Error Correction Based on Error Signatures applied to automatic speech recognition

Zur Erlangung des akademischen Grades eines
Doktors der Ingenieurwissenschaften
der Fakultät für Informatik
des Karlsruher Instituts für Technologie (KIT)

genehmigte

Dissertation

von

Dominic Telaar
aus Bocholt

Tag der mündlichen Prüfung:	21.12.2015
Erste Gutachterin:	Prof. Dr. T. Schultz
Zweiter Gutachter:	Prof. Dr. W. F. Tichy

Summary

Automatic speech recognition (ASR) systems are usually based on statistical machine learning methods and require a sufficient amount of training data to generalize to unseen test data. However, since the manual transcription of audio data is very costly and time consuming (e.g. the transcription of the recordings of the interviews used in this work required twelve times the length of the actual recordings), many speech recognition systems are trained on data whose transcriptions have been automatically created by an existing speech recognition system.

Errors that were made in this automatic transcription by the speech recognition system are included in the training data of the new speech recognition system. Thus, the newly trained speech recognition system learns not only to produce the same mistakes as the speech recognition system, which was used for automatic transcription, but will now generate these errors with a higher probability. To mitigate this problem, it is important to identify these systematic errors in the automatically transcribed data and fix them. Since current systems are trained on several thousand hours of speech data and consist of several components, the localization of frequent or systematic errors and their assignment to individual components poses a problem.

The aim of this thesis is to automatically detect and fix errors in speech recognition systems, to mitigate the problem of systematic errors described above. For this purpose, we developed a framework in which individual errors are assigned attributes and clustered. Each cluster is characterized by a set of attributes. A set of attributes is referred to as an *error signature*. The accumulation of errors and the computation of *error signatures* is the key contribution of this thesis and is based on algorithms for concept discovery in

the field of data-mining by means of which structures can be identified in data sets. These signatures are analyzed by an expert to derive improvements for the speech recognition system. This method can be divided into the following steps:

1. Error detection: in general, audio recordings that need to be transcribed have no existing reference transcriptions. Therefore, it is necessary to automatically assign error probabilities to each word in the hypothesis, based on the ASR system's output.

2. Error signature computation: In order to derive systematic errors of a system, potential errors have to be aggregated and clustered. In this thesis a method is presented which extracts a multitude of attributes from each word of all available transcriptions and introduce a clustering method to derive systematic errors. In contrast to existing methods in speech recognition, this method is able to identify commonalities between errors and to estimate the frequency of their occurrence.

3. Error analysis and error correction: The discovered *error signatures* can be examined by an expert to determine fixes for the ASR system's components. In order to reduce the labor required of an expert, it is possible to not only fix problems, but to also select signatures and exclude any data which matches these signatures from ASR training data.

We demonstrate the usefulness of the proposed framework on two speech corpora. The SEAME code-switching corpus, which was recorded in Singapore and consists of mixed English and Mandarin utterances. The ILSE corpus contains interview style recordings of about a thousand German speaking participants. We were able to significantly increase the performance of ASR systems on both corpora and only relied on automatically transcribed data. An additional contribution of this thesis is the development of a toolkit for automatic speech recognition in which the *error signature* framework has been integrated into.

Zusammenfassung

Automatische Spracherkennungssysteme (ASR Systeme) basieren in der Regel auf statistischen Lernverfahren und benötigen eine genügend große Menge an Trainingsdaten, um auf ungesehenen Testdaten zu generalisieren. Da jedoch die manuelle Transkription von Audiodaten sehr kosten- und zeitintensiv ist (z.B. die Verschriftung der Aufnahmen der in dieser Arbeit benutzten Interviews benötigte die zwölffache Dauer der eigentlichen Aufnahmen), werden viele Spracherkennungssysteme auf Daten trainiert deren Transkriptionen mit bereits bestehenden Spracherkennungssystemen automatisch erstellt wurden.

Fehler, die bei dieser automatischen Transkription durch das Spracherkennungssystem gemacht wurden, fließen in die Trainingsdaten des neuen Spracherkennungssystems ein. Das neu trainierte Spracherkennungssystem lernt damit nicht nur die gleichen Fehler zu produzieren, wie das Spracherkennungssystem, das zur automatischen Transkription genutzt wurde, sondern wird durch die zusätzlichen fehlerhaften Trainingsdaten diese jetzt auch mit höherer Wahrscheinlichkeit generieren. Um diesem Problem entgegenzuwirken ist es wichtig diese systematischen Fehler des Spracherkennungssystems zu identifizieren und zu beheben. Da aktuelle Systeme auf mehreren tausend Stunden an Sprachdaten trainiert werden und diese Systeme aus mehreren Teilkomponenten bestehen, stellt die Lokalisierung von häufig auftretenden bzw. systematischen Fehlern und deren Zuordnung zu einzelnen Komponenten ein Problem dar.

Ziel dieser Dissertation ist es, automatisch Fehlerquellen in Spracherkennungssystemen aufzudecken, diese zu beheben, und damit dem oben beschriebenen Problem der Verstärkung von systematischen Fehlern entgegen-

zutreten. Einzelnen Fehlern werden in dem vorgestellten Verfahren Attribute zugewiesen und anhand dieser zu Gruppen zusammengefasst. Jede Gruppe wird von einer Menge von Attributen charakterisiert. Eine Menge von Attributen wird dabei als *Fehlersignatur* (*Error Signature*) bezeichnet. Die Akkumulation von Fehlern und die Berechnung der *Fehlersignaturen* ist ein zentraler Beitrag dieser Arbeit und basiert auf Algorithmen zur Konzeptentdeckung aus dem Data-Mining mit deren Hilfe Strukturen in Datenmengen identifiziert werden können. Diese Signaturen werden durch einen Experten analysiert, um Verbesserungen für das Spracherkennungssystem abzuleiten. Das Verfahren lässt sich in folgende Schritte unterteilen:

1. Fehleridentifikation: Im Allgemeinen steht für die zu transkribierenden Audioaufnahmen keine Referenztranskription zur Verfügung. Daher ist es erforderlich anhand der generierten Hypothese jedem Wort automatisch eine Fehlerwahrscheinlichkeit zuzuordnen.

2. Fehlersignaturen berechnen: Es ist notwendig, einzelne potentielle Fehler zusammenzufassen, um daraufhin den zugrundeliegenden systematischen Fehler zu erschließen. In dem vorgestellten Verfahren werden zu diesem Zweck eine Vielzahl von Attributen eines jeden Wortes aller verfügbaren Transkriptionen extrahiert und eine Menge von Attributen berechnet, welche bestimmte systematische Fehler identifizieren. Im Gegensatz zu existierenden Methoden in der Spracherkennung ist es mit dem in dieser Arbeit vorgestellten Ansatz möglich Beziehungen zwischen Fehlern zu erkennen und die Häufigkeit ihres Auftretens abzuschätzen.

3. Analyse der Fehler und Fehlerbehebung: Um den Arbeitsaufwand für einen Experten bei der Sichtung der gefundenen *Fehlersignaturen* gering zu halten, ist es möglich, nicht nur direkt Probleme in den ASR Modellen zu beheben, sondern auch eine Menge an *Fehlersignaturen* auszuwählen mit denen die zusätzlichen Trainingsdaten vor dem ASR Training gefiltert werden können.

Die Nützlichkeit des Verfahrens wird an zwei Sprachkorpora demonstriert. Der SEAME Code-Switching Korpus wurde in Singapur aufgenommen und besteht aus Aufnahmen in denen Englisch und Mandarin gesprochen werden. Der ILSE Korpus enthält Aufnahmen von circa tausend deutschsprachigen Probanden, welche in Interviews erstellt wurden. Die Leistung der Spracherkennungssysteme auf beiden Korpora konnte, durch den Einsatz des vorgestellten Verfahrens, signifikant verbessert werden. Ein zusätzlicher Beitrag dieser Arbeit ist die Entwicklung eines Toolkits zur Spracherkennung, in welches das vorgestellte Verfahren integriert wurde.

Contents

List of Figures

List of Tables

Chapter 1

Introduction and Motivation

This chapter serves as motivation for this dissertation, giving a brief overview over the recent development of the field and presenting the major contributions of this thesis. In addition, the structure of this thesis is described.

1.1 Motivation

With the ever increasing prevalence of mobile devices, starting with the introduction of smart-phones in 2007, moving to increasingly smaller devices such as smart-watches, the conventional input method of typing on a keyboard is getting too cumbersome. The input method of choice is increasingly by voice. Advances in the capability of speech processing lead to the development of voice activated assistants such as Apple Siri, Google Now, and Microsoft Cortana. Any first step in such a device is the automatic speech recognition component which transforms the recorded audio into text for further processing. While a lot of research and progress has been made regarding the modeling of both speech and acoustic phenomena, the machine learning component tying all individual parts together is still a Hidden Markov Model (HMM) based sequence recognition system.

Therefore, while the models themselves vary and are tailored to the language or domain they are used in, the structure of the recognition process remains the same. Additionally, other applications such as motion recognition share

the same framework. Since the HMM framework ties various complex statistical models together, the interpretation of errors in the system and their linkage to their causes is not straight-forward and sometimes even impossible. To still improve HMM-based sequence recognition systems, researchers [Cha97, NRK99] in the field of speech recognition have proposed approaches to compare the reference to the generated result and deduce error categories from each recognition example.

While the available tools are sufficient to get an overview over which error categories are prevalent or which words/motions are commonly misrecognized, they fall short in giving a detailed summary of frequent errors and specific underlying causes. In order to alleviate this problem we propose a new approach to analyzing HMM-based sequence recognition systems and embed it into an iterative error correction process to finally improve speech recognition systems.

1.2 Contributions

The objective of this dissertation is to simplify the process of creating and improving an HMM-based sequence recognition system. The major contributions of this thesis are as follows:

1. We implemented a new easy to use HMM-based recognition toolkit, called BioKIT. It was developed during the duration of this thesis and was applied to various recognition tasks with state-of-the-art performance, such as automatic speech recognition, motion recognition and silent speech recognition. The modular C++ design and light-weight interfaces for the main components makes it easy to extend. The whole toolkit is exposed to the Python scripting language, such that BioKIT can easily be integrated with other Python libraries.

2. Integrated into BioKIT is a new approach to analyzing recognition systems, we call error signatures. The multitude of errors encountered on a specific data set are difficult to blame to error causes. In this thesis we create a novel framework, where various attributes are assigned to errors and sets of attributes are found to characterize a group of errors. These sets of attributes are referred to as error signatures in this thesis.

3. Since creating a reference for any task is a tedious and time-consuming exercise and data without reference transcriptions is often available in abundance, the analysis of ASR systems without reference transcrip-

tions is of great interest. To leverage off the data with no reference transcriptions for the analysis of an ASR system, we extend the error signatures introduced in this thesis to compute signatures and identify errors in an unsupervised fashion.

1.3 Structure of this Thesis

This thesis is separated into the following chapters:

Chapter 2 gives an introduction into the field of automatic speech recognition and HMM-based sequence recognition in general. It also introduces the core search algorithm used for finding the best sequence in a given HMM. Furthermore, the related work to this thesis is discussed.

Chapter 3 presents the data corpora that were used in this thesis and describes the challenges of the two corpora.

Chapter 4 presents our work on a novel toolkit for HMM-based sequence recognition, which combines various aspects of other toolkits into a new and flexible architecture.

Chapter 5 contains the error correction framework, which is at the core of this thesis. It introduces the notion of error signatures. Furthermore, it describes how the signatures are obtained and how they can be used for error correction.

Chapter 6 contains the experiments we conducted with error signatures. After an initial experiment, where reference transcriptions are available to compute error signatures, we show how the algorithm can be extended to not rely on the availability of reference transcriptions. Finally, we perform experiments on both, the SEAME and the ILSE corpus to demonstrate the usefulness of our approach without reference transcriptions.

Chapter 7 concludes the thesis with a summary of the results and discusses potential future research directions.

Chapter 2

Background

This chapter gives an introduction into the field of automatic speech recognition, as well as to all other fundamentals necessary for this thesis. Additionally, related work relevant to this thesis is discussed in the last sections of this chapter.

2.1 Automatic speech recognition

This section gives an overview of the components of an automatic speech recognition system. The goal is to find the most likely word sequence W^* given an audio recording. Mathematically the problem can be expressed as in equation 2.1:

$$\begin{aligned} W^* &= \arg\max_W P(W|O) \\ &= \arg\max_W \frac{P(O|W)P(W)}{P(O)} \\ &= \arg\max_W P(O|W)P(W) \end{aligned} \tag{2.1}$$

where $W = w_1, \ldots, w_n$ is a word sequence and $O = o_1, \ldots, o_T$ is the segmented and preprocessed audio recording, and is called the observation sequence. Transforming $P(W|O)$ using Bayes' rule and omitting $P(O)$ we get the final equation of $P(O|W)P(W)$.

A schematic overview of an ASR system is given in Figure 2.1. Input to the ASR system is an audio recording which is put into the signal-preprocessing component. The preprocessing segments the audio file into smaller chunks and extracts a vector of features from each. The resulting observation sequence O serves as input to the decoder. Output of the decoder process is the word transcription of the observation sequence. The decoder combines all components of the ASR system and conducts a Viterbi search to find the most likely word sequence using the Viterbi algorithm (refer to Section 2.1.1.1).

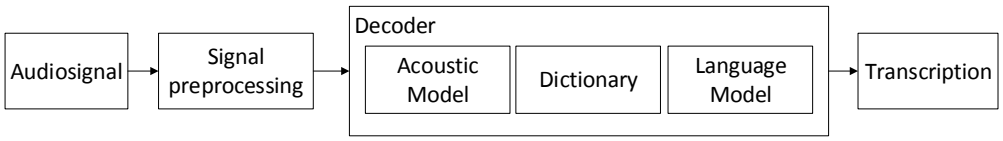

Figure 2.1: Overview of an automatic speech recognition system.

The subsequent sections will introduce the components of a speech recognition system in more detail.

2.1.1 Hidden markov models

Due to the reformulation of our problem in Equation 2.1, we can apply a generative model to our problem. A model which has been frequently used in the context of sequence recognition is the Hidden Markov Model (HMM). Each HMM is defined by the 5-tuple (S, A, V, B, π):

- Set of states $S = s_1, \ldots, s_N$
- $N \times N$ matrix of transition probabilities A, where a_{ij} is the probability of transitioning from state s_i to state s_j
- The vocabulary V containing all possible emissions
- Set of emission probability functions $B = b_1, \ldots, b_N$, where b_i is the emission probability function of state s_i with $b_i(o_t) = P(o_t|s_i)$ for an emission/observation o_t at time t in state s_i
- The probability distribution π over all states in S, modeling the probability of each state being the start of a state sequence

To relate the HMM to automatic speech recognition: each phone, which appears in the language to be recognized, is represented by its own HMM.

Usually, a left-to-right three state topology is employed as depicted in Figure 2.2.

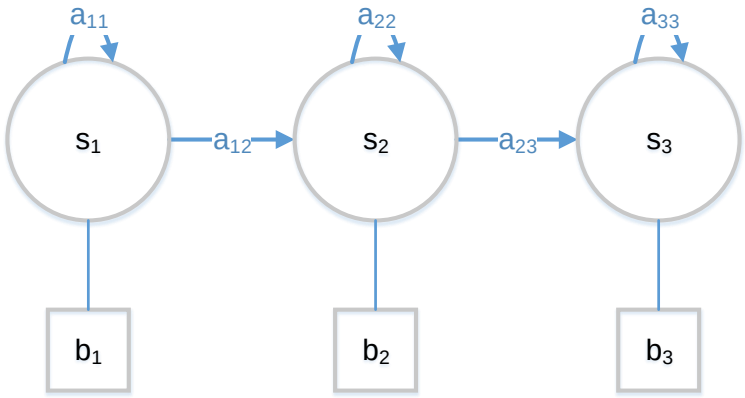

Figure 2.2: Three state left-to-right HMM, with s_1 being the only initial state, i.e. $\pi(s_1) = 1$.

This model has two assumptions: first, the model assumes a first order Markov process, implying that the probability of transitioning to the next state in the model only depends on the current state. Second, the emission probability function is only dependent on the current state. In addition, the state sequence generating the observation sequence O is not observable and therefore hidden. The most likely sequence of states can be extracted using the Viterbi algorithm. The algorithm is explained in more depth in the following subsection. A more detailed introduction of HMMs in general is given by Rabiner et al. [RJ86].

2.1.1.1 Viterbi path

The goal in speech recognition is to find the most likely word sequence given an observation sequence. By using the HMM framework as explained above we are interested in the most likely path in an HMM given the observation sequence, called the Viterbi path. The Viterbi algorithm used to compute the most likely path employs a dynamic programming approach. It uses the fact that for any two paths ending in the same state s_i at time t, only the best path in state s_i can be a part of the best path. Therefore, at any time t only N partial paths exist in the algorithm, where N is the total number of states in the HMM. Therefore, the algorithmic complexity of the algorithm is in $O(NT)$, where T is the number of observations in the observation sequence.

2.1.2 Signal preprocessing

The signal preprocessing component is responsible for converting the audio signal input to the observation sequence needed in the HMM model. The audio signal is in a first step cut into smaller segments of typically $16ms$ of audio, with a $6ms$ overlap between segments, also referred to as frames. In combination with the three state HMM topology, this segmentation enforces a minimum length of all phones of $30ms$. A common set of features that is extracted based on the segmented audio file are Mel-Frequency-Cepstral-Coefficients (MFCCs) [DM80]. MFCCs employ a mel scale filter bank to mimic the human hearing. Typically, the dimensionality of MFCCs is 13.

In order to include information on the audio context of each frame, neighboring frames are often stacked onto the current frame. Usually, the five neighboring frames to the left and right are stacked onto each frame. Due to the increase in the feature dimensionality, feature reduction techniques such as linear discriminant analysis (LDA) are applied [Bro87], which reduces the dimensionality of the feature space to 40.

2.1.3 Acoustic model

The emission probability functions B of the HMM are realized in the acoustic model (AM). Two types of models have been frequently used in the acoustic model for this purpose: Gaussian mixture models (GMM) and Deep Neural Networks (DNN).

2.1.3.1 Gaussian mixture model

Gaussian mixture models (GMM) are commonly used in conjunction with HMMs to model the emission probabilities. Each GMM consists of M Gaussian components and the emission probability $b_i(o_t)$ of emitting o_t in state s_i is defined as:

$$b_i(o_t) = \sum_{m=1}^{M} c_m \mathcal{N}(o_t|\mu_m, \Sigma_m) \tag{2.2}$$

where c_m is the weight of the m-th Gaussian. μ_m and Σ_m are the mean vector and covariance matrix of the m-th Gaussian respectively.

2.1.3.2 Deep neural network

A neural network consists of several perceptrons, which are interconnected. Each perceptron is a linear binary classifier, whose input is the weighted sum of all K input activations $X = x_1, \ldots, x_K$, as shown in Figure 2.3. Each input activation x_i is multiplied with its weight w_i before the activation function $f(z)$ of the perceptron is evaluated.

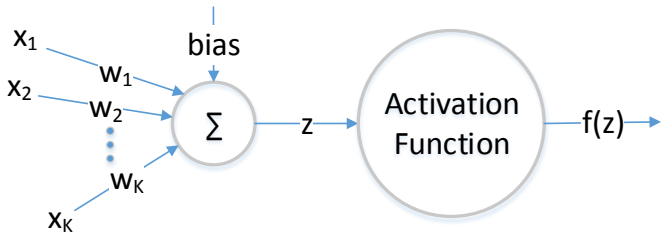

Figure 2.3: Single perceptron inside a neural network.

Perceptrons can be arranged in layers as depicted in Figure 2.4. The figure shows a feed-forward network, where no recurrences exist. An input layer receives the observation o_t, and is connected to the first hidden layer, which in turn can be connected to subsequent hidden layers. The last hidden layer is finally connected to the output layer.

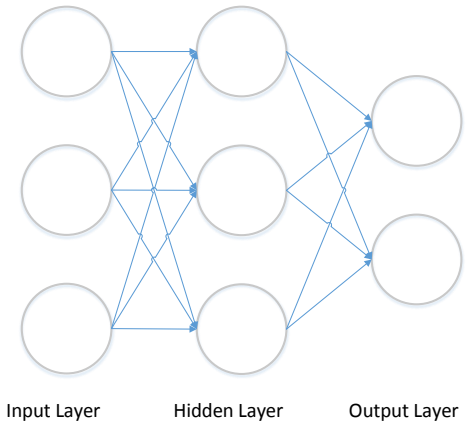

Figure 2.4: Feed-forward network with one hidden layer. The number of perceptrons in the input and hidden layer is three and the output layer consists of two perceptrons.

The name deep neural networks (DNN) stems from the many hidden layers they contain [SLY11]. The size of the input layer corresponds to the dimensionality of the input feature vector. The size of the output layer is determined by the number of unique states in all the HMMs.

All perceptrons in the DNN typically have the same activation function, which is commonly chosen to be the sigmoid function:

$$f(z) = \frac{1}{1 + e^{-z}} \tag{2.3}$$

where z is the weighted sum of the activations of all connected perceptrons from the previous layer, as shown in Figure 2.3. The perceptrons in the output layer form a softmax-layer, such that the output of the DNN is a probability distribution over the HMM states. The softmax activation function is defined as:

$$f_i(Z) = \frac{e^{z_i}}{\sum_{k=1}^{K} e^{z_k}} \tag{2.4}$$

where $Z = z_1, \ldots, z_K$ and z_i is the weighted sum of the activations of all connected perceptrons from the previous layer for the i-th perceptron in the output layer, and K is the total number of perceptrons in the output layer.

Since the DNN is a discriminative model, the output of the model has to be transformed to use it for modeling emission probabilities in the HMM:

$$P(o_t|s_i) = \frac{P(s_i|o_t)P(o_t)}{P(s_i)} \tag{2.5}$$

where $P(s_i|o_t)$ is the output of the DNN and $P(o_t)$ is the probability of observing the feature vector, which is omitted from the computation resulting in a pseudo likelihood. An evaluation of different optimization functions to train DNNs was performed by Veselỳ et al.[VGBP13].

2.1.3.3 Context-dependent modeling

Due to co-articulation effects between phones which are caused by the human vocal tract, modeling of phones based on their context increases the discriminability between them [SCK+85]. Commonly a decision tree is trained on the available training data, where at each node in the tree questions regarding

the phonetic context are posed [LHH+90]. The leaf nodes in the tree correspond to context-dependent acoustic models which are then realized with a GMM or as part of a DNN.

2.1.4 Dictionary

To be able to recognize words instead of phones, the dictionary contains a mapping from words to their respective pronunciation, expressed as a sequence of phones. A word-level HMM is constructed by simply concatenating the HMMs of single phones. If the acoustic model is context-dependent, the correct phone HMM has to be chosen such that context restrictions of the acoustic model are not violated.

2.1.5 Language model

The probability of word sequences is modeled by the language model (LM). A wide-spread approach is the usage of N-grams [Jel90], where the probability of a word depends on the context it has been observed in. The N-gram corresponds to an $n-1$-th order Markov chain. The probabilities of the N-grams are computed by counting the word sequences in a training text:

$$P(w_m|w_{m-1},\ldots,w_{m-(n-1)}) = \frac{Count(w_{m-(n-1)},\ldots,w_m)}{Count(w_{m-(n-1)},\ldots,w_{m-1})} \tag{2.6}$$

where $P(w_m|w_{m-1},\ldots,w_{m-(n-1)})$ is the probability of word w_m appearing in the word context of $w_{m-(n-1)},\ldots,w_{m-1}$.

To be able to compute the probability of word sequences whose N-grams have not been seen in the training text, backoff to lower order N-grams is usually performed [Kat87]. However, since no probability mass for backing off would be available if N-grams are estimated as given in Equation 2.6, discounting the word sequence counts of N-grams and thereby shifting probability mass to the backoff probability has to be performed. A very successful and widely used smoothing strategy was introduced by Kneser et al. [KN95], called

Kneser-Ney smoothing. As a result, the probability of a word w_m becomes:

$$
\begin{aligned}
P_{backoff}(w_m|h) &= \begin{cases} P_{discount}(w_m|h) & \text{if } Count(w_m, h) > k \\ \gamma(h) P_{backoff}(w_m|\tilde{h}) & \text{otherwise} \end{cases} \\
h &= w_{m-1}, \ldots, w_{m-(n-1)} \\
\tilde{h} &= w_{m-1}, \ldots, w_{m-(n-2)}
\end{aligned}
\tag{2.7}
$$

where the discounted N-gram probability $P_{discount}$ is used if the count of the N-gram exceeds a threshold k on the training set, and otherwise the backoff to the lower order N-gram is performed using the backoff probability $\gamma(h)$, such that we have a probability distribution over all words given the word history h.

2.1.6 Decoder

The decoder combines the acoustic model, dictionary, and language model introduced above and realizes the Viterbi algorithm to find the most likely word sequence given an observation sequence. Due to the high number of possible word sequences, heuristics are used during the Viterbi algorithm to limit the number of investigated word sequences. One of these methods is called beam search and employs pruning of partial word sequences, to ascertain that the scores of all investigated word sequences at any time are within beam range of the score of the best partial word sequence.

The HMM that is generated by combining all the model information is called a search network and depends on the type of the decoder. A static decoder combines acoustic model, dictionary, and language model, creating one network, where each node in the network is an HMM state. The search network of a dynamic decoder does not contain the language model information and is implemented as a word-loop. A word-loop contains all word-level HMMs of the ASR system and the last state of each word-level HMM is connected to the first state of each word-level HMM. Phonetic context constraints by the acoustic model have to be accounted for across word boundaries.

The search network is usually determinized and minimized to reduce the total number of states in the network [SS09]. However, this delays the point at which state in the network the identity of a word is certain, thereby delaying the integration of the language model probability into the score of the partial word sequence.

2.1.6.1 Language model lookahead

Since the integration of the language model knowledge is delayed by the determinization of the search network until a word is known, a heuristic called language model lookahead is used [OENC97]. The lookahead gives a best case estimate on the next word's language model probability. At any state in the search network the algorithm collects all words that are still possible to be recognized and returns the highest probability among all collected words. To limit the computation costs of the language model lookahead, the lookahead is often limited to the first dozen states in the search network. The parameter is called the lookahead depth.

2.1.7 Lattice

The lattice is a directed graph which can be extracted from the ASR system after decoding an utterance. It represents a compressed form of all hypotheses recognized in the decoding. A word-level lattice contains recognized words on the edges, along with their acoustic model and language model scores [PHB+12]. The nodes in the lattice correspond to points in time.

2.1.8 Confidence models

The estimation of confidences on the output of an ASR system has long been used for rescoring hypotheses of the system [XPMZ11] or just in general been applied to the task of error detection [All07].

The confidence score is estimated on the lattice, giving the posterior probability of each word in the hypothesis given the observation sequence $O = o_1, \ldots, o_T$:

$$p(w, t_b, t_e|O) = \arg\max_{t:t_b \leq t \leq t_e} \sum_{(t_i,t_j):t_i \leq t \leq t_j} \hat{p}(w, t_i, t_j|O) \tag{2.8}$$

in reference according to Wessel et al. [WMS98]. $p(w, t_b, t_e|O)$ gives the posterior probability of a word which was recognized in the time frames ranging from t_b to t_e. $\hat{p}(w, t_i, t_j|O)$ gives the posterior probability of the word w appearing in the exact time frame from t_i to t_j. The posterior probabilities are computed by using a forward-backward algorithm [BPSW70] on the lattice. Since the posterior probabilities are computed on the lattice

and by picking the maximum posterior probability, the word confidence score is an overestimate of the actual posterior probability.

2.1.9 Evaluation metric

Two evaluation metrics are used in this thesis and will be presented in this section.

2.1.9.1 Perplexity

In order to compare language models among each other without the necessity of evaluating a complete ASR system, the perplexity is used. It is the reciprocal probability of a test word sequence $W = w_1, \ldots, w_N$, normalized by the number of words [BDM+92].

$$\begin{aligned} PP(W) &= \sqrt[N]{\frac{1}{P(W)}} \\ &= \sqrt[N]{\prod_{i=1}^{N} \frac{1}{P(w_i|w_{i-1}, \ldots, w_1)}} \\ &= 2^{-\frac{1}{N}\sum_{i=1}^{N} log_2(P(w_i|w_{i-1},\ldots,w_1))} \end{aligned} \tag{2.9}$$

The perplexity (PP) can be interpreted as the average number of possible successor words after each word.

2.1.9.2 Word error rate

The default evaluation metric of ASR systems is the word error rate (WER). Given a hypothesis from the decoding and a reference for the same utterance, the Levenshtein distance between them is computed. Substitutions, deletions, and insertions have equal weight in the distance computation. The resulting distance is normalized by the number of words in the reference to obtain the word error rate. If smaller units are used for the distance computation, such as syllables or characters, the name of the metric changes

accordingly to syllable error rate (SER) or character error rate (CER):

$$WER(W^1_{hypo}, \ldots, W^M_{hypo} | W^1_{ref}, \ldots, W^M_{ref}) = \frac{\sum_{i=1}^{M} L(W^i_{hypo}, W^i_{ref})}{\sum_{i=1}^{M} |W^i_{ref}|} \cdot 100\% \tag{2.10}$$

where M is the number of utterances to compute the WER on, and W^i_{ref} is the reference transcription of the i-th utterance, and W^i_{hypo} the hypothesis of the i-th utterance. L is the Levenshtein distance.

2.2 Automatic speech recognition toolkits

A great number of toolkits exists, which focus on acoustic speech as their core modality, such as the HTK toolkit [YEG+06], Sphinx-4 [WLK+04], Julius [LKS01], and more recently the Kaldi toolkit [PGB+11]. A toolkit with a broader focus is the RWTH Aachen toolkit [RGH+09]. The RWTH Aachen toolkit is for example used for recognizing sign language [KNB13]. The Georgia Tech Gesture Toolkit [WBAS03] extends HTK to simplify the process of building HMM-based gesture recognizers.

Frameworks, which had a notable impact on the development of the toolkit presented in Chapter 4, called BioKIT, are the Janus Recognition Toolkit (JRTK) [FGH+97, SMFW01], the Attila toolkit [SSK10], and the Kaldi toolkit [PGB+11]. This section will give an overview over the various properties of these toolkits.

2.2.1 Decoder programming languages

The choice of programming language determines the ease of extending a toolkit, of setting up experiments, and of porting the framework to new platforms.

2.2.1.1 Ease of extension

All recent toolkits make use of programming languages which use an object-oriented programming paradigm, such as C++ and Java. Older toolkits such as the JRTK are written in C and are more difficult to extend [FGH+97].

Concerning C++ and Java: only the Sphinx toolkit uses a version which is implemented in Java [WLK+04], all other toolkits use C++.

2.2.1.2 Setup of experiments

A common approach among toolkits is to use a two-layer design of either C or C++ as the programming language for core algorithms, optimized for speed and low memory footprint and an outer scripting layer, embedding the core library. Toolkits following this two-layer design are for example JRTK [FGH+97] and Attila toolkit by Soltau et al. [SSK10]. Similar to the Attila toolkit, BioKIT has a Python scripting layer and a C++ layer for the toolkit's core library. The main concern is an easy to use scripting language, which is well supported in different operating systems and has a lot of developers contributing their libraries as a way of easily integrating additional features into experiments. Examples are the Matplotlib library [Hun07] for visualization of plots as well as the SciKit-Learn library [PVG+11] containing several machine learning methods.

2.2.1.3 Portability

Due to its extensive standard library and running in the Java Virtual Machine, makes toolkits using Java very ease to port to new platforms, such as mobile devices. In contrast C++ toolkits require the toolkit to be either compiled directly on the target platform or being cross-compiled on a different platform.

2.2.2 Static and dynamic decoder

Current toolkits usually offer either dynamic [RGH+09, SSK10] or static decoders [PGB+11]. Key difference is the search network. A static decoder integrates the linguistic knowledge of the language model directly into the search network, thereby increasing the size of the search network significantly compared to a dynamic decoder, which does not include the language model into the network [SS09].

While the decoding speed of the ASR system is usually faster using a static decoder, the build time of the search network especially with large vocabularies and big language models is considerably longer [SS09]. However, if the network does not change, as is the case for acoustic modeling research,

where other components are kept constant, the search network only needs to be created once and can be reused in experiments [PGB+11]. Since acoustic model, dictionary, and language model are integrated into the network, the decoder is less complex than a dynamic decoder. A drawback is that only language models which can be converted into a finite state transducer can be used. Dynamic decoders offer more flexibility for language modeling at the expense of more complex decoders [SSK10].

The programming overhead in dynamic decoders is caused by the search network. In static decoders computational overhead for each partial hypothesis is the same. However, since the language model information is not encoded in the network, a dynamic decoder has to use language model lookaheads to effectively prune partial hypotheses. The lookahead computation is a costly operation, requiring additional pruning thresholds for partial hypotheses to reduce the total number of lookahead queries to the language model. Furthermore, since the state does not automatically indicate the language model context, as in static decoders, each partial hypothesis has to know its own context, increasing the required amount of dynamically allocated memory.

If the vocabulary size or the language model is too big, the network of the static decoder will not fit into the memory of most desktop machines. The Kaldi toolkit [PGB+11] offers the possibility of creating a search network using a smaller language model and integrating the bigger language model on the fly into the decoding process. While this enables Kaldi to keep the size of the network in check and use big language models, the partial hypotheses of the decoder have to keep track of their own language model context, as in dynamic decoders. In this case, the smaller language model used for network construction serves as a language model lookahead similar to dynamic decoders, if weight pushing is performed on the search network, i.e. the language model probabilities are pushed towards the first states of the word-level HMMs.

2.3 Error blaming and error correction

Previous work in the field of error blaming and error correction relies on the availability of reference transcriptions. Ringger et al. [RA96] proposes a post-processing approach, wherein typical erroneous segments are learned on a training set, viewing the task as a translation of erroneous to corrected sentences. Other work investigates error correction in the context of user interfaces, where the user can correct the recognized utterance [SMW01].

Other investigations focus on the analysis of errors to correct mistakes in the models themselves. Among those is the work by Steinbiss et al. [SSN10] investigating the impact of pruning parameters of the decoding on the recognition results. The SCTK toolkit [SCT07], besides computing the word error rate of an ASR system, depicts alignments of reference transcriptions and hypotheses, collects information on word confusions, and offers statistical analysis tools for ASR systems.

In order to analyze where the encountered errors originated from, Chase [Cha97], and afterwards Nanjo et al. [NRK99], introduced a rule based system, wherein errors are assigned to error categories. This error blaming is further described in the following two subsections. As a preliminary step, a special alignment of reference transcription and hypothesis has to be computed. Subsequently, the rule based system can be applied to assign each error to an error category.

We focus on the work of Chase [Cha97] here, who was the first to publish work on error blaming and only few other papers were published in this area. The field of unsupervised model training, presented in the next section, garnered more attention, since ASR training data was scarce and improvements to ASR systems were partly achieved by increasing the training data for the acoustic model and language model [KW99]. However, early papers already observed a linear relationship between the word error rate of an ASR system and the logarithm of the amount of training data [Moo03]. Current state-of-the-art ASR systems are already trained on more than $10,000$ hours of speech data [MHM+14]. Thus, renewing the interest in identifying errors in ASR systems, such as the biennial ERRARE workshop.

2.3.1 Error regions

Differing from the alignment computed by the Levenshtein distance, the error blaming algorithm requires the reference transcriptions and hypotheses to be temporally aligned. Each segment in the alignment starts and ends at a word boundary and contains one or multiple words. If a segment contains an error, it is referred to as an *error region* [Cha97]. The computation of the *error regions* can be performed in two steps:

1. A temporal alignment of reference transcription and hypothesis is computed by using the Viterbi alignment of the reference and the Viterbi path of the hypothesis. A valid segment in the temporal alignment requires that the segment boundaries coincide with word boundaries of

reference and hypothesis words. A segment is padded with additional words from reference and hypothesis until this requirement is met. In the worst case the utterance consists of only one segment containing the complete utterance. The requirement can be relaxed by allowing a deviation in word boundaries at the start and end of segments between reference and hypothesis words. An example for a temporal alignment with no allowed deviation of reference and hypothesis word boundaries is given in Table 2.1.

2. Neighboring segments are merged if an error in a segment affects the next segment. In the case of an N-gram language model, an error affects the language model scores of the next $N-1$ words. Table 2.2 shows an example alignment using a 3-gram language model based on the temporal alignment depicted in Table 2.1. Segments one and two of Table 2.1 had to be merged in the alignment, since errors in segment one affect the language model score of the character sequence in segment two.

Table 2.1: *Temporal alignment of partial utterance from the SEAME corpus [LTCL10]. No deviation between reference and hypothesis word boundaries was allowed.*

Segment	1				2	3
Ref. frames	67–98	99–110	111–136	137–155	156–173	174–187
Hyp. frames	67–114		115–131	132–155	156–173	174–187
Ref.	b.	and	above	的话	我	就
Hyp.	piano		book	的话	我	就

Table 2.2: *Alignment of partial utterance from the SEAME corpus [LTCL10]. Segments one and two of Table 2.1 are merged into one segment.*

Segment	1					2
Ref. frames	67–98	99–110	111–136	137–155	156–173	174–187
Hyp. frames	67–114		115–131	132–155	156–173	174–187
Ref.	b.	and	above	的话	我	就
Hyp.	piano		book	的话	我	就

Table 2.3 shows the partial output of an alignment computed on a single utterance taken from the WSJ0 corpus [GGPP93]. The reference and hypothesis are aligned and split into *error regions*, no deviation between reference and hypothesis word boundaries was allowed. AM and LM scores are given for each segment. The scores are presented as negative log-likelihoods.

Table 2.3 shows three segments. In the third segment both acoustic and language model scores favor the erroneous hypothesis over the reference.

Table 2.3: *Utterance segment from the training set of the WSJ0 corpus [GGPP93]. The table shows component scores for reference (R) and hypothesis (H) of regions in the utterance. Error categories are listed in the last row of each region. Scores in table are negative log-likelihoods.*

Frames	73–85	86–103	104–147
Reference	with	that	surge has
Hypothesis	with	that	searches
Total score (R)	763.70	1054.16	2979.40
Total score (H)	763.70	1054.16	2862.51
AM score (R)	758.64	995.68	2799.51
AM score (H)	758.64	995.69	2725.93
LM cost (R)	5.06	58.48	179.89
LM cost (H)	5.06	58.48	136.58
Error-category	CORRECT	CORRECT	AM_LM_OVERWHELM

2.3.2 Blame assignment

After computing the *error regions* in an utterance, the decision tree as depicted in Figure 2.5 is used to determine the error category of each *error region* [Cha97]. For this purpose the computed scores of each region, as shown in Table 2.3 are used in the decision tree. Every segment which is not an *error region* is automatically assigned the category *CORRECT*, stating that there is no error in the segment. The following error categories exist:

- **Search error:** If the combined score of acoustic and language model is lower for the reference segment than for the hypothesis, the misrecognition is due to prematurely pruning the correct hypothesis from the decoding process.
- **Homophone:** The phone sequence of reference and hypothesis are the same in the segment. Therefore, only the language model is able to distinguish between the two word sequences, and the language model is at fault.
- **LM overwhelm:** If a wrong word sequence was recognized despite the fact that the acoustic model score of the reference is better than the hypothesis acoustic model score, the language model dominated

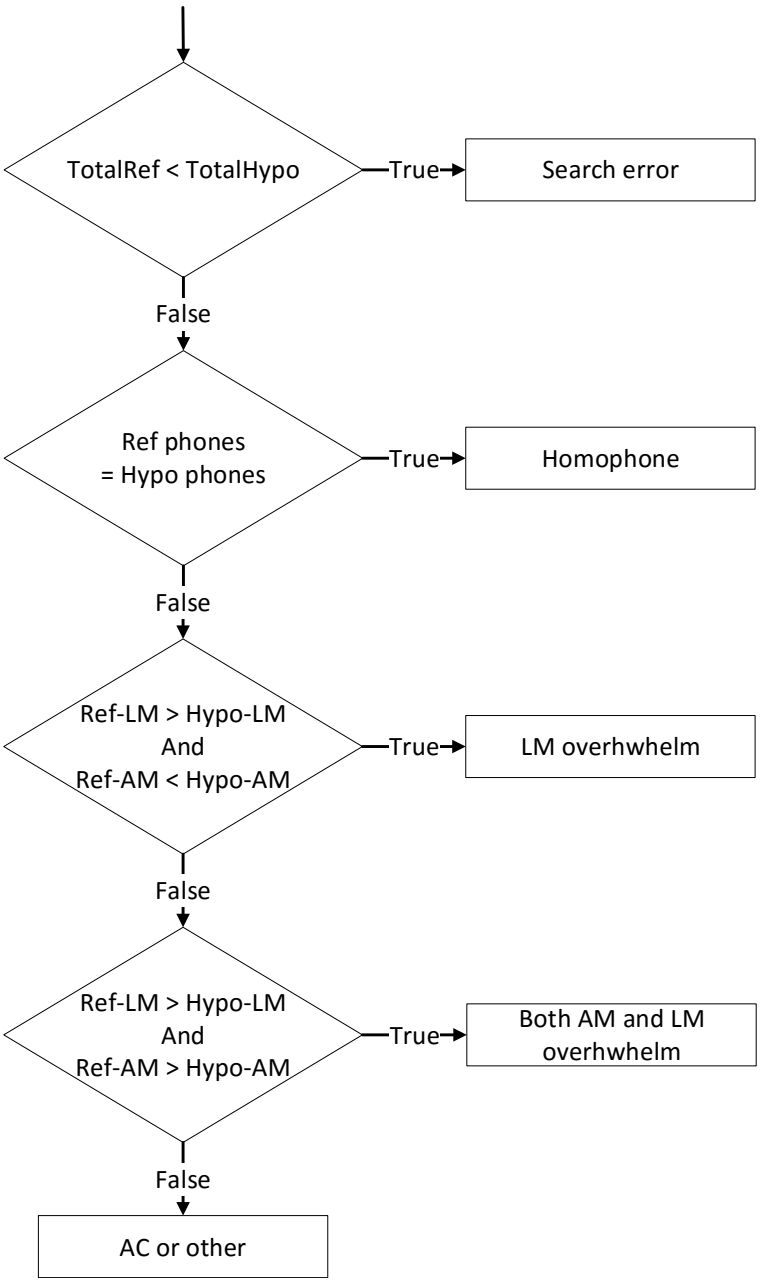

Figure 2.5: Binary decision tree used for assignment of error categories during error blaming, based on Chase [Cha97]. Scores for decision tree decisions are assumed to be negative log-likelihoods.

the acoustic model. Chase [Cha97] further distinguishes between LM errors that can be fixed by adjusting the language model weight and LM errors that cannot.

- **AM and LM overwhelm:** If both acoustic and language model assign a higher likelihood to the hypothesis, both models are at fault.
- **AM or other:** Errors that could not be assigned to any other category are handled in this category. Chase [Cha97] used additional rules to determine if the error was caused by an erroneous pronunciation in the dictionary, or if the acoustic models of reference and hypothesis are problematic.

In addition, by employing a phone-only decoding, Chase tried to detect out-of-vocabulary words in utterances.

2.4 Unsupervised training

In unsupervised training either an initial ASR system is already available to automatically transcribe data without reference transcriptions or some data with reference transcriptions is available to train an initial ASR system. There has been plenty of research applying this procedure, starting with the work by Zavaliagkos et al. [ZC98], conducting experiments on the Switchboard corpus [GHM92]. Kemp et al. [KW99] used a lattice-based confidence measure to select training data for unsupervised training of the acoustic model on broadcast news data. Subsequent work by Lamel et al. [LGA02] showed that only small amounts of data (as little as 10 minutes) are required to obtain improvements by using unsupervised acoustic model training. An additional more extensive study on broadcast news was conducted by Wessel et al. [WN05]. Previous work only considered the best hypothesis of each automatically transcribed utterance, Fraga-Silva et al. proposed to use the lattice generated from the automatic transcription process to be used instead, showing improvements on a broadcast news transcriptions task [FSGL11]. Vu et al. [VKS11] proposed to use several existing ASR systems to automatically transcribe data and combine the resulting hypotheses for unsupervised training. Finally, Laurent et al. [LHL14] showed the feasibility of unsupervised acoustic model training on a state-of-the-art system trained on Korean. They employ an iterative process, starting with an initial ASR system trained on a small amount of transcribed data. In each iteration the ASR system from the previous iteration is used to automatically transcribe

the training data. The automatic transcriptions are then filtered using an utterance-level confidence threshold, followed by training a new ASR system.

Chapter 3

Database

Several data corpora have been used throughout this thesis. They are introduced in ascending order of difficulty, where their respective difficulty is estimated based on the performance of ASR systems of each corpus.

3.1 GlobalPhone

The GlobalPhone corpus was created at the Karlsruhe Institute of Technology (KIT) by Schultz et al. [Sch02]. It is a collection of more than 20 languages from all around the world. Each language is comprised of high quality audio recordings of about 20 hours, including transcriptions. The corpus consists of read speech by about 100 speakers in each language. Pronunciation dictionaries as well as baseline N-gram language models are a part of the corpus. The data of each language is split into three parts. The training set is used to train the acoustic model and language model, the development set is used to tune parameters of the decoder such as pruning parameters, and the evaluation set is used to test the ASR system on.

In this thesis we employ the Bulgarian, Czech, German, and Vietnamese data collections from the GlobalPhone corpus. Information on the employed GlobalPhone languages and statistics are given in Table 3.1 and Table 3.2. The vocabulary size, dictionary size, and number of tokens for language model estimation for each language are given in Table 3.3. Further information on

the language model is given in Section 4.3.1.

Table 3.1: *Statistics of parts of the GlobalPhone database: Bulgarian (BG), Czech (CZ), German (GE), and Vietnamese (VN) languages.*

	Training		Development		Evaluation	
Language	[h:mm]	#Tokens	[h:mm]	#Tokens	[h:mm]	#Tokens
Bulgarian	16:48	119k	2:18	15k	2:00	14k
Czech	26:48	186k	2:24	17k	2:42	18k
German	14:54	116k	2:00	15k	1:30	12k
Vietnamese	22:18	164k	1:42	14k	1:30	16k

Table 3.2: *Number of speakers and utterances of parts of the GlobalPhone database: Bulgarian (BG), Czech (CZ), German (GE), and Vietnamese (VN) languages.*

	Training		Development		Evaluation	
Language	#Spk.	#Utt.	#Spk.	#Utt.	#Spk.	#Utt.
Bulgarian	63	6,890	7	816	7	816
Czech	82	10,367	10	1,028	10	1,030
German	65	8,185	6	1,073	6	826
Vietnamese	131	18,239	10	1,291	10	1,225

Table 3.3: *Statistics on the vocabulary, dictionary, and number of tokens to estimate the language model on, of parts of the GlobalPhone database.*

	#Vocab	#Dictionary entries	#Tokens for LM
Bulgarian	99,471	99,576	405M
Czech	32,942	33,035	508M
German	36,987	39,521	20M
Vietnamese	29,820	37,370	39M

3.2 SEAME corpus

The code-switching speech corpus we use is called SEAME (South East Asia Mandarin-English). It was recorded in Singapore and Malaysia by Lyu et al. [LTCL10] and contains spontaneously spoken interviews and conversations. Originally, it was used for the research project "Code-Switch" which

was jointly performed by the Nanyang Technological University (NTU) and the Karlsruhe Institute of Technology (KIT). Code-switching speech is defined as speech which contains more then one language. The code-switches in the SEAME corpus can appear at any point in an utterance. The corpus contains about 62 hours of audio data and manual transcriptions which are separated into training, development and evaluation set (refer to Table 3.4).

The words can be categorized into four language classes: Mandarin (58.6% of all tokens), English (34.4% of all tokens), particles (Singaporean and Malayan discourse particles, 6.8% of all tokens) and other languages (0.4% of all tokens). The average number of code-switching events between Mandarin and English is 2.6 per utterance. A code-switching event is defined as a switch between languages. The average duration of monolingual English and Mandarin segments is only 0.67 seconds and 0.81 seconds, respectively. In total, the corpus includes 9,210 unique English and 7,471 unique Mandarin types. Lyu et al. published an initial analysis on a subset of the corpus [LTCL10].

The vocabulary contains $29,173$ and the dictionary consists of $52,232$ entries. The language model is estimated on the training set transcriptions, which amount to $571k$ tokens. In addition, the language model was interpolated with monolingual language models built on $246M$ tokens of English NIST data and $195M$ tokens of Mandarin from the GALE project [VLW+12]. Further information on the language model is given in Section 6.1.1.

Table 3.4: *The SEAME database.*

	Training	Development	Evaluation
#Speakers	157	8	8
#Utterances	48,040	1,943	1,018
#Tokens	571k	23k	11k
Duration [h:mm]	58:24	2:30	1:06

3.2.1 Splitting training data

To be able to conduct the experiments in Chapter 6 on unsupervised model training we split the training data into two parts, since the SEAME corpus does not contain any untranscribed data. A third of the data was used to train the initial speech recognizer with which the remaining training data is decoded and used for the unsupervised experiments. Table 3.5 contains information on the two new training sets. Each speaker in the original training set is assigned to either transcribed training data or untranscribed training

set. Training data was split such that the ratio of male to female and Singaporean to Malayan speakers remains the same as in the whole training set.

Table 3.5: *SEAME training data for the unsupervised clustering.*

	Supervised Training	Unsupervised Training	Total
#Speakers	58	99	157
#Utterances	17,746	30,294	48,040
#Tokens	201k	370k	571k
Duration [h:mm]	19:04	39:20	58:24

3.2.2 Challenges

The SEAME corpus is not only challenging because of the code-switching events, which can appear mid-sentence. The speakers are Singaporean and Malayan who both have their own language specific pronunciation of the Mandarin characters and English words. Pronunciations that were available for the initial systems are based on an American English dictionary from CMU [Wei98] and the Mandarin GlobalPhone dictionary. Vu et al. tried to adapt the dictionary to the Singaporean/Malayan English by using applying several rules to the pronunciations [VLW+12]. Since code-switches are a spoken event and do not appear in written texts, the training data available for language modeling are limited to the audio transcriptions of the SEAME corpus, making the estimation of code-switching events especially challenging.

3.3 ILSE corpus

For the ILSE study [MM00, LTSM00], an interdisciplinary team at Heidelberg University was given the task of conduction a large-scale investigation into the impact of ageing on several aspects of daily life. Over the course of 20 years they conducted interviews and medical examinations on about 1,000 participants. The collected German speech corpus was partly digitized and transcribed. Details of the corpus and the subsets are given in Table 3.6. The vocabulary is derived from the training set transcriptions of the ILSE corpus and amounts to $71,928$ words. The pronunciations for the words in the vocabulary are either taken from the German GlobalPhone dictionary

or automatically generated. The total number of entries in the dictionary is $74,462$. The language model is estimated on the $2M$ tokens of the ILSE training set transcriptions and is further described in Section 6.3.2.2.

Table 3.6: *The ILSE corpus. The development and evaluation set statistics only contain the first interview session for each speaker.*

	Training	Development	Evaluation
#Speakers	68	10	14
#Utterances	30,647	936	1766
#Tokens	2,000k	73k	118k
Duration [h:mm]	265:24	9:00	14:24

3.3.1 Challenges

The audio recordings collected in the ILSE study pose several challenges to an ASR system. The list of challenges are given in the following list and are published by Weiner et al. [WFT+16]:

- **Transcription Quality:** While transcriptions exists for the data listed in Table 3.6, they do not reflect the verbatim content of the interview. Furthermore, speaker changes and utterance boundaries were transcribed but not marked in the interview recordings, and no time alignment between recording and transcript was created. Each transcription file is associated with 45min to 2 hours of recorded speech. 65% of the transcriptions were created without any post-checks and do not include any annotation of hesitations, back-channeling, or disfluencies.
- **Anonymized Transcriptions:** To comply with ethical, legal, and social responsibilities, all transcriptions were anonymized. Proper names, city names, and dates were substituted with generic place holders, resulting in a mismatch between transcript and actual spoken content.
- **Recording Quality:** No special attention was paid to the recording setup and the recording quality. Due to the start date of the study in 1993, no ASR approach to automatic transcription was envisioned. The setup consists of a single microphone placed in-between the interviewer and the study's participant. As a result, the recorded audio data contains noise, such as shuffling of paper, writing, and placement of objects, and often captures one speaker well, but the other not.

Furthermore, the recording device and recording quality was changed during the course of the study, such as switching from analogue to digitial recording devices.

- **Speaking Style:** While the interviewers were given specific questions to ask and are as a result well planned, the participants were encouraged to give lengthy spontaneous answers. In addition, due to the spontaneous nature of the interviews, cross-talk and disfluencies regularly appear during the course of a recording session. Also, due to personal content, emotional speech occurs.
- **Dialectal Speech:** The participants in the ILSE study are living in Leipzig and Heidelberg/Mannheim. In both regions a very distinct German dialect is spoken. While most parts of the transcriptions are written in the standardized German writing system, some portions of the transcriptions contain quasi-phonetic transcripts to write dialectal variants. The quasi-phonetic dialect variants were not marked and the standardized form of the dialectal variant was not annotated.

Chapter 4

BioKIT - Real-Time Decoder

The choice of the toolkit that is used for conducting the experiments is very important, since it dictates the possibilities with regards to experiments as well as ease of extension. Furthermore, to properly understand the encountered errors in a hypothesis and to know how to correct them, an in-depth understanding of the toolkit is very beneficial. This chapter introduces the toolkit BioKIT which we created during the course of this thesis and in which we integrated our error signature algorithm further described in the next chapter.

4.1 Introduction

Several toolkits for HMM-based sequence modeling have been developed over the last decades, especially in the area of automatic speech recognition, toolkits are well represented, as discussed in Section 2.2. Since the goal of this thesis is to promote error correction based on error signatures for HMM-based sequence recognition not only for speech signals, but biosignals in general, an easy to handle toolkit in which the algorithms are embedded is needed. However, most of the existing toolkits are tailored to the automatic speech recognition community with a very specific terminology. The following paragraph summarizes the key requirements that led us to develop a new toolkit we call BioKIT [TWG+14]:

- **Accessibility:** The terminology used in the toolkit should be suffi-

ciently generic to appeal to an interdisciplinary audience. Furthermore, interfaces and implemented modules, as well as the relation between them, should be well documented and tutorials should demonstrate common use-cases. In addition, a suite of integration and unit-tests should ensure that the implemented algorithms work correctly.

- **Error analysis:** During research it is important to analyze recognition errors to identify the responsible system components and room for improvements.

- **Flexible processing and modeling** of various types of signals, such as multi-channel bio-physiological signals like muscle-activity, high-resolution frequency based speech signals, and low-resolution time-based signals like accelerometer signals: All signal pre-processing modules should have generic interfaces that allow flexible use within arbitrary chains of processing modules for state-of-the-art decoding. Model assumptions should be minimal, e.g. flexible HMM architectures should be supported, including multiple streams, and hierarchical modeling.

- **Fast setup of experiments and flexible code-base:** The efforts of creating a new recognition setup and performing experiments should be small. Code should be designed in a modular fashion making it easy to combine and replace features or change behavior of existing algorithms. This should be possible during run-time and, in particular, without the necessity of recompiling and linking the complete code-base.

- **Processing of big amounts of data:** Sizes of data sets for ASR and other human-machine interaction applications, such as wearable computing and activity recognition, are increasing. Therefore, the toolkit should be able to handle massive amounts of data with correspondingly big models. Furthermore, parallelization which means utilizing clusters and multi-core machines efficiently (i.e. share memory across threads) is a requirement.

- **Online-capability:** The decoder needs to process a continuous data stream in real-time by chunk-wise decoding and outputting of intermediate results.

The remaining chapter gives an in-depth overview of the toolkit, its major interfaces, relationships between them, and the capabilities of the library in general. Finally, we present some experimental results comparing our toolkit to the state-of-the-art Kaldi toolkit [PGB+11], which is a freely available speech recognition library.

4.2 Overview

BioKIT was designed in such a fashion that algorithmic changes in the code can be made without the necessity of re-factoring the whole code-base. Figure 4.1 shows the dependency graph of the major classes and interfaces of the BioKIT core library. Classes with a dotted frame indicate an abstract class, having several implementations. To reduce the effort for new users to create their own experimental setup, BioKIT contains an extensive documentation including tutorials for several use-cases and a generic terminology to make it accessible to users beyond automatic speech recognition. Furthermore, a suite of unit- and integration-tests allows the user to verify the correctness of the implemented algorithms. The following sections elaborate on the major interfaces depicted in Figure 4.1, with Section 4.2.1 explaining our terminology and the dependency between the Objects listed in Figure 4.1.

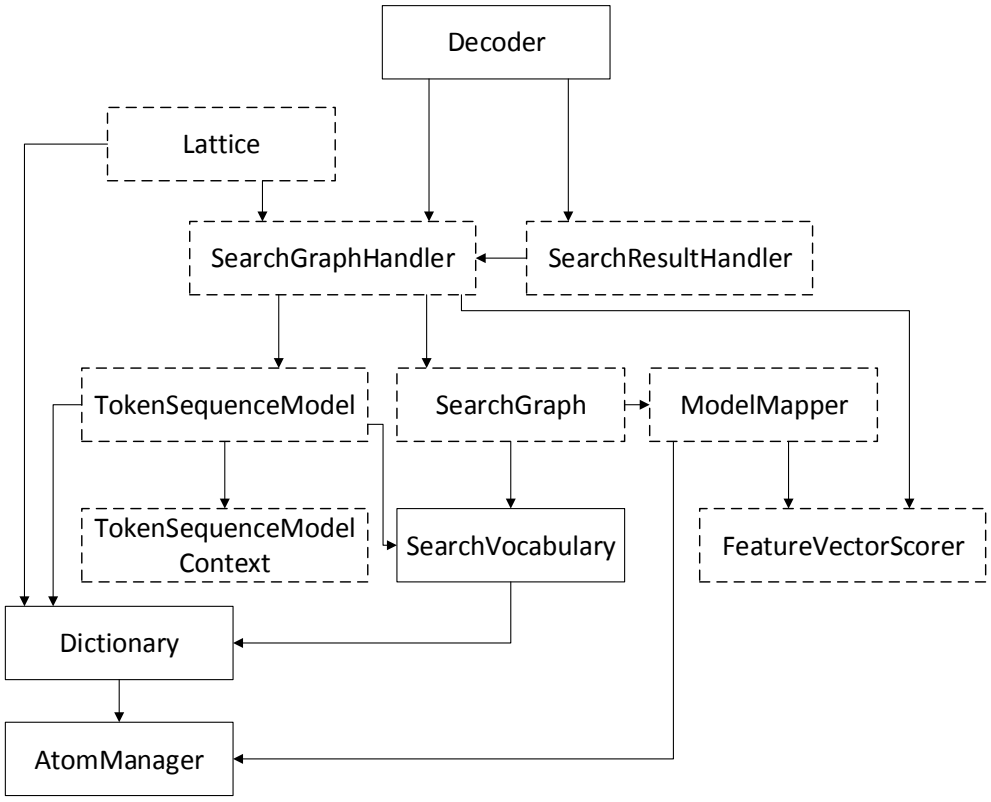

Figure 4.1: Dependency graph of BioKIT. Boxes with dotted lines indicate an abstract class.

4.2.1 Terminology

The terminology we employ in our toolkit abstracts from the specific terminology found in many speech recognition toolkits to appeal to a wide audience of researchers investigating different biosignals and input modalities. The goal of the recognition process in the BioKIT toolkit is to give the most likely sequence of *Tokens* given an observation sequence. In terms of speech recognition is the most likely word sequence given an audio signal as input.

The smallest unit in our system is an *Atom*. Each *Atom* is modeled by an HMM. Attributes of these *Atoms* as well as the *Atoms* themselves are managed by the *AtomManager*. A *Token* is a sequence of one or more *Atoms*, which also compose BioKIT's final recognition results. In the simplest case, one *Token* consists of one *Atom* which equates to modeling a *Token* with one distinct HMM. Each *Atom* can be associated with common attributes defined in the *AtomManager*. Examples of attributes in the *AtomManager* data-file for phonemes in the English WSJ0 corpus [GLF+93] are given in Table 4.1.

Table 4.1: *Examples of attributes in the* AtomManager *data-file for phonemes in the English WSJ0 Corpus. The attributes "Mid", "Low", and "Front" correspond to the position of the tongue.*

Attribute	List of phonemes
Vowel	/AA/ /AH/ /EH/
Mid	/AH/
Low	/AA/
Front	/EH/

The first entry in the list corresponds to an attribute and each following entry is a phoneme in the *AtomManager*. The *Dictionary* describes the mapping of *Tokens* to *Atom* sequences. Context-dependent attributes of *Atoms* can be defined here. In addition, each *Token* in the *Dictionary* can have any number of attributes which can be numbers (e.g. pronunciations weights), or lists of strings (e.g. features for the factored language model as described in Section 4.2.4). Name and form are left to the user as long as the appropriate attribute handler is registered in the dictionary. Examples of these context-dependent attributes for *Atoms* and attributes for *Tokens* are given in the following list:

- what - {/W/ WB} /AA/ {/T/ WB} [PronunWeight 1.61] [POS WDT]
- what - {/W/ WB} /AH/ {/T/ WB} [PronunWeight 0.22] [POS WDT]

The first entry in the dictionary is the *Token*, followed by the sequence of *Atoms*. Both *Atoms* in the beginning and end of the sequence have assigned a tag to them: "WB" for word boundary. Each of the two *Tokens* has two attributes appended to their sequence of *Atoms*, negative log-probability of its pronunciation and their corresponding part-of-speech tag (e.g. "WDT" refers to "Wh-determiner", for words such as "what", "who", "where" [MMS93]). The pronunciation weight is used to assign probabilities to pronunciation variations of the same token.

The *SearchVocabulary* can be used to restrict the *Tokens* which can be recognized during decoding and is an input to both the *SearchGraph*, constructing the search network, and the *TokenSequenceModel*. The *TokenSequenceModel* models occurrence probabilities of *Token* sequences. Emission probabilities of HMM states are computed by the *FeatureVectorScorer*. The *FeatureVectorScorer* computes the scores given a model id and feature vector of the input signal. The unit to create the search network is the *ModelMapper* which is responsible for mapping sequence of *Atoms* to their corresponding HMM.

To relate these terms to the common ASR terminology: A single *Atom* corresponds to a phone in speech. A sequence of *Atoms* equates to the pronunciation of a *Token*, i.e. a word. The *TokenSequenceModel* conforms to the language model. The *FeatureVectorScorer* corresponds to the acoustic model.

4.2.2 Preprocessing

The preprocessing is organized in modules which can be freely combined into preprocessing chains. Each module is able to process any kind of multi-channel data which are represented as lists of matrices where each matrix corresponds to a single channel. Typical modules in BioKIT are windowing, spectral analysis, linear transformation (e.g. computed by Linear Discriminant Analysis or feature-space Maximum Likelihood Linear Regression (fMLLR) [Gal98]).

Common preprocessing modules are implemented in the C++ layer, but could also be implemented directly in Python allowing for rapid prototyping of new preprocessing modules. As all matrices in the Python layer are represented as NumPy arrays, the NumPy and SciPy libraries [JOP+01] can be seamlessly integrated. This offers the whole range of mature and optimized signal processing algorithms from these libraries to the user. The

preprocessing chains are implemented in Python, enabling the user to modify and extend preprocessing operations on the fly. Processing modules can be reconfigured and exchanged during run-time.

4.2.3 FeatureVectorScorer interface

The *FeatureVectorScorer* manages the emission models in the HMM states and computes the emission probabilities. *FeatureVectorScorer* can be substituted with any valid implementation of the scorer interface to compute a score for a given model identifier and feature vector. We support Gaussian Mixture Models and Quicknet-trained neural networks[JEO+05]. In addition, if a compiled version of Kaldi is found during configuration of BioKIT, wrapper classes integrate a range of Kaldi models [PGB+11]. Currently supported adaptation methods are Maximum Likelihood Linear Regression (MLLR) [Gal98] and Maximum-A-Posteriori (MAP) adaptation [GL94].

4.2.4 TokenSequenceModel interface

The *TokenSequenceModel* (TSM) computes the probability of a *Token* given a sequence of previous *Tokens*. An important reason for us to develop a dynamic decoder is the ability to use arbitrary knowledge sources for modeling token sequences (concerning the type of model itself, as well as supporting very small to very large models). Currently, we support a grammar, a standard N-gram model in the ARPA format [Sto02], a factored language model [BK03], and the possibility to interpolate between *TokenSequenceModels*. In addition, we implemented wrapper classes for the TSM, such as the *CacheTokenSequenceModel* to cache queries to the TSM for faster turnaround times, and the *FillerWrapper* class, which handles queries for *Tokens* which are not modeled by the wrapped TSM. Custom *TokenSequenceModelContexts* for each TSM ensure that any knowledge source can be integrated. The only condition for *TokenSequenceModelContexts* to be valid is that they implement the *AbstractTokenSequenceModelContext's* interface, which consists of a hash function and a function to test two contexts for equality.

4.2.5 SearchGraphHandler interface

The decoder employs a token passing algorithm on a determinized and minimized search network [SS09]. Context-dependent modeling of *Atoms*, com-

monly used in ASR (see Section 2.1.3.3), is directly integrated into the network with unlimited context size. The only constraint of the network construction algorithm is to use left-to-right HMM topologies. Currently, two implementations of the *AbstractSearchGraph* interface exist. The *RestrictedSearchGraph* creates a search network consisting of a sequence of *Tokens*. The *RestrictedSearchGraph* is used to compute the Viterbi path of a reference transcription, given the corresponding observation sequence.

The *SearchGraph* implements the search network used for decoding and is a *Token*-loop. The *SearchGraphHandler* is responsible for conducting the actual search on any *AbstractSearchGraph* and is keeping a list of partial hypotheses. The search is implemented similar to Kaldi [PGB+11] with the exception that we have a dynamic decoder employing lookahead for the *TokenSequenceModel*. By implementing a *TokenSequenceModel* lookahead [OENC97], any *TokenSequenceModel* information can be integrated as soon as possible into the decoding process. A wrapper class implementing the *TokenSequenceModel* interface caches frequent queries employing a least recently used strategy and thus speeds up the search. Upon reaching the maximum size of the cache, the component removes the least recently used context and a new context is added in its stead.

4.2.5.1 Search network construction

The construction of the search network is split intro three stages. In the first stage of the network building process a graph consisting of all possible *atom sequences* based on the dictionary is assembled. *Token* Ids are added to the last node of the *body* of each *Atom sequence*. *Body* is defined here as the portion of an *Atom sequence* not influenced by *Atoms* of preceding or succeeding *Tokens* (as in [SS09]). The atom graph is determinized and minimized to reduce the total number of nodes as far as possible for the next stage. Each node at this stage represents one *Atom*.

As an example for a search network construction, we use the dictionary given in the list below, consisting of two different words (*Tokens*) with two pronunciations each. The examples are taken from the CMU pronunciation dictionary [Wei98]:

- what's(1) - /W/ /AA/ /T/ /S/
- what's(2) - /W/ /AH/ /T/ /S/
- wail - /W/ /EY/ /L/
- wale - /W/ /EY/ /L/

The pronunciations of the words "wail" and "wale" are identical, to visualize how we handle *Tokens* with the same sequence of *Atoms*, i.e. homophones in speech recognition.

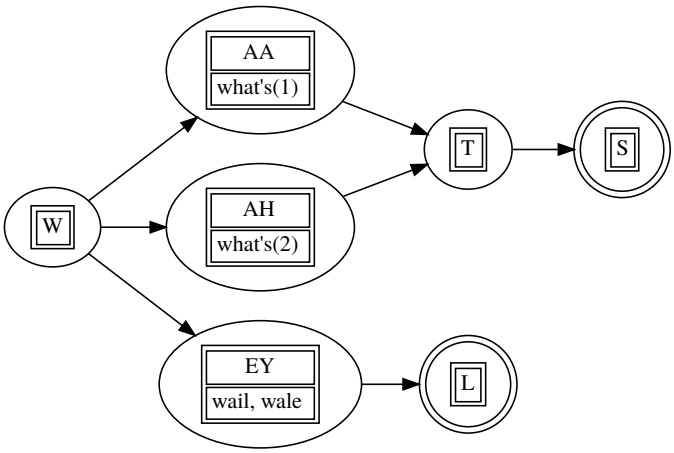

Figure 4.2: Stage 1: Atom graph as created by BioKIT during search network construction. Nodes with a double circle indicate the last node of a token.

Figure 4.2 shows the atom graph, generated from the example dictionary. The words "wail" and "wale" share all their nodes since their pronunciations are the same. Pronunciations for "what's" share all but one node.

In the second stage each node in the previously created graph is expanded based on its context of *Atoms* in the graph. *Atoms* at boundaries of *Token* are also expanded. If the employed *FeatureVectorScorer* is a context-independent model, the graph from stage two is the same as in stage one. Each node is expanded based on its context of *Atoms* in the atom graph. For all nodes which are at the beginning or end of the graph, we take all possible *Tokens* and their respective sequence of *Atoms* as possible successors/predecessors. At this point initial and final nodes of the graph are still not connected to each other.

After the expansion, each node contains an HMM for the *Atom* they are modeling and a representation of its context, refer to Figure 4.3 showing the example expanded graph of *Atoms*. To have a compact representation of contexts of *Atoms* we use a bitmask as proposed by Stoimenov et al. [SM06]. The bitmask is a matrix consisting of context offsets for the columns and all possible phonemes in the rows. Value of a field in the matrix is 1 if the phoneme in that row is possible at the context offset of the column. Initially, a bitmask is computed for every model in the *FeatureVectorScorer* based on the corresponding *ModelMapper* (i.e. decision tree in speech recognition).

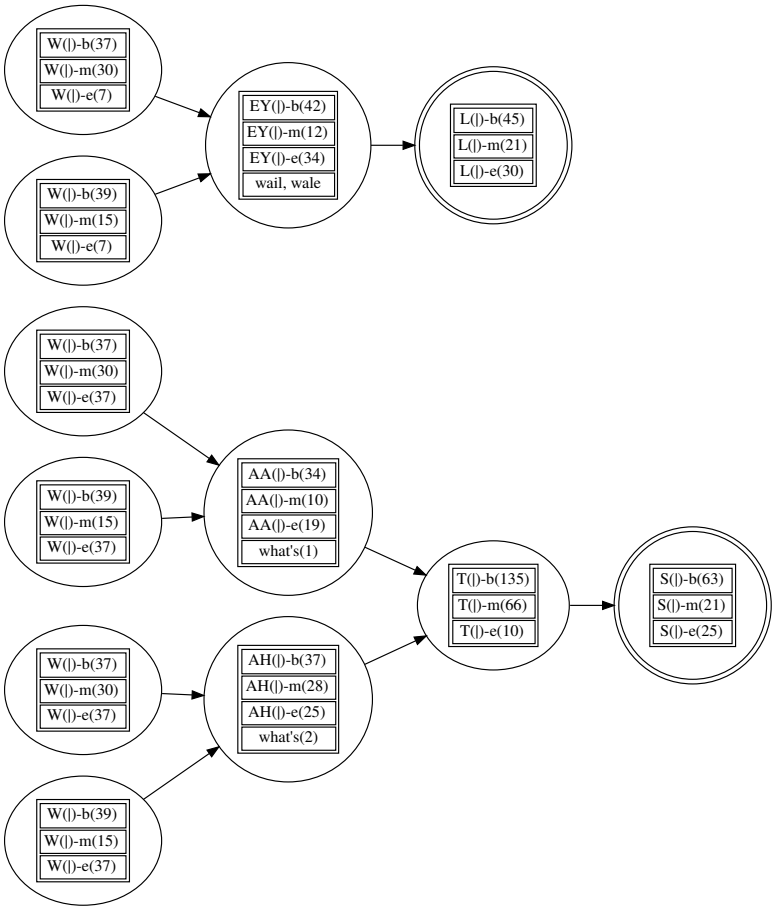

Figure 4.3: Stage 2: Context expanded atom graph as created by BioKIT during search network construction. Each node contains an HMM represented by the identifiers of each state's emission probability density function. Nodes with a double circle indicate the last node of a token.

The bitmask for an atom-level HMM, is then computed by combining the bitmasks of the individual models used in each state by applying the bitwise &-Operator. Table 4.2 shows the bitmask for the node modeling the phoneme "/L/" in the example. In the center, in polyphone position two, only the phoneme "/L/" is possible and thus is the only row with a "1" in that column. The left context of the phoneme are only the phonemes "/W/" and "/EY/", whereas the right context are the first two phonemes of all four words in the dictionary.

In the third stage, each expanded node is replaced by the states of their respective HMM. Afterwards, initial and final nodes are connected based

Table 4.2: *Example of a bitmask for the phoneme "/L/" of the word "wale" in the example shown in Figure 4.3. The acoustic model is a quintphone model.*

	Polyphone position				
Phoneme	0	1	2	3	4
/AA/	0	0	0	0	1
/AH/	0	0	0	0	1
/EY/	0	1	0	0	1
/L/	0	0	1	0	0
/S/	0	0	0	0	0
/T/	0	0	0	0	0
/W/	1	0	0	1	0

on their respective bitmasks. If after applying the bitwise &-Operator to a bitmask of an initial and a final node, a valid context remains in the bitmask then the two nodes are connected to each other (i.e. context is valid if at least one "1" in each column of the resulting bitmask remains).

After connecting final and initial nodes, all initial nodes with no predecessors and all final nodes with no successors are removed from the network. This process is repeated until only nodes with at least one successor and predecessor remain.

Finally, the graph is determinized, all *Token* labels are pushed as far as possible towards the initial nodes of the network, and the graph is minimized. Figure 4.4 shows the final search network constructed for our example, the connections between final and initial nodes have been omitted for clarity.

4.2.5.2 Pruning

Similarly to Povey et al. [PGB+11], absolute and relative pruning parameters are applied. Additional pruning parameters governing partial hypotheses in final states of the search network are used to limit the number of hypotheses in initial states of the network, since language model lookahead is comparably costly to compute. Pruning is applied after each frame:

- Hypothesis topN: only the N-best partial hypotheses are kept.
- Hypothesis beam: the score of each partial hypothesis has to be in beam range of the best partial hypothesis.

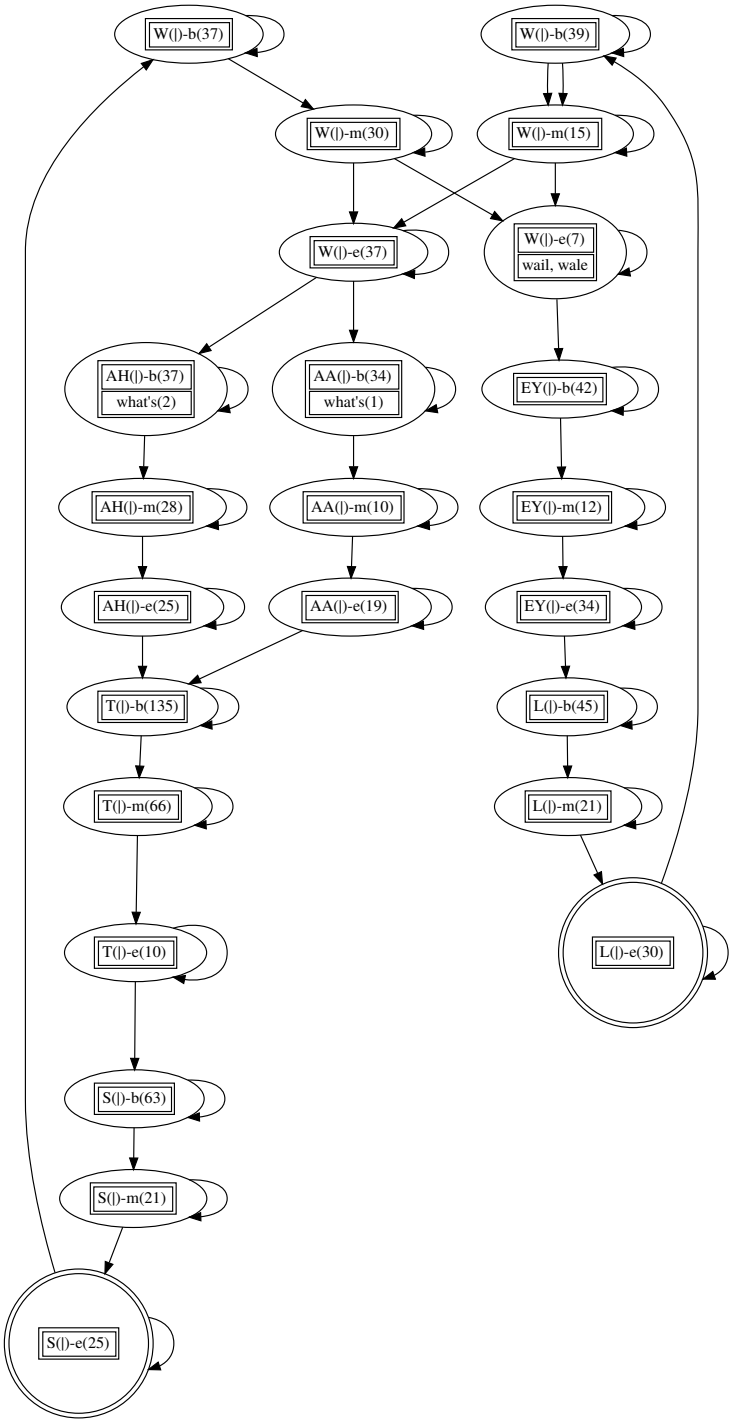

Figure 4.4: Stage 3: Final search network as created by BioKIT. Nodes with a double circle indicate the last state of a token.

- Final hypothesis topN: only the N-best partial hypotheses in a final state are kept.
- Final hypothesis beam: the score of each partial hypothesis in a final state has to be in beam range of the best partial hypothesis.
- Lattice beam: the total number of hypotheses in the lattice are limited, each hypothesis has to be withing beam range of the best hypothesis.

4.2.6 Lattice interface

At any-time during the decoding (in case of online decoding) or afterwards, the decoder can create a lattice, containing all hypotheses of the current decoder output.

Similar to Kaldi [PGB+11], we have two different implementations of the lattice, using different granularity. The *StateLattice* is a one-to-one mapping of the internal decoding state into the lattice format. Each node in the lattice presents a point in time and each edge, connecting the nodes, presenting a transition and state emission in the search network/HMM. The *TokenLattice* is a more compact representation, where the nodes also indicate a point in time, but each edge between nodes contains the state sequence belonging to a single *Token*. Furthermore, the *TokenLattice* is compressed such that each sequence of *Tokens* only appears once in the *TokenLattice*. Figure 4.5 shows an example lattice from the German part of the GlobalPhone database [SVS13].

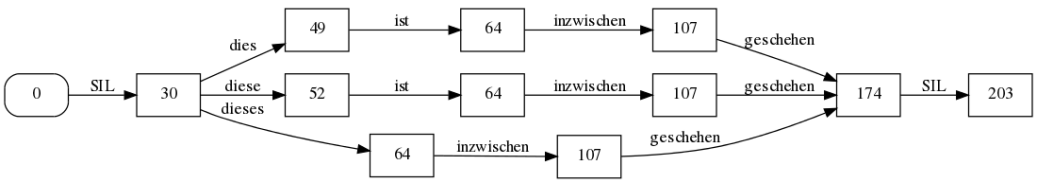

Figure 4.5: TokenLattice of utterance decoded with BioKIT, taken from the GlobalPhone database. Numbers in nodes indicate the current position in terms of time in the corresponding observation sequence.

Both lattices can be used to generate N-best lists containing only the N sequences of *Tokens* with the best score. Furthermore, each lattice can generate confidence scores for each *Token* of a given sequence of *Tokens*.

4.2.7 Parallelization

Similar to other toolkits BioKIT uses a sentence-level parallelization, where each sentence can be processed independently from the other sentences in the set to be decoded. However, our implementation differs from the parallelization used in for example Kaldi [PGB+11] or the JRTK [FGH+97]. While the latter two use sentence-level parallelization, they spawn an independent process for each additional computer core to be used. In contrast, BioKIT has only one process which creates the required number of threads, sharing memory between them. As a result, each abstract class, which is part of the decoding process, requires its implementation to implement a copy method for itself. Thus, thread safety is determined by each implementation individually, thereby reducing the memory overhead by sharing instances of objects if they are thread safe.

This feature is used on the Python layer by initially creating a prototype decoding process, loading all required models, and subsequently calling the copy method on the prototype's instance of the decoder class, returning a thread safe copy of the prototype decoder, sharing thread safe components between copies.

4.3 Experiments

We report on experimental results conducted with our decoder and compare the results to Kaldi [PGB+11] on a subset of the languages from the GlobalPhone database. Furthermore, we investigate the real-time factor necessary to obtain them. We chose Kaldi for our comparison since it offers state-of-the-art acoustic modeling and performance, and is freely available.

4.3.1 System setup

We report results on the Bulgarian (BG), Czech (CZ), German (GE), and Vietnamese (VN) part of the GlobalPhone database [SVS13]. Each system is trained on about 22 hours of read speech with the Kaldi toolkit. We present the results obtained on deep neural network (DNN) acoustic models from [VIP+14] using an fMLLR adaptation. The fMLLR transforms were separately created by Kaldi and BioKIT in a first pass by using Kaldi trained Gaussian Mixture Models. The input for the DNN is a 143 dimensional feature vector consisting of 11 stacked 13 dimensional MFCC vectors. The

DNN is initialized by training on a range of languages from the GlobalPhone database and is then re-trained with a target language training data. The neural network has 5 hidden layers with 1,500 nodes each.

Table 4.3: *Dictionary and language model information of tested GlobalPhone systems.*

language	#words	PPL	#highest N-grams
Bulgarian (BGs)	99,576	386	743k 3-grams
Bulgarian (BGb)	99,576	306	36,300k 3-grams
Czech (CZs)	32,942	1,644	2,906k 3-grams
Czech (CZb)	32,942	1,469	10,640k 3-grams
German (GEs)	37,301	673	990k 3-grams
German (GEb)	37,301	552	26,211k 3-grams
Vietnamese (VNs)	29,820	247	590k 3-grams
Vietnamese (VNb)	29,820	179	17,805k 5-grams

The dictionary sizes and language model (LM) information are given in Table 4.4. The number of highest order N-grams are presented in the table to give an estimate of the size of the model. The appended "s" and "b" indicate small and big language models, respectively. The out-of-vocabulary (OOV) rate for each system is less than 0.01% with the exception of Bulgarian where we have an OOV of 2.18%. For the experiments with BioKIT, the full language model lookahead was used for each system with an unlimited lookahead depth.

In order to limit network size and reduce memory consumption, Kaldi integrates big language models on the fly using the network build on the small LM. To be able to use DNNs in conjunction with our big LMs in Kaldi, we adapted the Kaldi decoder script.

4.3.2 ASR recognition performance

The error rates of our comparative experiment between BioKIT and Kaldi are given in Table 4.4.

The results for Vietnamese are given in syllable error rate (SER), and Bulgarian, Czech, and German in word error rate (WER). The results show that the Kaldi and BioKIT decoder achieve comparable error rates for all systems. Results of BioKIT and Kaldi systems were tested pair-wise for significance. The error rate differences between BioKIT and Kaldi systems were

Table 4.4: *Various GlobalPhone systems and their performance with Kaldi and BioKIT. Significant improvements are marked with * if $p < 0.05$.*

system	Error rate	BioKIT	Kaldi
Bulgarian (BGs)	WER	12.46%*	12.84%
Bulgarian (BGb)	WER	11.67%*	12.16%
Czech (CZs)	WER	8.99%	9.23%
Czech (CZb)	WER	8.66%	8.66%
German (GEs)	WER	10.88%	10.85%
German (GEb)	WER	9.63%	9.76%
Vietnamese (VNs)	SER	8.19%	8.10%
Vietnamese (VNb)	SER	6.97%	7.10%

not significant at a significance level of 0.05 except for the Bulgarian ASR systems.

4.3.3 Real-time factor

The real-time factor of a system sets the duration of the input sequence in relation to the time it takes the recognition system to compute the results. A real-time factor of one indicates that the duration of the input sequence and the required computation time are the same. In general, the real-time factor should be as low as possible. An online-capable system however requires a real-time factor of less than one, to avoid a backlog of unprocessed input data.

Figure 4.6 shows BioKIT's performance in terms of real-time factor over syllable error rate on the Vietnamese systems. Fortunately, the decoding speed of our decoder is not affected by the bigger language model. The decrease in error rate levels off after a real-time factor of about 0.3. Tests were run on an Intel Core i7-3770 with 3.4GHz and 4 cores. All tests were run with 4 threads in parallel. Real-time factor is derived from the sum of the decoding times of all 4 threads.

For the Vietnamese systems, the memory consumption for experiments with a single thread is 1.1GB (VNs) and 2.4GB (VNb). Using our parallelization with 4 threads, memory consumption increases to 3.3GB (VNs) and 4.6GB (VNb). Thus, the memory sharing between threads reduces memory consumption by approximately 25% for the Vietnamese system VNs and by 52% for the VNb system.

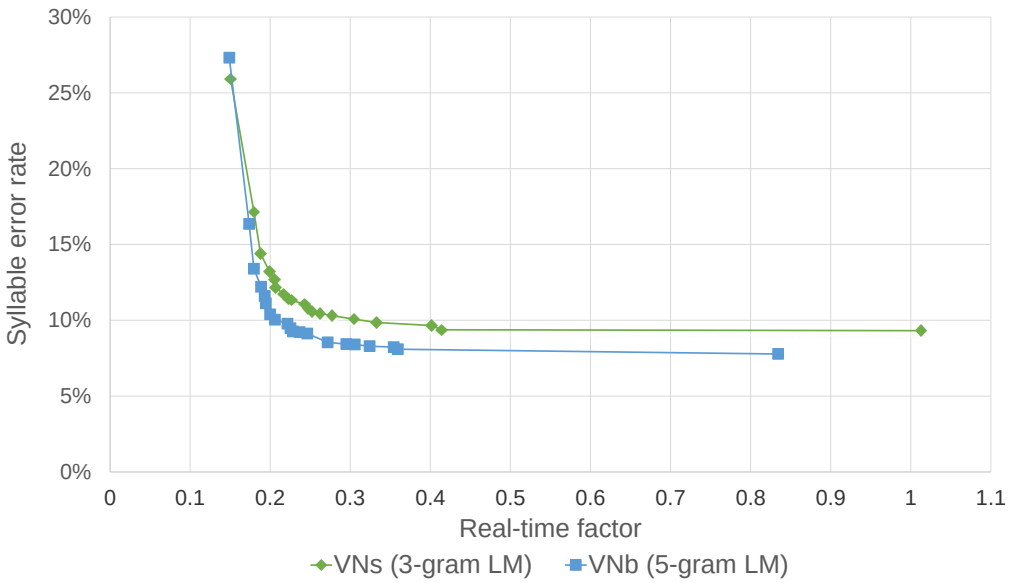

Figure 4.6: Real-time factor over SER for GlobalPhone Vietnamese system with 3-gram LM(VNs) and 5-gram LM(VNb).

4.4 Summary

We introduced our new toolkit BioKIT and gave an overview of its components. We showed that our toolkit is suitable for automatic speech recognition. We reported on first results of our decoder using Kaldi trained deep neural networks achieving comparable results. We showed that usage of large language models has no negative effect on real-time factor over error rate with BioKIT. Furthermore, by employing our parallelization strategy of sharing modules between threads, we are able to reduce the memory requirements significantly.

Chapter 5

Error Signatures

In this chapter we introduce a novel approach to analyzing HMM-based recognition systems. We describe the framework we developed to first accumulate occurrences of ASR errors, then to group these errors into similar clusters. The resulting clusters are used to pinpoint the reason of errors, and to finally manipulate the ASR system's components such that the system will no longer produce these errors. In the beginning of this chapter, the notion of error signatures is introduced. Each signature describes a group of errors using a set of features extracted from them. Subsequently, the algorithm to extract error signatures is outlined. Finally, we show how the obtained error signatures can be used to help an expert to correct errors in the system.

5.1 Motivation

The metric that is used to measure the performance of automatic speech recognition (ASR) systems is the word error rate (WER). It is the edit distance of the reference transcription to the hypothesis of the ASR system computed on a development or an evaluation set. Improving the WER of an ASR system usually requires input from an expert in the field, who has to identify and fix errors in the system. In order to gain the fastest improvements in terms of WER, the expert has to rank errors and their corresponding causes in descending order of frequency, fixing the most prevalent causes of errors first. However, determining which cause of error is the most prevalent

and even linking ASR errors to an error cause is not an easy task, since the WER of an ASR system only reveals the errors but not their underlying error cause.

Recent developments contribute to the difficulties in the analysis of ASR systems: ASR technology is applied to an increasing number of domains, Also, tasks have grown more difficult, combining resources of varying quality and quantity, ranging from smaller corpora [GLF+93] to increasingly difficult tasks [CCR+13], increasing the complexity of systems as a result and making their analysis more difficult (e.g. systems which use several decoding passes over an utterance to obtain the final hypothesis). Especially with the advent of improved algorithms for neural networks and the vast amounts of available training data [SLY11], identifying room for improvement for these systems can be a tedious task. These factors drive the demand for tools to systematically find causes of errors in ASR systems.

One of the most widespread tools that is available to analyze an ASR system is the SCTK toolkit [SCT07], giving sentence-by-sentence evaluations along with word confusion tables. Other research focused on directly assigning errors to specific components of an ASR system [Cha97, NRK99], or investigated the decoding process and the influence of pruning parameters on the accuracy [SSN10]. All available tools mentioned above have drawbacks that we address with our new framework: the above mentioned tools require that reference transcriptions of the data being analyzed is available, whereas we developed a toolkit that works without any additional data beyond the ASR systems decoding output. The approach of Chase [Cha97] analyzes each utterance on its own, therefore does not give any information on which error is prevalent. In our framework we group errors over all available utterances in order to rank the encountered errors by their prevalence and give additional information on the error cause by virtue of the groups of errors found. The SCTK toolkit [SCT07] gives purely statistical information on each word and their confusions and does not relate them to error causes or accounts for the fact that an error cause might impact several words (i.e. acoustic modeling problem of a certain phoneme). With our framework we analyze the encountered errors not only by word identity alone but also include fine-grained attributes. For this purpose we defined *error signatures*.

5.2 Definition of error signatures

Each error cause exhibits particular characteristics, depending on the ASR component the error cause originates from. For example, an error caused

by the language model appears in similar word contexts, a pronunciation problem appears in words sharing the same prefix or suffix phoneme sequence, and an acoustic model problem appears in the same phoneme or phonetic context. To identify these characteristics we use a set of attributes for which we aggregate errors on a training or development set:

- phonemes
- acoustic model ids
- word identity
- word context
- word confidence score
- language id
- language id context

Additional attributes, such as speaker identity, recording setup, speaking style, signal-to-noise ratio or part-of-speech tags, can be added on the fly to the error signature framework, by either supplying attributes and their values for a complete utterance during the collection of statistics on a given data set or by extending the dictionary and adding additional attributes for each word.

Each attribute can assume several different values, depending on the word, its context, and pronunciation. For a given English word "hello", the attribute *word identity* would have the value "hello". The pronunciation of the word is "/h/ /e/ /l/ /o/", the attribute *phonemes* would assume four different values.

In our novel framework, which is described in the next section, we group the attribute-value pairs encountered on the data set with the aim of finding a characteristic set of attribute-value pairs for a specific error. We call each set of attribute-value pairs an *error signature*. An example signature is given in Table 5.1. The signature indicates that if the combination of Mandarin character "的" and acoustic model 1333 are observed, the probability of an error is 0.93. This signature is further discussed in the next section.

The combination of attribute-value pairs of each signature, as in Table 5.1, can give a hint to the probable cause of the errors. Presented with *error signatures* the expert can fix the problems based on their frequency and impact, to rapidly gain improvements in WER.

Table 5.1: *Example of an* error signature *found on the training set of the SEAME corpus. The words on the right hand side of the word confusions column are the words in the 1-best hypothesis. Number of occurrences of the signature in the corpus are given along with the probability that the signature indicates an error (*ErrorRatio*).*

Sig.#	ErrorRatio	#Occ.	Error signature	Word confusions
1	0.93	120	AM No. 1333 0 =的	auntie → "and 的" novelty → "normal 的"

In the case that reference transcriptions are available we follow the approach proposed by Chase [Cha97]. The correctness of a word is not just based on the edit distance between reference and hypothesis: a correct word has to align to a matching word in the reference and must fall into a similar region of the audio file. The requirement of falling into the same region in the audio file can be relaxed by allowing for a deviation in overlap of the reference and hypothesis region. In addition, any word that is in the case of N-grams within $N-1$ words of an erroneous word is considered a part of the same error.

5.3 What information is available

This section presents the output of the *error signature* framework presented in this chapter, to make the remaining sections more accessible.

Table 5.1 gives an example of an *error signature* derived from the SEAME training set. Each signature is assigned a unique signature id to identify it more quickly in a sentence-by-sentence overview of the analyzed data set. In addition, the *ErrorRatio*, matching number of regions (#Occ.), and the signature, with its unique combination of attribute-value pairs, are displayed. The signature in Table 5.1 consists of two attribute-value pairs: acoustic model with id "1333" in combination with the word "的". To make sense of what the acoustic model in question is modeling, a list, giving a mapping between acoustic model ids and modeled phonemes, is given. In the example, acoustic model with id "1333" is modeling the phoneme "/d/". Mapping between acoustic model ids and phonemes can be retrieved from the *ModelMapper* in BioKIT. Furthermore, a list of word confusions is given, for the expert to spot patterns in the confusions.

Additional information is given in a sentence-by-sentence output of the analyzed data set. The output indicates the regions that where matched by a specific *error signature*. An example for a sentence output is given in Table 5.2. The example shows a segment from the SEAME corpus. In the rows of the table is given region-by-region information on the duration and location in number of frames, and acoustic model, language model, and total scores. In addition, signatures matching a region are indicated in the last row of the table for the expert to quickly find samples for a specific signature.

Table 5.2: *Segment from sentence from the training set of the SEAME corpus. Table shows component scores for reference (R) and hypothesis (H) of regions in the sentence. The signature id from Table 5.1 is indicated in the last cell. Scores in table are negative log-likelihoods.*

Frames	104–152	152–157	157–184	184–317	317–409
Reference	没有	[particle]	就	问	<SIL> auntie
Hypothesis	没有	[particle]	就	问	<SIL> and 的
TotalCost (R)	-13.06	1.41	-6.49	-11.87	-11.00
TotalCost (H)	-13.06	1.41	-6.49	-11.43	-15.15
AmScore (R)	-20.59	-1.30	-11.83	-53.74	-28.87
AmScore (H)	-20.59	-1.30	-11.83	-53.90	-33.49
LmCost (R)	7.54	2.71	5.33	41.87	17.87
LmCost (H)	7.54	2.71	5.33	42.46	18.34
Signatures					Sig. #1

5.4 Error signature algorithm

The process of finding *error signatures* is related to concept discovery in the field of data-mining [VMG04]. In data-mining concepts are discovered by finding patterns of jointly occurring attributes and deducing relationships between them. Transferring this concept to ASR, we are interested in finding a set of attribute-value pairs which jointly appear in erroneous words in an ASR hypothesis and are only rarely found in correct words.

The procedure of finding *error signatures* on a data set is separated into two steps. In the first step, erroneous and correct words in the hypothesis are marked and attribute-value pairs are assigned to each word. In the second step attribute-value pairs are clustered into *error signatures* for subsequent

presentation to the ASR system's expert. Subsection 5.4.1 describes the first step, while 5.4.2 introduces the second step of the *error signature* algorithm.

5.4.1 Collecting statistics of attributes

The first step to obtain *error signatures* is the assignment of attribute-value pairs to all words in the data set and the accumulation of statistics on all such attribute-value pairs. The process is divided into three parts. In the first part an alignment of reference and hypothesis has to be computed such that correct and erroneous words can be identified. In the second part, attributes are assigned to correct and erroneous words. In the third part, statistics on words and their attribute-value pairs are accumulated for each sentence.

5.4.1.1 Reference and hypothesis alignment

For each utterance in the data set to be analyzed, we require the utterance's forced alignment of the reference transcription to its audio file and the 1-best hypothesis from the decoding process. The algorithm to compute the alignment, as proposed by Chase [Cha97], is described in Section 2.3.1.

Table 5.3: *Example alignment of hypothesis and reference from the SEAME corpus. Table shows component scores for reference (R) and hypothesis (H) of regions in the sentence. Scores in table are negative log-likelihoods.*

Frames	0 – 70	70 – 83	83 – 205
Reference	<SIL> area five	的	total 是 <SIL>
Hypothesis	因为 area five	的	都 都是 <SIL>
TotalCost (R)	2.96	1.21	-15.17
TotalCost (H)	-9.02	1.21	-18.52
AmScore (R)	-20.02	-3.48	-39.58
AmScore (H)	-31.03	-3.48	-40.19
LmCost (R)	22.98	4.69	24.41
LmCost (H)	22.02	4.69	21.67
Error category	AM_LM_OVERWH.		AM_LM_OVERWH.

Table 5.3 contains an example of an utterance taken from the SEAME corpus [LTCL10]. The alignment algorithm separates the sentence into three regions. Each segment is referred to as a region with the first and third

region containing an error and therefore are called *ErrorRegions*, see Section 2.3.1. In the example, we used a trigram language model to decode the sentence. Hence, the error in the first and third region affects the succeeding two words which are therefore included in the same region. These succeeding words are included in the same *ErrorRegion* even if they have been correctly recognized.

Correctly recognized words, which are not affected by any errors in their vicinity, are in a region of their own. Combining sequences of correctly recognized words to one region and subsequently assign attribute-value pairs to whole regions would be detrimental to the computation of *error signatures*. An utterance, which has been correctly recognized, would therefore only consist of one region and would contain attribute-value pairs from the whole utterance and as a result match a multitude of *error signatures*, where the attribute-value pairs might stem from unrelated words in the beginning and end of the utterance and thus no longer be a meaningful correct example for these signatures. In contrast, correct regions consisting of only one word would only be assigned attribute-value pairs concerned with the word itself and its surroundings.

5.4.1.2 Assigning attributes to aligned segments

After every sentence is separated into regions we proceed to assign attribute-value pairs to each word in the hypothesis. Table 5.4 gives an excerpt of the attribute-value pairs assigned to the hypothesis segment "的" in region two of the example sentence in Table 5.3. The phonemes in the table are derived from the characters pronunciation in the dictionary. Since the word was correctly recognized the error category is "CORRECT", to indicate no error. All other categories, as proposed by Chase [Cha97], are presented in Section 2.3.2. The same definition of regions was chosen, since they do capture both acoustic modeling errors by using a temporal alignment, as well as language modeling errors by padding regions with neighboring words, if they are affected.

The reason that attribute-value pairs are assigned to words and not to regions lies with the clustering algorithm. The goal of the clustering algorithm is to find signatures that maximize the ratio of *ErrorRegions* to all matched regions. If attribute-value pairs were assigned to regions as a whole, the biggest difference between regions with and without errors would be the number of words in a region, since only regions with no error consist of one word. *ErrorRegions*, in case of an N-gram language model, would contain

Table 5.4: *Attributes for* 的 *from second region of utterance in Table 5.3.* -1 *and* $+1$ *refer to the preceding and succeeding word respectively. Different attribute-values are separated by "|".*

Attributes	Values
Word	的
Phonemes	/d/ | /i/
Acoustic models	1490 | 3081 | 2886 ...
Word context	-1=five | +1=都
Word confidence score	0.32
Language id	0=MAN
Language id context	-1=ENG | +1=MAN
Word confusion	-
Error category	CORRECT

at least $N-1$ words. Thus, the clustering algorithm would try to construct signatures which can only appear in regions with more than one word. For example, an *error signature* consisting of Mandarin and English phonemes could only appear if a region contains at least two words (one Mandarin, the other English). As a result, the signature would no longer be useful in identifying groups of errors.

Table 5.5: *Attributes for* 都是 *from third region of utterance in Table 5.3.* -1 *and* $+1$ *refer to the preceding and succeeding word respectively. Succeeding word is silence with no language id. Different attribute-values are separated by "|".*

Attributes	Values
Word	都是
Phonemes	/d/ | /o/ | /w/
Acoustic models	2548 | 1059 | 1183 ...
Word context	-1=都 | +1=<SIL>
Word confidence score	0.4
Language id	0=MAN
Language id context	-1=MAN | +1=NONE
Word confusion	是 → 都是
Error category	AM_LM_OVERWH.

In addition, to assigning attribute-value pairs on a word basis, the assignment of attribute-value pairs for correct words differs from the assignment for

erroneous words. The example given in Table 5.4 depicts the attribute-value pairs assigned to a correctly recognized word.

Table 5.5 shows the attribute-value pairs assigned to a word which was misrecognized. While the correct pronunciation of the character sequence "都是" in Table 5.5 is *"/d/ /o/ /w/ /S/ /i/"*, only the first three phonemes *"/d/ /o/ /w/"*, are assigned as attribute-value pairs, since they were confused with other phonemes. The remaining phonemes *"/S/ /i/"* are not added as attribute-value pairs so that the clustering process has to focus on problematic attribute-value pairs.

As a consequence, attribute-value pairs in *error signatures* indicate the problematic portions of the matching words. In the example in Table 5.4 and Table 5.5, "是" is confused with "都是". The Mandarin characters are assigned the error category acoustic model and language model overwhelm (AM_LM_OVERWHELM), since the components that dominate the score difference between reference and hypothesis for this error are both the acoustic model and the language model. The attributes "word confusion" and "error category" are excluded from the clustering process and are only added as an additional source of information for the expert reviewing the *error signatures*.

5.4.2 Computing error signatures

After attribute-value pairs have been assigned to each word in each hypothesis and their statistics have been collected, we employ a bottom-up greedy clustering algorithm to find the *error signatures* which best explain the errors we encounter in our data set.

A prerequisite for the clustering process is the ability to compare two *error signatures* and decide if one signature is a better fit for the observed errors than the other. In data mining two measures are used to compare two concepts with each other. They are a concept's support and confidence. The concept's support is the number of hypothesis regions matching the *error signature*, which will be referred to as the number of occurrences of an *error signature*. The confidence in an *error signature* is derived from the ratio of *ErrorRegions*, to all regions matching that signature:

$$\begin{aligned} ErrorRatio(S) &= P(Error|S) \\ &= \frac{|\{errorRegion|S \in errorRegion\}|}{|\{region|S \in region\}|} \end{aligned} \tag{5.1}$$

where S defines a set of attribute-value pairs and corresponds to a single *error signature*, $errorRegion$ denotes an *ErrorRegion* and $region$ is any region in the data set being profiled. We refer to the confidence in an *error signature* as the *ErrorRatio*. It can be interpreted as the probability that a hypothesis word with that signature was misrecognized.

The greedy clustering algorithm starts with one *error signature* for every attribute-value pair whose number of occurrences exceeds a threshold. New *error signatures* are created by taking the list of attribute-value pairs from previous *error signatures* and add an additional attribute-value pair to each list. Criteria for these newly created *error signatures* are:

1. An *error signature* must cover a minimum number of *ErrorRegions* (min. number of occurrences)
2. *ErrorRatio* has to exceed a certain threshold (min. *ErrorRatio*)
3. There is no *error signature* which consists of a subset of attribute-value pairs of the *error signature* in question and has a higher *ErrorRatio* (removal of generic *error signatures*)
4. There is no *error signature* which represents the same group of *ErrorRegions* and has a higher *ErrorRatio* (removal of dominated *error signatures*)

The clustering process stops if no new *error signature* can be created without violating one of the previously mentioned criteria. To speed up the clustering, all attribute-value pairs are processed in ascending order of occurrences. Processed in such a way attribute-value pairs with low occurrence count can be processed quickly since their *error signatures* quickly fall below the required minimum number of occurrences.

The greedy recursive algorithm is presented in Algorithm 1. The algorithm starts with the procedure *computeSignatures*, which calls the recursive function *expandSignature*. Initially, a signature S is created for each attribute-value pair, which is then recursively used to create new signatures by expanding S until one of the above criteria is violated. The removal of dominated *error signatures* and signatures with an *ErrorRatio* below the threshold is performed at the end of the *computeSignatures* procedure. The function *computeErrorRatio* calculates and returns the result of Equation 5.1.

The algorithm to obtain the *error signatures* is integrated into our BioKIT toolkit [TWG+14]. The benefit of integrating the algorithm with BioKIT is that it can be easily integrated into any existing experimental BioKIT setup.

A stand-alone tool however would require the user to conform to a specific input format.

Algorithm 1 Error signature clustering algorithm

```
1: procedure computeSignatures(minOcc, minErrorRatio,
   attrValuePairs, regions)
2:     signatures = []
3:     for attrValuePair in attrValuePairs do
4:         S = createErrorSignature(attrValuePair)
5:         regions_S = computeMatches(S, regions)
6:         occurrences_S = |regions_S|

7:         if occurrences_S > minOcc then
8:             addToSignatures(S, signatures)
9:             expandSignature(S,minOcc,attrValuePairs,regions,signatures)
10:    removeSignaturesViolatingCriteria(minErrorRatio, signatures)
11:    return signatures

12: procedure expandSignature(S, minOcc, attrValuePairs,
    regions, signatures)
13:    regions_S = computeMatches(S, regions)
14:    ratio = computeErrorRatio(S, regions_S)
15:    for attrValuePair in attrValuePairs do
16:        S_exp = createErrorSignature(S, attrValuePair)
17:        regions_exp = computeMatches(S_exp, regions)
18:        occurrences_exp = |regions_exp|
19:        ratio_exp = computeErrorRatio(S_exp, regions_exp)

20:        if occurrences_exp > minOcc and ratio_exp > ratio then
21:            addToSignatures(S_exp, signatures)
22:            expandSignature(S_exp,minOcc,attrValuePairs,regions,signatures)
23:    return
```

5.4.3 Discussion of the clustering algorithm

The choice of clustering algorithm crucially impacts the results of any clustering task. Several requirements have to be met by a clustering algorithm to be useful in the context of *error signatures*:

- The clusters derived from the algorithm need to be interpretable: clusters should indicate an error and by themselves contain information on the error.
- The clustering algorithm needs to handle discrete features: each attribute-value pair is either present or absent in a word.
- The algorithm needs to handle high dimensionality: the number of attribute-value pairs on the SEAME corpus is $8,973$ and $24,810$ on the ILSE corpus.
- The vectors characterizing the *error signatures* are sparse: only a few attribute-value pairs are used to describe an error.

In general, the complexity of the problem is such that not all possible *error signatures* can be analyzed. Given a set of N attribute-value pairs, the total number of possible *error signatures* amounts to $2^N - 1$. Therefore, the evaluation of all *error signatures* is not feasible and suitable heuristics have to be employed.

The goal in data-mining applications is to find meaningful clusters of data points which are closer to each other with regards to a distance measure than to other data points not belonging to a particular cluster. Standard algorithms use for example the Jaccard coefficient to determine the similarity between two data points consisting of attribute-value pairs similar to our case. The Jaccard coefficient between two data points T_1 and T_2 is defined as:

$$Jaccard(T_1, T_2) = \frac{|T_1 \cap T_2|}{|T_1 \cup T_2|} \tag{5.2}$$

where $|T_1 \cap T_2|$ is the number of common attribute-value pairs and $|T_1 \cup T_2|$ is the total number of attribute-value pairs of T_1 and T_2. Several other algorithms based on a similarity measure on data points exist that are introduced in a comprehensive overview of clustering algorithms used in data-mining by Berkhin [Ber06]. These algorithms are unsuitable for our problem since, the created clusters are not necessarily interpretable and the quality of the created clusters is hampered by the high dimensionality of our feature space. Furthermore, these algorithms do not account for our actual optimization criterion for clustering to find clusters with a high *ErrorRatio*.

Co-clustering algorithms which do not only cluster data points but also cluster attribute-value pairs are more adequate [Ber06]. A related algorithm used to cluster text documents is quite similar to our problem [BM98]. Given a

set of documents and class labels the goal is to increase the classification accuracy of new documents to their corresponding class using a naive Bayes approach. The attribute-value pairs (in this case words) are chosen such that the probability of the correct class label is maximized.

Due to the employed greedy algorithm, we are potentially not able to detect the best *error signatures* with regards to the *ErrorRatio* on a data set. This is true for any *error signature* which consists of $M + 2$ attribute-value pairs and of which any subset of attribute-value pairs of size $M + 1$ has a worse *ErrorRatio* than any possible subset of attribute-value pairs of size M. It would be possible that for example only a combination of three or more phoneme attribute-value pairs are indicative of an error but any subset of two phonemes would not. The problem is mitigated by the usage of context-dependent acoustic models, since they depend on several preceding and/or succeeding phonemes.

5.4.4 Discretization of continuous attributes

The clustering algorithm we employ to find the *error signatures* uses only discrete features, testing for the presence or absence of an attribute-value pair. In order to make use of continuous-valued attributes, such as word confidence scores, the value range has to be discretized before clustering. We chose to use the minimum entropy partitioning algorithm introduced by Fayyad et al. [FI93]. This greedy algorithm recursively chooses the partition of the value range with the biggest decrease in entropy. The algorithm starts with one partition containing the complete value range and proceeds to partition the value range until either the information gain of an additional partition or the occupancy count of the resulting partitions fall below a threshold. The above described algorithm was chosen since it performed best against several other partitioning algorithms on a range of partitioning tasks [DKS95].

5.5 Error correction

After *error signatures* have been computed it is up to the expert to scan the list of signatures, and deduce which component of the system could be responsible for producing the observed errors in the hypotheses. Furthermore, the expert has to conclude if the problem can be fixed by modifying a component of the ASR system and, if that is the case, which changes are most likely to fix the encountered problem. An approach which an expert

might follow to improve an ASR system using *error signatures* is presented in Section 5.5.4.

To keep the required amount of manual work to a minimum, the *error signatures* presented to the expert have to be limited to a meaningful selection. Therefore, we employ heuristics to filter the computed *error signatures*. The data for the figures in this section are derived from the training set of the SEAME corpus using the baseline system trained on the same set, which is presented in Section 6.1. The total number of *error signatures* without filtering, with the exception of a minimum number of matching regions per signature (min. support) as defined in the *error signature* algorithm in Section 5.4, is $27,615$.

5.5.1 Error ratio filter

Figure 5.1 depicts all *error signatures* and their corresponding *ErrorRatio*, as they were found in the training set of the SEAME corpus. It shows that only a relative small group of *error signatures* exceed a threshold of 0.5. Since the signatures need to be handled by an expert, they need to be significant, i.e. each signature should have a high *ErrorRatio*. By applying a threshold of 0.5 to the ErrorRatio of each error signature, the number of error signatures which are of interest to an expert for error correction are reduced from $27,615$ to 717.

A more detailed overview of different *ErrorRatio* thresholds is given in Figure 5.2. Increasing the threshold to 0.7 further decreases the number of selected signatures to 316. Higher *ErrorRatio* thresholds in conjunction with the frequency filter, introduced in the next section, would leave too little signatures to present to the expert. As a result a threshold of 0.7 was chosen in the following experiments on *error signatures*.

5.5.2 Frequency filter

Since the number of error signatures that can be viewed by an expert should be limited to a select meaningful few, the potential impact of each error signature on the error rate of the system should be as big as possible. Thus, any *error signature* selected for further investigation has to represent a high number of errors of the data set being analyzed. Figure 5.3 shows all *error signatures* sorted by their number of matching errors in the training set. A suitable filter threshold can be derived from the amount of data the *error*

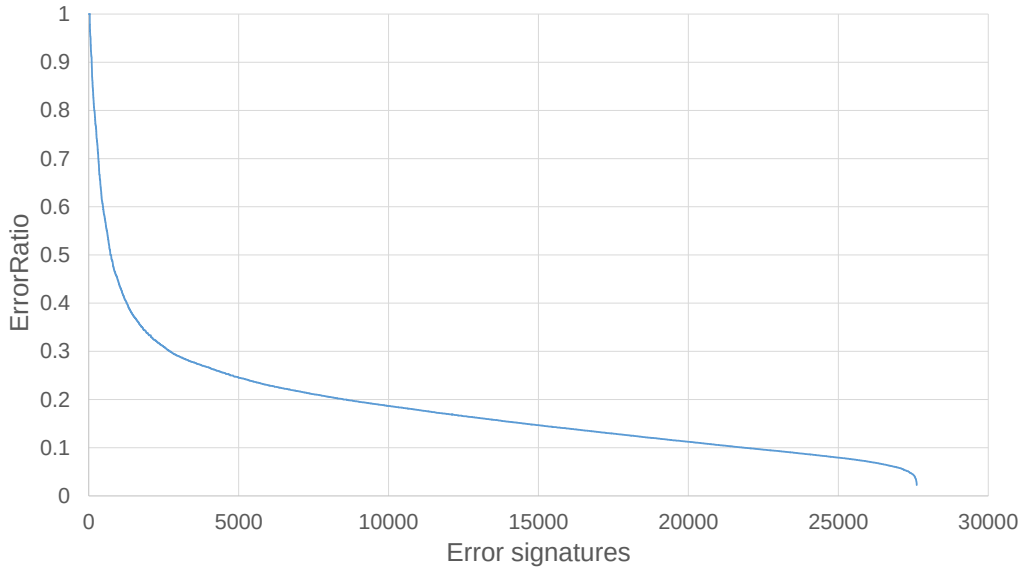

Figure 5.1: *Error signatures* on the training set of the SEAME corpus sorted by their *ErrorRatio*.

signatures are estimated on, requiring *error signatures* with a high impact to match a percentage of the data. The total number of words in the training set amounts to 681,348 distributed over 48,040 utterances. Since a change to a word will also impact its surrounding words in an utterance, we require *error signatures* to have an impact on a minimum number of utterances from the chosen data set.

Figure 5.4 shows different filter thresholds applied to the *error signatures* extracted on the SEAME training set. Each threshold is given as a percentage of affected utterances in the training set. For example, limiting the minimum number of affected utterances to at least 0.2% would reduce the 27,615 *error signatures* to 4,456.

Choosing a higher threshold would result in signatures which affect a higher number of utterances. However, an increase in affected utterances leads to less specific *error signatures*, potentially representing more than one error cause or no error cause at all. Figure 5.5 depicts the combination of frequency filter and chosen *ErrorRatio* filter of 0.7. The figure shows that selecting *error signatures* with matching errors exceeding 1% of training data would result in four signatures whereas a lower threshold of 0.2% of affected utterances still leads to 23 signatures. As a result, a threshold of 0.2% was chosen.

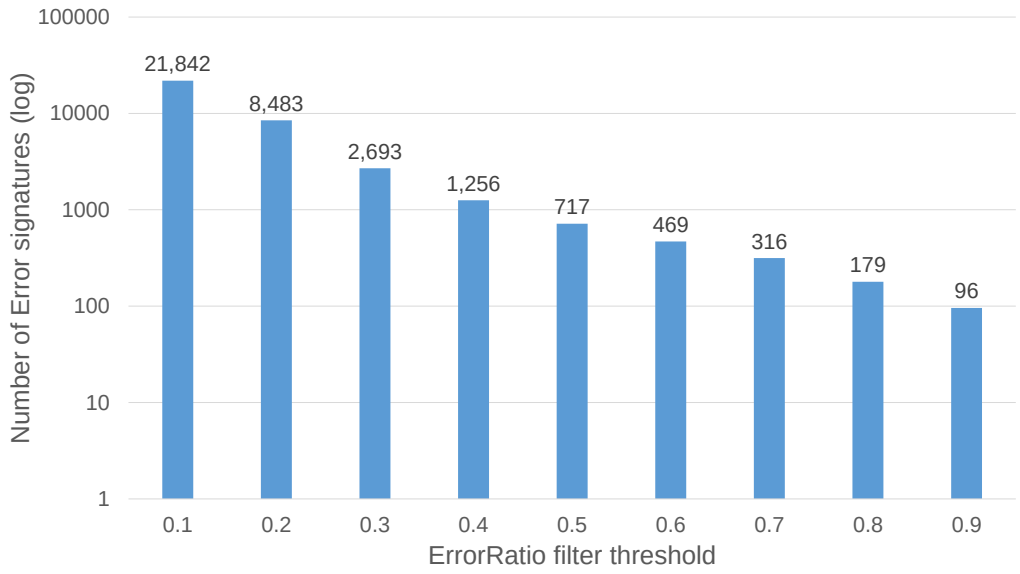

Figure 5.2: Different thresholds for the *ErrorRatio* and the number of *error signatures* which exceed the threshold based on the signatures depicted in Figure 5.1. Y-axis is in logarithmic scale.

5.5.3 Cluster similar error signatures

The algorithm to compute *error signatures* checks for "dominated" *error signatures*. An *error signature* $e1$ is dominated if there exists another signature $e2$ whose matching regions are a super-set of the regions matching $e1$ with $e2$ having a lower *ErrorRatio*. If the matching regions of two *error signatures* are neither subset nor super-set, the algorithm will keep both signatures even though their underlying error cause might be the same. As a remedy, we use a heuristic to detect similar *error signatures* and bundle them together to present them as a group to the expert.

To find similar *error signatures* we use a hierarchical agglomerative clustering, where at each step of the algorithm the two most similar *error signatures* are merged, until no pair of signatures with a similarity exceeding the chosen similarity threshold remain. We define similarity as the ratio of regions which match both signatures to all regions matched by any of the two signatures. An example of an *error signature bundle* consisting of two signatures is shown in Table 5.6. In the example signature #1 is more specific than signature #2, requiring an additional character (然) in the previous token compared to it being a Mandarin character sequence (-1=MAN). Nonetheless, both signatures point to the same errors, which can been seen in the

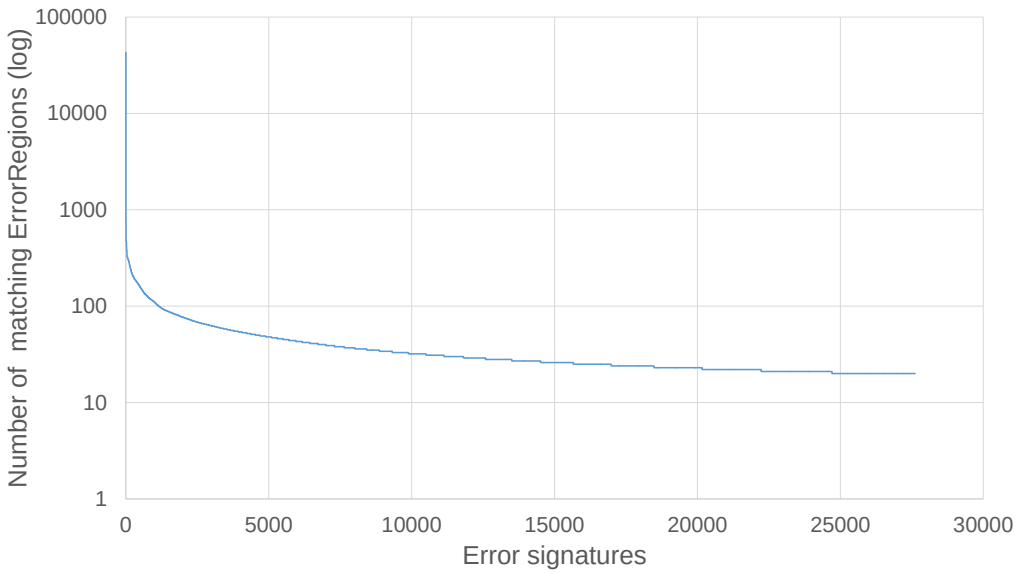

Figure 5.3: Error signatures on the training set of the SEAME corpus sorted by the number of matching ErrorRegions. Y-axis is in logarithmic scale.

column "word confusions" of the table.

Table 5.6: *Example of bundled* error signatures *found on the training set of the SEAME corpus. The words on the right hand side of the word confusions column are the words in the 1-best hypothesis.*

Sig.#	ErrorRatio	#Occ.	ErrorSig.	Word confusions
1	0.82	125	-1=然	然后 → "然后 [particle]"
			-1=后	[noise] → "然后 [particle]"
			0=[particle]	<SIL> → "<SIL> [particle]"
2	0.71	169	-1=MAN	然后 → "然后 [particle]"
			-1=后	[noise] → "然后 [particle]"
			0=[particle]	<SIL> → "<SIL> [particle]"

Figure 5.6 shows the number of *error signature bundles* after applying varying thresholds for the similarity measure. The lower the required treshold for signatures to be considered similar the more signatures are bundled together.

Applying varying similarity thresholds to signatures computed on the SEAME corpus further reduces the 23 signatures to 16 at a similarity threshold of 0.8, see Figure 5.7. The choice of threshold was conservatively high, since the example only contains few examples and clustering results did not

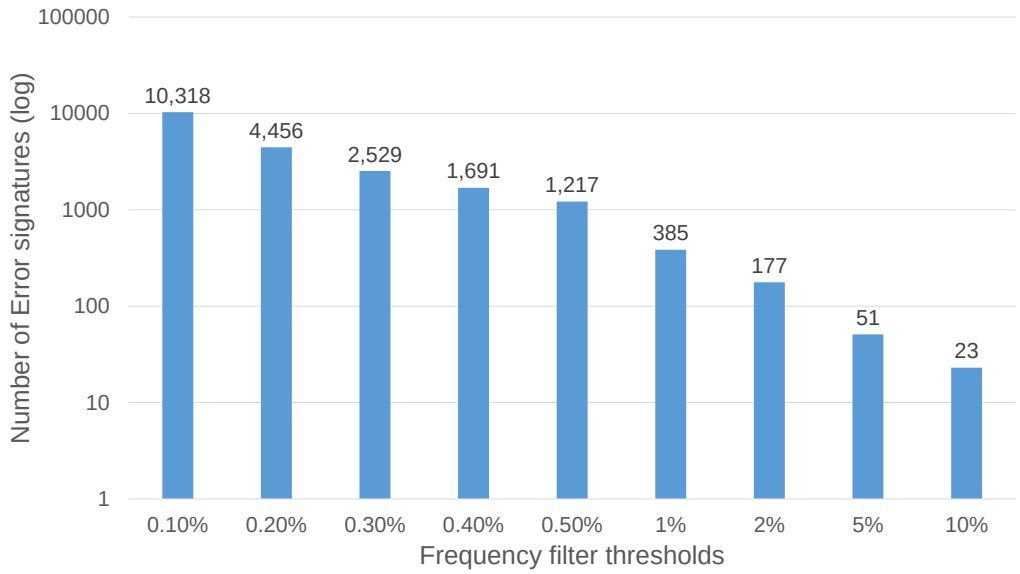

Figure 5.4: Different thresholds for minimum number of matching Error-Regions as a percentage of affected training set utterances based on the signatures depicted in Figure 5.3. Y-axis is in logarithmic scale.

change between a threshold of 0.8 and 0.7.

5.5.4 Identifying underlying causes

After applying the filter steps, as described in the preceding sections, the remaining *error signatures* and *error signature bundles* are sorted by their *ErrorRatio* and number of matching *ErrorRegions* and are presented to an expert. This section will give a recipe on how the information from the *error signatures* can be exploited.

Due to the multitude of possible error causes, any recipe to arrive at an error fix from an *error signature* is of a general nature and requires an expert in the field of speech recognition to review the signatures. Typically, errors encountered with a speech recognizer can be blamed to one of three components: the acoustic model, the language model, and the pronunciation dictionary. Errors which are caused by environmental noise or errors in the preprocessing were not found or fixed with the proposed approach. Types of errors are described in the sections of Chapter 6. To identify errors concerning the acoustic model, language model, and pronunciation dictionary we found that the following steps lead to the identification of the underlying

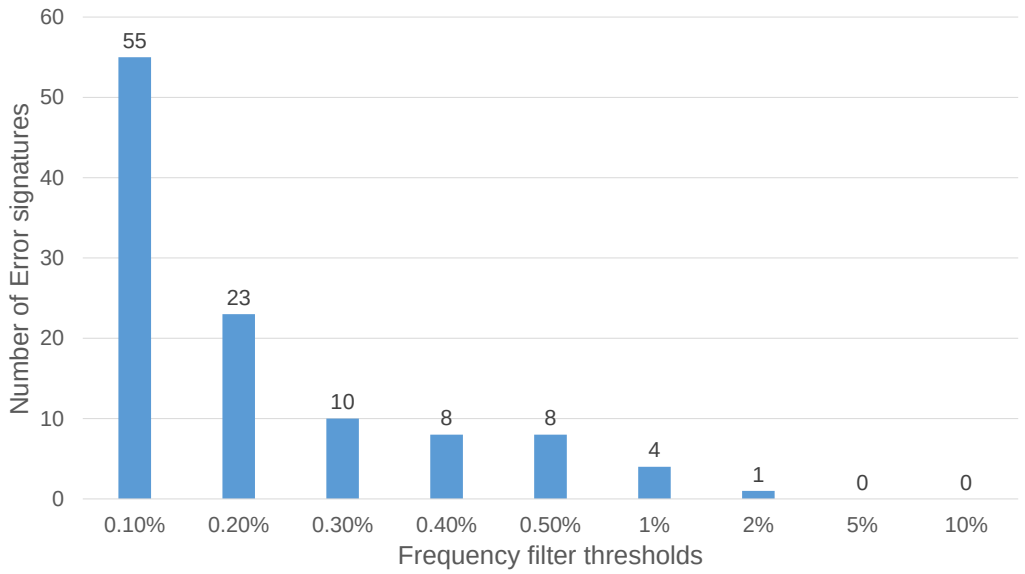

Figure 5.5: Different thresholds for minimum number of matching ErrorRegions as a percentage of affected training set utterances based on the signatures depicted in Figure 5.3. Additionally, enforcing a threshold of 0.7 on the *ErrorRatio*.

cause of an *error signature*:

1. Listen to examples matching the *error signature* being analyzed.
2. Compare Viterbi path of 1-best hypothesis and spectrogram of audio file to look for systematic mismatches in the given examples: A tool which we used for this purpose is Wavesurfer by Sjölander et al. [SB00].
3. Review the pronunciations of both hypothesis and reference words from the word confusions, to find systematic mispronunciations: especially in the case of automatically generated pronunciation dictionaries, using a grapheme-to-phoneme model. A list of word confusions with phonetically similar prefixes or suffixes is a strong indicator of a pronunciation problem.
4. Review N-grams of hypothesis and reference examples to look for irregularities: While word context attributes are an indicator for a language model problem, knowledge of the language being analyzed is necessary to be able to identify and fix problems in the language model.

In general, not all of these steps have to be looked at in order to derive the error and its error fix. For one, the *error signature* itself can be a strong in-

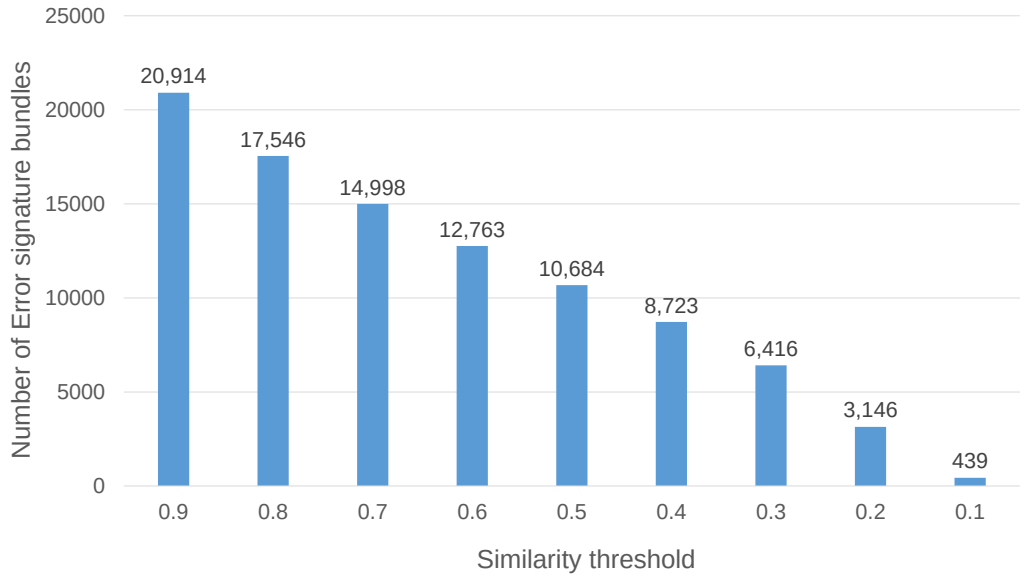

Figure 5.6: Number of *error signature bundles* after clustering with different similarity thresholds.

dicator for the error cause, since word identity, phoneme, and acoustic model attributes are only assigned to erroneous words if the concerned word/phoneme/acoustic model has been confused. Thus, with regard to the example signature given in Table 5.1, the combination of acoustic model with id 1333, modeling phoneme */d/*, and the phonetically similar sounding word confusions indicate either an acoustic model or pronunciation dictionary problem.

Once the cause of the error has been identified, fixing the problem relies on the knowledge of the expert, but a few general points can be given:

- **Acoustic model problem:** Fixing the alignments in the problematic segments and retraining the acoustic model can help to alleviate the problem. For example, in case of erroneous boundaries between speech and non-speech segments, the application of a voice activity detection component can result in a better Viterbi alignment.
- **Pronunciation dictionary:** In case of systematic errors with the grapheme-to-phoneme model, rules can be derived to fix the problematic pronunciations in the dictionary or other approaches such as the pronunciation extraction from internet sources, such as Wiktionary can be used to help fix problematic word pronunciations [SOS10].
- **Language model problem:** Excluding problematic sentences from

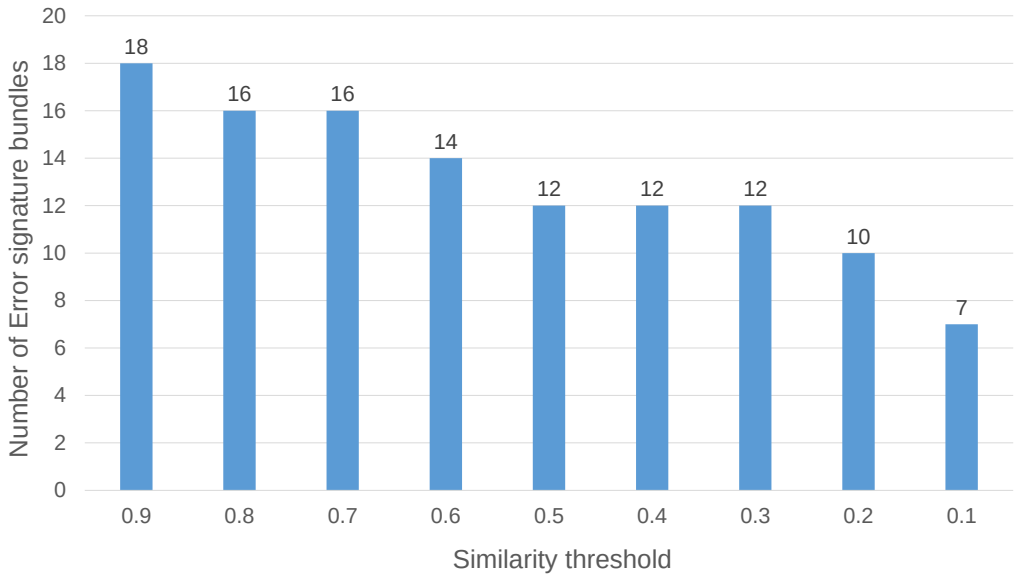

Figure 5.7: Number of *error signature bundles* after clustering with different similarity thresholds. Additionally, an *ErrorRatio* filter of 0.7 and a frequency filter of 0.2% of affected training set utterances are used.

language model training or searching for additional text resources [SGVS13] and subsequently re-estimating the language model can help to reduce the impact of the problem.

5.6 Summary

In this chapter we introduced a novel framework to group errors made by an ASR system on a specific data set. We defined the term *error signatures* and showed how they can be extracted from the decoding results of an ASR system. Furthermore, the algorithm to extract *error signatures* was described in detail.

We presented methods to filter the extracted *error signatures* and as a result reduced the effort asked of an expert viewing the signatures. As an example, we used the *error signatures* obtained from the training set of the SEAME corpus and were able to reduce the signatures from 27,615 to 19. Experiments on the remaining signatures are shown in Section 6.1. Finally, we showed what information is available to the expert as a result from our approach and how the information can be used to derive error cause and

fix from an *error signature*. The impact of *error signature* extraction and subsequent error fixing is described in the next chapter.

Chapter 6

ASR System improvements with Error Signatures

In this chapter we present the experiments we conducted using our error signature *framework introduced in the previous chapter. First, we present an initial experiment applying the* error signature *framework to the SEAME corpus using the reference transcriptions, to show the feasibility of our approach. Second, we extend our algorithm to obtain* error signatures *without relying on reference transcriptions. And third, we show that we are able to improve ASR systems on the SEAME and ILSE corpus without any reference transcriptions. For this purpose we used* error signatures *derived from unsupervised training. Finally, we discuss the overall usefulness of our approach with regards to the manual work that is needed to make use of the* error signature *framework.*

6.1 Supervised system improvements

To show that improving an ASR system using *error signatures* is feasible, we chose to conduct an initial experiment on the SEAME code-switching corpus presented in Section 3.2. For this purpose we first train a baseline system whose setup is described in Section 6.1.1. Afterwards, we extract *error signatures* on the training and development set of the SEAME corpus. Finally, we derive error causes from the computed *error signatures*, implement fixes

for the error causes, and compare the resulting system to our baseline.

6.1.1 System setup

The baseline system we aim to improve is a multi-pass system trained with the Kaldi toolkit [PGB+11]. The last pass is a deep neural network (DNN) acoustic model using a feature-space maximum likelihood linear regression transformation (fMLLR) [Gal98] estimated on the decoding results of the previous pass. The fMLLR transformations were created in a first pass by using Kaldi trained Gaussian Mixture Models. The input for the DNN are stacked MFCC feature vectors, which have been transformed with an LDA transform and stacked again after the transform has been applied to receive the final 440-dimensional feature vector. The preprocessing from audio input to the acoustic model state posterior probabilities is summarized in Figure 6.1.

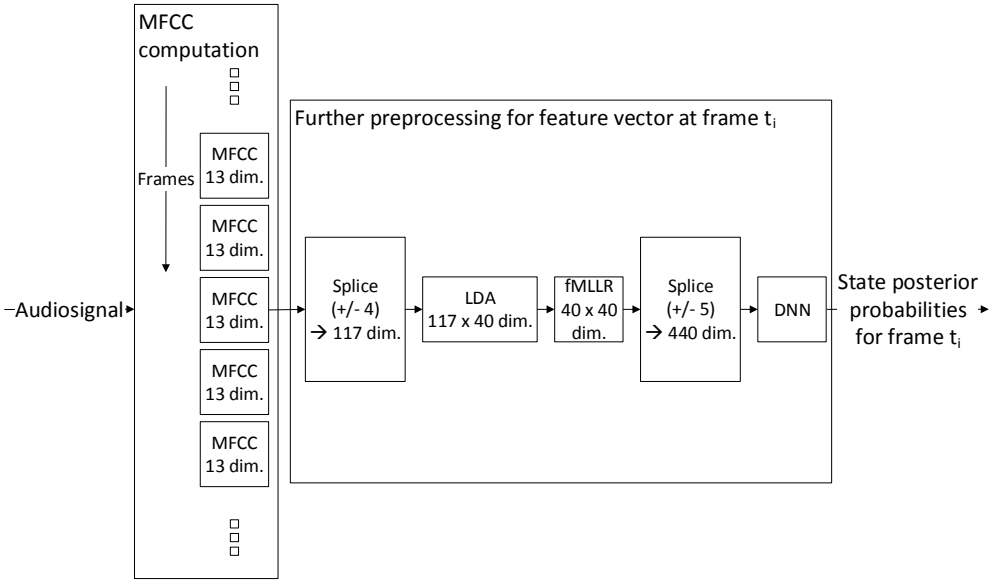

Figure 6.1: Preprocessing used for the last pass of the SEAME multi-pass system. Splice function is stacking +/- x frames of the preceding preprocessing step.

The DNN is trained by first performing pre-training, followed by 13 iterations optimizing cross-entropy and finally four iterations of state-level minimum Bayes risk (sMBR) sequence training as described by Veselỳ et al.[VGBP13]. The neural network consists of 6 hidden layers with 2,048 nodes each and an output layer with 3,194 nodes.

The employed language model is a Kneser-Ney 3-gram estimated on the transcriptions of the SEAME training text and interpolated with language models built on monolingual texts. Perplexity on the development and evaluation set are 268.39 and 282.86 respectively. Decoding results were obtained using BioKIT [TWG+14]. We used the mixed error rate (MER) as an evaluation measure [VLW+12]. In this measure English words are treated the same as in the word error rate calculation, but Mandarin character segments are split into their individual characters. Thus, enabling us to potentially compare error rates across different segmentations of Mandarin.

The baseline system achieves a mixed error rate of 30.61% on the development set and a mixed error rate of 24.50% on the evaluation set. This system is a strong baseline as it improves by 18.3% relative the MER on the evaluation set compared to the former best system reported using the same language model and by 16.1% relative compared to our former best overall system as reported in [AVK+13].

6.1.2 Discovered error signatures

Due to the limited size of the development set we computed the *error signatures* on both training and development set. Since the language model was estimated on the training set transcriptions, the ASR system is expected to produce different errors on the training set compared to the development set. Therefore, the computation of the signatures is conducted separately for training and development set.

Table 6.1 gives a summary of the error categories assigned to the regions in the training and development set, showing that, as expected, the relative amount of errors on the training set is lower compared to the development set. The small amount of search errors on the development set of 0.9% indicates that gains by increasing the pruning parameters will be marginal. Therefore, sizable gains in recognition accuracy will only be achieved by fixing problems in the ASR models.

A selection of *error signatures* as a result of our clustering process are presented in Table 6.2 and Table 6.3. Each row in the tables corresponds to a single *error signature*. Signatures #1 and #2 are taken from the training set and signatures #3–5 from the development set. The structure of the tables is the same as in the example given in Table 5.1.

The following list describes the causes for each *error signature* presented in Table 6.2 and Table 6.3 and the steps we took to fix them. In Section 6.1.3

Table 6.1: *Error categories of regions in the training and development set.*

Category	Dev set		Train set	
Acoustic model	16.5%	(1,705)	5.9%	(23,862)
Language model	9.8%	(1,013)	3.6%	(14,346)
Homophone	2.5%	(254)	1.5%	(5,982)
Search error	0.9%	(92)	0.4%	(1,574)
Correct	70.4%	(7,288)	88.7%	(358,226)
Total	100%	(10,352)	100%	(403,990)

Table 6.2: *Examples for* error signatures *found in the training set of the SEAME corpus.*

Sig.#	ErrorRatio	#Occ.	Error signature	Word confusions
1	0.93	120	AM No. 1333 0=的	auntie → “and 的” novelty → “normal 的”
2	0.78	1398	-1=<SIL> 0=[particle]	<SIL> → [particle] <NOISE> → [particle]

we then describe the impact on the MER.

- **Signature #1:** Character “的” (pronunciation: /d/ /i/) is hypothesized. In addition, acoustic model with id “1333”, modeling phoneme /d/, is being confused with another acoustic model based on the alignment with the reference. Confusions show that phoneme /t/ of the pronunciations for words such as “auntie” and “novelty” is being confused.
  Solution: To alleviate this problem additional pronunciation variations for English words with pronunciations ending in “/t/ /i/” were added. The resulting pronunciations for the word “auntie” are now: the already existing pronunciation of “/ao/ /n/ /t/ /i/”, and the new

Table 6.3: *Examples for* error signatures *found in the development set of the SEAME corpus.*

Sig.#	ErrorRatio	#Occ.	Error signature	Word confusions
3	1.0	20	0=so	说 → so
4	0.97	73	0=就说	就是说 → 就说
5	0.95	38	0=okay	ok → okay k. → okay

pronunciation "/ao/ /n/ /d/ /i/".

- **Signature #2:** Frequently "[particle]" ("0=[particle]") was recognized after silence ("-1=<SIL>"). By inspecting the training dictionary we discovered that one of the noise tokens was erroneously mapped to the same acoustic model as "[particle]".
  Solution: To solve the problem we changed the pronunciation for the noise token and retrained the acoustic model.

- **Signature #3:** A specific pronunciation variation of the word "so" was wrongly recognized.
  Solution: The problematic pronunciation variation of "so" ("/s/ /ao/") was removed from the dictionary, leaving only the pronunciation "/s/ /o/".

- **Signature #4:** Investigating the confusion of the segment "就说" we found that in audio examples which fit the signature the middle character of "就是说" is not pronounced.
  Solution: To fix the error we added a pronunciation for "就是说" omitting the phoneme sequence of "/S/ /i/" for the middle character "是".

- **Signature #5:** This signature indicates a normalization problem of the word "okay". The word appears in several versions in the development set: "okay", "ok", and "o. k.".
  Solution: We fixed this problem by normalizing the word "okay" in the reference transcriptions and re-estimated the language model.

6.1.3 Impact on error rate

In order to verify the usefulness of the applied fixes we retrained and tested the updated system. Results are given in Table 6.4. We present separate results for the retrained model just fixing signature #2 from Table 6.2 and for all signatures fixed. An individual system just fixing signature #2 was trained due to its possible high impact on the error rate. In all cases the correction of errors lead to MER gains. While the error rate differences are significant between all systems on the development set with a value of $p < 0.001$, error rate gains on the evaluation set are not significant ($p = 0.15$) using the matched pair sentence segment test [GC89], implemented in the SCTK toolkit [SCT07]. The reason is that the discovered *error signatures* are found and sorted based on their impact on the respective set which they are extracted on. Thus, they are not guaranteed to generalize to other sets.

Table 6.4: *Mixed error rates of baseline system and systems after fixed derived from* error signatures *have been applied. In addition, relative improvements for each set to the baseline (S0_sup) are given. Significant improvements are marked with * if $p < 0.05$ and ** if $p < 0.01$.*

System	Dev set	(Rel. gain)	Eval set	(Rel. gain)
Baseline (S0_sup)	30.35%	(0%)	24.25%	(0%)
Signature #2 fix (S1_sup)	29.62%**	(2.4%)	24.10%	(0.6%)
All signatures fix (S2_sup)	**29.24%****	**(3.7%)**	**23.98%**	**(1.1%)**

6.1.4 Summary

We showed in an initial experiment that we can find useful *error signatures* and derive error fixes from them. We were able to improve the performance of our state-of-the-art baseline system by 3.66% and 1.11% relative on the development and evaluation set respectively. Furthermore, we found that *error signatures* do not generalize well, if the underlying data being analyzed contains only small amounts of data, in the case of the development set, or only few errors, as is the case for the training set. Our experiment with supervision was conducted as a first test if ASR systems can be improved using *error signatures*. As a result, only a few *error signatures* on the baseline system were investigated.

6.2 Unsupervised training scenario

Much of the related work on the topic of system analysis and error assignment need supervision in form of reference transcriptions. Our first implementation of the *error signature* algorithm relies on the availability of reference transcriptions as well. However, large-scale ASR systems trained today are based on thousands of hours of speech without reference transcriptions [MHM+14]. Usually, new data is transcribed by an already existing ASR system and the resulting automatic transcriptions are used for training a new ASR system as described in [LHL14]. Confidence models are employed to exclude potentially erroneous segments from the training data [LHL14]. In order to be able to analyze ASR systems without reference transcriptions, we have to extend our *error signature* algorithm.

In this section we will explain how we extended our framework. We present the results from unsupervised model training in combination with the knowl-

edge derived from the signatures on the SEAME corpus. Furthermore, we describe what the shortcomings of our implementation are and propose how to fix them. Finally, we give the results of the latest *error signature* algorithm in relation to our first implementation and point out their differences. Further experiments applying our improved algorithm to the SEAME and ILSE corpus are shown in the next section.

6.2.1 Computing error signatures without references

The algorithm to obtain error signatures, which was introduced in Section 5.4, requires a forced alignment of an ASR system's hypothesis with the corresponding reference transcriptions. To accommodate the more frequent scenario of a large data corpus with no reference transcription, the algorithm has to be modified. The two following sections explain the changes we made to our framework.

6.2.1.1 Assigning attribute-value pairs

We used the definition of *ErrorRegions* as introduced by Chase [Cha97] in our work. Since no reference transcriptions are available, the computation of *ErrorRegions* is no longer possible. Therefore, in the modified framework we chose that each word in the hypothesis becomes a region of its own. An example segmentation of a hypothesis for the unsupervised case is given in Table 6.5.

Table 6.5: *Example of segmentation of hypothesis into regions from the SEAME corpus.*

Frames	0–21	21–48	48–70	70–83	83–103	103–197	197–205
Hypothesis	因为	area	five	的	都	都是	<SIL>

Table 6.6 contains attribute-value pairs assigned to a word from the hypothesis presented in Table 6.5. All the information is derived from the hypothesis not from the reference. Since we no longer know which word was misrecognized and as a result which attribute-value pairs are being confused, we simply add all possible attribute-value pairs to each word. Thus, instead of adding three phonemes *"/d/, /o/, /w/"* as in Table 5.5, for the character sequence "都是", we add all five phonemes *"/d/, /o/, /w/, /S/, /i/"*. The segment is preceded by a Mandarin word (-1=MAN) and specifically

preceded by the character “都” (-1=都). Furthermore, the word is assigned the language id “MAN” for Mandarin.

Table 6.6: *Attributes for character sequence “都是” from segment in Table 6.5. −1 and +1 refer to the preceding and succeeding word respectively. Succeeding word is silence with no language id. Different attribute-values are separated by “|”. Information is derived from the hypothesis only, since no reference is available.*

Attributes	Values
Word confidence score	0.4
Phonemes	/d/ | /o/ | /w/ | /S/ | /i/
Acoustic model id	2548 | 1059 | 1183 | ...
Word	都是
Word context	-1=都 | +1=SIL
Language id	0=MAN
Language id context	-1=MAN | +1=NONE

6.2.1.2 Adapting the error signature algorithm

To determine the quality of an *error signature* in our original framework we used the *ErrorRatio*, defined in Equation 5.1, as the probability that the signature indicates an error. However, without reference transcriptions the distinction between regions with and without errors is no longer possible. Thus, we can no longer compute the *ErrorRatio* in its original form. We adapt the equation to estimate the *ErrorRatio* based on the probability that each matching region is erroneous. To estimate erroneousness we use the word confidence score, derived from the word lattice of an ASR system. We assume that a low word confidence score indicates an error. Thus, we extend our *ErrorRatio* as follows:

$$\begin{aligned} ErrorRatio(S) &= P(Error|S) \\ &= \frac{\sum_{region|S \in region} (1 - Confidence(region))}{|\{region|S \in region\}|} \end{aligned} \tag{6.1}$$

where S is an *error signature* and $region$ denotes any region/word in the data set being analyzed. $Confidence$ refers to the word confidence score, which is extracted from the word lattice of an ASR system for every hypothesis

as described by Wessel et al. [WMS98]. The *computeErrorRatio* function used in lines 14 and 19 of the algorithm described in Algorithm 1 has to be adapted to use the modified equation presented in Equation 6.1. The remainder of the *error signature* algorithm can be used in the same way as if reference transcriptions were available.

Since no reference transcriptions are available, the attribute "word confusion", comparing reference and hypothesis can no longer be extracted as in the original framework. Thus, the expert can no longer rely on the word confusions as a source of information to derive the error cause from. As a remedy we compute probable word confusions, derived from the lattice. We compute possible word confusions by comparing the best hypothesis word confidence scores to word confidence scores of other, less likely, hypotheses. Possible word confusions are with those words which have a higher word confidence score than the corresponding word in the best hypothesis.

6.2.2 Experiments on SEAME

To verify if our approach generates useful *error signatures* without reference transcriptions, we performed experiments on the SEAME corpus. For this purpose we trained a new ASR system on a subset of the training data and used the remaining data as an unsupervised training set on which we could compute the *error signatures*. To determine if the improvements derived from our signatures are significant, we trained three new systems.

The first is the new baseline system *S0_unsup* which is only trained on the 19 hours of transcribed training data. The second system *S1_unsup_noES* uses the 19 hours of transcribed data and in addition uses a subset of the data from the 39 hours of automatically transcribed data. The subset was selected by computing the confidence score of each utterance and only select utterances for training whose confidence exceed a threshold, as described by Laurent et al. [LHL14]. The third system *S1_unsup_ES* uses the same training data as *S1_unsup_noES*, but also is improved by applying error fixes derived from our *error signatures*.

If our *error signatures* are useful, we expect that the error rate of system *S1_unsup_ES* is lower than the error rate of system *S1_unsup_noES*. The following sections explain the experiment in more detail and report on our results.

6.2.2.1 System setup

The setup of the baseline system *S0_unsup* is the same as in Section 6.1.1. The system is trained only on the 19 hours of the supervised training set as opposed to the previous system in Section 6.1, which is trained on the complete 60 hour training set. The training set was split as described in Section 3.2.1. The baseline language model is a Kneser-Ney trigram estimated on the supervised training set's text and is interpolated with monolingual English and Mandarin texts using the SRILM toolkit [Sto02]. The perplexity is 349.6 and 356.6 on the SEAME development set and evaluation set respectively. The perplexity on the unsupervised training set is 413.9. Decoding results were obtained using BioKIT [TWG+14].

6.2.2.2 Baseline

In a first step, the baseline system *S0_unsup* trained on 19 hours of speech is used to decode the unsupervised training set of 39 hours. The mixed error rate (MER) on that unsupervised training set is 37.85%. The MER on the development and evaluation set is 35.74% and 29.60% respectively. This is deterioration of 23.4% relative compared to the best system trained on all data in the previous experiment in Section 6.1. The degradation in performance is caused by the smaller amount of training data, which both impacts the acoustic model and the language model. Perplexity of the language model in *S0_unsup* rose by 30% on the development set compared to the language model of *S0_sup*, which is estimated on all transcriptions of the full 60 hour training set. The MER deterioration on the unsupervised training set compared to the development set and evaluation set is also caused by the language model, which has a 18% higher perplexity on the unsupervised training set compared to the development set.

Figure 6.2 shows the confidence distribution in deciles over the unsupervised training set. The confidence of each utterance in the unsupervised training set is computed based on the geometric mean of the confidence of each word in the utterance's hypothesis [LHL14]. It shows that only 7.8 hours of the 39 hours of data exceed a confidence threshold of 0.8, increasing the amount of data of both *S1_unsup_noES* and *S1_unsup_ES* to almost 27 hours. A threshold of 0.8 was chosen since Laurent et al. [LHL14] achieved good results for both GMM and DNN based acoustic models.

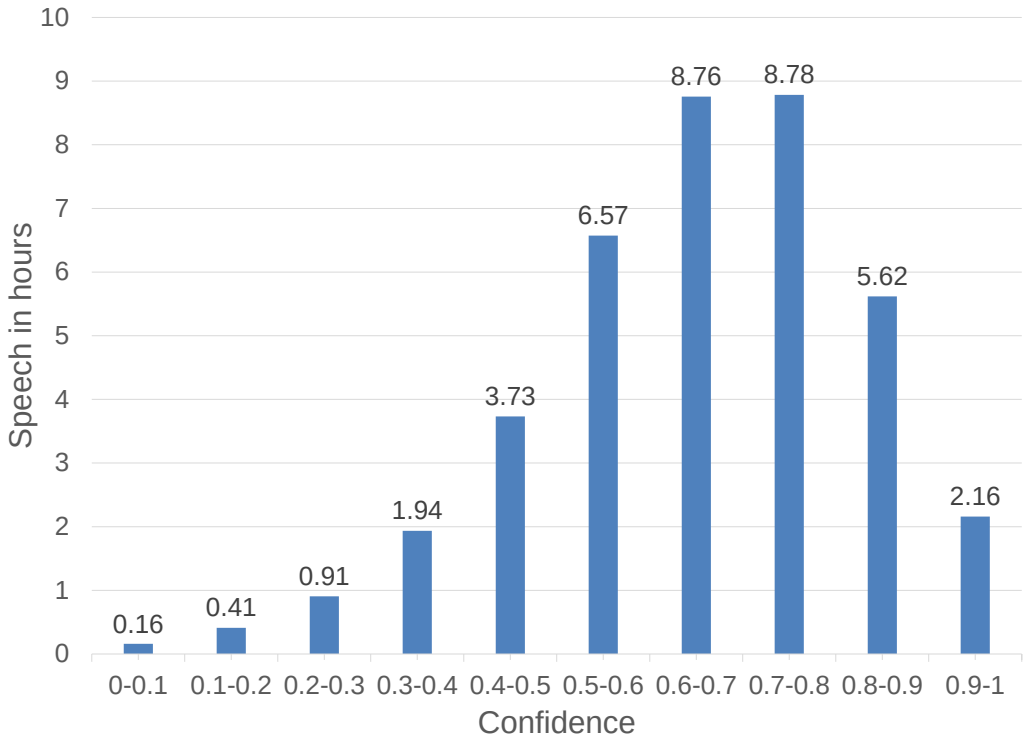

Figure 6.2: Unsupervised training data split into decile confidence bins after decoding and utterance-level confidence estimation.

6.2.2.3 Finding error signatures

After decoding the unsupervised training set, we apply the error signature algorithm and filter the resulting error signatures according to the steps described in Section 5.5. Only 4 signatures remain by applying the same thresholds for filtering as described in Section 5.5. The lack of signatures is a major drawback of the implementation of the modified *error signature* algorithm and will be addressed in Section 6.2.3.

From the 4 *error signatures* we picked the three *error signatures* with the highest *error ratios* for further analysis. The selected *error signatures* are shown in Table 6.7 and the following list describes the steps we took to improve the model, based on these *error signatures*:

- **Signature #1:** Investigating the dictionary, we found that pronunciations for "like" and "kite" where "/l/ /ay/" and "/k/ /ay/" respectively. The cause for these faulty pronunciations was due to an error in

Table 6.7: *Examples for* error signatures *found on the unsupervised training set of the SEAME corpus. Word on the right hand side of word confusion's column is the word in the 1-best hypothesis. Attribute "min. dur." refers to minimum duration of the specific token.*

Sig.#	ErrorRatio	#Occ.	Error signature	Word confusion
1	0.79	124	+1=</s> AM no. 274	的 → kite direct → like
2	0.76	147	+2=</s> 0="而已"	yen → "而已" "来了" → "而已"
3	0.75	382	min. dur. [noise]	"现在" → [noise] center → [noise]

the automatic generation of pronunciations which also affected other words in the vocabulary, which ended in plosives such as /t/, /k/, and /p/.
Solution: Faulty pronunciations, such as "/l/ /ay/" and "/k/ /ay/" for "like" and "kite", were only present in the decoding dictionary and not found in the training dictionary. Therefore, in a first step, we only kept pronunciations of English words in the decoding dictionary which are already present in the training dictionary. Since not all faulty pronunciations could be removed this way, we counted frequencies of all words that appeared during decoding of the 39 hour unsupervised training set in a second step. Afterwards, we applied a threshold to how often each pronunciation of each word had to appear relative to the total number of appearances of the respective word. After scanning the list of word appearance counts, we set a threshold of 15% of how frequently each pronunciation had to appear to remain in the final decoding dictionary. The number of entries in the decoding dictionary was reduced from $52,225$ to $35,645$ while keeping the vocabulary size fixed.

- **Signature #2-#3:** No specific cause could be identified for signatures #2 and #3 by following the approach described in Section 5.5.4, possibly due to a lack of knowledge of the Mandarin language.
  Solution: Utterances matching #3 were excluded from unsupervised acoustic model training. Utterances matching signature #2 were truncated to exclude the last two words. The remainder of each utterance is added to the training data, if the utterance-level confidence exceeds the threshold of 0.8.

6.2.2.4 Results

To estimate the impact of our fixes we compare *S1_unsup_noES* and *S1_unsup_ES*. Mixed error rates of the systems are presented in Table 6.8 and Table 6.9 for the development and evaluation set, respectively.

Table 6.8: *Mixed error rates on the development set, comparing S1_unsup_noES and S1_unsup_ES on the SEAME corpus. Significant improvements are marked with * if $p < 0.05$ and ** if $p < 0.01$.*

System	S1_unsup_noES	S1_unsup_ES	(Rel. gain)
AM retrained	34.65%	34.65%	(0%)
AM & LM retrained	34.21%	34.51%	(-0.9%)
AM & LM retrained, dict fix	-	**33.95%**	**(0.76%)**

Table 6.9: *Mixed error rates on the evaluation set, comparing baseline system with system using error signatures on the SEAME corpus. Significant improvements are marked with * if $p < 0.05$ and ** if $p < 0.01$.*

System	S1_unsup_noES	S1_unsup_ES	(Rel. gain)
AM retrained	28.22%	28.52%	(-1.1%)
AM & LM retrained	28.24%	28.06%	(0.6%)
AM & LM retrained, dict fix	-	**27.44%****	**(2.8%)**

The impact of selecting different amounts of data for unsupervised acoustic modeling and language modeling adaptation based on signatures #2-#3 led to differences in performance on the development set. These differences were not significant on the development set at a significance level of 0.05. Only changing the decoding dictionary had a noticeable impact on system performance. By applying the fixes to the dictionary we achieved a significant reduction in MER ($p = 0.009$) from 28.24% on the adapted baseline system to 27.44% when using error signatures.

6.2.3 Shortcomings of implementation

A major drawback of the current implementation is the limited number of *error signatures* that are presented to the expert. Even though some of the signatures could be used to derive fixes from them to significantly improve the ASR system's performance, changes have to be made to the algorithm to obtain more signatures, as described in this section.

Compared to the supervised computation of *error signatures*, the current unsupervised algorithm cannot differentiate between correct and erroneous words, due to the lack of reference transcriptions. As a result, all possible attribute-value pairs of a word are assigned to it regardless if the word was misrecognized or not, leading to a less focused clustering.

6.2.4 Improving error signature algorithm without references

We want the *error signatures* and their attribute-value pairs to be as informative as possible, to not only indicate commonalities between errors, but also help to deduce the error cause. For this purpose we assigned attribute-value pairs in the supervised case depending on whether the region in question contained any errors or was correctly recognized. Since this information is not available in the unsupervised case we chose to assign confidence scores to each attribute-value pair instead, indicating how likely they are to be correct.

Table 6.10: *Phoneme-level confidence for character sequence "都是" of utterance in Table 6.5.*

Phoneme	Confidence
/d/	0.80
/o/	0.45
/w/	0.45
/S/	1.0
/i/	1.0

Table 6.10 shows phoneme-level confidence scores of the character sequence "都是" from the utterance given in Table 6.5. The reference of the sentence states that the last character "是", with pronunciation "/S/ /i/", are correctly recognized. However, in the current implementation, a signature only containing the phoneme attribute-value pair "/S/" or "/i/" would attribute a confidence score of 0.4 to this word, since all confidence scores are computed on the word-level. Table 6.10 indicates that the phoneme-level confidence score would assign a higher confidence to the correctly recognized phonemes "/S/" and "/i/" than to the erroneous phonemes.

Therefore, we change the algorithm to compute *error signatures* such that attribute-value pairs can have individual confidence scores assigned to them. Therefore, the computation of the *ErrorRatio* is not dependent on the word-level confidence score, but rather on the confidence scores assigned to each

attribute-value pair of a matching region/word. Thus, the equation for the *ErrorRatio* from Equation 6.1 is adapted to:

$$\begin{aligned} ErrorRatio(S) &= P(Error|S) \\ &= \frac{\sum_{region|S \in region} (1 - Confidence(region, S))}{|\{region|S \in region\}|} \end{aligned} \tag{6.2}$$

where the confidence computation is now depending on the region/word and the specific *error signature*, instead of on the region/word alone as in Equation 6.1. The *computeErrorRatio* function used in lines 14 and 19 of the algorithm described in Algorithm 1 has to be adapted to use the improved equation presented in Equation 6.2. In order to compute the confidence score we employ the generalized mean as defined, here:

$$M_p(x_1, \ldots, x_N) = (\frac{1}{N} \sum_{i=0}^{N} x_i^p)^{\frac{1}{p}} \tag{6.3}$$

where M_p is the generalized mean depending on the parameter p, which can be freely chosen, and x_i is the confidence score of the i-th attribute-value pair of the *error signature* for a specific region/word. In our experiments we chose p to be $-\infty$. By choosing p in such a manner the equation of the generalized mean simplifies to:

$$M_{-\infty}(x_1, \ldots, x_N) = \min(x_1, \ldots, x_N) \tag{6.4}$$

Other choices for p would have been feasible as well, for example choosing $p = 0$ would result in the geometric mean. The geometric mean would have resulted in higher confidence scores for $Confidence(region, S)$ in Equation 6.2 compared to the minimum. However, we chose $p = -\infty$, based on work conducted by Souvignier et al. who compared several methods to derive utterance-level confidence scores and showed that the minimum performed best in their experiments [SW99].

6.2.4.1 Experiments on SEAME with improved algorithm

To verify if the changes to the signature computation had a positive impact, we re-computed the *error signatures* on the unsupervised training set of the

SEAME corpus to compare the resulting signatures to the experiment in Section 6.2.2.

Table 6.11: *Examples for* error signatures *found on the unsupervised training set of the SEAME corpus with the improved* error signature *algorithm. Word on the right hand side of word confusion's column is the word in the 1-best hypothesis.*

Sig.#	ErrorRatio	#Occ.	Error signature	Word confusion
1	0.96	118	+2=</s>, /R/ AM no. 1905	comply → 而已 生日 → 而已
2	0.96	1077	+1=</s> AM no. 797	而 → just place → release
3	0.94	548	+1=</s> AM no. 1341	slides → it lend → event
4	0.94	124	+1=</s> AM no. 274	的 → kite direct → like

A selection of the *error signatures* that were found with the improved algorithm are shown in Table 6.11. The *ErrorRatios* of the computed *error signatures* are higher than in the previous implementation, as expected. The total number of *error signatures* and *error signature* bundles after applying our filter and similarity methods amount to 1,456. The following list contains the steps we took to improve the model based on the signatures presented in Table 6.11.

- **Signature #1:** This signature is the same as signature #2 in Table 6.7.
  Solution: Utterances matching the signature were truncated to exclude the last two words. The remainder of the utterance is added to the training data, if the utterance-level confidence exceeds the threshold of 0.8.

- **Signature #2:** Words ending with the phoneme "/s/" on inspection of Viterbi path and spectrogram matched to silence frames at the end of utterances.
  Solution: We used a voice activity detection (VAD) model to determine silence segments at the beginning and end of each training utterance. The VAD model consists of just two GMMs which were trained on speech and non-speech segments respectively. The detected silence segments at sentence beginning and end were mapped to the silence acoustic model and the acoustic model was retrained.

- **Signature #3:** The phoneme "/t/" is often confused with the phoneme "/d/" at word ends of English words.
  Solution: We added additional pronunciations to the dictionary for English words where phoneme "/t/" at the word end is replaced with the phoneme "/d/".
- **Signature #4:** This signature is the same as signature #1 in Table 6.7.
  Solution: The dictionary was filtered in the same manner as described in the solution to the signature in Table 6.7, page 80.

6.2.4.2 Results

Table 6.12 compares the MER results of system *S1_unsup_ES* to the system *S1_unsup_ES_impr*, which is using the fixes derived from signatures from the improved algorithm. While the first approach yields a system with nominal lower MER the difference in MER is not significant at a significance level of 0.05. The marginal difference in error rate is explained by the similarity of the top signatures found with both approaches. In addition, the proposed fix to signature #2 for the improved algorithm did not seem to have an impact on the error rate. However, since the top signatures were similar to the first implementation and due to the higher number of signatures generated by the improved algorithm, experiments in succeeding sections will use the new implementation.

Table 6.12: *Mixed error rates on the evaluation (development) set, comparing the system using the improved* error signature *algorithm to the initial implementation.*

System	S1_unsup_ES	S1_unsup_ES_impr
AM retrained	28.52% (34.65%)	28.28% (35.10%)
AM & LM retrained	28.06% (34.51%)	28.31% (34.68%)
AM & LM retrained, dict fix	**27.44% (33.95%)**	27.77% (34.06%)

6.2.5 Summary

We adapted our algorithm to compute *error signatures* without the availability of reference transcriptions. We showed the shortcomings of our initial

implementation for the unsupervised case and updated our algorithm accordingly. Finally, we compared the two implementations to each other and could show that we could still find the same errors as with the initial implementation and achieve similar MER results with the fixed systems. In addition, the improved algorithm yields 1,456 signatures compared to only 4 signatures using the signature filters described in Section 5.5.

6.3 Unsupervised iterative system improvements

The approach described in Section 6.2 to add training data by automatically transcribing an untranscribed data set and select data based on utterance-level confidence, can be integrated into an unsupervised iterative training scheme. The framework we use in this section is depicted in Figure 6.3. We want to investigate if adding our *error signature* algorithm to this training scheme has a positive impact on the ASR system's performance.

As a baseline system for our experiments we use the ASR systems which are generated by performing unsupervised model training as described by Laurent et al. [LHL14] and shown in Figure 6.3 part "(a)". We extend the iterative training process with our algorithm for *error signature* computation as presented in Figure 6.3 part "(b)" and integrate the fixes derived at each iteration into the ASR model training.

While investigating only one *error signature* at each iteration would be ideal to show the impact of each error fix, we chose to investigate several signatures in each iteration due to the high computational cost of each iteration. However, since the error correction process is reliant on an expert to derive a fix and implement it, we capped the total number of investigated signatures in each iteration to get feedback on the impact of the implemented fixes. As a result, we decided to investigate four to five signatures in each iteration.

We perform experiments on both the SEAME corpus and the ILSE corpus. The initial transcribed data as shown in Figure 6.3 is the 19 hour training set for SEAME, and a 44 hour training set for ILSE containing small audio segments from the training data.

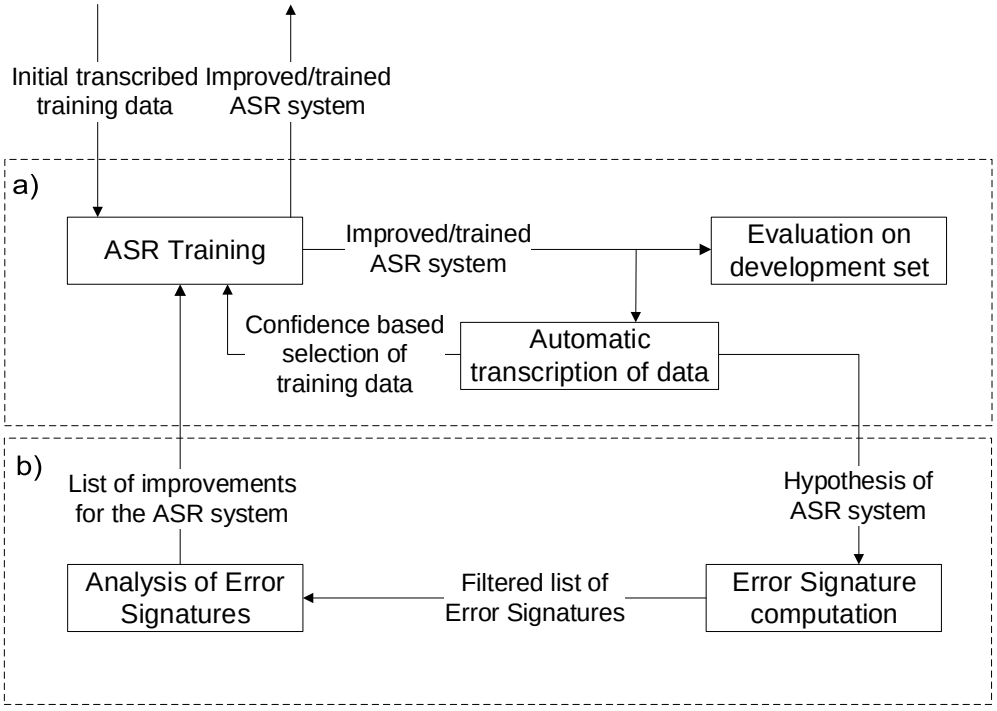

Figure 6.3: Iterative unsupervised training process as employed in the experiments in this section. Baseline iterative process is indicated in "(a)", additional steps for *error signatures* are shown in "(b)". Input is an initial set of training data and output is the trained ASR system.

6.3.1 Iterative unsupervised system improvements on SEAME

This section extends the experiments we already conducted in Section 6.2 on the SEAME corpus. The baseline system is *S0_unsup* already presented in the unsupervised training results in Section 6.2.2.2.

6.3.1.1 Iteration one

The first iteration system in the iterative training framework is the system *S1_unsup_ES_impr*, which was created by adding automatically transcribed data to the 19 hours of transcribed data and applying the fixes derived from the found *error signatures* in Section 6.2.4.1. The first iteration baseline system is *S1_unsup_noES*, which was created by adding automatically tran-

scribed data to the 19 hours of transcribed data in, as explained in Section 6.2.2.2.

6.3.1.2 Iteration two

In the beginning of iteration two the 39 hour unsupervised training set is re-decoded with both systems from iteration one (*S1_unsup_noES* and *S1_unsup_ES_impr*). The baseline *S2_unsup_noES* is trained based on the 19 hours of transcribed data and the automatically transcribed utterances by *S1_unsup_noES*, which exceed an utterance-level confidence threshold of 0.8.

For *S2_unsup_ES_impr* new signatures are extracted. The signatures are computed on the automatically transcribed data of the unsupervised training set by *S1_unsup_ES_impr*. We extract a set of 918 new *error signatures* and *error signature bundles* after filtering. Notably, the number of signatures after filtering is greatly reduced from the 1,456 signatures in iteration one. After filtering the signatures with the highest *ErrorRatio* are reviewed. The investigated signatures are listed in Table 6.13.

Table 6.13: *Examples for* error signatures *found on the unsupervised training set of the SEAME corpus in iteration two. The words on the right hand side of word confusion's column are the words in the 1-best hypothesis.*

Sig.#	ErrorRatio	#Occ.	Error signature	Word confusion
1	0.93	297	AM no. 1189	[noise] → him
			AM no. 2296	four → them
2	0.92	106	+2=</s>	和 → after
			AM no. 445	enough → hour
3	0.92	292	+2=</s>	搬 → 回家
			AM no. 1369	what → area
4	0.92	124	AM no. 141	[particle] → 因为
			AM no. 256	k. → 因为
			min. dur. /n/	words → 因为
5	0.91	557	AM no. 141	家 → 因为
			min. dur. /e/	s → 因为
			min. dur. /n/	futures → 因为

The following list shows the errors we associated with the found signatures and what steps we took to remedy the problems:

- **Signature #1:** Both acoustic models in the signature model the phoneme "/m/" for different states of the phoneme. By looking at the alignment of spectrogram and 1-best decoding path, we found that the models in the signature do not model the phoneme "/m/", but rather silence at the investigated segments.
  Solution: Based on the path of the 1-best hypothesis we used a threshold on the power to find silence segments between words in order to reduce the number of silence frames that are modeled by speech models such as models for phoneme "/m/".

- **Signature #2:** The acoustic model in the given signature is modeling the phoneme "/er/". Investigating examples for the signature, we found that the "schwa" is being confused with the phoneme "/er/". The confusions mainly appear at the word end preceding the token for a discourse particle "[particle]". The problem could stem from both a lack of acoustic modeling and language modeling data.
  Solution: By removing the matching utterances from the unsupervised training set and creating a new language model using the filtered unsupervised data, the combination of English word followed by a discourse particle becomes less likely and we avoid training the acoustic model on erroneous data.

- **Signature #3:** Phoneme "/a/" is modeled by the acoustic model given in the signature. By inspecting several examples, we found that the words are mostly correctly recognized. However, some frames at the end of words are falsely assigned to the phoneme "/a/" and should be silence.
  Solution: Silence and speech boundaries are already fixed by the solution to signature #1. Therefore, the examples with the given signature are just excluded from training.

- **Signature #4-#5:** Both acoustic models in the signature model the phoneme "/y/" for different states of the phoneme in signatures #4 and #5. The character sequence of "因为" can be translated to the English word "because" and has a high probability of appearing at the start of utterances in our language model. In the investigated examples, we found that the segment is often wrongly inserted in the beginning of utterances in partial segments of silence. The minimum duration of the phonemes indicates that the acoustic model does not ascribe a high probability to observing these phonemes at this point. Hence, the language model is forcing the word into this position.
  Solution: Ideally, additional training data for the language model is

necessary to better estimate the probability of N-grams at the start of utterances. However, since no additional data is available, we chose to mitigate this problem by excluding the utterances matching these signatures from the unsupervised training data for language modeling, reducing the probability of the 2-gram "<s> 因为".

6.3.1.3 Iteration three

Using the systems from iteration two (*S2_unsup_noES* and *S2_unsup_ES_impr*), we re-decoded the unsupervised training data. *Error signatures* were extracted on the automatic transcriptions produced by *S2_unsup_ES_impr*. The number of *error signatures* in the third iteration after bundling and filtering is further reduced to 785 compared to iteration two with 918. The signatures with the highest *ErrorRatio* were investigated and are listed in Table 6.14.

Table 6.14: *Examples for* error signatures *found on the unsupervised training set of the SEAME corpus in iteration three. Word on the right hand side of word confusion's column is the word in the 1-best hypothesis.*

Sig.#	ErrorRatio	#Occ.	Error signature	Word confusion
1	0.95	350	AM no. 2290 min. dur. /h/ min. dur. /o/	有请 → 然后 because → 然后 梦 → 然后
2	0.94	191	+1=</s> AM no. 1827	spread → australia self → 一下
3	0.93	1,036	+1=</s> AM no. 817	till → students us → shoes
4	0.91	152	AM no. 1854 AM no. 196 min. dur. /y/	想 → 了 不好意思 → 了 不知 → 了然

The analysis and identified error causes of the investigated *error signatures* in iteration three are listed below:

- **Signature #1:** Acoustic model with id 2290 is modeling the phoneme /r/. By investigating examples of the signature, we found that some speakers would omit some phonemes in the standard pronunciation of the two characters in the hypothesis ("然后").
  Solution: We added an additional pronunciation omitting these phonemes to the dictionary.

- **Signature #2:** An acoustic model for phoneme /ah/ has very low confidence score at the end of utterances. No direct cause of error could be identified, most of the reviewed examples were correctly recognized.
  Solution: While no error cause could be identified, utterances matching the signature were excluded from the unsupervised training data.
- **Signature #3:** In this signature the acoustic model for phoneme /s/ has a low confidence score at word ends. Reviewing the pronunciations in the dictionary, we found that pronunciations of English words ending in "/s/" and "/z/" are not consistent and do not match the pronunciations found in the CMU English dictionary [Wei98].
  Solution: To fix the dictionary entries, English words ending with phoneme "/s/" or "/z/" were replaced with the matching pronunciation in the CMU English dictionary.
- **Signature #4:** The acoustic models in the signature model the phonemes /l/ and /a/ respectively. Inspection of the dictionary led to the discovery that pronunciations of tokens with the character ˩ did not have the correct tone assigned to their phonemes. While the phonemes should be assigned tone three, the dictionary contained additional pronunciations with tone four and five.
  Solution: As a remedy, pronunciation variations containing tones four and five where removed from the dictionary.

6.3.1.4 Results

The MER results for all iterations are presented in Table 6.15. While the improvements on the systems of iteration one and two are not significant, the third iteration system *S3_unsup_ES_impr* is significantly better than *S3_unsup_noES*. Notably, the performance of *S3_unsup_noES* deteriorates compared to *S2_unsup_noES*. Comparing *S3_unsup_ES_impr* and *S2_unsup_noES*, the difference in error rate is not significant.

Table 6.16 depicts the training data of each ASR system for each iteration, showing that the amount of data is similar for *SX_unsup_noES* and *SX_unsup_ES_impr* systems. Therefore, the difference in error rate is not due to an increased amount of training data, to the contrary amount of training data of *SX_unsup_ES_impr* systems is lower than for *SX_unsup_noES* systems.

Based on the signatures #2, #4, and #5 from Table 6.13, we found that the language model requires additional data to not lead the search astray.

Table 6.15: *Mixed error rates of evaluation (development) set of three iterations of unsupervised training of the baseline system and systems after fixes derived from all investigated* error signatures *have been applied. In addition, relative improvements of SX_unsup_ES_impr to SX_unsup_noES are given. Significant improvements are marked with * if $p < 0.05$ and ** if $p < 0.01$.*

Iter.#	SX_unsup_noES		SX_unsup_ES_impr		Rel. gain	
0	29.60%	(35.74%)			0%	(0%)
1	28.24%	(34.21%)	27.77%	(34.06%)	1.7%	(0.4%)
2	27.84%	(33.75%)	27.64%	(33.61%)	0.7%	(0.4%)
3	28.18%	(34.14%)	**27.39%***	**(33.57%*)**	**2.8%**	**(1.7%)**

Table 6.16: *Training data in hours of speech ([hh:mm]) for the SEAME ASR systems trained in each iteration.*

Iter.#	SX_unsup_noES	SX_unsup_ES_impr
0	[19:04]	
1	[26:49]	[26:32]
2	[31:54]	[31:31]
3	[33:06]	[32:18]

It seems that the language model is currently the biggest problem in both *SX_unsup_noES* and *SX_unsup_ES_impr* ASR systems. In order to show that the fixes we made to the *SX_unsup_ES_impr* ASR systems, we decode the development and evaluation set with the best baseline ASR system (*S2_unsup_noES*) and *S3_unsup_ES_impr* using the baseline language model from the supervised ASR system (*S0_sup*) trained on all 60 hours of the SEAME training data. The MER results are shown in Table 6.17. While the differences between the two systems are significant on the development set with $p = 0.011$, the differences on the evaluation set are not, even though

Table 6.17: *Mixed error rates of evaluation and development set of S2_unsup_noES and S3_unsup_ES_impr using the baseline language model of the supervised system S0_sup of Section 6.1.1, page 70. Significant improvements are marked with * if $p < 0.05$ and ** if $p < 0.01$.*

Test set #	S2_unsup_noES	S3_unsup_ES_impr	(Rel. gain)
Dev set	32.82%	**32.21%***	**(1.9%)**
Eval set	26.80%	**26.22%**	**(2.2%)**

they have relatively increased compared to the results in Table 6.15.

6.3.1.5 Introducing improvements to the supervised system

To further investigate if the fixes derived from the computed *error signatures* during unsupervised training were beneficial, we introduce the fixes to our best system trained on the complete training set of the SEAME corpus and train a new system (*S3_sup*).

Fixes that were included here beyond the ones already depicted in Section 6.1 are predominantly changes to the dictionary (such as removal of wrong and infrequent pronunciations, correction of the pronunciation of English words ending in phonemes /s/ and /z/), and the application of the voice activity detection model and power thresholding resulting in better initial alignments for acoustic model training.

Table 6.18: *Mixed error rates of baseline system and systems after fixes derived from all* error signatures *have been applied. Significant improvements are marked with* * *if* $p < 0.05$ *and* ** *if* $p < 0.01$.

System	Dev set	(Rel. gain)	Eval set	(Rel. gain)
S0_sup	30.35%	(0%)	24.25%	(0%)
S2_sup	29.24%**	(3.7%)	23.98%	(1.1%)
S3_sup	28.66%**	(5.6%)	22.82%**	(5.9%)
S3_sup & improved LM	27.94%**	(8.0%)	22.22%**	(8.4%)

The MER results of the supervised systems are given in Table 6.18. *S0_sup* and *S2_sup* are the baseline system and improved system from the supervised signature experiment in Section 6.1.

It shows that the total improvements achieved by using *error signatures* amount to 5.90% relative improvement on the evaluation set. A majority of the performance gain on the evaluation set stems from the fixes we derived from the *error signatures* computed in an unsupervised fashion. The resulting differences are significant with $p < 0.001$ between *S0_sup* and *S3_sup*, and *S2_sup* to *S3_sup*. Additionally, by using the improved language model by Adel et al. [AKV+14] we achieve the best MER results to date on the SEAME corpus of 22.22%. We conclude that the fixes derived from the *error signatures*, found during the unsupervised model training, have a significant impact on the ASR system's performance, but due to an overall lack of training data were not able to impact the unsupervised system's performance in

the same degree.

6.3.1.6 Analysis

While we were able to achieve additional improvements compared to the baseline system when using knowledge gained from our *error signatures*, it is unclear how many errors where fixed as indicated by the signatures and which errors remain and were either not covered by our fixes or did not even match an *error signature* after the signature filters haven been applied.

Table 6.19: *Error categories of regions in the unsupervised SEAME training set on the latest system (S3_unsup_ES_impr).*

Category	Regions with signatures	(Rel. %)	Regions without signatures	(Rel. %)
AM overwhelm	24,185	(27.0%)	7,317	(10.4%)
LM overwhelm	12,790	(14.3%)	5,085	(7.2%)
Homophone	699	(0.8%)	763	(1.1%)
Search error	1,441	(1.6%)	958	(1.4%)
Correct	50,438	(56.3%)	56,089	(79.9%)
Total regions	89,553	(100%)	70,212	(100%)

To investigate which types of errors are covered by our signatures after filtering, we computed *error signatures* as before on our latest unsupervised system on the unsupervised SEAME training set (*S3_unsup_ES_impr*). Afterwards, we used the reference transcriptions of the unsupervised training set, to compute supervised regions and deduced their error categories. Subsequently, we iterate over all signatures which were previously computed in an unsupervised fashion and assign them to all the matching supervised regions to validate our approach. Finally, we count the error categories for regions which had been assigned an unsupervised *error signature* and regions without signatures. The distribution over error categories is depicted in Table 6.19.

It is evident that we are not able to cover "Homophone" errors and "Search errors" to the same degree as errors supposedly caused by the language model or acoustic model. However, we observe that the proportion of erroneous to all regions is higher for regions with *error signatures* than for regions without, indicating that the clustering process is working correctly.

To further analyze the *ErrorRatio*, we investigated signatures and their matching regions for different *ErrorRatio* thresholds. Figure 6.4 lists the per-

centage of erroneous regions covered by the found signatures. We find that the *ErrorRatio* and percentage of matched erroneous regions to all matched regions is positively correlated with a correlation coefficient of 0.95.

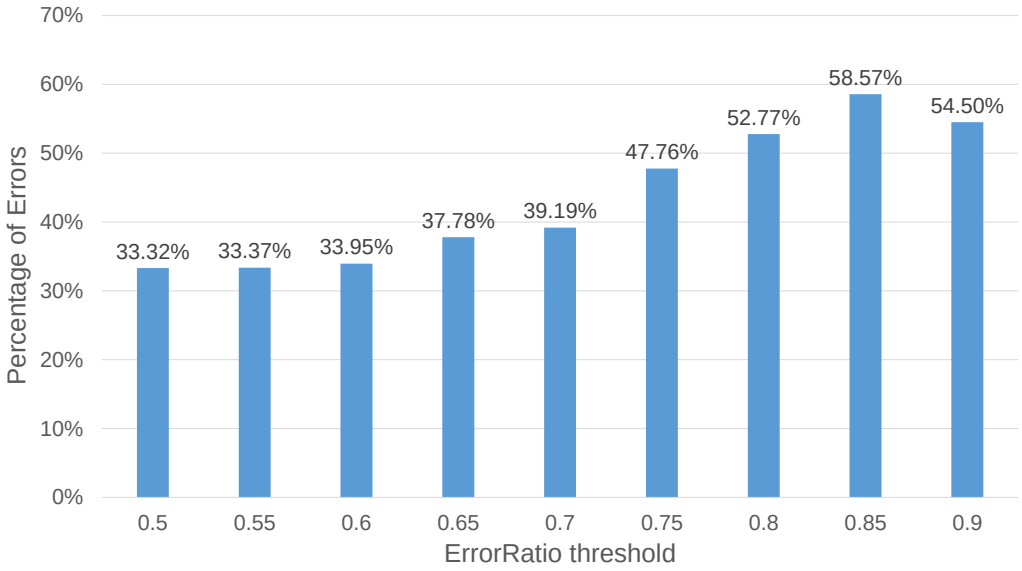

Figure 6.4: Percentage of erroneous regions among all matching regions of the unsupervised training set, using reference transcriptions and different *ErrorRatio* thresholds for selecting signatures.

Although the absolute number of matched errors dwindles with a higher *ErrorRatio* threshold in Figure 6.5, some of the fixes, such as the removal of wrong and infrequent word pronunciations from the decoding dictionary and the application of the power threshold and VAD model to obtain better initial ASR training alignments, have an impact beyond the regions that match the specific signature.

Thus, even though we achieved a reduction in error rate on the SEAME system using the knowledge gained from the *error signatures*, the low ratio of matched errors in Figure 6.4 begs the question if the *ErrorRatio* or the confidence score used for the *ErrorRatio's* computation is at fault. For this purpose we investigated the average confidence score and standard deviation of the supervised regions, as depicted in Table 6.20. Even though the average confidence score of errors is considerably lower than for regions associated with the category "CORRECT", the average confidence score for each category is within one standard deviation of each category. Therefore, confusions between regions belonging to the category "CORRECT" and other categories have to at least partly stem from the confidence score.

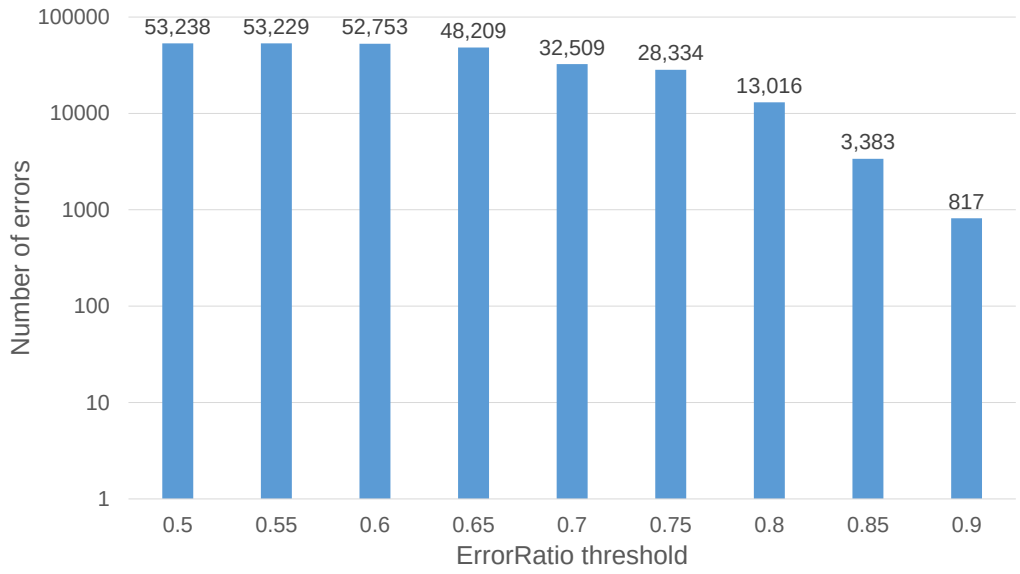

Figure 6.5: Absolute number of errors of the development set matched by the computed *error signatures* using different *ErrorRatio* thresholds.

With regards to the errors that were fixed using the knowledge from the computed *error signatures*, we compare the two systems *S2_unsup_noES* and *S3_unsup_ES_impr* of the unsupervised training scheme with each other in the following analysis. In addition, due to the small difference in performance of the two systems, we compare the systems trained on the complete 60 hour training set presented in Section 6.3.1.5 as well. We compare the system *S2_sup* with an MER of 23.98% to the *S3_sup*natures fixed" with an MER of 22.82%:

- We removed wrong and infrequent pronunciations of English and Mandarin words from the decoding dictionary, thereby reducing the absolute number of errors concerning these words from $3,255$ to $3,137$, which amounts to 3.6%. The number of errors for this group of words in the supervised systems is reduced from $2,772$ to $2,540$, which is 8.4% relative.

- By using a VAD model and a power threshold to produce better initial alignments for model training, we expect words ending in fricatives and nasals to be less often misrecognized. While there is no difference in performance for these group of words in the unsupervised systems, the errors in the supervised system decrease from $2,266$ to $2,191$, or 3.3% relative.

Table 6.20: *Error categories of regions in the unsupervised training set and the average word confidence and standard deviation of regions associated with the error category.*

Category	Average confidence	Standard deviation
Acoustic model	0.66	0.26
Language model	0.65	0.28
Homophone	0.76	0.23
Search error	0.77	0.25
Correct	0.88	0.24
All	0.83	0.26

- Pronunciation of English words ending with either phoneme "/s/" and "/z/" were inconsistent with the pronunciation found in the CMU dictionary. Errors concerning these words was reduced from 630 to 571 or 9.4% relative. The errors in the supervised system decrease from 572 to 504 or 11.9% relative.

In conclusion, we found that *error signatures* derived in an unsupervised fashion have a significant impact on the error rate of both unsupervised and supervised SEAME ASR systems. Furthermore, we found that improvements were achieved for words matching our signatures. With regards to the computation of the *ErrorRatio* and the confidence score, a confidence model specifically trained to detect errors on the SEAME corpus could potentially improve the quality of the found signatures. However, this would require additional data to train the confidence model on.

6.3.2 Iterative unsupervised system improvements on ILSE

In this section we present the experiments conducted on the ILSE corpus and the improvements we achieved by applying fixes to errors found with *error signatures*. Since the condition of the manual transcriptions of the training set is such that they only loosely match the actual content, we chose to conduct our experiments on the complete 265 hour training set. Due to the extensive size of both development and evaluation set we only report word error rates on a subset of both development and evaluation set. From the original development and evaluation sets, we selected the first 45 minutes of each interview as development and evaluation set.

6.3.2.1 System setup

The acoustic model is a deep neural network with a preprocessing similar to the SEAME ASR systems presented in this chapter. Since no speaker information is available in the interview files, the fMLLR transform is omitted, resulting in a one pass system. The vocabulary is chosen based on the transcriptions in the training data and contains 71,889 words. The dictionary stems from the German GlobalPhone dictionary. Words not contained in the dictionary were automatically generated. The dictionary contains 74,423 entries. The language model is build on the reference transcriptions of the complete training set. While the transcriptions contain no timestamps, they contain some punctuations and speaker turns, which were used to extract sentences from the transcriptions to estimate the language model on. The language model is a 3-gram Kneser-Ney language model with a perplexity of 199.38 and 179.69 on the development and evaluation set respectively.

6.3.2.2 Baseline

Due to the long audio files of up to 45 minutes and their accompanying transcriptions being erroneous and not verbatim with no timestamps, the forced alignment cannot be applied successfully to train an initial system. Therefore, to extract training data for an initial ASR system, Weiner et al. [WFT+16] used a long audio alignment approach to extract short segments. In this approach each 45 min. interview is decoded with a vocabulary restricted to the words found in the corresponding transcriptions. In a post-processing step each decoded segment that matches a portion in the transcriptions is added to the training data. Ultimately, the baseline system *I0_unsup* we employ is trained on 44 hours of training data.

6.3.2.3 Iteration one

Similar to our experiments on the SEAME corpus we trained two systems based on the utterance-level confidence scores depicted in Figure 6.6 (*I1_unsup_noES* and *I1_unsup_ES_impr*). The total amount of training data for the new models is 47 hours in the first iteration. Only a few segments exceed our threshold of 0.8, since each segment was a complete interview of up to 45 minutes in length.

Table 6.21 shows the error signatures we found and implemented fixes for. The total number of *error signatures* is 2,601. The investigated *error signa-*

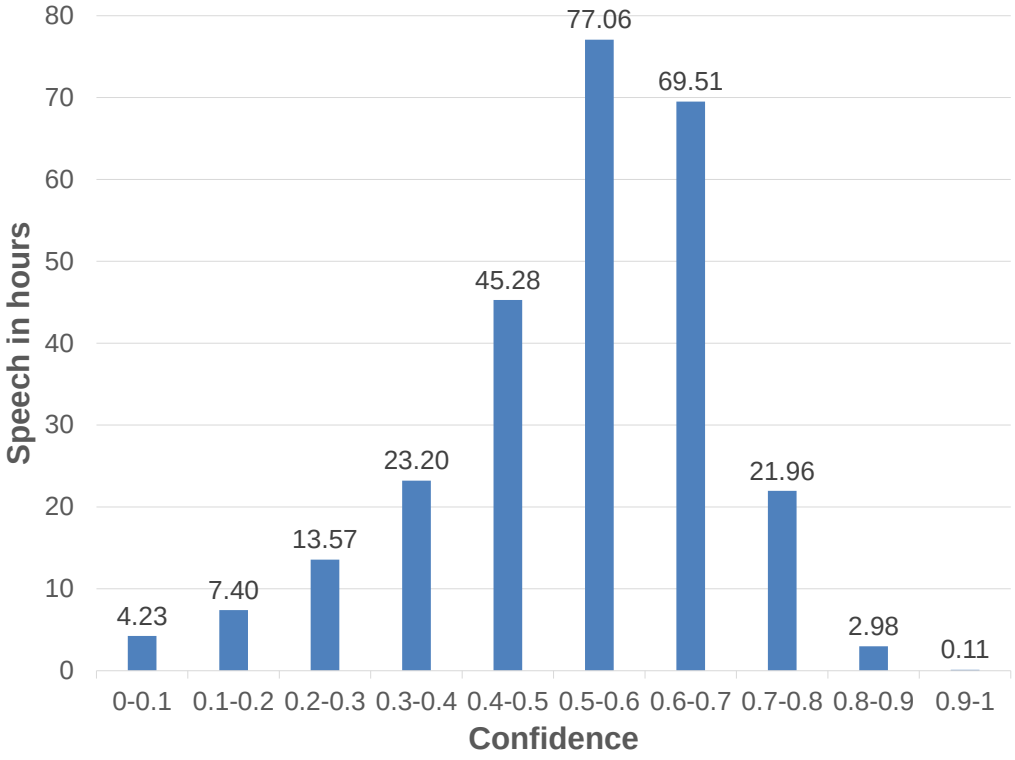

Figure 6.6: Training data split into decile confidence bins after decoding and confidence estimation for iteration one.

tures were among the 32 most promising signatures in terms of *ErrorRatio*. 14 of the omitted signatures were also concerned with the noise acoustic model. The remaining signatures were similar to the investigated signatures, were however not bundled together. The following list shows the fixes we implemented to remedy the problems identified by the *error signatures*:

- **Signature #1:** The acoustic model with id 140 models the phoneme /n/. Frequent potential confusions of signature #1 are with the word "und" which should only rarely appear at the end of utterances. Investigating the language model probabilities for the bigram "und </s>", turned out that this bigram is far more likely than most other 2-grams with the word "und". By further looking into the training text used for estimating the language model probabilities, we found that the text contains a sizable number of partial sentences ending with the word "und".
  Solution: To remedy this situation we removed sentence end tags from all utterances ending in the word "und". The language model was then

Table 6.21: *Examples for* error signatures *found on the training set of the ILSE corpus in iteration one. The words on the right hand side of word confusion's column are the words in the 1-best hypothesis.*

Sig. #	ErrorRatio	#Occ.	Error signature	Word confusion
1	0.99	1,251	+1=</s>	die → und
			AM no. 140	möbeln → und
2	0.98	1,149	AM no. 2542	SIL → [noise]
3	0.98	773	AM no. 2673	[redacted] → [noise]
			AM no. 850	
4	0.98	10,038	AM no. 349	beste → [noise]
			AM no. 4531	bleibe → [noise]
5	0.97	2,381	AM no. 4458	[redacted] → [noise]
6	0.96	1,240	0=/h/	[unk] → haben
			AM no. 277	daheim → haben
			min. dur. /etu/	oben → haben
			min. dur. /n/	feilen → haben
7	0.94	634	0=oder	[redacted] → oder
			AM no. 2805	tätigkeit → oder
8	0.94	763	AM no. 1244	drin → besonderen
			AM no. 3098	marschiert → marschieren
			AM no. 4360	überall → waren
			min. dur. /r/	das → verfügbaren
9	0.94	1,902	AM no. 1334	hatten → werden
			AM no. 3098	freunden → geworden
			AM no. 697	gut → wurden
			min. dur. /etu/	zum → worden

re-estimated.

- **Signature #2-#5:** The noise token frequently appears in areas of low confidence. We noticed that signatures regarding the noise token contained more than the expected five acoustic models for the five state noise HMM. Further investigation of the acoustic model turned out that the noise model had erroneously been trained as a context-dependent model.
  Solution: In the new model we retrained noise as a context-independent acoustic model.

- **Signature #6:** This signature mostly concerns the word "haben" with only one pronunciation in the dictionary: "/h/ /a/ /b/ /etu/

/n/". The last two phonemes are minimum duration only (duration of three frames), which indicates that the actual pronunciation contains less phonemes.
Solution: As a solution we added another colloquial pronunciation of "haben" ("/h/ /a/ /m/") to the dictionary.

- **Signature #7:** The acoustic model with id 2805 in the *error signature* models the phoneme /r/. The signature shows that the word "oder", which is quite frequent in the German language, is often confused with other words. The dictionary contained two pronunciations for this word, the first one being "/ol/ /d/ /atu/" and the second one "/ol/ /d/ /etu/ /r/".
Solution: Since the second pronunciation is rather unlikely of being correct it was removed from the dictionary.

- **Signature #8-#9:** The last two signatures are words whose pronunciations either end in "/r/ /etu/ /n/" or "/d/ /etu/ /n/".
Solution: As a fix to this problem we added additional pronunciations to the dictionary which more closely match the actual pronunciation by omitting the phoneme /etu/.

The results are explained in Section 6.3.2.6.

6.3.2.4 Iteration two

Since the selected audio data was very little for both *I1_unsup_noES* and *I1_unsup_ES_impr*, caused by the length of the training segments, we decided to cut training segments at silence boundaries starting with iteration two. We chose to cut a training segment at silence segments which are at least 0.2 seconds long. The threshold of 0.2 seconds has already been used to cut the audio files of the development and evaluation set into smaller parts. In addition, we required each new training segment to be of a certain length to be accepted. Figure 6.7 shows the amount of speech attributed to confidence bins depending on varying thresholds on minimum segment length. As a compromise between selecting as much data as possible and selecting data of high quality we chose a threshold of 3 seconds to be the minimal length of a segment.

Table 6.22 presents the signature we investigated for the second iteration. The total number of *error signatures* is $3,103$. Due to the possible high impact of the fix derived from the signature we chose to only work on one signature in this iteration:

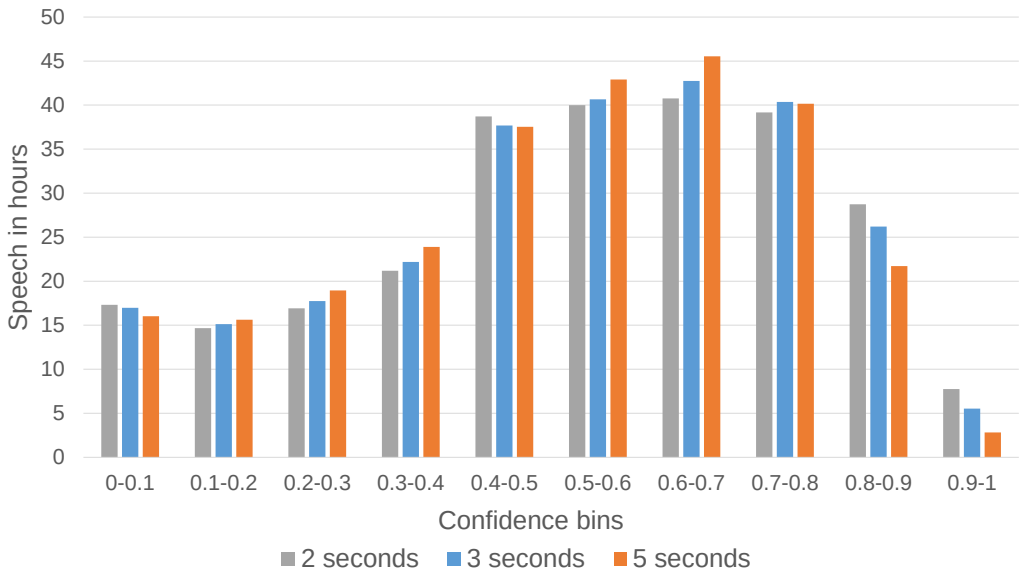

Figure 6.7: Training data split into decile confidence bins after decoding and confidence estimation, using three thresholds for minimum length of training segments for segmentation.

Table 6.22: *The* error signature *investigated in iteration two of the ILSE corpus. Word on the right hand side of word confusion's column is the word in the 1-best hypothesis.*

Sig.#	ErrorRatio	#Occ.	Error signature	Word confusion
1	0.99	3,933	AM no. 575 AM no. 798	[redacted] → und das → und

- **Signature #1:** The acoustic models of the signature presented in Table 6.22 model the phonemes "/n/" and "/u/". Listening to the examples of the signatures, we found that the word "und" is recognized instead of filler words, such as "ähm" and "aha". We found that "ähm" and "aha" are already part of the dictionary, but are modeled with the language model. Since filler words can be inserted at any point in an utterance they should not be modeled by the language model, but disregarded as filler, similar to noise and silence tokens. In addition, since filler words are mostly not transcribed, the probability in the language model is comparably low. Thus, filler words were never recognized.
  Solution: We removed the words from the language model and designated them as filler words such that they do not affect the language

model context during decoding and are handled like noise and silence tokens.

Results are explained in Section 6.3.2.6.

6.3.2.5 Iteration three

Table 6.23: *Examples for* error signatures *found on the training set in iteration three of the ILSE corpus. The words on the right hand side of word confusion's column are the words in the 1-best hypothesis.*

Sig.#	ErrorRatio	#Occ.	Error signature	Word confusion
1	0.99	1,888	AM no. 2323	eine → [emhm]
			AM no. 838	ums → [emhm]
2	0.99	2,591	AM no. 2483	man → [ehm]
			AM no. 838	irgenwie → [ehm]
3	0.99	661	AM no. 2712	dann → [emhm]
			AM no. 2956	hingehen → [emhm]
			AM no. 838	drinnen → [emhm]
4	0.99	606	AM no. 2675	[redacted] → [ähä]
			AM no. 3243	später → [ähä]
			AM no. 4591	steg → [ähä]

The process for data selection in the third iteration is the same as for iteration two. The total number of *error signatures* is 3,367. The *error signatures* that we investigated in the third iteration are presented in Table 6.23:

- **Signature #1-#4:** The *error signatures* for the third iteration mostly concern the filler words introduced in the last iteration. The acoustic models of the signatures presented in Table 6.23 model phonemes of the filler words. Upon inspection of examples matching the signatures, we found that filler words are often recognized in segments of uncertainty, since they do not affect the language model context and have only a filler penalty assigned to them.
  Solution: To reduce the number of filler words that are recognized, we require each filler word to either have a preceding or succeeding segment of silence. We base our decision on work by Stouten et al., who found that 80% of filler words appeared in the context of silence in their experiments [SDMW06].

6.3.2.6 Results

Word error rates of the systems trained iteration one, two, and three are shown in Table 6.24.

Table 6.24: *Mixed error rates of the evaluation (development) set of three iterations of unsupervised training of the baseline system and systems after fixes derived from all* error signatures *have been applied. In addition, relative improvements of IX_unsup_ES_impr to IX_unsup_noES are given. Significant improvements are marked with * if $p < 0.05$ and ** if $p < 0.01$.*

Iter.#	IX_unsup_noES		IX_unsup_ES_impr		Rel. gain	
0	64.76%	(66.98%)			0%	(0%)
1	64.04%	(66.42%)	63.66%*	(65.54%**)	0.6%	(1.3%)
2	62.16%	(64.56%)	62.06%	(63.80%**)	0.2%	(1.2%)
3	62.33%	(64.84%)	**61.90%****	**(63.79%**)**	**0.7%**	**(1.6%)**

While we achieved a significant ($p = 0.005$) word error rate reduction of 0.69% relative on the evaluation set comparing *I3_unsup_ES_impr* to *I3_unsup_noES*, the difference between the *I3_unsup_ES_impr* to *I2_unsup_noES* is not significant at a significance level of 0.05.

6.3.2.7 Analysis

The consistently high error rates and the amount of speech data selected for unsupervised training in Table 6.25 point to fundamental problems within the ILSE corpus, namely the reduced amount of training data between the third and second iteration of both systems. The expectation is that the amount of training data increases in each iteration as is the case for the SEAME unsupervised training, refer to Table 6.16 on page 92. Among the problems in the ILSE corpus are erroneous transcriptions, dialectal speech, changing recording setup, and missing annotations of speaker turns in the transcriptions. We already alluded to these problems in the description of the database in Chapter 3.

The setup of the recordings is such that two speakers, the interviewer and interviewee, speak to each other in turns. The interviewer however is often interjecting filler words, such as “aha” and “emhm” into the interviewees turns. Since the interviews are spontaneous, the interviewee is often interrupted with additional questions by the interviewer in an effort to focus

Table 6.25: *Total training data in hours of speech ([hh:mm]) for the ILSE ASR systems trained in each iteration.*

Iter.#	IX_unsup_noES	IX_unsup_ES_impr
0	[44:17]	
1	[47:27]	[47:28]
2	[89:48]	[76:22]
3	[75:22]	[70:07]

the interviewee's answers. Therefore, a system first segmenting the data into speaker turns would be very beneficial and would add the possibility of speaker normalization.

With regards to the errors that were fixed using the knowledge from the computed *error signatures*, we compare the best baseline system (*I2_unsup_noES*) with *I3_unsup_ES_impr* in the following analysis:

- The word "und" was mistakenly recognized at the end of utterances, as well as falsely substituted for filler words. The baseline system counts a total of $4,339$ errors with the word "und", the system using signatures only $3,672$, a reduction of 15.4%.

- An unlikely pronunciation for the word "oder" was removed from the dictionary, reducing errors with the word by 11.1%.

- We added an additional pronunciation for the word "haben" to the dictionary, reducing the errors concerning the word by 5.5%.

- Added additional pronunciations for words ending in "-den" to the dictionary, reducing errors in that group by 2.3%.

- Additional pronunciations for words ending in "-ren" were added to the dictionary, reducing errors among the group by 2%.

The absolute reduction in errors by the items given in the above list amounts to 912 errors, corresponding to 0.8% of words in the evaluation set. However, judging by the signatures in iteration three, filler words cause additional deletion errors in the system, thereby reducing the performance lead of our best system to the baseline.

6.3.3 Summary

Using the SEAME and ILSE corpus, we integrated our *error signature* framework into an unsupervised training scheme. While the signature improved systems all produced lower error rates on the evaluation sets compared to the baseline systems, only some of the improvements were found to be significant. The ILSE corpus lacks speaker segmentation and high quality reference transcriptions and as such produces overall poor results. Upon closer inspection of the SEAME corpus we found that a general lack of data for acoustic modeling and language modeling were the major cause for the poor performance. Integrating the fixes found during the SEAME corpus unsupervised training into the supervised system trained on the complete 60 hour training set reduced the MER by 1.16% absolute to the supervised signature improved system or 4.84% relative.

6.4 Summary and impact of the human factor

We could show in our experiments that we are able to extract useful information from the *error signatures*. However, the actual feasibility of our approach undoubtedly depends on the amount of effort required from the expert reviewing the computed signatures.

While it would be an option to use the signatures as a way of automatically filtering the training data selected for unsupervised model training, the benefit of the signatures is caused by the expert using them to fix issues in the ASR system. Since a user study comparing the progress of experts with and without signatures is beyond this work, we can point to the development history of ASR systems concerning the two investigated SEAME and ILSE corpora.

The start of the project for the SEAME corpus was in the year 2012 with initially only a subset of the acoustic modeling data [VLW+12]. The corpus was extended in the year 2013 to its current size. For two years since then, researchers have tried to improve the performance of the system. Yet, even after training a new state-of-the-art system based on the previous research we were still able to find significant room for improvement using our *error signature* framework.

The ILSE corpus [MM00, LTSM00] is a more recent project ailing from a

multitude of problems as indicated in Section 3.3, requiring a lot of manual effort to fix. While we started at the end of 2014 to improve the system, improvements were only marginal. By using the unsupervised model training we were able to decrease the error rate compared to the baseline system by 4% relative. Using the *error signatures* to fix errors in the system we were able to reduce the WER by 4.4% relative to the baseline.

Chapter 7

Conclusion and Future Directions

This chapter summarizes the results of this thesis. The contributions of this work are concluded and possible future research directions are outlined.

7.1 Summary of results

This thesis has two major contributions. The first one is the implementation and introduction of a new HMM-based sequence recognition toolkit BioKIT, the second is the creation of an *error signature* framework to accumulate, group, and analyze errors made by HMM-based sequence recognition systems.

7.1.1 BioKIT

We implemented a new easy to use toolkit for HMM-based sequence recognition [TWG+14]. By omitting the focus on automatic speech recognition, the toolkit is appealing to a wider group of researchers, such as research in speech recognition using Electrocorticography [HHd+15], Electromyography, or gesture recognition [GAS15], using Electromyography and an Inertial Measurement Unit.

Due to its modular design, the C++ core library is easy to extend with new

algorithms concerning HMM-based sequence recognition. By using a two-layer structure of the C++ core library and Python scripting layer, the need for recompilation only arises if new modules need to implemented, but is not required for experimental setups. By enabling threads to share memory between them, the memory requirements especially for tasks with big language models of several gigabyte in size is greatly reduced, and allows the user to fully exploit multi-core computers. BioKIT was applied throughout this thesis and advanced by *error signatures*.

7.1.2 Error signatures

We developed a novel framework to analyze decoding output of HMM-based sequence recognition systems using the example of automatic speech recognition. While previous research relied on the availability of reference transcriptions to conduct their analysis, we could show that our approach is feasible even if no reference transcriptions are available.

We introduced the notion of *error signatures* grouping errors together by a set of interpretable attribute-value pairs. Based on the *error signatures* we deduced the underlying cause of the errors. In addition, the framework allows the user to add additional attributes, such as speaker id or part-of-speech tags, to the clustering process to obtain the *error signatures*.

The algorithm to obtain *error signatures* is the same for supervised and unsupervised data with the exception of the *ErrorRatio* computation. However, despite different computations the resulting measure for the supervised and the unsupervised case is the same. Thus, both decoding output with reference transcriptions and without reference transcriptions can be used at the same time for *error signature* computation.

We observed that the baseline systems for both the SEAME and the ILSE corpus deteriorated with the third iteration (*S3_unsup_noES* and *I3_unsup_noES*), whereas the systems improved by fixes derived from *error signatures* have a decreasing error rate. In conclusion, both *S3_unsup_ES_impr* and *I3_unsup_ES_impr* are significantly better than the comparable baseline *S3_unsup_noES* and *I3_unsup_noES*. Furthermore, the fixes from the SEAME corpus, which were derived in an unsupervised fashion, were helpful in significantly reducing the MER of the supervised SEAME system by 4.84% relative.

7.2 Potential future research

Several avenues of research and additional interesting features can be pursued based on the results of this thesis. The following section will explore these for both the BioKIT toolkit, as well as the *error signature* framework.

7.2.1 BioKIT

While toolkits can always be extended to use state-of-the-art algorithms and models, the task of decoding spontaneous speech, especially from the ILSE corpus, is a demanding application. The detection of word repetitions and false starts requires attention. Additionally, as is the case in German, handling of multi-words needs to be addressed. Added flexibility in combining class-based N-gram language models with grammars would be beneficial to recognizing spontaneous speech. These potential research directions apply to both the ILSE corpus and BioKIT.

Furthermore, to increase the portability of BioKIT to other platforms, such as hand-held devices, the usage of external libraries needs to be reduced as far as possible. In addition, possible restrictions in terms of memory, processing power, e.g. reduction of floating point operations, have to be investigated. Possible extension of BioKIT to work as a static decoder should be pursued to increase the performance on for example hand-held devices by avoiding language model lookahead.

7.2.2 Error signatures

While the BioKIT tool is already used for research beyond automatic speech recognition, *error signatures* have not yet been applied to systems using other modalities, such as motion (e.g. Airwriting [AGS14]), Electromyography, or Electrocorticography.

Beyond different input modalities, the investigation of different attributes, which might also enable the recognition of preprocessing related problems, are of interest. Furthermore, combined usage of decoding output with available reference transcriptions and without reference transcriptions might yield better *error signatures*. Weighting schemes for the different utterances depending on the availability and the quality of their reference transcriptions could be explored.

Another avenue of research would be to change the *error signature* algorithm, such that it no longer focuses on errors as a whole, but rather detects idiosyncrasies between different corpora of speech. For example, an ASR system trained on native speakers is expected to perform worse for non-native speakers of the language. Instead of investigating the errors the ASR system produces in general, the algorithm could be used to identify signatures for non-native speakers to identify problematic words or phonemes appearing in non-native speech and not in native speech.

Bibliography

[AGS14] Christoph Amma, Marcus Georgi, and Tanja Schultz. Airwriting: A Wearable Handwriting Recognition System. *Personal and ubiquitous computing*, 18(1):191–203, 2014.

[AKV+14] Heike Adel, Katrin Kirchhoff, Ngoc Thang Vu, Dominic Telaar, and Tanja Schultz. Comparing Approaches to Convert Recurrent Neural Networks into Backoff Language Models For Efficient Decoding. In *Proc. of Interspeech*, pages 651–655, 2014.

[All07] Alexandre Allauzen. Error Detection in Confusion Network. In *Proc. of Interspeech*, pages 1749–1752, 2007.

[AVK+13] Heike Adel, Ngoc Thang Vu, Franziska Kraus, Tim Schlippe, Haizhou Li, and Tanja Schultz. Recurrent Neural Network Language Modeling for Code Switching Conversational Speech. In *Proc. of ICASSP*, pages 8411–8415. IEEE, 2013.

[BDM+92] Peter F Brown, Peter V Desouza, Robert L Mercer, Vincent J Della Pietra, and Jenifer C Lai. Class-Based N-gram Models of Natural Language. *Computational linguistics*, 18(4):467–479, 1992.

[Ber06] Pavel Berkhin. A Survey of Clustering Data Mining Techniques. In *Grouping multidimensional data*, pages 25–71. Springer, 2006.

[BK03] Jeff A Bilmes and Katrin Kirchhoff. Factored Language Models and Generalized Parallel Backoff. In *Proc. of HLT-NAACL*, volume 2, pages 4–6, 2003.

[BM98] L Douglas Baker and Andrew Kachites McCallum. Distributional Clustering of Words for Text Classification. In *ACM SIGIR conference on Research and development in information retrieval*, pages 96–103. ACM, 1998.

[BPSW70] Leonard E Baum, Ted Petrie, George Soules, and Norman Weiss. A Maximization Technique Occurring in the Statistical Analysis of Probabilistic Functions of Markov Chains. *The annals of mathematical statistics*, pages 164–171, 1970.

[Bro87] Peter F Brown. *The Acoustic-Modeling Problem in Automatic Speech Recognition*. Computer Science Department, Carnegie-Mellon University, 1987.

[CCR+13] Jia Cui, Xiaodong Cui, Bhuvana Ramabhadran, Janice Kim, Brian Kingsbury, Jonathan Mamou, Lidia Mangu, Michael Picheny, Tara N Sainath, and Abhinav Sethy. Developing Speech Recognition Systems for Corpus Indexing under the IARPA Babel Program. In *Proc. of ICASSP*, pages 6753–6757. IEEE, 2013.

[Cha97] Lin L Chase. *Error-Responsive Feedback Mechanisms for Speech Recognizers*. PhD thesis, Pittsburgh, PA: Carnegie Mellon University, 1997.

[DKS95] James Dougherty, Ron Kohavi, and Mehran Sahami. Supervised and Unsupervised Discretization of Continuous Features. In *Machine learning*, volume 12, pages 194–202, 1995.

[DM80] Steven B Davis and Paul Mermelstein. Comparison of Parametric Representations for Monosyllabic Word Recognition in Continuously Spoken Sentences. *Proc. of ICASSP*, 28(4):357–366, 1980.

[FGH+97] Michael Finke, Petra Geutner, Hermann Hild, Thomas Kemp, Klaus Ries, and Martin Westphal. The Karlsruhe-Verbmobil Speech Recognition Engine. In *Proc. of ICASSP*, volume 1, pages 83–86, 1997.

[FI93] Usama Fayyad and Keki Irani. Multi-Interval Discretization of Continuous-Valued Attributes for Classification Learning. *Proc. of IJCAI*, pages 1022–1027, 1993.

[FSGL11] Thiago Fraga-Silva, Jean-Luc Gauvain, and Lori Lamel. Lattice-Based Unsupervised Acoustic Model Training. In *Proc. of ICASSP*, pages 4656–4659. IEEE, 2011.

[Gal98] Mark JF Gales. Maximum Likelihood Linear Transformations for HMM-based Speech Recognition. *Computer Speech & Language*, 12(2):75–98, 1998.

[GAS15] Marcus Georgi, Christoph Amma, and Tanja Schultz. Recognizing Hand and Finger Gestures with IMU based Motion and EMG based Muscle Activity Sensing. In *Proc. of Biosignals*, pages 99–108, 2015.

[GC89] Laurence Gillick and Stephen J Cox. Some Statistical Issues in the Comparison of Speech Recognition Algorithms. In *Proc. of ICASSP*, pages 532–535. IEEE, 1989.

[GGPP93] John Garofalo, David Graff, Doug Paul, and David Pallett. Continous Speech Recognition (CSR-I) Wall Street Journal (WSJ0) News, Complete. Technical report, Linguistic Data Consortium, Philadelphia, 1993.

[GHM92] John J Godfrey, Edward C Holliman, and Jane McDaniel. SWITCHBOARD: Telephone Speech Corpus for Research and Development. In *Proc. of ICASSP*, volume 1, pages 517–520. IEEE, 1992.

[GL94] Jean-Luc Gauvain and Chin-Hui Lee. Maximum a Posteriori Estimation for Multivariate Gaussian Mixture Observations of Markov Chains. *Speech and audio processing, IEEE transactions on*, 2(2):291–298, 1994.

[GLF+93] John S Garofolo, Lori F Lamel, William M Fisher, Jonathon G Fiscus, and David S Pallett. DARPA TIMIT Acoustic-Phonetic Continuous Speech Corpus CD-ROM. NIST Speech Disc 1-1.1. *NASA STI/Recon Technical Report N*, 93:27403, 1993.

[HHd+15] Christian Herff, Dominic Heger, Adriana de_Pesters, Dominic Telaar, Peter Brunner, Gerwin Schalk, and Tanja Schultz. Brain-to-text: Decoding Spoken Phrases from Phone Representations in the Brain. *Name: Frontiers in Neuroengineering*, 8(6), 2015.

[Hun07] John D. Hunter. Matplotlib: A 2D Graphics Environment. *Computing In Science & Engineering*, 9(3):90–95, 2007.

[Jel90] Fred Jelinek. Self-Organized Language Modeling for Speech Recognition. *Readings in speech recognition*, pages 450–506, 1990.

[JEO+05] David Johnson, Dan Ellis, Chris Oei, Chuck Wooters, and Philipp Faerber. Quicknet, 2005.

[JOP+01] Eric Jones, Travis Oliphant, Pearu Peterson, et al. SciPy: Open Source Scientific Tools for Python, 2001.

[Kat87] Slava M Katz. Estimation of Probabilities from Sparse Data for the Language Model Component of a Speech Recognizer. *Acoustics, Speech and Signal Processing, IEEE Transactions on*, 35(3):400–401, 1987.

[KN95] Reinhard Kneser and Hermann Ney. Improved Backing-Off for M-gram Language Modeling. In *Proc. of ICASSP*, volume 1, pages 181–184. IEEE, 1995.

[KNB13] Oscar Koller, Hermann Ney, and Richard Bowden. May the force be with you: Force-Aligned Signwriting for Automatic Subunit Annotation of Corpora. In *Automatic Face and Gesture Recognition (FG)*, pages 1–6. IEEE, 2013.

[KW99] Thomas Kemp and Alex Waibel. Unsupervised Training of a Speech Recognizer: Recent Experiments. In *Proc. of Eurospeech*, pages 827–830, 1999.

[LGA02] Lori Lamel, Jean-Luc Gauvain, and Gilles Adda. Unsupervised Acoustic Model Training. In *Proc. of ICASSP*, volume 1, pages 874–877. IEEE, 2002.

[LHH+90] Kai-Fu Lee, Satoru Hayamizu, Hsiao-Wuen Hon, Cecil Huang, Jonathan Swartz, and Robert Weide. Allophone Clustering for Continuous Speech Recognition. In *Proc. of ICASSP*, pages 749–752. IEEE, 1990.

[LHL14] Antoine Laurent, William Hartmann, and Lori Lamel. Unsupervised Acoustic Model Training for the Korean Language. In *ISCSLP*, pages 469–473. IEEE, 2014.

[LKS01] Akinobu Lee, Tatsuya Kawahara, and Kiyohiro Shikano. Julius - An Open Source Real-Time Large Vocabulary Recognition Engine. In *Proc. of Eurospeech*, pages 1691–1694, 2001.

[LTCL10] Dau-Cheng Lyu, Tien-Ping Tan, Eng-Siong Chng, and Haizhou Li. An Analysis of a Mandarin-English Code-Switching Speech Corpus: SEAME. *Age*, 21:25–33, 2010.

[LTSM00] Ursula Lehr, Hans Thomae, Marina Schmitt, and Elisabeth Minnemann. Interdisziplinäre Längsschnittstudie des Erwachsenenalters: Geschichte, theoretische Begründung und ausgewählte Ergebnisse des 1. Messzeitpunktes. In Peter Martin, Klaus Udo Ettrich, Ursula Lehr, Dorothea Roether, Mike Martin, and Antje Fischer-Cyrulies, editors, *Aspekte der Entwicklung im mittleren und höheren Lebensalter: Ergebnisse der Interdisziplinären Längsschnittstudie des Erwachsenenalters (ILSE)*, pages 1–16. Steinkopff, 2000.

[MHM+14] Erik McDermott, Georg Heigold, Pedro Moreno, Andrew Senior, and Michiel Bacchiani. Asynchronous Stochastic Optimization for Sequence Training of Deep Neural Networks: Towards Big Data. In *Interspeech*, pages 1224–1228, 2014.

[MM00] Peter Martin and Mike Martin. Design und Methodik der Interdisziplinären Längsschnittstudie des Erwachsenenalters. In Peter Martin, Klaus Udo Ettrich, Ursula Lehr, Dorothea Roether, Mike Martin, and Antje Fischer-Cyrulies, editors, *Aspekte der Entwicklung im mittleren und höheren Lebensalter: Ergebnisse der Interdisziplinären Längsschnittstudie des Erwachsenenalters (ILSE)*, pages 17–27. Steinkopff, 2000.

[MMS93] Mitchell P Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. Building a Large Annotated Corpus of English: The Penn Treebank. *Computational linguistics*, 19(2):313–330, 1993.

[Moo03] Roger K Moore. A Comparison of the Data Requirements of Automatic Speech Recognition Systems and Human Listeners. In *INTERSPEECH*, pages 2581–2584, 2003.

[NRK99] Hiroaki Nanjo, Akinobu Ri, and Tatsuya Kawahara. Automatic Diagnosis of Recognition Errors in Large Vocabulary Continuous Speech Recognition System. *Joho Shori Gakkai Kenkyu Hokoku*, 99(64):41–48, 1999.

[OENC97] Stefan Ortmanns, Andreas Eiden, Hermann Ney, and Norbert Coenen. Look-Ahead Techniques for Fast Beam Search. In *Proc. of ICASSP*, volume 3, pages 1783–1786, 1997.

[PGB+11] Daniel Povey, Arnab Ghoshal, Gilles Boulianne, Lukas Burget, Ondrej Glembek, Nagendra Goel, Mirko Hannemann, Petr Motlicek, Yanmin Qian, and Petr Schwarz. The Kaldi Speech

Recognition Toolkit. In *Proc. of ASRU workshop*, pages 1–4, 2011.

[PHB+12] Daniel Povey, Mirko Hannemann, Gilles Boulianne, Lukáš Burget, Arnab Ghoshal, Miloš Janda, Martin Karafiát, Stefan Kombrink, Petr Motlicek, Yanmin Qian, et al. Generating Exact Lattices in the WFST Framework. In *ICASSP*, pages 4213–4216. IEEE, 2012.

[PVG+11] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-Learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[RA96] Eric K Ringger and James F Allen. Error Correction via a Post-Processor for Continuous Speech Recognition. In *Proc. of ICASSP*, volume 1, pages 427–430. IEEE, 1996.

[RGH+09] David Rybach, Christian Gollan, Georg Heigold, Björn Hoffmeister, Jonas Lööf, Ralf Schlüter, and Hermann Ney. The RWTH Aachen University Open Source Speech Recognition System. In *Proc. of Interspeech*, pages 2111–2114, 2009.

[RJ86] Lawrence R Rabiner and Biing-Hwang Juang. An Introduction to Hidden Markov Models. *ASSP Magazine*, 3(1):4–16, 1986.

[SB00] Kåre Sjölander and Jonas Beskow. Wavesurfer - An Open Source Speech Tool. In *Proc. of Interspeech*, pages 464–467, 2000.

[Sch02] Tanja Schultz. Globalphone: A Multilingual Speech and Text Database Developed at Karlsruhe University. In *Interspeech*, pages 345–348, 2002.

[SCK+85] R Schwartz, YL Chow, Ol Kimball, S Roucos, M Krasner, and J Makhoul. Context-Dependent Modeling for Acoustic-Phonetic Recognition of Continuous Speech. In *Proc. of ICASSP*, volume 10, pages 1205–1208. IEEE, 1985.

[SCT07] The NIST Speech Recognition Scoring Toolkit (SCTK) Version 2.4.0. http://www.nist.gov/speech/tools, 2007.

[SDMW06] Frederik Stouten, Jacques Duchateau, Jean-Pierre Martens, and Patrick Wambacq. Coping with Disfluencies in Spontaneous Speech Recognition: Acoustic Detection and Linguistic Context Manipulation. *Speech Communication*, 48(11):1590–1606, 2006.

[SGVS13] Tim Schlippe, Lukasz Gren, Ngoc Thang Vu, and Tanja Schultz. Unsupervised Language Model Adaptation for Automatic Speech Recognition of Broadcast News using Web 2.0. In *Proc. of Interspeech*, pages 2698–2702, 2013.

[SLY11] Frank Seide, Gang Li, and Dong Yu. Conversational Speech Transcription Using Context-Dependent Deep Neural Networks. In *Proc. of Interspeech*, pages 437–440, 2011.

[SM06] Emilian Stoimenov and John McDonough. Modeling Polyphone Context with Weighted Finite-State Transducers. In *Proc. of ICASSP*, volume 1, pages 121–124, 2006.

[SMFW01] Hagen Soltau, Florian Metze, Christian Fügen, and Alex Waibel. A One-Pass Decoder Based on Polymorphic Linguistic Context Assignment. In *Proc. of ASRU workshop*, pages 214–217, 2001.

[SMW01] Bernhard Suhm, Brad Myers, and Alex Waibel. Multimodal Error Correction for Speech User Interfaces. *ACM transactions on computer-human interaction (TOCHI)*, 8(1):60–98, 2001.

[SOS10] Tim Schlippe, Sebastian Ochs, and Tanja Schultz. Wiktionary as a Source for Automatic Pronunciation Extraction. In *Proc. of Interspeech*, pages 2290–2293, 2010.

[SS09] Hagen Soltau and George Saon. Dynamic Network Decoding Revisited. In *Proc. of ASRU workshop*, pages 276–281, 2009.

[SSK10] Hagen Soltau, George Saon, and Brian Kingsbury. The IBM Attila Speech Recognition Toolkit. In *Proc. of SLT*, pages 97–102, 2010.

[SSN10] Volker Steinbiss, Martin Sundermeyer, and Hermann Ney. Direct Observation of Pruning Errors (DOPE): A Search Analysis Tool. In *Proc. of Interspeech*, pages 214–217, 2010.

[Sto02] Andreas Stolcke. SRILM —An Extensible Language Modeling Toolkit. In *Proc. of Interspeech*, pages 901–904, 2002.

[SVS13] Tanja Schultz, Ngoc Thang Vu, and Tim Schlippe. GlobalPhone: A Multilingual Text & Speech Database in 20 Languages. In *Proc. of ICASSP*, 2013.

[SW99] Bernd Souvignier and Andreas Wendemuth. Combination of Confidence Measures for Phrases. In *les actes de IEEE Au-*

tomatic Speech Recognition and Understanding Workshop, Budapest, Hungary, pages 217–220, 1999.

[TWG+14] Dominic Telaar, Michael Wand, Dirk Gehrig, Felix Putze, Christoph Amma, Dominic Heger, Ngoc Thang Vu, Mark Erhardt, Tim Schlippe, Matthias Janke, Christian Herff, and Tanja Schultz. BioKIT - Real-Time Decoder for Biosignal Processing. In *Proc. of Interspeech*, pages 2650–2654, 2014.

[VGBP13] Karel Veselỳ, Arnab Ghoshal, Lukás Burget, and Daniel Povey. Sequence-Discriminative Training of Deep Neural Networks. In *Proc. of Interspeech*, pages 2345–2349, 2013.

[VIP+14] Ngoc Thang Vu, David Imseng, Daniel Povey, Petr Motlicek, Tanja Schultz, and Hervé Bourlard. Multilingual Deep Neural Network based Acoustic Modeling For Rapid Language Adaptation. In *Proc. of ICASSP*, 2014.

[VKS11] Ngoc Thang Vu, Franziska Kraus, and Tanja Schultz. Rapid Building of an ASR System for Under-Resourced Languages Based on Multilingual Unsupervised Training. In *Interspeech*, pages 3145–3148. Citeseer, 2011.

[VLW+12] Ngoc Thang Vu, Dau-Cheng Lyu, Jochen Weiner, Dominic Telaar, Tim Schlippe, Fabian Blaicher, Eng-Siong Chng, Tanja Schultz, and Haizhou Li. A First Speech Recognition System for Mandarin-English Code-Switch Conversational Speech. In *Proc. of ICASSP*, pages 4889–4892. IEEE, 2012.

[VMG04] Petko Valtchev, Rokia Missaoui, and Robert Godin. Formal Concept Analysis for Knowledge Discovery and Data Mining: The New Challenges. In *Concept lattices*, pages 352–371. Springer, 2004.

[WBAS03] Tracy Westeyn, Helene Brashear, Amin Atrash, and Thad Starner. Georgia Tech Gesture Toolkit: Supporting Experiments in Gesture Recognition. In *ICMI*, pages 85–92, 2003.

[Wei98] Robert L. Weide. The CMU Pronunciation Dictionary, release 0.6, 1998.

[WFT+16] Jochen Weiner, Claudia Frankenberg, Dominic Telaar, Britta Wendelstein, Johannes Schröder, and Tanja Schultz. Towards Automatic Transcription of ILSE –an Interdisciplinary Longitu-

dinal Study of Adult Development and Aging. In *submitted to LREC*, 2016.

[WLK+04] Willie Walker, Paul Lamere, Philip Kwok, Bhiksha Raj, Rita Singh, Evandro Gouvea, Peter Wolf, and Joe Woelfel. Sphinx-4: A Flexible Open Source Framework for Speech Recognition. *Sun Microsystems Technical Report*, 2004.

[WMS98] Frank Wessel, Klaus Macherey, and Ralf Schlüter. Using Word Probabilities as Confidence Measures. In *Proc. of ICASSP*, volume 1, pages 225–228. IEEE, 1998.

[WN05] Frank Wessel and Hermann Ney. Unsupervised Training of Acoustic Models for Large Vocabulary Continuous Speech Recognition. *Speech and Audio Processing, IEEE Transactions on*, 13(1):23–31, 2005.

[XPMZ11] Haihua Xu, Daniel Povey, Lidia Mangu, and Jie Zhu. Minimum Bayes Risk Decoding and System Combination Based on a Recursion for Edit Distance. *Computer Speech & Language*, 25(4):802–828, 2011.

[YEG+06] Steve J Young, Gunnar Evermann, Mark J.F. Gales, Dan Kershaw, Gareth Moore, Julian J. Odell, Dave G. Ollason, Dan Povey, Valtcho Valtchev, and Phil C. Woodland. The HTK Book Version 3.4. *Cambridge University*, 2006.

[ZC98] George Zavaliagkos and Thomas Colthurst. Utilizing Untranscribed Training Data to Improve Performance. In *DARPA Broadcast News Transcription and Understanding Workshop*, pages 301–305. Citeseer, 1998.

Publications

1. Dominic Telaar, Jochen Weiner, Tanja Schultz, Error Signatures to identify Errors in ASR in an unsupervised fashion, Errare Workshop, 2015
2. Christian Herff, Dominic Heger, Adriana de Pesters, Dominic Telaar, Peter Brunner, Gerwin Schalk, Tanja Schultz, Brain-to-Text: Decoding spoken phrases from phone representations in the brain, Frontiers in Neuroscience, Vol. 9, No. 217, 2015
3. Heike Adel, Ngoc Thang Vu, Katrin Kirchhoff, Dominic Telaar, Tanja Schultz, Syntactic and Semantic Features For Code-Switching Factored Language Models, IEEE/ACM Transactions on Audio, Speech and Language Processing, Vol. 23, No. 3, 2015
4. Dominic Telaar, Michael Wand, Dirk Gehrig, Felix Putze, Christoph Amma, Dominic Heger, Ngoc Thang Vu, Mark Erhardt, Tim Schlippe, Matthias Janke, Christian Herff, Tanja Schultz, BioKIT - Real-time Decoder For Biosignal Processing, Interspeech, 2014
5. Heike Adel, Katrin Kirchhoff, Ngoc Thang Vu, Dominic Telaar, Tanja Schultz, Comparing Approaches to Convert Recurrent Neural Networks into Backoff Language Models For Efficient Decoding, Interspeech, 2014
6. Heike Adel, Kartin Kirchhoff, Dominic Telaar, Ngoc Thang Vu, Tim Schlippe, Tanja Schultz, Features for Factored Language Models for Code-Switching Speech, Interspeech 2014
7. Heike Adel, Dominic Telaar, Ngoc Thang Vu, Katrin Kirchhoff, Tanja Schultz, Combining Recurrent Neural Networks and Factored Language Models During Decoding of Code-Switching Speech, Interspeech, 2014
8. Dominic Telaar, Mark C Fuhs, Accent- and Speaker-Specific Polyphone Decision Trees for Non-Native Speech Recognition, Interspeech, 2013
9. Ngoc Thang Vu, Dau-Cheng Lyu, Jochen Weiner, Dominic Telaar, Tim Schlippe, Fabian Blaicher, Eng-Siong Chng, Tanja Schultz, Haizhou Li, A First Speech Recognition System For Mandarin-English Code-Switch Conversational Speech, ICASSP, 2012
10. Jochen Weiner, Ngoc Thang Vu, Dominic Telaar, Florian Metze, Tanja Schultz, Dau-Cheng Lyu, Eng-Siong Chng, Haizhou Li, Integration Of Language Identification Into A Recognition System For Spoken Conversations Containing Code-Switches, SLTU, 2012