as expected (see: chapter 9). Therefor the idea of a system combination was changed to the idea of model combination. Which means that for each used model involved in the decision process of a speech recognizer a single weight was trained. These models are the language model and the acoustic model, consisting of the emission probabilities and transition probabilities of the HMM and the pronunciation probability. This leads to the following equations:

$$\begin{aligned} p_\Lambda(w_1^N|x_1^T) &= C(\Lambda, x_1^T) \prod_{m=1}^{M} p_m(w_1^N, x_1^T)^{\lambda_m} \\ &= C(\Lambda, x_1^T) \prod_{m=1}^{M} \left[p_m^{\alpha_m}(w_1^N) p_m(x_1^T|w_1^N)\right]^{\lambda_m} \end{aligned}$$

The joint probability is decomposed into a language model and an acoustic model. α_m denotes the language model factor used in system m. In a further step the acoustic model is decomposed into it's components:

$$\begin{aligned} p_\Lambda(w_1^N|x_1^T) &= C(\Lambda, x_1^T) \prod_{m=1}^{M} \Bigg[p_m^{\alpha_m}(w_1^N) \max_{\tilde{w}_1^N} p_m^{\beta_m}(\tilde{w}_1^N|w_1^N) \cdot \\ &\qquad \cdot \sum_{s_1^T} p_m^{\gamma_m}(x_1^T|s_1^T) p_m^{\delta_m}(s_1^T|\tilde{w}_1^N) \Bigg]^{\lambda_m} \\ &= C(\Lambda, x_1^T) \max_{\tilde{w}_1^N} \prod_{m=1}^{M} \Big[p_m^{\alpha_m}(w_1^N) p_m^{\beta_m}(\tilde{w}_1^N|w_1^N) \cdot \\ &\qquad \cdot \max_{s_1^T} p_m^{\gamma_m}(x_1^T|s_1^T) p_m^{\delta_m}(s_1^T|\tilde{w}_1^N) \Big]^{\lambda_m} \end{aligned} \tag{5.3}$$

Every element of the acoustic model is scaled by an individual factor that is provided by each system. In equation 5.3 the maximum approximation is applied to the sequence of HMM states and to the pronunciation variants. Note that the sequence of states depends on the used acoustic model. Like in the case of lattice-based system combination the chain rule is applied. $s_{\tilde{w}_{n,m}}$ denotes the sequence of states for word

w_n depending on system m:

$$p_\Lambda(w_1^N|x_1^T) = C(\Lambda, x_1^T) \max_{\tilde{w}_1^N} \prod_{m=1}^{M} \left[\prod_{n=1}^{N} p_m^{\alpha_m}(w_n|w_1^{n-1}) p_m^{\beta_m}(\tilde{w}_n|w_n) \cdot \right.$$
$$\left. \cdot \max_{s_{|\tilde{w}_{n,m}}} p_m^{\gamma_m}(x_{|\tilde{w}_{n,m}}|s_{|\tilde{w}_{n,m}}) p_m^{\delta_m}(s_{|\tilde{w}_{n,m}}|\tilde{w}_n) \right]^{\lambda_m} \tag{5.4}$$

Interchanging the order of the product symbols allows an arc-wise application of the model combination:

$$\begin{aligned} p_\Lambda(w_1^N|x_1^T) &= C(\Lambda, x_1^T) \max_{\tilde{w}_1^N} \prod_{n=1}^{N} \prod_{m=1}^{M} \left[p_m^{\alpha_m}(w_n|w_1^{n-1}) p_m^{\beta_m}(\tilde{w}_n|w_n) \cdot \right. \\ &\quad \left. \cdot \max_{s_{|\tilde{w}_{n,m}}} p_m^{\gamma_m}(x_{|\tilde{w}_{n,m}}|s_{|\tilde{w}_{n,m}}) p_m^{\delta_m}(s_{|\tilde{w}_{n,m}}|\tilde{w}_n) \right]^{\lambda_m} \\ &= C(\Lambda, x_1^T) \max_{\tilde{w}_1^N} \prod_{n=1}^{N} \prod_{m=1}^{M} p_m^{\alpha'_m}(w_n|w_1^{n-1}) \prod_{m=1}^{M} p_m^{\beta'_m}(\tilde{w}_n|w_n) \cdot \\ &\quad \cdot \prod_{m=1}^{M} \max_{s_{|\tilde{w}_{n,m}}} p_m^{\gamma'_m}(x_{|\tilde{w}_{n,m}}|s_{|\tilde{w}_{n,m}}) p_m^{\delta'_m}(s_{|\tilde{w}_{n,m}}|\tilde{w}_n) \end{aligned} \tag{5.5}$$

In equation 5.5 the coefficient λ_m is combined with the scales for every model, e.g. $\alpha'_m = \alpha_m + \lambda_m$ for the language model. The combination can be carried out for every element of the distribution individually, hence the number of applied language models may differ from the number of pronunciation and acoustic models:

$$\begin{aligned} p_\Lambda(w_1^N|x_1^T) &= C(\Lambda, x_1^T) \max_{\tilde{w}_1^N} \prod_{n=1}^{N} \prod_{j=1}^{L} p_j^{\alpha'_j}(w_n|w_1^{n-1}) \prod_{j=1}^{P} p_j^{\beta'_j}(\tilde{w}_n|w_n) \cdot \\ &\quad \cdot \prod_{j=1}^{A} \max_{s_{|\tilde{w}_{n,j}}} p_j^{\gamma'_j}(x_{|\tilde{w}_{n,j}}|s_{|\tilde{w}_{n,j}}) p_j^{\delta'_j}(s_{|\tilde{w}_{n,m}}|\tilde{w}_n) \end{aligned} \tag{5.6}$$

Equation 5.6 expresses that L language models, P pronunciation, and A acoustic models can be combined in a log-linear way at the level of word lattices. Note that the *lattice-based system combination (LSC)* from equation 5.2 is a special case of the *lattice-based model combination (LMC)* (equation 5.6).

6. Lattice Rescoring

System combination requires results, that have to be combined. Generation of these results is time consuming, as state-of-the-art speech recognition systems have a high real-time factor. An efficient approach to create theses different results is *rescoring*. A recognition with an initial model is performed to generate a reusable (approximative) representation of the search space. Subsequent models search only this noticeable reduced search space. Typically word graphs or N-best lists are chosen for representation, as mentioned in chapter 2.4.1. There are two general methods for rescoring: *full search* and *exact match*.

6.1. Full Search

In full search rescoring a complete recognition task is performed, with the only difference, that the search space is limited to hypotheses from a previous recognition. Since the search space is considerably reduced, results produced by different models can be generated faster than with a full recognition process. To yield useful results the remaining search space should contain enough different hypotheses, which can be selected by the other models. Usage of full search rescoring results in lattices or N-best lists with the same set of hypotheses but with different scores and word boundaries.

6.2. Exact Match

In exact match rescoring every arc a of a given lattice is rescored individually. Arcs are labeled with a word w_i together with its phoneme sequence, the word boundaries $(t_i, t_{i+1} - 1)$ and an acoustic score. Word boundaries and the acoustic score were determined in the initial recognition process:

$$a = (t_i, t_{i+1} - 1, w_i, score(w_i, t_i, t_{i+1} - 1)) \text{ , with:}$$
$$score(w_i, t_i, t_{i+1} - 1) = -\log p(x_{t_i}^{t_{i+1}-1}, s_{t_i}^{t_{i+1}-1} | w_i)$$

The sequence of HMM states $s_{t_i}^{t_{i+1}-1}$ is determined by the phoneme sequence of w_i and the used acoustic model. With the information provided by an arc a new score for that arc can be calculated using a different acoustic model. Within this type of rescoring the word boundaries determined by the initial model remain the same. This may negatively influence the performance of results generated by a rescoring, since for a different acoustic model other word boundaries may be optimal.

Exact match is chosen within this work, as it is faster than full search. A further advantage of exact match is that the structure of the rescored word lattices remains unchanged. Thus the combination of the lattices can be done arc-wise by traversing the lattices in the same way. The small loss of performance caused by the fixed word boundaries is compensated with a gain in runtime of the rescoring procedure.

6.2.1. Within-Word Rescoring

Given a word lattice produced during a previous recognition process exact match rescoring has to perform following steps for each arc in the graph. Firstly the features within the word boundaries $x_{t_i}^{t_{i-1}-1}$ have to be extracted using the signal analysis procedure of the new model. Secondly the HMM for the word, respectively for the phoneme sequence of the word at the arc w_i has to be built. In the last step a time alignment between the feature sequence and the states of the HMM has to be generated. The HMM state sequence of the this alignment ${s'}_{t_i}^{t_{i+1}-1}$ is used to calculate the new score for the arc:

$$a = (t_i, t_{i+1} - 1, w_i, score'(w_i, t_i, t_{i+1} - 1)) \text{ , with:}$$

$$score'(w_i, t_i, t_{i+1} - 1) = -\log p'(x_{t_i}^{t_{i+1}-1}, {s'}_{t_i}^{t_{i+1}-1} | w_i)$$

Figure 6.1 shows the elements involved in within-word rescoring.

Implementation Of Within-Word Rescoring

The rescoring is carried out using transducer based algorithms. Roughly speaking, a transducer is a finite state automaton, extended with output labels and weights for every arc; details can be found in [Mohri & Pereira+ 00]. An efficient implementation of transducer algorithms is provided by the RWTH FSA-toolkit [Kanthak & Ney 04]. The rescoring will be performed within this framework. Three transducers are used in the speech recognizer: the first transducer P converts a word into a sequence of phonemes, according to the pronunciation lexicon. A second transducer C transforms the sequences of phonemes to a sequence of context-dependent phones i.e. triphones. The mapping from this sequence of context-dependent phones to a sequence of distributions is done by a transducer denoted with H [Mohri & Pereira+ 00]. To obtain an HMM for a word w the composition

$$H \circ C \circ P \circ w$$

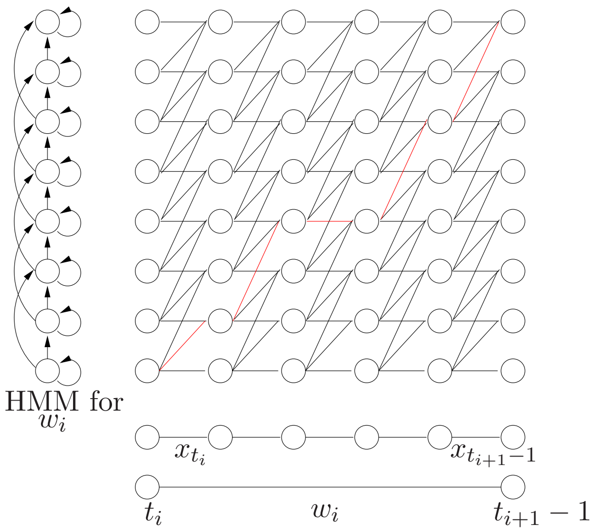

Figure 6.1.: Time alignment step of a sequence of features $x_{t_i}^{t_{i+1}-1}$ belonging to word w_i and the corresponding HMM representation of w_i in within-word rescoring

is calculated. Here, the composition of two transducers is denoted by $\circ$. If the exact pronunciation of the word w is already known only transducer $H \circ C$ has to be applied to build the desired HMM. This is exactly what is done during rescoring. The arcs in the word lattice are labeled with the used pronunciation (in addition to the word). This pronunciation is utilized to build a small transducer $phon(w_i)$ that simply represents the phoneme sequence of the pronunciation. In order to gain HMM for w_i

$$\text{HMM}(w_i) = H \circ C \circ phon(w_i)$$

is built. Now the time alignment between the HMM and the feature sequence can be computed. The best scoring path in this alignment (marked red in figure 6.1) is used to calculate the new score for the processed arc.

6.2.2. Across-Word Rescoring

Across-word models are standard in large vocabulary recognition systems. Thus, a rescoring performed on word lattices has to be capable of using these models. Since the information provided by one arc does not suffice to calculate the correct across-word score the adjacent phonemes have to be taken into account.

Due to the adjacent phonemes figure 6.2 has to be modified. The HMM is not solely built for the word w_i but also for the so-called across-word context: the surrounding

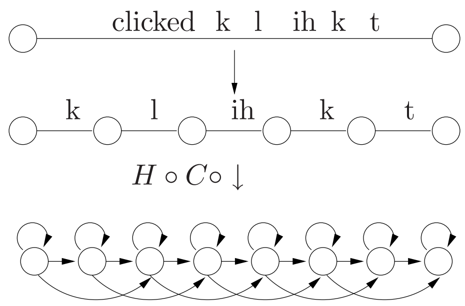

Figure 6.2.: Steps that are carried out to build an HMM out of the information provided by an arc in within-word rescoring

phonemes. Resulting in an HMM for ${}_{c_r}w_{i_{c_l}}$ where c_l, c_r represent the phonemes hypothesized before and after the word w_i.

Implementation Of Across-Word Rescoring

To perform across-word acoustic rescoring, the context information has to be included into the HMM for word w_i. Therefor the phoneme sequence of the word w_i is extended by the adjacent phonemes. Disambiguation symbols are used do distinguish the phonemes belonging to the word form the surrounding ones: $phon'(w_i) = c_l, \#_1, phon(w_i), \#_2, c_r$. These disambiguation symbols are used to mark the position in the HMM where the across word context begins. This information provides the possibility to cut the HMM at this position. This has to be done, because just the context information of theses phonemes are needed, not the phonemes themselves. A composition of $H \circ C$ and $phon'(w_i)$ yields to an HMM of c_l, w_i, c_r, but this HMM can not be aligned with the features for the word w_i, as it begins with the phoneme c_l and context w_{i_1} and not with w_{i_1} and contexts c_l, w_{i_2}. That also applies for the ending of the word.

To remove this unwanted states, the HMM is searched for the disambiguation symbols $\#_1$ and $\#_2$. Once the positions are determined, all states before $\#_1$ and after $\#_2$ are removed from the HMM. Resulting in the desired HMM for w_i with across-word context. The following steps are the same as for within-word rescoring.

Final Implementation Of Across-Word Rescoring

The first implementation worked perfectly for the emission scores but not for the TDP scores, because some important transitions which were removed during the removal of the unwanted states. To solve this problem the transducer $C \circ phon(w_i)$ is directly

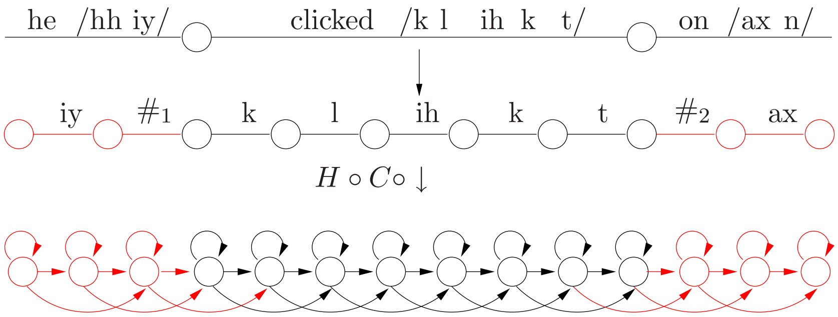

Figure 6.3.: Steps that lead to an HMM for word w_i with across-word context

constructed with the across-word context build$CP(_{c_l} w_{i c_r})$ to be independent from the transducer $H \circ C$. The HMM is obtained by computing

$$HMM(_{c_l} w_{i c_r}) = H \circ \text{build}CP(_{c_l} w_{i c_r}).$$

With this implementation, both, the emission scores as well as the TDP scores were correct. And a benefit in runtime was gained, because the search for the disambiguation symbols for every word arc could be omitted.

Further improvements in runtime performance were achieved by exploiting redundancies, that were introduced by the language model to the word lattice. In figure 6.4 the two arcs labeled with b may have exactly the same word boundaries, but because of the language model context the two arcs can not be combined. But an execution of the whole rescoring process is not necessary, if the word boundaries $t_i, t_{i+1} - 1$ as well as the across-word context for two arcs are identical. Therefor the once calculated value for $(w_i, t_i, t_{i+1} - 1, c_l, c_r)$ is cached and looked up on demand, to take advantage of these kind of redundancies.

6.2.3. Language Model Rescoring

Language model rescoring is an easier task. For every arc the $n-1$ predecessor words are determined, and the n-gram language model probability $p(w_i|w_{i-n+1}^{i-1})$ can be extracted and used as the new score, or added to the acoustic score at the arc. Unfortunately, this straight forward approach applies only to n-grams of a lower or same order than the initially used one, if the structure of the lattice has to be kept. If higher order n-grams are used, the history w_{i-n+1}^{i-1} stored in the lattice is not unique any more.

Figure 6.4 shows a part of a word lattice which was generated using a 2-gram language model. The probability for seeing c can be reestimated, using another 2-gram language

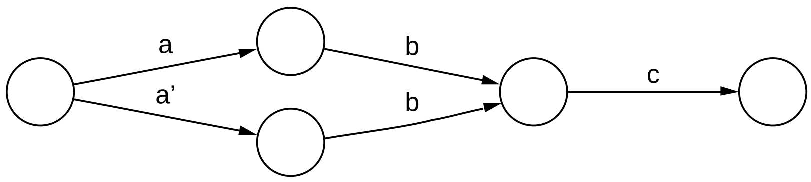

Figure 6.4.: Toy word lattice built with a 2-gram language model

model: $p'(c|b)$. But if a 3-gram language model is to be used, c has two different histories, thus two different probabilities: $p'(c|a, b), p'(c|a', b)$. This problem can be solved by assigning the maximum of the possible probabilities to the processed arc, since in the decoding step the maximum approximation is applied. In this work the same language model was used for generation and rescoring, thus no such approximation was used.

Because rescoring is used to generate the lattices for lattice-based system combination, the initial scheme presented in figure 5.1 has to be extended by the rescoring step (see figure 6.5).

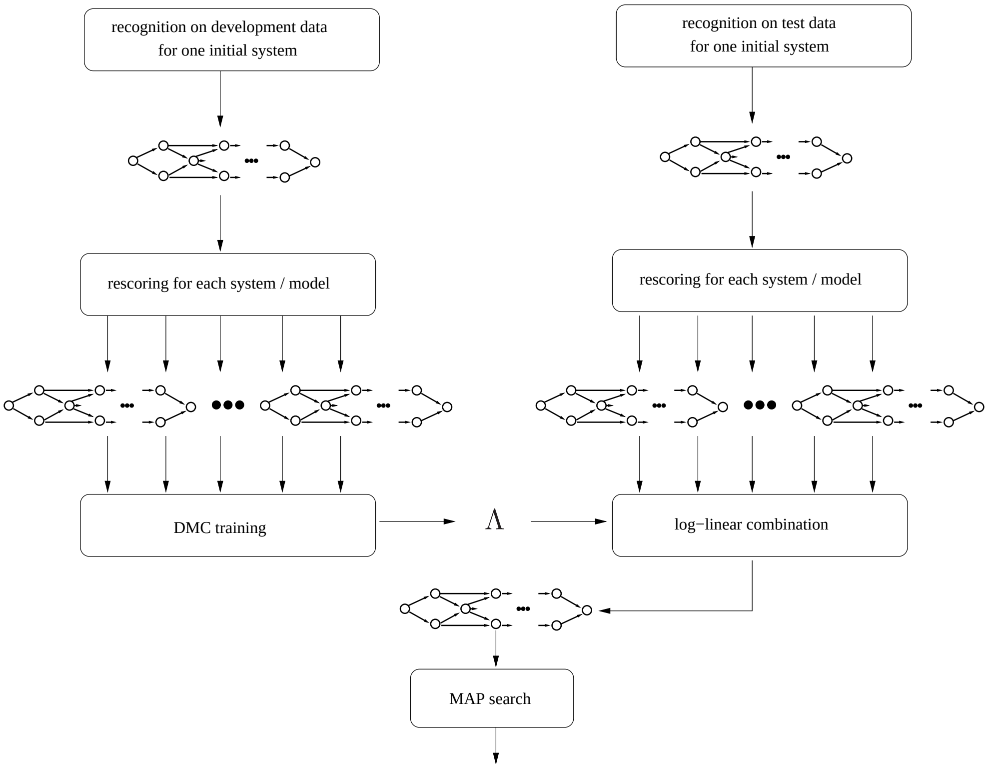

Figure 6.5.: Initial scheme for log-linear system combination at the level of word lattices, extended by a rescoring step to generate the needed lattices

7. Acoustic Models

System combination needs systems to combine, due to this fact, a variety of techniques which are used in automatic speech recognition, were applied to the baseline system, to generate variations. The baseline acoustic model consists of a CART, a LDA matrix and the trained weights for the Gaussian mixture models (GMM). To produce variations it is possible to change some settings in these parts: generate a different CART, allowing more or less leafs; estimate a different LDA matrix with more or less dimensions; generate more or less mixture densities. Other methods to retrieve different recognition results are training on different audio data, application of speaker adaptation methods, usage of different or additional features, utilization of a different pronunciation lexicon and many others.

Additional to variations of the acoustic model, the language model can be changed as well, by estimating the probabilities on different data, usage of a different context lengths or application of different smoothing techniques to handle unseen events.

Available standard methods presented in this chapter will be used to build different recognition systems for the lattice-based system combination.

7.1. Vocal Tract Length Normalization

Vocal tract length normalization (VTLN) is a speaker adaptation technique. Because the extracted feature vectors should ideally not contain information about the speaker, but only about what was said, adaptation techniques are used to minimize the characteristics of a speech signal. One element that is involved in the characteristics of a particular voice is the speakers vocal tract length. To remove the influence of the vocal tract length, a normalization factor has to be computed for a speaker's speech data, and normalization has to be performed using this factor.

VTLN In Training

VTLN must be used during the training process as well, since VTLN transforms the used features. During the training procedure, for each speech segment all possible normalization factors are applied and the best scoring one is used to extract the features for training:

$$\begin{aligned}\hat{\alpha}_r &= h(X_r, W_r, \vartheta) && (7.1)\\ &= \underset{\alpha}{\operatorname{argmax}}\, p_\vartheta(X_r|W_r;\alpha) \\ &\cong \underset{\alpha}{\operatorname{argmax}}\, \tilde{p}_\vartheta(X_r^\alpha|W_r) && (7.2)\end{aligned}$$

Estimation of the warping factor for a training speaker is done by applying equation 7.1, where X_r is a sequence of feature vectors belonging to this speech segment, W_r is the transcription, and $\hat{\alpha}_r$ is the acquired warping factor. The scores are calculated using the initial model $p_\vartheta(X_r|W_r)$. In equation 7.2 X_r^α denotes the feature vectors normalized with α.

Once the normalization factors for each speech segment are determined, the standard training procedure can be carried out:

$$\begin{aligned}X_r^{\hat{\alpha}_r} &:= f_\alpha(X_r)\\ \hat{\vartheta} &= \underset{\vartheta}{\operatorname{argmax}} \prod_{r=1}^{R} p_\vartheta(X_r|W_r;\hat{\alpha}_r)\\ &\cong \underset{\vartheta}{\operatorname{argmax}} \prod_{r=1}^{R} \tilde{p}_\vartheta(X_r^{\hat{\alpha}_r}|W_r)\end{aligned}$$

VTLN In Recognition

During the recognition process the warping factor has to be determined as well, but in a way this is more complicated than in training, because neither the speaker nor the word sequence is known. A straight forward application of Bayes' decision rule leads to a computational infeasible solution, since for every warping factor a single recognition process has to be performed:

$$\begin{aligned}W &= \underset{W'}{\operatorname{argmax}}(p(W')\max_\alpha p_{\hat{\vartheta}}(X|W';\alpha))\\ &\cong \underset{W'}{\operatorname{argmax}}(p(W')\max_\alpha \tilde{p}_{\hat{\vartheta}}(X^\alpha|W'))\end{aligned}$$

Using this technique the real-time-factor (RTF) of an automatic speech recognition system is multiplied by the number of used warping factors, to reduce complexity a text dependent function $h(X, W, \hat{\vartheta})$, similar to the function in training, is used to estimate the warping factor. Since this function is text dependent a preliminary transcription

W' is needed, it can be produced by applying an unadopted recognition pass, hence the system's RTF doubles.

Since it is observed that the distribution of acoustic vectors in the feature space varies for speakers with different vocal tract lengths independently of what was said, a text independent function $h(X, \hat{\vartheta})$ can be used to determine the warping factor. To estimate the warping factor, a GMM is estimated:

$$\begin{aligned} \tilde{\Upsilon} &= \operatorname*{argmax}_{\Upsilon} \prod_{r=1}^{R} p_{\Upsilon}(X_r|\alpha) \\ &\cong \operatorname*{argmax}_{\Upsilon} \prod_{r=1}^{R} p_{\Upsilon}(X_r^{\alpha}) \end{aligned}$$

$$\begin{aligned} \tilde{p}_{\tilde{\Upsilon}}(x) &= \sum_{l=1}^{L} c_l \mathcal{N}(x|\mu_l, \Sigma; \tilde{\Upsilon}), \quad \sum_{l=1}^{L} c_l = 1 \\ &\approx \max_{l} c_l \mathcal{N}(x|\mu_l, \Sigma; \tilde{\Upsilon}) \end{aligned}$$

Because this GMM is trained with normalized speech data, the discrimination between the different warping factors is possible. So the warping factor used for recognition will be:

$$\hat{\alpha} = \operatorname*{argmax}_{\alpha} \tilde{p}_{\tilde{\Upsilon}}(X^{\alpha})$$

Another approach is to model a distribution for each warping factor, which is estimated on unadopted data:

$$\Upsilon_{\alpha} = \operatorname*{argmax}_{\Upsilon} \prod_{r:\alpha_r=\alpha} p_{\Upsilon}(X_r),$$

hence the warping factor that will be used, is computed by the following equation:

$$\hat{\alpha} = \operatorname*{argmax}_{\alpha} p_{\Upsilon_{\alpha}}(X)$$

In [Molau 03] it is reported that this method performs better.

Usage of text independent detection of the warping factor needs only a negligible computational effort and is therefor a good idea.

VTLN Warping Function

The detected warping factor has to somehow be taken into account, this is done by the application of a warping function f_{α} on the frequency. A *piecewise warping rule* has following form:

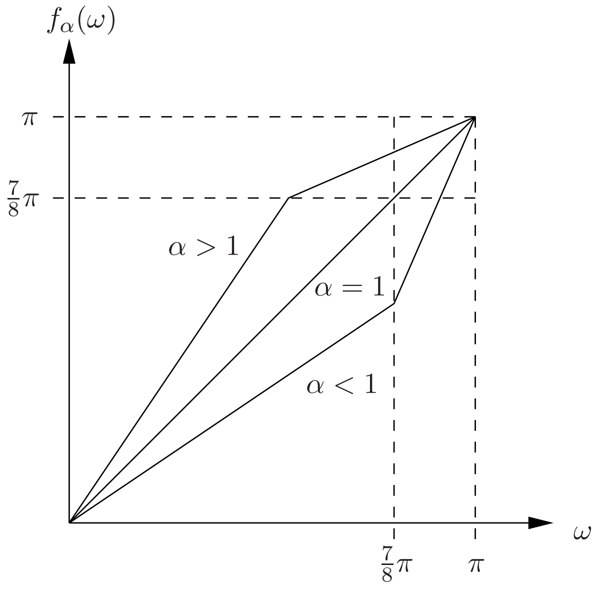

Figure 7.1.: Example of a piecewise warping function used in VTLN

$$f_\alpha(\omega) = \begin{cases} \alpha \cdot \omega & \omega \leq \omega_0 \\ \omega_0 + \frac{\pi - \omega_0}{\pi - \omega_0/\alpha} \cdot (\omega - \omega_0/\alpha) & \omega > \omega_0 \end{cases}$$

where ω_0 depends on α:

$$\omega_0 = \begin{cases} \frac{7}{8}\pi & \alpha \leq 1 \\ \frac{7}{8 \cdot \alpha}\pi & \alpha > 1 \end{cases}$$

Further details can be found in [Zahn & Waibel 97, Molau 03].

7.2. Voicedness Features

In addition to the standard MFCC feature extraction process described in chapter 2.1 a voiced-unvoiced feature is used, since these sounds form complementary classes. Utilization of a feature providing information about the voicedness of a time frame can help to better distinguish phonemes and consequently improve recognition results. Details are described in [Zolnay & Schlüter$^+$ 02, Zolnay & Schlüter$^+$ 03]. In this work those voicedness features are not solely used to improve the recognition process, but also to produce different results compared to a system not benefiting from those features.

7.3. Maximum Likelihood Linear Regression

Maximum likelihood linear regression (MLLR) is another speaker adaptation technique, which tries to transform the means of the acoustic model by applying a set of linear transforms, so finally the trained and transformed distributions fit the acoustic signal significantly better.

As described in chapter 2.2 the emission probabilities of the HMMs are modeled by multiple Gaussian mixture densities, a single Gaussian has following form:

$$p(\mathbf{x}|s,w,l) = \frac{1}{(2\pi)^{\frac{n}{2}}|\Sigma|^{\frac{1}{2}}} \exp\left\{-\frac{1}{2}(\mathbf{x}-\mu_{s,l})^T\Sigma^{-1}(\mathbf{x}-\mu_{s,l})\right\}$$

Let $\nu_{s,l} = [1, \mu_{s,l_1}, \ldots, \mu_{s,l_n}]^T$ be the extended mean vector, and $A \in R^{n\times(n+1)}$ the adaptation matrix, then the new mean for the Gaussian mixture is

$$\hat{\mu}_{s,l} = A\nu_{s,l},$$

thus the adapted probability distribution, used during the recognition process is

$$p(\mathbf{x}|s,w,l) = \frac{1}{(2\pi)^{\frac{n}{2}}|\Sigma|^{\frac{1}{2}}} \exp\left\{-\frac{1}{2}(\mathbf{x}-A\nu_{s,l})^T\Sigma^{-1}(\mathbf{x}-A\nu_{s,l})\right\}.$$

The matrices are estimated to maximize the likelihood of generating the adaptation data using the transformed models. Details can be found in [Legetter & Woodland 95, Pitz 05].

Before the recognition process each speech segment is assigned to a particular regression class. This is done by clustering the corpus. A set of transformation matrices will be estimated.

In the RWTH system MLLR is performed in two steps: firstly an initial standard recognition is performed using the initial acoustic model G. The transcript W, received in this way, is aligned with the acoustic features X. Alignment, features and initial model are used to estimate the MLLR matrices A, now the initial acoustic model is adopted using this set of matrices, resulting in a new acoustic model G'. A recognition process using G' results in a different transcript W' (figure 7.2). This transcript may be used as the final result for the recognition process, or it is used together with G' to produce a new set of matrices, A'', and therefor a new acoustic model G'' and transcription W''.

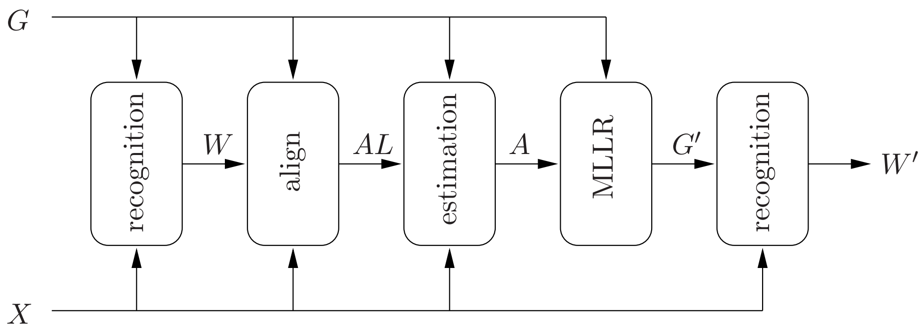

Figure 7.2.: Scheme for one step of MLLR

7.4. noLDA

Inspired by the system description of the CU-HTK 2003 system [Evermann & Chan+ 04] which makes use of an acoustic model, that is omitting the application of Linear Discriminant Analysis (or Heteroscedastic Linear Discriminant Analysis [Liu & Gales+ 03] in that case). The derivatives of the cepstral coefficients are taken into account during the feature extraction process, since the LDA that "learned" the is no longer part of the feature extraction process. As expected the performance of the recognition system is negatively influenced by omitting this part of the feature extraction process. Though the system performs worse than the baseline system, it is useful for taking it into account when performing system combination, as results in chapter 9 show.

7.5. Single Pronunciation Lexica

Another variant are lexica that provide only a single pronunciation per word. As proposed in [Soltau & Yu+ 04, Evermann & Chan+ 04] the use of single pronunciation lexica in training and recognition results in a small benefit. The single pronunciation lexica used in this work were generated by omitting all but the most frequent pronunciation variant from the lexicon. These frequencies were calculated on the recognition results of the development data.

8. Experimental setup

8.1. The NIST Error Rate

The standard error measure in speech recognition is the word error rate. As mentioned in chapter 2 the WER is calculated by counting the number of edit operations needed to transform the correct transcription w_1^N into the recognized word sequence $\hat{w}_1^{\hat{N}}$ divided by the number of words in the reference sentence:

$$\text{WER} = \mathcal{L}(w_1^N, \hat{w}_1^{\hat{N}})/N$$

Another error measure is the NIST error rate, which expresses basically the same as the WER. In addition to the calculation of the edit distance a mapping rule m is applied to the recognized word sequence. This mapping is used to convert the upper and lower case of recognized words, to merge words ("president in office" $\rightarrow$ "president-in-office"), or to separate words. The calculation of the word error rate is performed on the converted sequence of recognized words:

$$\text{NIST} = \mathcal{L}(w_1^N, \tilde{w}_1^{\tilde{N}})/N \quad \text{where: } \tilde{w}_1^{\tilde{N}} = m(\hat{w}_1^{\hat{N}})$$

Because the mapping allows aggregation and separation of words, calculation of the NIST error rate is not trivial on word lattices, thus the standard WER is used to calculate the lattice error rate.

8.2. Finding Useful Combinations

As a rule of thumb useful systems have a comparable error rate but produce different errors. A simple method to identify such systems is, to compare the recognition results of two or more systems.

In the first step of this comparison process the individual recognition results are aligned to the correct word sequence, after that the two correct word sequences are aligned and the recognition results are arranged according to that alignment.

The structure of the resulting multiple alignment corresponds to the example's structure in figure 8.2, now it so possible to simply count the kinds of errors every system

reference	the dog barks loud	⇒	the	dog	barks	-	loud
system 1	a frog barks a loud		a	frog	barks	a	loud
reference	the dog barks loud	⇒	the	-	dog	barks	loud
system 2	the old dog bark loud		the	old	dog	bark	loud

⇓

reference	the	-	dog	barks	-	loud
system 1	a	-	frog	barks	a	loud
system 2	the	old	dog	bark	a	loud

Figure 8.1.: Multiple alignment of the reference sentence and the recognition results of two different systems

produces. For example system1 produces two substitutions and one insertion, system2 produces one substitution and two insertions, both systems have an word error rate of 75% in this sentence. If the errors of the systems are compared each to another, it can be seen that system1 makes two errors which are not made by system2, the same holds for system2. If the use of an oracle that reveals which system decided correct would be accepted, then only the insertion made by both systems would remain, thus the word error rate reduces to 25%.

The *oracle word error rate* calculated by the comparison of two single-best outputs generated by different systems, is an indicator for the usability of two systems in a combination. If the optimized word error rate is not significantly less than one of the single-best outputs, the mistakes of the two systems are too equal, thus there will not be a large benefit from such a combining. Note that the oracle WER has to be compared to the true WER and not to the NIST error rate, as no mapping rules are applied.

8.3. Generation Of Word Lattices

To reduce the computational effort for the rescoring steps of the approach, the density of the generated lattice has to be in a proper range. During initial experiments the density was reduced by *forward pruning* in the recognition process, but this may lead to suboptimal results if the pruning is too tight. In further experiments “full” lattices were created during the recognition process and pruned with an implementation of the *forward-backward algorithm* [Kanthak & Ney 04] to yield proper lattice densities and maintain the best results.

In initial experiments lattices had been generated, using the baseline system. Rescoring was done with following variations: VTLN, VTLN+voicedness features, and noLDA, all these models are described in chapter 7.

As acoustic rescoring with MLLR is not supported in the RWTH system yet, lattices for further experiments were generated using the full setup: baseline augmented with VTLN, voicedness features and MLLR. Rescoring was executed to generate lattices for the baseline model, the baseline with VTLN, the baseline with VTLN and voicedness features, baseline without LDA, and for some tasks baseline using a single pronunciation lexicon.

8.4. Removal Of Redundant Arcs

For some tasks the acoustic model is extended by so-called *noise models* that are used to detect different types of interruptions of a speech signal or background noises. Unfortunately if a noise is to be hypothesized, one noise arc for every noise model is introduced in the final word lattice. This is an disadvantage, since for all arcs in the word lattice an acoustic rescoring has to be performed, thus the redundant noise arcs are not excluded from that process. A noise arc is called redundant, if another noise hypothesizing arc exists, that has a higher probability and the same source and target state. Performing the whole rescoring process for the redundant arcs is only a loss in performance, as there is no significant gain in information (figure 8.4.a). Redundant noise arcs form a more severe disadvantage, because redundant paths are introduced to the lattice, and redundant paths lead to a deferment of posterior probability of a hypothesis as described in [Macherey 98]. To solve this problem, all but the most probable noise arc are removed from the lattice in a post-processing step. As experiments showed, removing these arcs did not effect the results of the acoustic rescoring.

When no noises were hypothesized by separate models, an easy to solve redundancy was created by the silence model itself. In some cases a word was able to describe a sequence of features for a certain number of time frames, and a concurring hypothesis for the same sequence was formed by the same word with an adjacent silence arc. But since the structure of this alternate path is simple (figure 8.4.b), it could be removed by a trivial algorithm.

To evade further problems with redundant paths (8.4.c) the system weights were trained on N-best lists for the EPPS and HUB4 tasks.

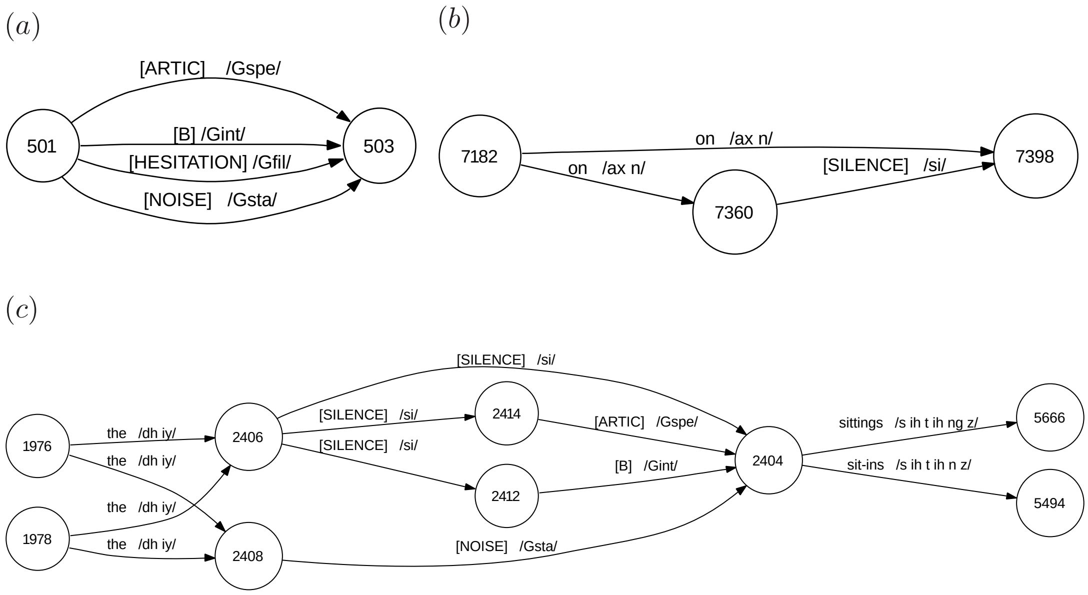

Figure 8.2.: Examples for redundant paths generated by used silence and noise models. (a) shows different noises, hypothesized for the same time frame; (b) presents a redundant alternative introduced by utilization of silence; (c) displays a set of redundant paths between states, introduced by the use of silence and noise models

8.5. Generation Of N-best Lists

Due to the problems occurred while working with word graphs, N-best lists were produced. Initially 1000-best lists were generated for every speech segment in the development corpus, later 2000-best lists were tested, but as results in chapter 9 show, there was no significant benefit. Due to the fact that the runtime of the DMC training algorithm heavily depends on the size of the used N-best lists, lists with $N = 1000$ were preferred.

Since the algorithm for producing N-best lists performs better on less dense lattices, the word graphs are heavily pruned before running the N-best extraction. Because of the *on-demand* implementation of the FSA algorithms [Kanthak & Ney 04], it can be checked quickly how many different word sequences are contained in the pruned word graph. The pruning threshold is raised until the desired number of hypotheses is held by the word graph. Then the extraction of the N-best lists is started

The final scheme for performing lattice-based system combination, where the individual system coefficients are trained on N-best lists is presented in figure 8.3.

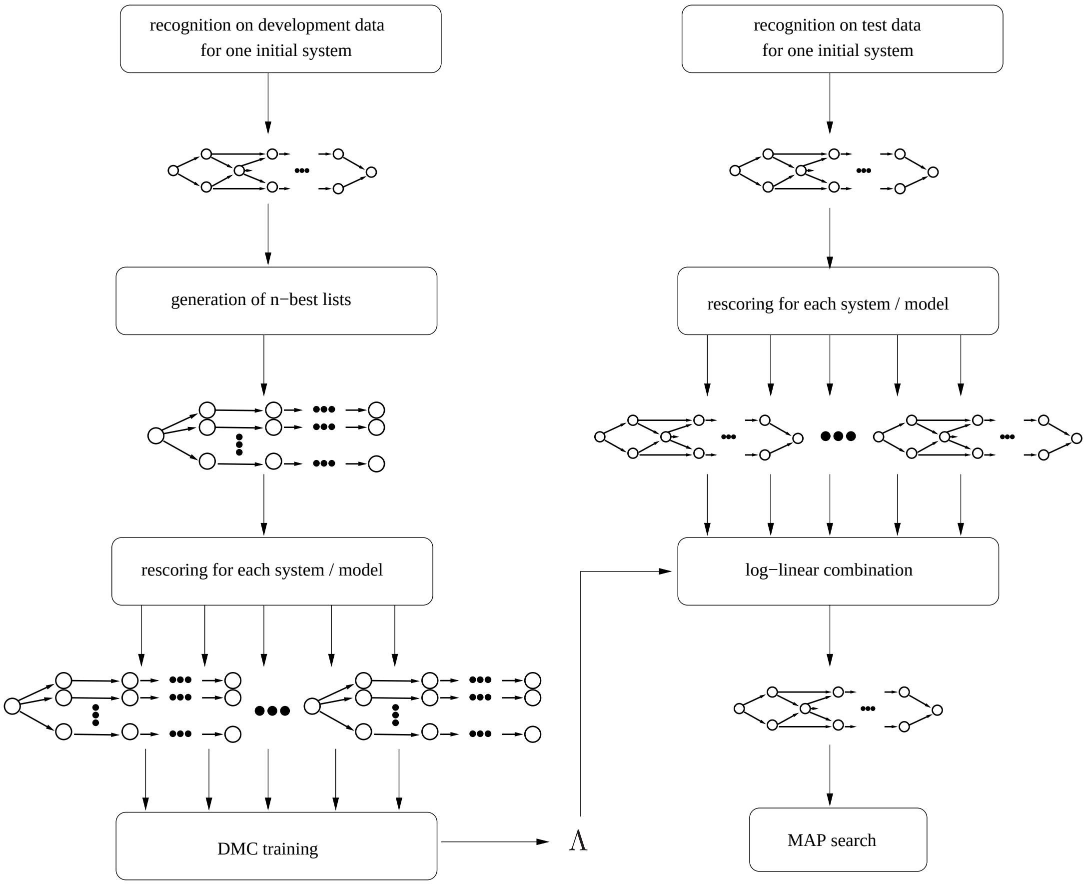

Figure 8.3.: The advanced scheme for the steps of the lattice-based system combination is extended by N-best list generation for the development corpus

8.6. Generation Of Acoustic Models

EPPS English

For initial tests and experiments only four different systems were used. The baseline system was generated by applying the training steps described in chapter 2.5. For the training 40.8h of audio data were used. Performance: 15.5% NIST. This model will be denoted as *baseline* in the following sections. One iteration of the complete training program took approximately seven days on an AMD Opteron with 2.2 GHz. The language model used for the experiments on the EPPS English task is a trigram language model, absolute discounting with interpolation was applied to model unseen events. Resulting in a perplexity of 85.51 on the development data. Details on the used language model and the EPPS task can be read in [Gollan & Bisani$^+$ 05].

	total [h]	speech [h]	# segments	# running words	# OOV
train-05	57.3	40.8	21,497	358,408	0
devel-05	13.8	3.7	1,400	35,474	391
eval-05	14.1	3.5	1,449	34,277	862

Table 8.1.: Statistics of the used corpora for training, development and evaluation in the EPPS English task. total denotes the total length of the audio data and speech the length of the data with spoken language. OOV denotes the number out-of-vocabulary words

The CART of the baseline system has 4500 leafs and seven splits of the single Gaussians yield to 446110 mixture densities. The LDA applied during the feature extraction reduces the dimension of the feature space to 33.

The second system is the baseline model extended by *vocal tract length normalization* described in chapter 7.1. VTLN was used during training and recognition. The error rate (NIST) was reduced by 4.5% relative, from 15.5% to 14.8% on the development set, and 15.7% to 15.0% on the evaluation data. Systems trained with VTLN are denoted as *vtln*.

The third system is briefly described in chapter 7.2 and includes VTLN and *voicedness features* in addition to the baseline setup, unfortunately these additional features do not have much effect on this task. Neither the performance on the development data, nor on the evaluation set was improved. But utilization of this additional features did not worsen the results as well. Since now all features are included, this model will be denoted as *complete*.

A suboptimal system was trained by omitting the application of LDA, as explained in chapter 7.4. As expected the performance on both data sets decreases, a raise of the

	devel-05	eval-05
initial	15.5	15.7
vtln	14.8	15.0
complete	14.8	15.0
mllr	13.9	14.1
spron (only training)	15.5	n/a
spron	15.7	n/a

Table 8.2.: Performance of used systems on development and evaluation data. Performance is given in NIST (%)

NIST error rate of 5% respectively 8.8% relative, compared to the baseline system, was observed on development and evaluation data.

These four systems were initially used to determine effects of acoustic rescoring, and to perform initial tests in system combination.

At a later stage two additional systems were introduced. The set of models were expanded by one, that makes use of *maximum likelihood linear regression* (described in chapter 7.3). MLLR was used in addition to the *complete* model, and is denoted as *mllr*. Utilization of MLLR decreased the word error rate by 6% relative compared to the *complete* model on both development and evaluation data.

A further acoustic variation was generated by altering the training and recognition lexicon. As illustrated in chapter 7.5 all but one pronunciation variant were removed from the lexica. The standard training lexicon holds 14732 pronunciations and it was reduced by 1097, from the recognition lexicon 3841 pronunciation entries were removed (initially 58110 pronunciations were contained). Usage of a single pronunciation lexicon in both training and recognition performs slightly worse than the baseline system. But the runtime of the training procedure is reduced drastically: instead of spending approximately three days, the seven splitting and aligning steps were done in less than one day (both on a AMD Opteron with 2.2 GHz). When the single pronunciation lexicon was only used in training, no loss in performance was observed.

EPPS Spanish

Since initial tests were performed with the EPPS English task, the gained experience was used and directly the five different systems were generated. The creation of a single pronunciation lexicon was set aside, as the existing lexicon does not contain a considerable number of pronunciation variants, hence no single pronunciation model was trained. Error rates (NIST) for the used acoustic variations are listed in table 8.4.

A recognition for the *nolda* system was not carried out, as no significant knowledge is expected from this experiment. A trigram language model was applied in the experiments, the same smoothing as in the English task was chosen. Perplexity on the development corpus is 85.77.

The setup for the baseline model is in general the same as for the EPPS English task: The CART of the baseline system has 4500 leafs and seven splits of the single Gaussians yield to 419487 mixture densities. The LDA applied during the feature extraction reduces the dimension of the feature space to 33.

	total [h]	speech [h]	# segments	# running words	# OOV
train-05	54.2	41.4	27,750	333,227	0
devel-05	13.8	3.7	1,188	32,438	297
eval-05	14.0	3.8	1,690	31,925	365

Table 8.3.: Statistics of the used corpora for training, development and evaluation in the EPPS Spanish task

	devel-05	eval-05
initial	13.2	14.5
vtln	12.4	13.7
complete	12.3	13.5
mllr	11.7	12.9

Table 8.4.: Performance of trained systems on the development and evaluation corpus of the EPPS Spanish task. Performance is given in NIST (%)

HUB4 English

The standard hub4 acoustic training material includes 144h of audio data, but only 97h are used for training, since it was observed that the additional audio material has no significant effect on the resulting models. Because of this large amount of training material, one iteration of the acoustic training procedure lasts approximately nine days on a AMD Opteron with 2.2 GHz. Instead of an initial alignment gained by linear segmentation, an already existing model was used to generate the time alignment, as some speech segments in the training data are longer than 100 seconds. Processing these segments with the initial acoustic model generated by linear segmentation exceeded the available main memory. The used language model has a perplexity of 118 on the development set, details can be found in [Beyerlein & Aubert^{+} 99].

The CART of the baseline system has 4500 leafs and seven splits of the single Gaussians yield to 537858 mixture densities. The LDA applied during the feature extraction reduces the dimension of the feature space to 45.

According to the EPPS task, the baseline was extended by VTLN and voicedness features. The standard training lexicon contains 91094 pronunciation and the recognizer lexicon contains 85715, a large number are pronunciation variants (15985 respectively 19445), thus a single pronunciation lexicon was generated for training and for recognition. The effect of the accelerated training procedure could as well be observed. But due to limited time, no experiments could be carried out with these systems. Performances of the individual systems are listed in table 8.6.

	total [h]	speech [h]	# segments	# running words	# OOV
training	200	96.7	26,167	1,108,623	0
eval-97 (devel)	3.05	2.90	657	32576	351
eval-98 (eval)	3.00	2.92	827	32223	393

Table 8.5.: Statistics of the used corpora for training, development and evaluation in the HUB4 English task

	eval-97	eval-98
initial	26.78	25.27
vtln	-	24.14
complete	-	23.97

Table 8.6.: Performance of trained systems on the development and evaluation corpus of the HUB4 English task. Performance is given in WER (%)

9. Experimental Results

9.1. Rescoring Results

Alignment Strategies

An approach to reduce the runtime of the rescoring algorithm was to share the time alignment made by one model for all models that should be used in rescoring. Results of this approach are listed in table 9.1. It is easy to see that the shared alignment does not perform well, even the performance of more sophisticated models decreases, therefor a disjoint time alignment was used for the experiments.

	shared alignment	disjoint alignment
initial	19.69	19.69
vtln	19.96	18.90
complete	19.85	18.95
nolda	21.81	20.56

Table 9.1.: Comparison of two alignment strategies used in rescoring. Performance is given in WER (%)

Runtime Of Acoustic Rescoring

To make the first implementation of the across-word acoustic rescoring usable, the TDP rescoring was performed without taking the across-word context into account. Due to this fact, two different time alignments have to be calculated, one for the emission probabilities of the HMM, using the across-word context, and one for the TDPs. The second implementation works correct and redundancies in the processed word lattices are exploited (see: chapter 6.2.2).

As results in table 9.2 show, the first implementation is rather slow, and the runtime is even slower than a full recognition, whereas the second implementation is much quicker and calculates the correct TDP scores.

	lattice density	time [h]
recognition	-	≈ 63
first implementation	282	≈ 81
second implementation	333	≈ 10

Table 9.2.: Runtime of the rescoring process on word lattices, compared to a full recognition of the EPPS English evaluation corpus, on a AMD Opteron with 2.2 GHz

9.2. EPPS English

Oracle WER

Analysis of the pairwise calculated oracle WER (see: table 9.3) showed for the six created systems that five of them are useful for system combination. The use of the single pronunciation system was set aside, as the WER for the baseline system (19.7 %) is not reduced very much compared to the oracle WER of baseline+spron. The best oracle WER was generated by the use of mllr and nolda.

	vtln	complete	mllr	nolda	spron
baseline	15.37	15.27	14.74	15.33	17.91
vtln	-	15.98	15.27	14.79	-
complete	-	-	16.02	14.47	-
mllr	-	-	-	14.22	-

Table 9.3.: Pairwise calculated oracle WER (%) on the rescoring results for the EPPS English development corpus

9.2.1. Initial Experiments

The initial experiments were carried out using only four systems to determine effects of the rescoring process, and to test different learning criteria. The lattices were generated by performing a recognition with the baseline model and the acoustic rescoring was executed for: vtln, complete and nolda. Results of the rescoring processes are listed in table 9.5 for the EPPS English development and evaluation corpora. The correct transcription is merged into the development lattices, to ensure that it is contained by the lattice. This is needed for the training with the MMI criterion, thus the lattice error rate is zero.

	devel-05		eval-05
	lattice	1000-best	lattice
baseline	19.72		20.91
lattice error	0.0	10.47	7.71
lattice density	406	-	282

Table 9.4.: Quality of used lattices and N-best lists given as lattice error rate and lattice density for initial experiments on EPPS English. Performance is given in WER (%)

	devel-05		eval-05	
	recognition	lattice (rescore)	recognition	lattice (rescore)
initial	15.5	-	16.0	-
vtln	14.8	15.1	15.0	15.3
complete	14.8	15.1	15.0	15.1
nolda	n/a	16.7	n/a	17.0

Table 9.5.: Comparison of a full recognition and a rescoring carried out on lattices for the EPPS English task. Lattice were generated with the baseline system. Performance is given ins NIST (%)

No recognition was performed for the nolda model, because no significant information was expected by carrying out this experiment. The rows for vtln and complete show that the once determined word boundaries are usable for the rescoring process, without loosing too much performance.

MMI On Lattices

The first tested criterion is the MMI criterion (see: chapter 4.2). As results in 9.6 show, training of the coefficients Λ using the MMI criterion was not successful, since the error rate raises on the evaluation corpus for all tested combinations. A combination of all four models was not tested, because preliminary results were not encouraging. These results are not surprising, as it was reported in [Beyerlein 00] that the MMI criterion performs not well on discriminative model combination.

MWE On Lattices

Initially it was tried to train the coefficients Λ on the development lattices, but during the training iterations the error rate was not reduced, even on the development data. As a possible reason the redundant noise-silence paths could be determined, because they defer the posterior probability for a hypothesis in a lattice. Because of redundant paths a hypothesis is contained multiple times in the lattice. According to the explanations in chapter 8, N-best lists are used for further experiments.

MWE On N-best Lists

Estimating the weights Λ for the models was successful on N-best lists, using the MWE criterion. Obtained improvements are listed in table 9.6. The best combination is given by combining vtln, complete, and nolda. This combination results in 14.4% NIST on the evaluation data. Application of ROVER results in 14.8% NIST, the best ROVER combination was given by vtln, complete and nolda, too.

		devel-05	eval-05
best system	complete	15.1	15.1
MMI	- baseline, vtln, complete	16.7	17.6
	- baseline, vtln, nolda	16.5	17.5
	- baseline, complete, nolda	16.6	17.5
	- vtln, complete, nolda	16.2	17.4
MWE lattice	- baseline, vtln, complete	17.2	17.1
MWE N-best	- baseline, vtln, complete	14.2	14.6
	- baseline, vtln, nolda	14.4	14.7
	- baseline, complete, nolda	14.7	14.6
	- vtln, complete, nolda	**13.7**	**14.4**
	- baseline, vtln, complete, nolda	16.1	15.2

Table 9.6.: Compared performance of the MMI and MWE criteria on lattices and N-best lists. Performance is given in NIST (%)

It turned out that 30 iterations of iterative improvement of the system coefficients assure converged weights, these 30 iterations last approximately 20 hours on a AMD Opteron with 2.2 GHz. Figure 9.1 shows the development of the error rate during the iterative improvement of the coefficients Λ for development and evaluation corpus. For all combinations carried out the same initial values for the iterations were chosen, for the lattice-based model combination the coefficient for the language model was set to 1, the remaining coefficients were set to 1/20. This value is the multiplicative inverse to the language model scale used in recognition. In the case of lattice-based system combination all coefficients were set to 1/20.

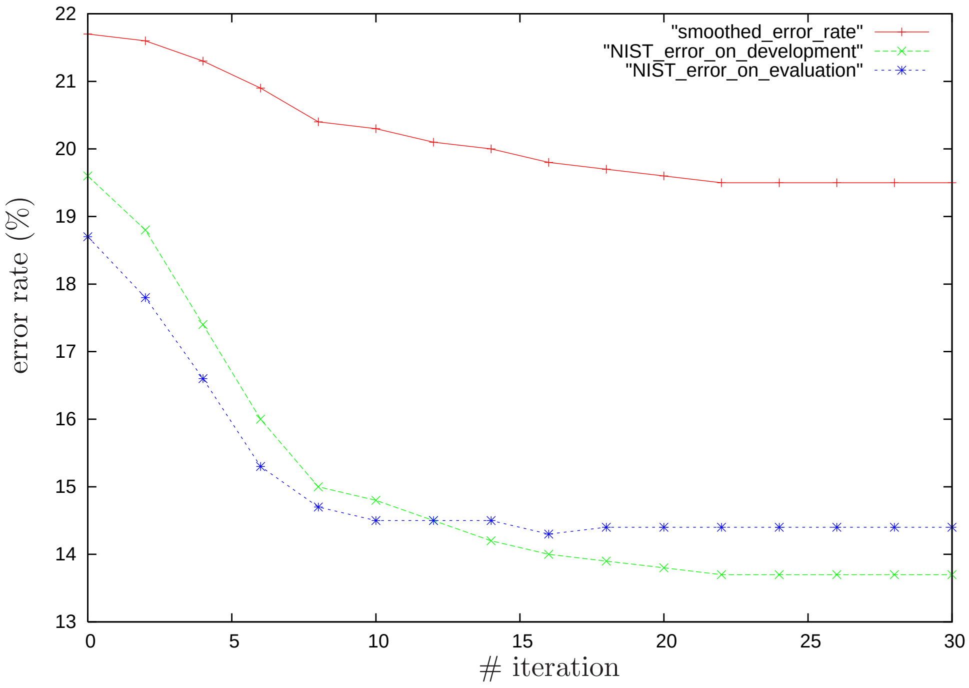

Figure 9.1.: Development of error rate using iteratively improved Λs with the MWE criterion on 1000-best lists for *vtln+complete+nolda* on devel-05 (green), eval-05 (blue) in NIST (%) and smoothed error rate on devel-05 (red)

9.2.2. Further Experiments

With the information acquired by the initial experiments, the aim was to include the system with MLLR into the combination. Since no rescoring with MLLR was possible, new lattices had to be generated and used as initial lattices. Table 9.7 displays the lattice error rate and the density for the development lattices and N-best lists. Table 9.8 shows the NIST error rates of the recognition and the gained results by rescoring the development lattices and N-best lists with the other systems. Tables 9.9, 9.10 give the corresponding results for the evaluation corpus.

The results on the N-best lists compared to the ones gained on word lattices, point out that stored search space is too small to acquire the same, or almost the same, results by rescoring. For example there is a large gap between the performance of rescoring carried out on N-best lists and lattice for the nolda system, which is by far too good on N-best lists. But due to the computational effort for generating N-best lists, no $N > 2000$ was used, the generation of 1000-best lists took approximately one and a half day on nine AMD Opteron with 2.2 GHz, creation of 2000-best lists consumed the double time.

	recognition	pruned	1000-best	2000-best
mllr	17.84			
lattice error	4.59	5.10	9.90	9.53
lattice density	2487.9	291.7	-	-

Table 9.7.: Quality of the used N-best lists and lattices, given in lattice error (WER %) and lattice density compared to the *mllr* system for the EPPS development corpus

	recognition	lattice (rescore)	1000-best	2000-best
initial	15.5	15.8	15.1	15.1
vtln	14.8	15.2	14.8	14.8
complete	14.8	15.2	15.0	15.0
mllr	13.9	-	-	-
nolda	n/a	16.6	15.1	15.1

Table 9.8.: Performance of rescoring on lattices and N-best lists compared to a full recognition on the EPPS English development corpus. Performance given in NIST (%)

	recognition	pruned
mllr	19.12	
lattice error	5.37	5.83
lattice density	2545.1	333

Table 9.9.: Quality of the used N-best lists and lattices for the EPPS English evaluation corpus, given in lattice error (WER %) and lattice density compared to the *mllr* system

	recognition	rescoring
initial	15.7	15.9
vtln	15.0	15.7
complete	15.0	15.5
mllr	14.1	-
nolda	n/a	17.3

Table 9.10.: Performance of rescoring on lattices compared to a full recognition on the EPPS English evaluation task. Performance given in NIST (%)

As results in tables 9.7 and 9.9 show, the forward-backward pruning step applied to the word lattices from the recognition, raises the lattice error rate negligible, but the size (lattice density) decreases drastically. Tables 9.8 and 9.10 show that the rescoring results gained by rescoring deviate significantly from the recognition results. This effect was not observed during initial experiments. It is assumed that the word boundaries selected by the mllr system do not suit the other systems.

Lattice-Based System Combination

The weights for the combinations listed in table 9.11, were estimated, using the MWE criterion on N-best lists. The combination was done according to equation 5.2. Results of combinations that are not listed in table 9.11 can be found in the appendix.

	smoothed error in training	devel-05	eval-05
baseline, mllr	19.59	13.9	14.0
nolda, mllr	19.40	13.7	14.0
baseline, vtln, mllr	19.58	13.9	14.0
baseline, nolda, mllr	19.42	13.7	14.0
baseline, vtln, nolda, mllr	19.47	13.8	14.0
vtln, complete, nolda, mllr	19.47	13.7	14.1
baseline, vtln, complete, nolda, mllr	19.51	13.8	14.1

Table 9.11.: Performance of the lattice-based system combination on EPPS English

Some of the combinations were able to perform better than the best single system (mllr), but the gained performance is not significant. A possible reason is that the rescoring results deviate too much from the ones of a full recognition, therefor the trained coefficients Λ do not reflect the true conditions.

Lattice-Based Model Combination

To improve the combinatorial results gained by lattice-based system combination, the individual systems were split into their knowledge sources, and weights were trained for each of these models, according to equation 5.6. In table 9.12 some of the gained results are listed (additional results in appendix).

	smoothed error in training	devel-05	eval-05
baseline, mllr	20.60	13.6	13.9
nolda, mllr	20.34	13.3	14.0
baseline, vtln, mllr	20.63	13.7	13.9
baseline, complete, mllr	20.69	13.7	13.9
baseline, nolda, mllr	20.34	13.3	13.9
vtln, nolda, mllr	20.35	13.3	13.9
baseline, vtln, complete, mllr	20.75	13.9	14.0
baseline, vtln, nolda, mllr	20.41	13.4	13.9
baseline, vtln, complete, nolda, mllr	20.79	14.1	14.5

Table 9.12.: Performance of the lattice-based system combination on EPPS English. Performance given in NIST (%)

One of the best combination, according on the development data, is given by combining nolda and mllr, but the benefit, observed on the development data could not be transfered to the evaluation data. As a reason, the overestimated performance of nolda on the N-best lists could be determined. The gain in performance on a combination that does not make use of nolda was transfered to the evaluation data (*baseline+mllr*). The trained weights do not reflect the distribution of the evaluation data.

9.3. EPPS Spanish

Oracle WER

Analogous to the EPPS English task, five systems were used: baseline, vtln, complete, nolda, and mllr. At the beginning, the oracle WER for every pair of these systems was calculated to determine, how promising a combination is. The results in table 9.13 show that all systems can be taken into account for lattice-based system combination as the error rate (WER) for the mllr (13.66%) system is reduced significantly by all combinations, especially for nolda+mllr.

	vtln	complete	mllr	nolda
initial	11.97	11.79	11.40	11.57
vtln	-	12.29	11.73	11.44
complete	-	-	12.43	11.39
mllr	-	-	-	11.07

Table 9.13.: Oracle WER (%) calculated on the rescoring results of the EPPS Spanish development corpus

Rescoring

According to the English task, the initial lattices were generated, using the mllr model. Lattices densities and results of the rescoring are listed in tables 9.14 to 9.17. Again the benefit on 2000-best lists was not huge enough to use them for the training procedure, thus 1000-best lists were applied for the training of the system coefficients. The gap between rescoring on lattices and N-best lists could be observed as well, especially for the nolda system.

	recognition	pruned	1000-best	2000-best
mllr	13.66			
lattice error	3.09	3.39	6.51	6.25
lattice density	2333.8	420	-	-

Table 9.14.: Quality of the used N-best lists and lattices, given in lattice error (WER %) and lattice density compared to the *mllr* system. Results for the EPPS Spanish development corpus

	recognition	lattice (rescoring)	1000-best	2000-best
initial	13.2	13.1	12.6	12.6
vtln	12.4	12.4	12.2	12.3
complete	12.3	12.3	12.3	12.3
mllr	11.7	-	-	-
nolda	n/a	14.4	13.1	13.2

Table 9.15.: Performance of rescoring on lattices and N-best lists compared to a full recognition on the EPPS Spanish task. Performance given in NIST (%)

	recognition	pruned
mllr	15.21	
lattice error	4.11	4.47
lattice density	2387.7	400

Table 9.16.: Quality of the used lattices, given in lattice error (WER %) and lattice density compared to the *mllr* system. Results for the EPPS Spanish evaluation corpus

	recognition	rescoring
initial	14.5	14.6
vtln	13.7	13.7
complete	13.5	13.5
mllr	12.9	-
nolda	n/a	15.5

Table 9.17.: Performance of rescoring on lattices compared to a full recognition on the EPPS Spanish evaluation corpus. Error rate is given in NIST (%)

Lattice-Based System Combination

Like in the English task, the lattice-based system combination does not bring any benefit, the best performing combinations are at least as good as the best single model.

	smoothed error in training	devel-05	eval-05
baseline, mllr	14.23	11.6	13.0
vtln, mllr	14.26	11.7	13.0
nolda, mllr	14.23	11.6	12.9
baseline, vtln, mllr	14.24	11.7	12.9
baseline, complete, mllr	14.29	11.7	13.0
baseline, nolda, mllr	14.21	11.6	12.9
vtln, nolda, mllr	14.26	11.7	12.9
baseline, vtln, nolda, mllr	14.23	11.7	12.9
vtln, complete, nolda, mllr	14.32	11.7	12.9
baseline, vtln, complete, nolda, mllr	14.29	11.8	12.9

Table 9.18.: Performance of the lattice-based system combination on EPPS Spanish in NIST (%)

Lattice-Based Model Combination

Table 9.19 shows the results for lattice-based model combination on the EPPS Spanish task. The best combination turned out to be nolda, mllr, which corresponds to the English task. But the error, produced by training the weights on N-best lists is smaller than in the English task. Therefor the benefit can be transfered to the evaluation corpus. Additional results can be found in the appendix. The approach worked on this task, because the results gained by rescoring do not deviate much, except for the nolda system, form the actual recognition.

9.4. HUB4 English

Oracle WER

Four systems were created for the HUB4 English task: baseline, vtln, complete and nolda. In the first step was to determine which systems will be used in the combination. The pairwise calculated oracle WER (table 9.20) indicates that all systems provide useful information, as the error rate acquired by a recognition with the baseline system on the development data (table 8.6) was significantly reduced.

	smoothed error in training	devel-05	eval-05
vtln, mllr	15.30	11.7	12.6
complete, mllr	15.46	11.9	12.7
nolda, mllr	**14.96**	**11.5**	**12.4**
baseline, complete, mllr	15.25	11.9	12.7
baseline, nolda, mllr	14.95	11.8	12.7
vtln, nolda, mllr	14.94	11.8	12.6
complete, nolda, mllr	15.01	11.8	12.5
baseline, complete, nolda, mllr	14.98	12.0	12.8
vtln, complete, nolda, mllr	14.98	12.0	12.7
baseline, vtln, complete, nolda, mllr	15.25	13.0	13.6

Table 9.19.: Performance of the lattice-based model combination on EPPS Spanish in NIST (%)

	vtln	complete	nolda
baseline	20.53	20.55	21.60
vtln	-	21.53	20.20
complete	-	-	20.38

Table 9.20.: Oracle WER calculated on the rescoring results of the HUB4 development corpus

Rescoring

Due to limited time a full recognition was not carried out for all systems on the development corpus. Tables 9.21 and 9.23 list the quality of the created lattices and N-best lists for the evaluation and development corpus. The 1000-best lists were used for the training of the coefficients, but no significant benefit of the resulting combinations was expected, as the N-best lists were by far to small to expect any improvements. The lattice generated by a recognition on the evaluation corpus were not pruned any further as the lattice density was already in a proper range. Tables 9.22 and 9.24 show the results of the acoustic rescoring compared to a full recognition.

	recognition	pruned	1000-best	2000-best
baseline	26.73			
lattice error	8.05	8.27	20.90	20.52
lattice density	∞	∞	-	-

Table 9.21.: Quality of the used N-best lists and lattices, given in lattice error (WER %) and lattice density compared to the *baseline* system. Results for the HUB4 development corpus

	recognition WER	lattice) WER	1000-best WER
baseline	26.73	-	-
vtln	-	25.87	25.74
complete	-	25.88	25.51
nolda	-	30.45	27.01

Table 9.22.: Performance of rescoring on lattices and N-best lists on the HUB4 development corpus. Performance given in WER (%)

	recognition
baseline	25.45
lattice error	9.93
lattice density	204

Table 9.23.: Quality of the used lattices, given in lattice error (WER %) and lattice density compared to the *baseline* system. Results for the HUB4 evaluation corpus

	recognition	rescoring
initial	25.45	-
vtln	24.14	25.05
complete	23.97	24.87
nolda	-	28.86

Table 9.24.: Performance of rescoring on lattices compared to a full recognition on the HUB4 evaluation task. Performance given in WER (%)

Lattice-Based System Combination

	smoothed error in training	eval-97	eval-98
baseline, vtln	25.70	25.23	24.32
baseline, complete	25.57	25.19	24.33
baseline, nolda	26.52	26.77	25.63
vtln, complete	25.56	25.50	24.53
vtln, nolda	25.72	25.66	24.88
complete, nolda	25.60	25.68	24.67
baseline, vtln, complete	25.53	25.07	24.19
baseline, vtln, nolda	25.68	25.20	24.51
baseline, complete, nolda	25.57	25.10	24.38
vtln, complete, nolda	25.54	25.35	24.54
baseline, vtln, complete, nolda	25.52	25.05	24.32

Table 9.25.: Results of the lattice-based system combination on the HUB4 corpora. Performance given in WER (%)

As in the previous tested cases the lattice-based system combination is not capable of improving the performance of the overall system. In general two reasons are responsible for this result: (1) the used N-best lists are by far too small, which worsens the quality for the trained coefficients; (2) the rescoring results deviate 1% absolute from the full recognition (see: table 9.24). This may worsen the evaluation results as well.

Lattice-Based Model Combination

No improvements could be achieved by using the lattice-based model combination instead of the lattice-based system combination. None of the combination performed better than the single *complete* system.

	smoothed error in training	eval-97	eval-98
baseline, vtln	25.95	26.29	24.35
baseline, complete	25.71	26.50	24.73
baseline, nolda	26.31	28.08	25.93
vtln, complete	25.72	26.29	24.43
vtln, nolda	25.75	25.72	24.33
complete, nolda	25.64	26.02	24.23
baseline, vtln, complete	25.77	26.39	24.47
baseline, vtln, nolda	25.85	26.01	24.47
baseline, complete, nolda	25.70	26.39	24.54
vtln, complete, nolda	25.65	25.84	24.10
baseline, vtln, complete, nolda	25.80	27.13	25.13

Table 9.26.: Results of the lattice-based model combination on the HUB4 corpora. Performance is given in WER (%)

10. Conclusion and Outlook

Within this work the effects of acoustic rescoring were analyzed, and how the results of a rescoring process can be combined to improve the performance of a speech recognition system. The acoustic rescoring with fixed word boundaries, which was applied to generate congruent lattices, consumes less computational resources than a full recognition process. The fixed word boundaries are a small disadvantage, which is negligible. If a new a new acoustic model has to be tested, and an idea of the performance of this model shall be obtained. This rescoring method can be used to generate lattices that have be combined later. The combination of lattices to improve the overall systems performance was the second part of this work. It turned out that the lattice-based combination performs better than the ROVER approach on the EPPS English corpus. This is not surprisingly, as the used ROVER algorithm only performed a majority vote over the provided single-best sequences. Unfortunately the lattices-bases system combination could not be compared to the confusion network combination, because it was not available. This comparison would be interesting, since the CNC is as well able to take alternative output of a recognition system into account.

Two different criteria were tested. The MWE criterion turned out to be useful, in order to estimate weights for the system combination that are robust enough to transport the observed benefit from the development data to evaluation data. Whereas the MMI criterion was not useful to reduce the amount of word errors.

A further interesting result is that taking suboptimal systems in combination with sophisticated systems, improves the over-all performance of a combined system. Performance of the EPPS English system could be improved by 1.4% relative, from 14.1% NIST, generated by a system, that makes use of VTLN, voicedness features and MLLR, to 13.9% NIST, by combining this system with i.e. the baseline. A 3.8% relative reduction of the NIST error rate on the EPPS Spanish task could be achieved, the full setup has 12.9% NIST and was reduced by the a combination of mllr system together with a suboptimal system (baseline without LDA) to 12.4% NIST. On the HUB4 task no improvements could be observed.

It emerged that the investigated approach heavily depends on the quality of the used N-best lists during training, since the used N-best lists were relatively small. Due to the computational effort for the generation of N-best lists, no $N > 2000$ was tested. Lists with $N = 1000$ were used to estimate the coefficients for the log-linear combination, because the lattice error rate of larger N-best lists were not significantly better. The negative effect of too small N-best lists could be observed on the EPPS English task, there was a gap of 1.5% absolute between the rescoring result on 1000-best list and word lattices. Coefficients trained on the basis of these lists do not reflect the conditions on the evaluation data, thus the benefit is just small or not existent as in the case of HUB4. The HUB4 N-best lists showed that the lattice error rate has to be significantly below the single-best error rate to make DMC usable.

It turned out that using all available models in one combination does not necessarily minimize the error rate. In fact, combinations which did use only two models performed better. The iterative optimization converge into a local minimum when using five systems, thus the combination of two performs better.

Combining whole speech recognition systems, or the individual knowledge bases, is a promising path which should be followed to improve the performance of a speech recognition system. The different models that could be used in that combination are generated during the creation of a good single-pass system anyway. There is no reason to waste that additional performance, though it may be small.

Future experiments could explore, how combinations perform, whose coefficients were trained on larger N-best lists or even on word lattices. The influence of different language models and more acoustic models i.e. quinphones could be investigated.

Bibliography

[Bahl & Jelinek+ 83] L. R. Bahl, F. Jelinek, R. L. Mercer. A maximum likelihood approach to continuous speech recognition. *PAT*, Vol. 5, pp. 179–190, March 1983.

[Baker 75] J. K. Baker. Stochastic modeling for automatic speech understanding. In D. R. Reddy, editor, *Speech Recognition*, pp. 512–542. Academic Press, New York, NY, USA, 1975.

[Bakis 76] R. Bakis. Continuous speech word recognition via centisecond acoustic states. In *Proc. of the 91st Meeting Acoustical Society of America 1976*, Washington, DC, USA, April 1976.

[Baum & Petrie 66] L. E. Baum, T. Petrie. Statistical inference for probabilistic functions of finite state markov chains. In *Ann. Math. Stat.*, Vol. 37, pp. 1554–1563, 1966.

[Bellman 57] R. E. Bellman. *Dynamic Programming*. Princeton University Press, Princeton, NJ, 1957.

[Beyerlein & Aubert+ 99] P. Beyerlein, X. L. Aubert, R. Haeb-Umbach, M. Harris, D. Klakow, A. Wendemuth, S. Molau, M. Pitz, A. Sixtus. The philips/rwth system for transcription of broadcast news. In *Sixth European Conference on Speech Communication and Technology*, Vol. 2, pp. 647–650, Budapest, Hungary, Sept. 1999.

[Beyerlein 98] P. Beyerlein. Discriminative model combination. In *Proceedings of ICASSP*, Vol. 1, pp. 481–484, Seattle, WA, 1998.

[Beyerlein 00] P. Beyerlein. *Diskriminative Modellkombination in Spracherkennungssystemen mit grossem Wortschatz*. Ph.D. thesis, Aachen, Germany, Oct 2000.

[Cooley & Tukey 65] J. W. Cooley, J. W. Tukey. An algorithm for machine calculation of complex fourier series. *Mathematics of Computation*, Vol. 19, pp. 297–301, April 1965.

[Davis & Biddulph+ 52] K. H. Davis, R. Biddulph, S. Balashek. Automatic speech recognition of spoken digits. *JASA*, Vol. 24, No. 6, pp. 637–642, 1952.

[Duda & Hart 73] R. O. Duda, P. E. Hart. *Pattern Classification and Scene Analysis*. John Wiley & Sons, New York, NY, USA, 1973.

[Evermann & Chan+ 04] G. Evermann, H. Chan, M. Gales, T. Hain, X. Liu, D. Mrva, L. Wang, P. Woodland. Defelopment of the 2003 cu-htk conversational telephone speech transcription system. In *Proceedings of ICASSP2004*, Vol. 1, pp. 249–252, 2004.

[Evermann & Woodland 00] G. Evermann, P. Woodland. Posterior probability decoding, confidence estimation and system combination. In *Proceedings. of the NIST Speech Transcription Workshop*, 2000.

[Fiscus 97] J. Fiscus. A post-processing system to yield reduced word error rates: Recogniser output voting error reduction. In *Proc. IEEE ASRU Workshop*, Santa Barbara, 1997.

[Fry 59] D. B. Fry. Theoretical apects of mechanical speech recognition. *Journal British Institution of Radio Engineering*, Vol. 19, No. 4, pp. 211–229, 1959.

[Fukunaga 90] K. Fukunaga. *Introduction to Statistical Pattern Recognition*. Academic Press, Boston, 2nd edition, 1990.

[Gollan & Bisani+ 05] C. Gollan, M. Bisani, S. Kanthak, R. Schlüter, H. Ney. Cross domain automatic transcription on the tc-star epps corpus. In *In Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, Philadephia, PA, March 2005.

[Jelinek 69] F. Jelinek. A fast sequential decoding algorithm using a stack. *IBM Journal of Research and Development*, Vol. 13, pp. 675–685, Nov. 1969.

[Jelinek 76] F. Jelinek. Continuous speech recognition by statistical methods. *Proceedings of the IEEE*, Vol. 64, No. 10, pp. 532–556, April 1976.

[Juang & Chou+ 95] B. H. Juang, W. Chou, C. H. Lee. Statistical and discriminative methods for speech recognition. In J. M. L. S. A. J. Rubio Ayuso, editor, *Speech Recognition and Coding - New Advances and Trends*, pp. 41–55. Springer Verlag, Berlin-Heidelberg, 1995.

[Kanthak & Ney 04] S. Kanthak, H. Ney. Fsa: An efficient and flexible c++ toolkit for finite state automata using on-demand computation. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL 2004)*, pp. 510–517, Barcelona, Spain, July 2004.

[Lee 89] K.-F. Lee. The development of the sphinx system. Kluwer Academic Publishers, 1989.

[Legetter & Woodland 95] C. Legetter, P. Woodland. Flexible speaker adaptation using maximum likelihood linear regression. In *ARPA Spoken Language Technology Workshop*, pp. 104–109, 1995.

[Levenshtein 65] V. I. Levenshtein. Binary codes capable of correcting deletions, insertions and reversals. *Doklady Akademii nauk SSSR (in Russian)*, Vol. 163, No. 4, pp. 845–848, 1965. Also in Cybernetics and Control Theory, 10(8), pp 707-710, 1966.

[Levinson & Rabiner+ 83] S. E. Levinson, L. R. Rabiner, M. M. Sondhi. An introduction to the application of the theory of probabilistic functions of a markov process to automatic speech recognition. *Bell System Technical Journal*, Vol. 62, No. 4, pp. 1035–1074, April 1983.

[Liu & Gales+ 03] X. Liu, M. Gales, P. Woodland. Automatic complexity control for hlda systems. In *Proceedings ICASSP*, 2003.

[Macherey 98] W. Macherey. Implementation and comparison of discriminative training methods for automatic speech recognition. Master's thesis, Lehrstuhl für Informatik VI RWTH Aachen, Aachen, Germany, Nov 1998.

[Mangu & Brill+ 99] L. Mangu, E. Brill, A. Stolcke. Finding consensus among word error minimizatioin. In *Proceedings of Eurospeech*, 1999.

[Mohri & Pereira+ 00] M. Mohri, F. Pereira, M. Riley. Weighted finite state transducers in speech recognition. In *In ISCA ITRW Automatic Speech Recognition: Challenges for the Millenium*, pp. 97–106, Paris, France, Sept. 2000.

[Molau 03] S. Molau. *Normalization in the Acoustic Feature Space for Improved Speech Recognition*. Ph.D. thesis, Aachen, Germany, Feb 2003.

[Ney & Essen+ 94] H. Ney, U. Essen, R. Kneser. On structuring probabilistic dependencies in language modeling. *CSL*, Vol. 2, No. 8, pp. 1–38, 1994.

[Ney & Haeb-Umbach+ 92] H. Ney, R. Haeb-Umbach, B.-H. Tran, M. Oerder. Improvements in beam search for 10000-word continuous speech recognition. In *ICASSP*, Vol. 1, pp. 9–12, San Francisco, CA, USA, March 1992.

[Ney & Mergel+ 87] H. Ney, D. Mergel, A. Noll, A. Paeseler. A data-driven organization of the dynamic programming beam search for continuous speech recognition. In *ICASSP*, pp. 833–836, Dallas, TX, USA, April 1987.

[Ney & Noll 88] H. Ney, A. Noll. Phoneme modeling using continuous mixture densities. In *ICASSP*, pp. 437–440, New York, NY, USA, April 1988.

[Ney 84] H. Ney. The use of a one-stage dynamic programming algorithm for connected word recognition. *IEEE Transactions on Acoustic, Speech and Signal Processing*, Vol. 32, No. 2, April 1984.

[Ney 90] H. Ney. Acoustic modeling of phoneme units for continuous speech recognition. In L. Torres, E. Masgrau, M. A. Lagunas, editors, *Signal Processing V: Theories and Applications, Fifth European Signal Processing Conference*, pp. 65–72. Elsevier Science Publishers B. V., Barcelona, Spain, 1990.

[Ney 02] H. Ney. Spracherkennung. Ausarbeitung zur Vorlesung an der RWTH Aachen. Ausgearbeitet von: Margit Generet und Sonja Nießen, 2002.

[Ortmanns & Ney 95] S. Ortmanns, H. Ney. An experimental study of the search space for 20000-word speech recognition. In *EUROSPEECH*, Vol. 2, pp. 901–904, Madrid, Spain, Sept. 1995.

[Paul 91] D. B. Paul. Algorithms for an optimal A^* search and linearizing the search in the stack decoder. In *ICASSP*, Vol. 1, pp. 693–696, Toronto, Canada, May 1991.

[Pitz & Ney 03] M. Pitz, H. Ney. Vocal tract normalization as linear transformation of mfcc. In *EUROSPEECH*, pp. 1445–1448, Geneva, Switzerland, Sept. 2003.

[Pitz 05] M. Pitz. *Investigations on Linear Transformations for Speaker Adaptation and Normalization.* Ph.D. thesis, Aachen, Germany, Mar 2005.

[Rabiner 89] 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, Feb. 1989.

[Schwartz & Austin 91] R. Schwartz, S. Austin. A comparison of several approximate algorithms for finding multiple (N-best) sentence hypotheses. In *ICASSP*, Vol. 1, pp. 701–704, Toronto, Canada, May 1991.

[Schwartz & Chow 90] R. Schwartz, Y.-L. Chow. The N-best algorithm: An efficient and exact procedure for finding the N most likely sentence hypotheses. In *ICASSP*, pp. 81–84, Albuquerque, NM, USA, April 1990.

[Soltau & Yu$^+$ 04] H. Soltau, H. Yu, F. Metze, C. Fügen, Q. Jin, S.-C. Jou. The 2003 isl rich transcription system for conversational telephony speech. In *Proceedings ICASSP 2004*, pp. 773–776. IEEE, 2004.

[Steinbiss & Tran$^+$ 94] V. Steinbiss, B.-H. Tran, H. Ney. Improvements in beam search. In *ICSLP*, Vol. 4, pp. 2143–2146, Yokohama, Japan, Sept. 1994.

[van Kampen 92] N. G. van Kampen. *Stochastic Processes in Physics and Chemistry.* Elsevier Science Publishers B. C., Amsterdam, The Netherlands, 1992.

[Vintsyuk 71] T. K. Vintsyuk. Elementwise recognition of continuous speech composed of words from a specified dictionary. *Kibernetika*, Vol. 7, pp. 133–143, March 1971.

[Viterbi 67] A. Viterbi. Error bounds for convolutional codes and an asymptotically optimal decoding algorithm. *IEEE Transactions on Information Theory*, Vol. 13, pp. 260–269, 1967.

[Wakita 77] H. Wakita. Normalization of vowels by vocal tract length and its application to vowel identification. In *ICASSP*, Vol. ASSP-25, No. 2, pp. 183–192, April 1977.

[Zahn & Waibel 97] P. Zahn, A. Waibel. Vocal tract length normalization for large vocabulary continous speech recognition. In *Technical Report CMU-LTI-97-150*, Carnegie Mellon University, Center for Machine Translation, May 1997.

[Zolnay & Schlüter+ 02] A. Zolnay, R. Schlüter, H. Ney. Robust speech recognition using a voiced-unvoiced feature. In *Proceedings International Conference on Spoken Language Processing*, Vol. 2, pp. 1065–1068, Denver, Colorado, Sept. 2002.

[Zolnay & Schlüter+ 03] A. Zolnay, R. Schlüter, H. Ney. Extraction methods of voicing features for robust speech recognition. In *Proceedings of Eurospeech*, Vol. 1, pp. 497–500, Geneva, Swizerland, Sept. 2003.

A. Notations

- (X_r, W_r) training data $r = 1, \ldots, R$
- w_1^N sequence of spoken words $w_1, \ldots, w_T$
- x_1^T sequence of feature vectors $x_1, \ldots, x_T$
- $\mathcal{L}(\cdot, \cdot)$ Levenshtein distance of two word sequences
- $Pr(w_1^N)$ language model (a-priori) probability
- $Pr(x_1^T | w_1^n)$ emission probabilities
- s state, l mixture index
- ϑ set of acoustic parameters
- $c_{s,l}$ mixture weight of density l in state s
- $\mu_{s,l}$ mean of density l in state s
- $\sigma_{s,l}^2$ variance vector of density l in state s
- $g(x_1^T, w_1^N, {w'}_1^{N'})$ discriminant function
- $l(X_i, W_i, \Lambda)$ loss function
- $d(X_i, W_i, \Lambda)$ misclassification function
- Λ set of coefficients used in the model combination
- H, C, P transducer used in rescoring
- ${}_{c_l}w_{i_{c_l}}$ word w_i with left and right phoneme context

B. Abbreviations

ASR	Automatic Speech Recognition
CART	Classification and Regression Tree
CNC	Confusion Network Combination
CND	Confusion Network Decoding
DMC	Discriminant Model Combination
EM	Expectation Maximization
FFT	Fast Fourier Transformation
GMM	Gaussian Mixture Model
HLDA	Heteroscedastic Linear Discriminant Analysis
HMM	Hidden Markov Model
LDA	Linear Discriminant Analysis
LM	Language Model
MFCC	Mel-frequency Cepstrum Coefficients
MAP	Maximum a-posteriori
MCE	Minimum Classification Error
ML	Maximum Likelihood
MLLR	Maximum Likelihood Linear Regression
MMI	Maximum Mutual Information
MWE	Minimum Word Error
ROVER	Recognizer Output Voting Error Reduction
RTF	Real-Time-Factor
RWTH	Rhein-Westfälische Technische Hochschule
SAT	Speaker Adaptive Training
TDP	Time Distortion Penalty
VTLN	Vocal Tract Length Normalization
WER	Word-error rate

C. Detailed Derivations

C.1. Derivation Of The MCE Iterative Function

This section describes the detailed derivation of an iterative formula used to optimize the system weights λ_m based on a modified MCE criterion, which was already done by Beyerlein in [Beyerlein 00]

Based in [Juang & Chou+ 95] the misclassification function is:

$$
\begin{aligned}
d(X_i, W_i, \Lambda) &= -\log p_\Lambda(W_i|X_i) + \log \left(\frac{1}{K} \sum_{k=1}^{K} \exp \left\{ \eta \cdot \log p_\Lambda(\hat{W}_{i,k}|X_i) \right\} \right)^{\frac{1}{\eta}} \\
&= -\log p_\Lambda(W_i|X_i) + \log \left(\frac{1}{K} \sum_{k=1}^{K} \left[P_\Lambda(\hat{W}_{i,k}|X_i) \right]^{\eta} \right)^{\frac{1}{\eta}} \\
&= \log \left(\frac{1}{K} \sum_{k=1}^{K} \left[\frac{p_\Lambda(\hat{W}_{i,k}|X_i)}{p_\Lambda(W_i|X_i)} \right]^{\eta} \right)^{\frac{1}{\eta}} .
\end{aligned}
\tag{C.1}
$$

In order to include the distance of concurrent word sequences to the correct word sequence the Levenshtein distance $\mathcal{L}(\hat{W}_{i,k}, W_i)$ is included as a weighting factor:

$$
d(X_i, W_i, \Lambda) = \log \left(\frac{1}{K} \sum_{k=1}^{K} \exp \left\{ \eta \cdot \mathcal{L}(\hat{W}_{i,k}, W_i) \log \frac{p_\Lambda(\hat{W}_{i,k}|X_i)}{p_\Lambda(W_i|X_i)} \right\} \right)^{\frac{1}{\eta}} . \tag{C.2}
$$

In this equation the influence on the calculation of the misclassification function of a hypotheses with a high error rate is increased, whereas good hypotheses with few errors slightly effect the misclassification measure. If the value η is high ($\eta \to \infty$), then:

$$
d(X_i, W_i, \Lambda) = \max_{\hat{W}_{i,k}} \mathcal{L}(\hat{W}_{i,k}, W_i) \log \frac{p_\Lambda(\hat{W}_{i,k}|X_i)}{p_\Lambda(W_i|X_i)},
$$

thus the word error rate influences the misclassification measure linearly.

For the loss function $l(...)$ used in 4.10 conversions lead to:

$$\begin{aligned} l(X_i, W_i, \Lambda) &= (1 + \exp\{-a(b + d(\dots))\})^{-1} \\ &= (1 + \exp\{-ab\}(\exp\{-d(\dots)\})^{a})^{-1} \end{aligned}$$

Inserting equation C.2 into the converted loss function results in:

$$l(X_i, W_i, \Lambda) = \left(1 + A\left(\frac{1}{K}\sum_{k=1}^{K}\left[\frac{p_\Lambda(\hat{W}_{i,k}|X_i)}{p_\Lambda(W_i|X_i)}\right]^{\eta\cdot\mathcal{L}(\hat{W}_{i,k},W_i)}\right)^{-\frac{B}{\eta}}\right)^{-1}, \quad \text{(C.3)}$$

where $A = \exp\{-ab\}, B = a$.

With this equation it is possible to calculate the derivative regarding parameter λ_m of function E_{MCE} (equation 4.9).

$$\frac{\partial E_{MCE}(\Lambda)}{\partial \lambda_m} = \frac{1}{I}\sum_{i=1}^{I}\frac{\partial l(X_i, W_i, \Lambda)}{\partial \lambda_m} \quad \text{(C.4)}$$

Application of the chain rule for derivatives:

$$\begin{aligned} \frac{\partial l(X_i, W_i, \Lambda)}{\partial \lambda_m} &= \frac{\partial}{\partial \lambda_m}\left(1 + A(\exp\{-d(X_i, W_i, \Lambda)\})^{B}\right)^{-1} \\ &= -\left(1 + A(\exp\{-d(X_i, W_i, \Lambda)\})^{B}\right)^{-2} \cdot \\ &\quad \cdot B \cdot A(\exp\{-d(X_i, W_i, \Lambda)\})^{B-1} \cdot \\ &\quad \cdot(-\exp\{-d(X_i, W_i, \Lambda)\}) \cdot \frac{\partial d(X_i, W_i, \Lambda}{\partial \lambda_m} \\ &= \left(1 + A(\exp\{-d(X_i, W_i, \Lambda)\})^{B}\right)^{-2} \cdot \\ &\quad \cdot A(\exp\{-d(X_i, W_i, \Lambda)\})^{B} \cdot B\frac{\partial d(X_i, W_i, \Lambda}{\partial \lambda_m} \end{aligned}$$

Since $(1+x)^{-2}x = (1+x)^{-1}(1-(1+x)^{-1})$:

$$\frac{\partial l(X_i, W_i, \Lambda)}{\partial \lambda_m} = B \cdot l(X_i, W_i, \Lambda)(1 - l(X_i, W_i, \Lambda)) \cdot \frac{\partial d(X_i, W_i, \Lambda)}{\partial \lambda_m} \quad \text{(C.5)}$$

For the derivative of d holds:

$$\frac{\partial d(X_i, W_i, \Lambda)}{\partial \lambda_m} = \frac{\partial}{\partial \lambda_m}\log\left(\frac{1}{K}\sum_{k=1}^{K}\left[\frac{p_\Lambda(\hat{W}_{i,k}|X_i)}{p_\Lambda(W_i|X_i)}\right]^{\eta\cdot\mathcal{L}(\hat{W}_{i,k},W_i)}\right)^{\frac{1}{\eta}}$$

Further transformations lead to:

$$\begin{aligned}
\frac{\partial d(X_i, W_i, \Lambda)}{\partial \lambda_m} &= \frac{1}{\eta} \frac{\frac{1}{K}\sum_{k=1}^{K} \frac{\partial}{\partial \lambda_m} \left[\frac{p_\Lambda(\hat{W}_{i,k}|X_i)}{p_\Lambda(W_i|X_i)}\right]^{\eta \cdot \mathcal{L}(\hat{W}_{i,k}, W_i)}}{\frac{1}{K}\sum_{k=1}^{K} \left[\frac{p_\Lambda(\hat{W}_{i,k}|X_i)}{p_\Lambda(W_i|X_i)}\right]^{\eta \cdot \mathcal{L}(\hat{W}_{i,k}, W_i)}} \\
&= \frac{1}{\eta} \frac{\sum_{k=1}^{K} \frac{\partial}{\partial \lambda_m} \left[\frac{\prod_{m=1}^{M} p_m(\hat{W}_{i,k}|X_i)^{\lambda_m}}{\prod_{m=1}^{M} p_m(W_i|X_i)^{\lambda_m}}\right]^{\eta \cdot \mathcal{L}(\hat{W}_{i,k}, W_i)}}{\sum_{k=1}^{K} \left[\frac{p_\Lambda(\hat{W}_{i,k}|X_i)}{p_\Lambda(W_i|X_i)}\right]^{\eta \cdot \mathcal{L}(\hat{W}_{i,k}, W_i)}} \\
&= \left(\eta \sum_{k=1}^{K} \left[\frac{p_\Lambda(\hat{W}_{i,k}|X_i)}{p_\Lambda(W_i|X_i)}\right]^{\eta \cdot \mathcal{L}(\hat{W}_{i,k}, W_i)}\right)^{-1} \cdot \\
&\quad \cdot \sum_{k=1}^{K} \frac{\partial}{\partial \lambda_m} \left(\frac{p_m(\hat{W}_{i,k}|X_i)}{p_m(W_i|X_i)}\right)^{\eta \cdot \mathcal{L}(\hat{W}_{i,k}|W_i) \cdot \lambda_m} \cdot \\
&\quad \cdot \left[\prod_{j=1, j\neq m}^{M} \left(\frac{p_j(\hat{W}_{i,k}|X_i)}{p_j(W_i|X_i)}\right)^{\lambda_j}\right]^{\eta \cdot \mathcal{L}(\hat{W}_{i,k}, W_i)}
\end{aligned}$$

Because $\frac{\partial}{\partial x} a^{bx} = (b \log a) a^{bx}$ does apply:

$$\begin{aligned}
\frac{\partial d(X_i, W_i, \Lambda)}{\partial \lambda_m} &= \left(\eta \sum_{k=1}^{K} \left[\frac{p_\Lambda(\hat{W}_{i,k}|X_i)}{p_\Lambda(W_i|X_i)}\right]^{\eta \cdot \mathcal{L}(\hat{W}_{i,k}, W_i)}\right)^{-1} \cdot \\
&\quad \cdot \sum_{k=1}^{K} \eta \cdot \mathcal{L}(\hat{W}_{i,k}, W_i) \log\left(\frac{p_m(\hat{W}_{i,k}|X_i)}{p_m(W_i|X_i)}\right) \left(\frac{p_m(\hat{W}_{i,k}|X_i)}{p_m(W_i|X_i)}\right)^{\eta \cdot \mathcal{L}(\hat{W}_{i,k}|W_i) \cdot \lambda_m} \cdot \\
&\quad \cdot \left[\prod_{j=1, j\neq m}^{M} \left(\frac{p_j(\hat{W}_{i,k}|X_i)}{p_j(W_i|X_i)}\right)^{\lambda_j}\right]^{\eta \cdot \mathcal{L}(\hat{W}_{i,k}, W_i)} \\
&= \frac{\sum_{k=1}^{K} \mathcal{L}(\hat{W}_{i,k}, W_i) \log\left(\frac{p_m(\hat{W}_{i,k}|X_i)}{p_m(W_i|X_i)}\right) \left[\prod_{j=1}^{M} \left(\frac{p_j(\hat{W}_{i,k}|X_i)}{p_j(W_i|X_i)}\right)^{\lambda_j}\right]^{\eta \cdot \mathcal{L}(\hat{W}_{i,k}, W_i)}}{\sum_{k=1}^{K} \left[\frac{p_\Lambda(\hat{W}_{i,k}|X_i)}{p_\Lambda(W_i|X_i)}\right]^{\eta \cdot \mathcal{L}(\hat{W}_{i,k}, W_i)}} \\
&= \frac{\sum_{k=1}^{K} \mathcal{L}(\hat{W}_{i,k}, W_i) \log\left(\frac{p_m(\hat{W}_{i,k}|X_i)}{p_m(W_i|X_i)}\right) \left[\frac{p_\Lambda(\hat{W}_{i,k}|X_i)}{p_\Lambda(W_i|X_i)}\right]^{\eta \cdot \mathcal{L}(\hat{W}_{i,k}, W_i)}}{\sum_{k=1}^{K} \left[\frac{p_\Lambda(\hat{W}_{i,k}|X_i)}{p_\Lambda(W_i|X_i)}\right]^{\eta \cdot \mathcal{L}(\hat{W}_{i,k}, W_i)}} \qquad \text{(C.6)}
\end{aligned}$$

Combining equations C.4, C.5 and C.6 leads to:

$$\begin{aligned}\frac{\partial E_{MCE}(\Lambda)}{\partial \lambda_m} &= \frac{B}{I}\sum_{i=1}^{I} l(X_i, W_i, \Lambda)(1 - l(X_i, W_i, \Lambda)) \cdot \\ &\cdot \frac{\sum_{k=1}^{K} \mathcal{L}(\hat{W}_{i,k}, W_i) \log\left(\frac{p_m(\hat{W}_{i,k}|X_i)}{p_m(W_i|X_i)}\right) \left[\frac{p_\Lambda(\hat{W}_{i,k}|X_i)}{p_\Lambda(W_i|X_i)}\right]^{\eta\cdot\mathcal{L}(\hat{W}_{i,k},W_i)}}{\sum_{k=1}^{K}\left[\frac{p_\Lambda(\hat{W}_{i,k}|X_i)}{p_\Lambda(W_i|X_i)}\right]^{\eta\cdot\mathcal{L}(\hat{W}_{i,k},W_i)}}\end{aligned} \tag{C.7}$$

By setting $U = I$ (identity matrix) in equation 4.12 forms together with the results of equation C.7 the iterative Formula for the coefficients of the log-linear combination of the distributions (equation 4.14):

$$\begin{aligned}\lambda_m^{(0)} &= 0 \\ \lambda_m^{(J+1)} &= \lambda_m^{(J)} - \varepsilon \sum_{i=1}^{I} l'(X_i, W_i, \Lambda^{(J)}(1 - l'(X_i, W_i, \Lambda^{(J)})) \cdot \\ &\quad \frac{\sum_{k=1}^{K} \mathcal{L}(\hat{W}_{i,k}, W_i) \log\left(\frac{p_m(\hat{W}_{i,k}|X_i)}{p_m(W_i|X_i)}\right) \left[\frac{p_{\Lambda^{(J)}}(\hat{W}_{i,k}|X_i)}{p_{\Lambda^{(J)}}(W_i|X_i)}\right]^{\eta\mathcal{L}(\hat{W}_{i,k},W_i)}}{\sum_{k=1}^{K}\left[\frac{p_{\Lambda^{(J)}}(\hat{W}_{i,k}|X_i)}{p_{\Lambda^{(J)}}(W_i|X_i)}\right]^{\eta\mathcal{L}(\hat{W}_{i,k},W_i)}} \\ \Lambda^{(J)} &= (\Lambda_1^{(J)}, \ldots, \Lambda_M^{(J)})^T\end{aligned} \tag{C.8}$$

C.2. Minimizing The Smoothed Word Error Rate

In this section the optimization criterion $E(\Lambda)$ is approximated by a smoothed target function.

$$E_{MWE}(\Lambda) = \frac{1}{\sum_{i=1}^{I} N_i} \sum_{i=1}^{I}\sum_{k=1}^{K} \mathcal{L}(\hat{W}_{i,k}|W_i) S(\hat{W}_{i,k}, i, \Lambda) \tag{C.9}$$

The indicator function S is replaced by a smoothed indicator function to receive differentiable criterion:

$$S(\hat{W}_{i,k}, i, \Lambda) = \frac{\left(\frac{p_\Lambda(\hat{W}_{i,k}|X_i)}{p_\Lambda(W_i|X_i)}\right)^\eta}{\sum_{W'}\left(\frac{p_\Lambda(W'|X_i)}{p_\Lambda(W_i|X_i)}\right)^\eta} \tag{C.10}$$

The function $E_{MWE}(\Lambda)$ derived with respect to λ_m:

$$\frac{\partial E_{MWE}(\Lambda)}{\partial \lambda_m} = \frac{1}{\sum_{i=1}^{I} N_i} \sum_{i=1}^{I} \sum_{k=1}^{K} \mathcal{L}(\hat{W}_{i,k}|W_i) \frac{\partial}{\partial \lambda_m} S(\hat{W}_{i,k}, i, \Lambda) \tag{C.11}$$

So, the derivative of S regarding λ_m is needed:

$$\begin{aligned}
\frac{\partial S(\hat{W}_{i,k}, i, \Lambda)}{\partial \lambda_i} &= \frac{\eta \log \frac{p_m(\hat{W}_{i,k}|X_i)}{p_m(W_i|X_i)} \exp\left\{\eta \log \frac{p_\Lambda(\hat{W}_{i,k}|X_i)}{p_\Lambda(W_i|X_i)}\right\} \sum_{W'} \exp\left\{\eta \log \frac{p_\Lambda(W'|X_i)}{p_\Lambda(W_i|X_i)}\right\}}{\left(\sum_{W'} \exp\left\{\eta \log \frac{p_\Lambda(W'|X_i)}{p_\Lambda(W_i|X_i)}\right\}\right)^2} - \\
&\quad \frac{\exp\left\{\eta \log \frac{p_\Lambda(\hat{W}_{i,k}|X_i)}{p_\Lambda(W_i|X_i)}\right\} \sum_{W'} \eta \frac{p_m(\hat{W}_{i,k}|X_i)}{p_m(W_i|X_i)} \exp\left\{\eta \log \frac{p_\Lambda(W'|X_i)}{p_\Lambda(W_i|X_i)}\right\}}{\left(\sum_{W'} \exp\left\{\eta \log \frac{p_\Lambda(W'|X_i)}{p_\Lambda(W_i|X_i)}\right\}\right)^2} \\
&= S(\hat{W}_{i,k}, i, \Lambda) \frac{\eta \frac{p_m(\hat{W}_{i,k}|X_i)}{p_m(W_i|X_i)} \sum_{W'} \exp\left\{\eta \log \frac{p_\Lambda(W'|X_i)}{p_\Lambda(W_i|X_i)}\right\}}{\sum_{W'} \exp\left\{\eta \log \frac{p_\Lambda(W'|X_i)}{p_\Lambda(W_i|X_i)}\right\}} - \\
&\quad S(\hat{W}_{i,k}, i, \Lambda) \frac{\sum_{W'} \eta \frac{p_m(W'|X_i)}{p_m(W_i|X_i)} \exp\left\{\eta \log \frac{p_\Lambda(W'|X_i)}{p_\Lambda(W_i|X_i)}\right\}}{\sum_{W'} \exp\left\{\eta \log \frac{p_\Lambda(W'|X_i)}{p_\Lambda(W_i|X_i)}\right\}} \\
&= S(\hat{W}_{i,k}, i, \Lambda) \eta \frac{p_m(\hat{W}_{i,k}|X_i)}{p_m(W_i|X_i)} - S(\hat{W}_{i,k}, i, \Lambda) \sum_{W'} \eta \frac{p_m(W'|X_i)}{p_m(W_i|X_i)} S(W', i, \Lambda) \\
&= \eta S(\hat{W}_{i,k}, i\Lambda) \left(\frac{p_m(\hat{W}_{i,k}|X_i)}{p_m(W_i|X_i)} - \sum_{W'} \frac{p_m(W'|X_i)}{p_m(W_i|X_i)} S(W', i, \Lambda) \right) \qquad \text{(C.12)}
\end{aligned}$$

Combining equations C.11 and C.12 results in:

$$\begin{aligned}
\frac{\partial E_{MWE}(\Lambda)}{\partial \lambda_m} &= \frac{1}{\sum_{i=1}^{I} N_i} \sum_{i=1}^{I} \sum_{k=1}^{K} \mathcal{L}(\hat{W}_{i,k}, W_i) \eta S(\hat{W}_{i,k}, i, \Lambda) \cdot \\
&\quad \left(\frac{p_m(\hat{W}_{i,k}|X_i)}{p_m(W_i|X_i)} - \sum_{W'} \frac{p_m(W'|X_i)}{p_m(W_i|X_i)} S(W', i, \Lambda) \right) \\
&= \frac{1}{\sum_{i=1}^{I} N_i} \left\{ \sum_{i=1}^{I} \sum_{k=1}^{K} \mathcal{L}(\hat{W}_{i,k}, W_i) \eta S(\hat{W}_{i,k}, i, \Lambda) \frac{p_m(\hat{W}_{i,k}|X_i)}{p_m(W_i|X_i)} - \right. \\
&\quad \left. \sum_{i=1}^{I} \sum_{k=1}^{K} \mathcal{L}(\hat{W}_{i,k}, W_i) \eta S(\hat{W}_{i,k}, i, \Lambda) \sum_{W'} \frac{p_m(W'|X_i)}{p_m(W_i|X_i)} S(W', i, \Lambda) \right\}
\end{aligned} \tag{C.13}$$

$$
\begin{aligned}
&= \frac{1}{\sum_{i=1}^{I} N_i} \left\{ \sum_{i=1}^{I} \sum_{k=1}^{K} \mathcal{L}(\hat{W}_{i,k}, W_i) \eta S(\hat{W}_{i,k}, i, \Lambda) \frac{p_m(\hat{W}_{i,k}|X_i)}{p_m(W_i|X_i)} - \right. \\
&\quad \left. \sum_{i=1}^{I} \sum_{W'} \sum_{k=1}^{K} \mathcal{L}(\hat{W}_{i,k}, W_i) \eta S(\hat{W}_{i,k}, i, \Lambda) \frac{p_m(W'|X_i)}{p_m(W_i|X_i)} S(W', i, \Lambda) \right\} \\
&= \frac{\eta}{\sum_{i=1}^{I} N_i} \left\{ \sum_{i=1}^{I} \sum_{k=1}^{K} S(\hat{W}_{i,k}, i, \Lambda) \frac{p_m(\hat{W}_{i,k}|X_i)}{p_m(W_i|X_i)} \mathcal{L}(\hat{W}_{i,k}, W_i) - \right. \\
&\quad \left. \sum_{i=1}^{I} \sum_{W'} S(W', i, \Lambda) \frac{p_m(W'|X_i)}{p_m(W_i|X_i)} \sum_{k=1}^{K} \mathcal{L}(\hat{W}_{i,k}, W_i) S(\hat{W}_{i,k}, i, \Lambda) \right\}
\end{aligned}
\tag{C.14}
$$

By exchanging indices $\hat{W}_{i,k}$ and W', where $\{W'\} = \{\hat{W}_{i,1}, \ldots, \hat{W}_{i,K}\}$, and subsequently $\sum_{W'} = \sum_{k=1}^{K}$:

$$
\begin{aligned}
\frac{\partial E_{MWE}(\Lambda)}{\partial \lambda_m} &= \frac{\eta}{\sum_{i=1}^{I} N_i} \left\{ \sum_{i=1}^{I} \sum_{k=1}^{K} S(\hat{W}_{i,k}, i, \Lambda) \frac{p_m(\hat{W}_{i,k}|X_i)}{p_m(W_i|X_i)} \mathcal{L}(\hat{W}_{i,k}, W_i) - \right. \\
&\quad \left. \sum_{i=1}^{I} \sum_{k=1}^{K} S(\hat{W}_{i,k}, i, \Lambda) \frac{p_m(\hat{W}_{i,k}|X_i)}{p_m(W_i|X_i)} \sum_{W'} \mathcal{L}(W', W_i) S(W', i, \Lambda) \right\} \\
&= \frac{\eta}{\sum_{i=1}^{I} N_i} \sum_{i=1}^{I} \sum_{k=1}^{K} S(\hat{W}_{i,k}, i, \Lambda) \frac{p_m(\hat{W}_{i,k}|X_i)}{p_m(W_i|X_i)} \cdot \\
&\quad \cdot \left(\mathcal{L}(\hat{W}_{i,k}, W_i) - \sum_{W'} S(W', i, \Lambda) \mathcal{L}(W', W_i) \right)
\end{aligned}
\tag{C.15}
$$

The final equation is:

$$
\frac{\partial E_{MWE}(\Lambda)}{\partial \lambda_m} = \frac{\eta}{\sum_{i=1}^{I} N_i} \sum_{i=1}^{I} \sum_{k=1}^{K} S(\hat{W}_{i,k}, i, \Lambda) \frac{p_m(\hat{W}_{i,k}|X_i)}{p_m(W_i|X_i)} \left(\mathcal{L}(\hat{W}_{i,k}, W_i) - \overline{\mathcal{L}(W_i)} \right)
$$

where:

$$
\overline{\mathcal{L}(W_i)} = \sum_{W'} S(W', i, \lambda) \mathcal{L}(W', W_i) \tag{C.16}
$$

is the smoothed error rate of observation X_i.

D. Additional Results

D.1. EPPS English

All results of lattice-based system combination on EPPS English

	smoothed error in training	devel-05	eval-05
baseline, vtln	20.12	14.6	14.8
baseline, complete	20.10	14.4	14.6
baseline, nolda	20.25	15.1	15.4
baseline, mllr	19.59	13.9	14.0
vtln, complete	20.23	14.5	14.7
vtln, nolda	19.97	14.4	14.9
vtln, mllr	19.71	13.9	14.1
complete, nolda	19.94	14.1	14.6
complete, mllr	19.85	14.1	14.3
nolda, mllr	19.40	13.7	14.0
baseline, vtln, complete	19.98	14.3	14.5
baseline, vtln, nolda	19.84	14.4	14.6
baseline, vtln, mllr	19.58	13.9	14.0
baseline, complete, nolda	19.85	14.2	14.5
baseline, complete, mllr	19.64	13.9	14.1
baseline, nolda, mllr	19.42	13.7	14.0
vtln, complete, nolda	19.81	14.1	14.5
vtln, complete, mllr	19.77	14.0	14.3
vtln, nolda, mllr	19.44	13.7	14.1
complete, nolda, mllr	19.46	13.7	14.1
baseline, vtln, complete, nolda	19.74	14.1	14.4
baseline, vtln, complete, mllr	19.63	13.9	14.1
baseline, vtln, nolda, mllr	19.47	13.8	14.0
baseline, complete, nolda, mllr	19.49	13.7	14.1
vtln, complete, nolda, mllr	19.47	13.7	14.1
baseline, vtln, complete, nolda, mllr	19.51	13.8	14.1

Table D.1.: Results of lattice-based system combination on EPPS English.Performance given in NIST (%)

All results of lattice-based model combination on EPPS English

	smoothed error in training	devel-05	eval-05
baseline, vtln	20.98	14.6	15.0
baseline, complete	20.97	14.3	14.5
baseline, nolda	20.79	14.8	15.2
baseline, mllr	20.60	13.6	13.9
vtln, complete	21.07	14.3	14.7
vtln, nolda	20.62	14.1	14.7
vtln, mllr	20.71	13.9	14.2
complete, nolda	20.64	13.9	14.6
complete, mllr	20.81	13.9	14.4
nolda, mllr	20.34	13.3	14.0
baseline, vtln, complete	20.96	14.4	14.7
baseline, vtln, nolda	20.60	14.1	14.6
baseline, vtln, mllr	20.63	13.7	13.9
baseline, complete, nolda	20.67	14.1	14.5
baseline, complete, mllr	20.69	13.7	13.9
baseline, nolda, mllr	20.34	13.3	13.9
vtln, complete, nolda	20.65	14.0	14.4
vtln, complete, mllr	20.82	13.9	14.3
vtln, nolda, mllr	20.35	13.3	13.9
complete, nolda, mllr	20.43	13.4	14.0
baseline, vtln, complete, nolda	20.70	14.1	14.6
baseline, vtln, complete, mllr	20.75	13.9	14.0
baseline, vtln, nolda, mllr	20.41	13.4	13.9
baseline, complete, nolda, mllr	20.50	13.5	14.1
vtln, complete, nolda, mllr	20.49	13.5	14.1
baseline, vtln, complete, nolda, mllr	20.79	14.1	14.5

Table D.2.: Results of lattice-based model combination on EPPS English. Performance given in NIST (%)

D.2. EPPS Spanish

All results of lattice-based system combination on EPPS Spanish

	smoothed error in training	devel-05	eval-05
baseline, vtln	14.76	12.4	13.5
baseline, complete	14.74	12.2	13.4
baseline, nolda	14.88	12.7	13.9
baseline, mllr	14.23	11.6	13.0
vtln, complete	14.72	12.2	13.3
vtln, nolda	14.78	12.3	13.4
vtln, mllr	14.26	11.7	13.0
complete, nolda	14.74	12.2	13.3
complete, mllr	14.34	11.7	13.1
nolda, mllr	14.23	11.6	12.9
baseline, vtln, complete	14.64	12.1	13.3
baseline, vtln, nolda	14.64	12.2	13.4
baseline, vtln, mllr	14.24	11.7	12.9
baseline, complete, nolda	14.63	12.1	13.3
baseline, complete, mllr	14.29	11.7	13.0
baseline, nolda, mllr	14.21	11.6	12.9
vtln, complete, nolda	14.64	12.1	13.2
vtln, complete, mllr	14.33	11.8	13.0
vtln, nolda, mllr	14.26	11.7	12.9
complete, nolda, mllr	14.31	11.8	13.0
baseline, vtln, complete, nolda	14.56	12.1	13.3
baseline, vtln, complete, mllr	14.29	11.7	13.0
baseline, vtln, nolda, mllr	14.23	11.7	12.9
baseline, complete, nolda, mllr	14.26	11.7	13.0
vtln, complete, nolda, mllr	14.32	11.7	12.9
baseline, vtln, complete, nolda, mllr	14.29	11.8	12.9

Table D.3.: Results of lattice-based system combination on EPPS Spanish. Performance given in NIST (%)

All results of lattice-based model combination on EPPS Spanish

	smoothed error in training	devel-05	eval-05
baseline, vtln	15.30	12.5	13.3
baseline, complete	15.35	15.1	13.1
baseline, nolda	15.09	12.8	13.5
baseline, mllr	15.21	11.7	12.8
vtln, complete	15.44	12.2	12.9
vtln, nolda	15.01	12.3	13.1
vtln, mllr	15.30	11.7	12.6
complete, nolda	15.10	12.4	12.9
complete, mllr	15.46	11.9	12.7
nolda, mllr	**14.96**	**11.5**	**12.4**
baseline, vtln, complete	15.30	12.3	13.1
baseline, vtln, nolda	14.96	12.2	13.0
baseline, vtln, mllr	15.21	11.9	12.8
baseline, complete, nolda	15.02	12.2	12.8
baseline, complete, mllr	15.25	11.9	12.7
baseline, nolda, mllr	14.95	11.8	12.7
vtln, complete, nolda	15.00	12.2	12.9
vtln, complete, mllr	15.33	11.8	12.6
vtln, nolda, mllr	14.94	11.8	12.6
complete, nolda, mllr	15.01	11.8	12.5
baseline, vtln, complete, nolda	15.03	12.3	13.0
baseline, vtln, complete, mllr	15.26	12.1	12.8
baseline, vtln, nolda, mllr	14.94	12.0	12.8
baseline, complete, nolda, mllr	14.98	12.0	12.8
vtln, complete, nolda, mllr	14.98	12.0	12.7
baseline, vtln, complete, nolda, mllr	15.25	13.0	13.6

Table D.4.: Results of lattice-based model combination on EPPS Spanish. Performance given in NIST (%)