Karlsruhe Institute of Technology

Temporal Patterns (TRAPs) in Janus Recognition Toolkit

Student Project of

Tatiana Glushkova

At the Department of Informatics
Institute for Anthropomatics (IFA)
Prof. Dr. Alex Waibel Department

Reviewer:	Prof. Dr. Alex Waibel
Second reviewer:	Dr. Sebastian Stüker
Advisor:	Kevin Kilgour

Duration:: 01.08.2012 – 31.10.2012

Ich erkläre hiermit, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Karlsruhe, den 31. Oktober 2012

Acknowledgments

This project would not have been possible without the kind support and help of many people. I would like to extend my sincere thanks to Prof. Alex Waibel, Dr. Sebastian Stüker and Prof. Florian Metze for the opportunity to work on this project as well as for the inspiring and guiding discussions. I wish to express my special thanks to Yury Titov for sharing his enthusiasm and knowledge as well as for supervising me at the first steps in this field. My deepest gratitude is for Kevin Kilgour for overtaking the supervising, for helping to complete the experiments and to arrange the text of this project. Additionally I would like to thank Udhay Nallasamy for providing his sample scripts and helping to understand them. My thanks and appreciations also go to my student fellows Henrich Kolkhorst, Jochen Weiner and Alexey Mokin for the cheerful atmosphere in the lab as well as for the helpful advice. Finally, I wish to express my heartfelt thanks to my family and friends for their patience, support and optimism.

Abstract

In the last years application of artificial neural networks (ANN) in the acoustic modeling (AM) of automatic speech recognition (ASR) systems has drawn serious attention. One of these techniques is TempoRAl Patterns (TRAP) which allows to extract input features for a gaussian mixture model - hidden markov model (GMM-HMM) recognizer. Showing good experimental results, TRAPs feature extraction method was included in the state-of-the-art systems for large vocabulary continuous speech recognition (LVCSR) as a standard component [1], [2].

The objective of this student project is to reproduce the basic TRAPs technique and to try out some of its modifications for the Janus recognition toolkit (JRTK).

The most of the published experiments with TRAPs are done on English data. Therefore it is also interesting to study how well this technique would work for other languages. In this project experiments were done for Arabic and German languages.

Contents

1. Introduction

Automatic speech recognition (ASR) technology receives much interest in both scientific circles as well as in wide public. The main advantages of the communication between people and computers through speech are naturality, speed and portability. That is why there are so many different applications from simple speech commands to advanced speech-to-speech translation.
One of the ASR applications is data mining in spoken data. An example is the project Quaero `www.quaero.org`, [3], which aims to develop indexing and information retrieval tools for multilingual multimedia. This opens up possibilities to search not only in text but also in audio and video data on the Internet.
Another application is the Lecture Translation project [4]. This project has an objective of providing simultaneous translation of lectures that will support foreign students. The lecturer's speech will be first recognized and converted to text. Then the text will be translated into the target language and presented to the audience e.g. on a display.
Although ASR applications are computationally intensive in general, as the examples above, some of them are capable of running on mobile devices. The smartphone application Jibbigo, `http://www.jibbigo.com`, provides speech-to-speech translation to twelve different languages. Its portability is especially useful for travelers.
All these applications are results of many years of profound research in speech technologies. In spite of the current achievements, there are still many challenges. One of them is to make ASR systems more robust and accurate. To achieve this goal, many different techniques are being developed to improve each individual part of a system. This project is focused on improving the acoustic model of an ASR system by using artificial neural networks.
The project thesis is organized as follows. Chapter 1 gives a brief background on typical ASR system components and its training cycle with an emphasis on the preprocessing and the acoustic modeling. Chapter 2 summarizes the previous work on feature extraction techniques used in the preprocessing. It introduces the Temporal Patterns (TRAP) and the bottle-neck (BN) techniques. It also presents some of their modifications in architecture and processing. Chapter 3 is devoted to the experimental part of the project. It describes the baseline system, the configurations of chosen experimental setups and the implementation details. Chapter 4 presents and interprets the experimental results. It also mentions problems and challenges, ideas for improvement and suggestions for future work. Chapter 5 summarizes the work done on this project and draws conclusions from the analysis of the results.

1.1. ASR System

This paragraph presents a typical statistical ASR system, its components and training cycle. As it may be seen in the figure 1.1, an ASR system contains a preprocessing component (also called a front end) and a decoder (or a recognizer). The decoder uses information from an acoustic model (AM), a pronunciation dictionary and a language model (LM).

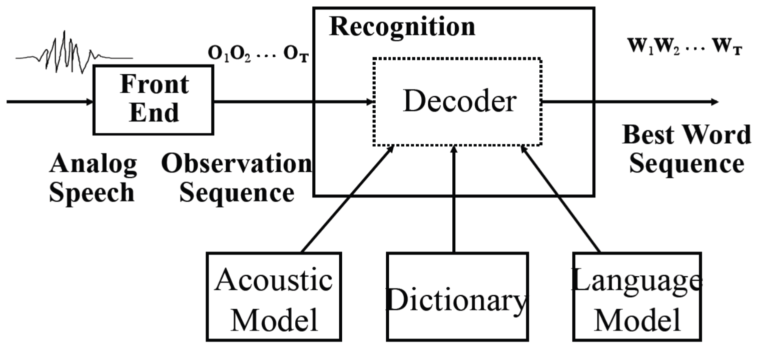

Figure 1.1.: Components of an ASR System, [5].

Recognition process runs as follows. The front end receives analog speech, converts it into a digital format and applies several transformations to eliminate the unnecessary information and to convert the remained essential information into a format suitable for the decoder. This forms an input feature vector (or an observation sequence) for the decoder. Given the extracted input feature vector, the decoder searches for the most probable word sequences and outputs one or several of the best found sequences. The probability of a word sequence given a sound observation is provided by the acoustic and the language models using information from the dictionary. A sound observation consists of utterances that corresponds to phonemes of the word.
Thus the AM delivers probability of a phoneme given an utterance. The dictionary contains information on how to build words from phonemes. The LM gives the probability of a word given several previous ones.
The probability of a word given an utterance can be computed using the Bayes rule. This gives the fundamental formula of the statistical ASR:

$$w^* = \arg\max_w p(w|X) = \arg\max_w \frac{p(X|w)*P(w)}{p(X)} = \arg\max_w p(X|w)*P(w)$$

Generally, this is a classification problem where pattern X is given and to find is the class w^* with the maximum probability given the observed pattern. In our case the pattern is an utterance and classes are words.
The X here stands for an utterance. The word w^* maximizes the conditional probability density function $p(w|X)$ over all the words w. Due to the Bayes rule, this density can be expressed through the inverted conditional density function $p(X|w)$ and the probabilities of the variables it contains. Since the maximization is being done over w, the $P(X)$ is a constant and therefore can be omitted. So, the w^* depends only on $p(X|w)$ and $P(w)$. As it was mentioned above, the conditional density function $p(X|w)$ is computed by the AM and the probability of a word $P(w)$ by the LM.
Since the number of possible words is countably infinite, a smart searching strategy should be applied to prune the search space. This strategy reveals the promising word sequences and eliminates the improbable ones. The best suitable word sequence will be found after searching among the most probable word sequences that are generated using the AM, the

dictionary and the LM.
This was a general overview of the whole system. For better understanding the techniques used in this project, front end component and acoustic model will be examined in detail.

1.2. Preprocessing and Acoustic Modeling (AM)

This project is focused on the preprocessing step and, since it influences the AM configuration, also on AM training. The preprocessing step contains the transformation from the speech recorded in a sound file into the features suitable for the AM input.

Typical preprocessing steps are the following. An analog speech is recorded and digitized, having the 16 kHz sampling rate and the 16 bit resolution. The speech will be then cut in small fragments by applying some window, e.g. overlapping Hamming windows with the width of 16 ms and the shift of 10 ms. Each of these fragments forms a frame.
For each frame a discrete Fourier-transformation (FT) will be done and a power spectrum will be calculated. After that, Mel-scale filterbanks will be applied and logarithm will be taken. Then, cepstral coefficients will be gained by applying an inverse FT or a discrete cosine transformation (DCT). Obtained features are mel-frequency cepstral coefficients (MFCC). In order to make these features more suitable for a decoder, the linear discriminant analysis (LDA) will be used. LDA is a linear feature space transformation that makes different classes better separated from each other. Here the classes are some phonetic units, e.g. phonemes. The features that correspond to the same phoneme became more compact being more distant to the features of other phonemes. So, the gained MFCC-LDA features are ready input for a decoder.

The acoustic model consists of a hidden markov model (HMM) used with Gaussian mixture models (GMM) to represent emission probabilities of HMM states. A recognizer that employs such AM is called HMM-GMM recognizer.
There are phoneme HMMs and word HMMs in the AM as shown in the figure 1.2. Phoneme HMM have often 3 states corresponding to the beginning, middle and end part of a phoneme. Emission probabilities for all these states are modeled by GMMs. Word HMMs are the combinations of phoneme HMMs into complete words. Several concatenated word HMMs represent a word sequence. The word transitrion in this sequence are chosen by the LM.

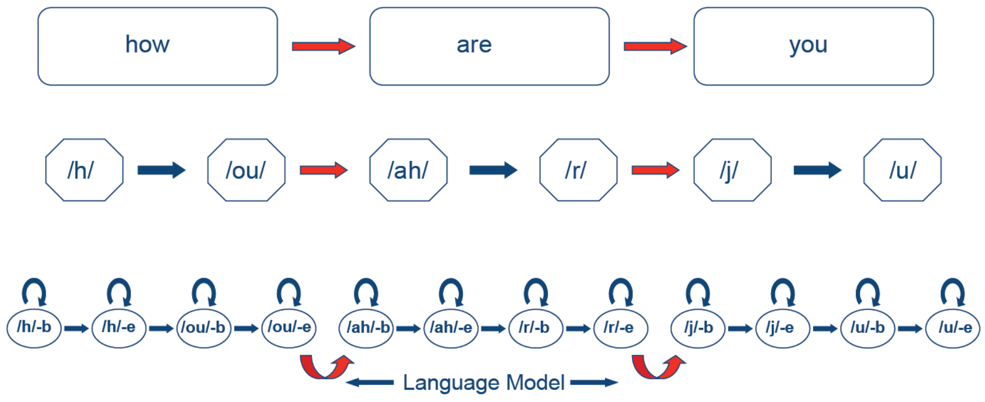

Figure 1.2.: HMMs in acoustic modeling, [5].

There are several ways to represent a GMM. The same mixture of Gaussins can be used for many different HMM states. In this case there is a number of Gaussian mixtures called codebooks. Each HMM state can use one of these codebooks, adjusting it by mixture coefficients that weight each Gaussian in the codebook. Thus different states can share the same distributions as shown in the figure 1.3.

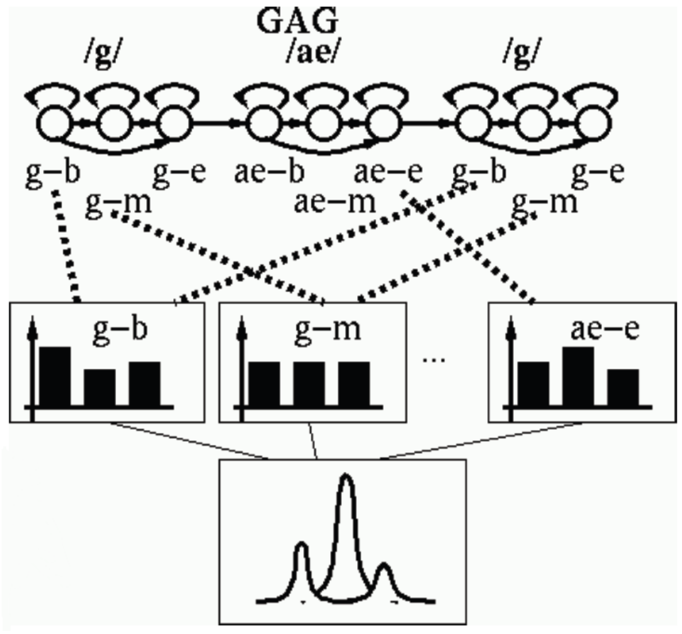

Figure 1.3.: GMMs in acoustic modeling, [5].

1.3. Training Cycle

Typical training cycle of an ASR system is shown in the figure 1.4. The following paragraphs roughly describe the steps should be performed to train a complete system. The focus here lies on the AM training that was used in the experiments of this project.

First of all, training data for AM and LM should be collected. AM training data is usually about 100 hours of transcribed recordings from different speakers. LM training data is texts on both specific domains as well as general topics. All the data should be separated into training, cross-validation, developing and evaluation set.

The AM training includes two parts. The first one is a context-independent (CI) model training and the second one is a context-dependent (CD) model training. Before the training, all the necessary configuration should be defined. These are e.g. a set of phonemes, a kind of preprocessing, the maximal number of Gaussians in mixtures, a HMM topology. Then, the forced alignment (or time alignment) can start. This is the process of writing labels to provide the correspondence between training data and HMM states.

The CI acoustic model will be trained by the expectation maximization (EM) algorithm. There are several steps that precede this training. The linear discriminant analysis (LDA) matrix should be computed, that is required for the preprocessing part, and the GMMs should be initialized by the k-means algorithm. After that, the EM training can begin.

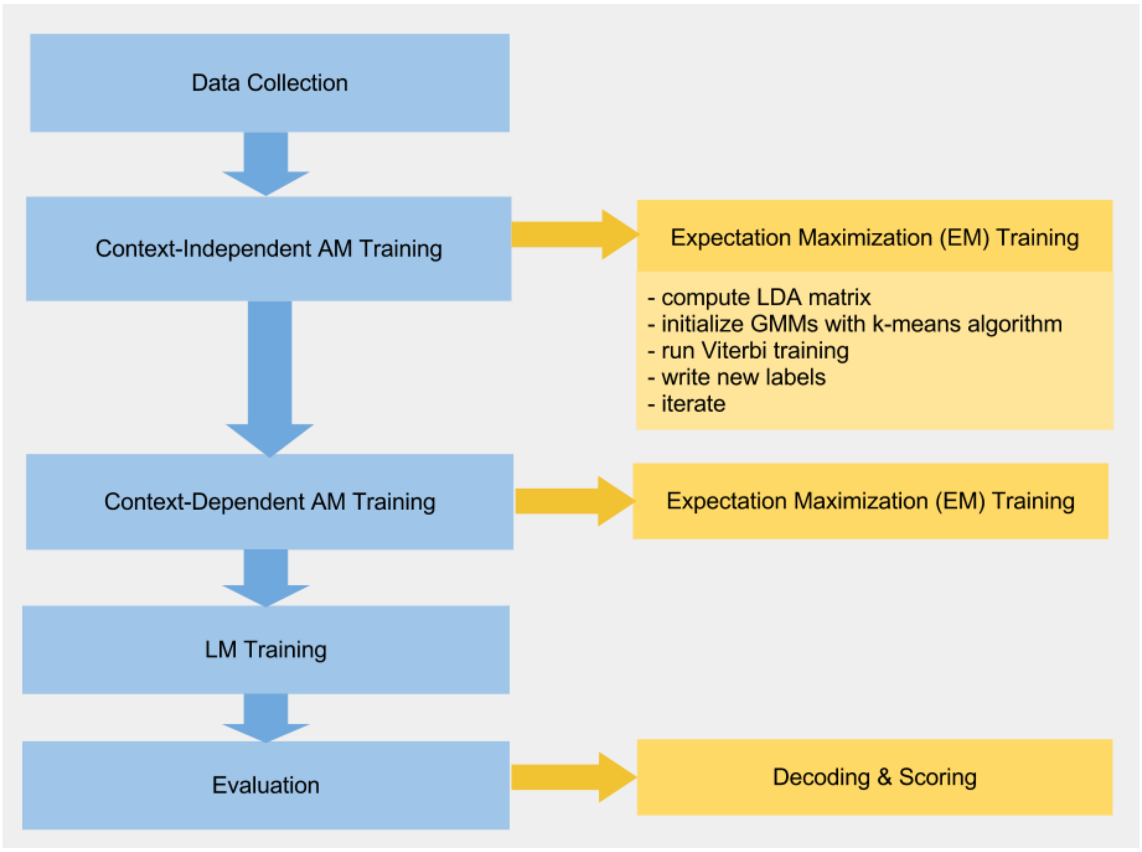
Data Collection
Context-Independent AM Training
Expectation Maximization (EM) Training
- compute LDA matrix
- initialize GMMs with k-means algorithm
- run Viterbi training
- write new labels
- iterate
Context-Dependent AM Training
Expectation Maximization (EM) Training
LM Training
Evaluation
Decoding & Scoring

Figure 1.4.: ASR system training steps

Generally, the EM is an iterative method to find the maximum likelihood parameter estimates for a statistical model that have some unobserved variables. It consists of the expectation step (E-step), which computes the expectation function with the current parameters, and maximization step (M-step), which computes the parameters that will maximize the expectation function founded in the previous step. These two steps should iterate in the interchanging way.
In the AM training, the Viterbi training corresponds to the M-step and writing new improved labels corresponds to the E-step. After several iteration, the CI acoustic model is trained.

While the context-independent AM has 3 HMM states for the phoneme parts, the context-independent uses the parts of a polyphone instead. A polyphone represents a single phoneme with the influence of the previous and succeeding phonemes. Often this context is formed by one preceding and one following phoneme. There is a big number of possible polyphones. That is why they will be first grouped by their context forming a decision tree which is called polyphone tree.
After the polyphone tree is created, the CD acoustic model training proceed analogous to the CI training.

Having the AM and LM models trained, there are still some parameters that should be adjusted before the system is ready to decode. For example, to get a good balance between the AM and the LM, their scores can be adjusted by the appropriate weights found on the development data.

Finally, the performance of the trained system will be measured. The ASR system evaluation consists of two steps. The first step is the decoding of some test set. The text, that is obtained by the system from the given test speech, is called hypotheses. The second step is the scoring, where the hypotheses will be compared with the transcription of the test set.
In order to measure the recognition quality, the word error rate (WER) will be computed. The system can be then compared to its previous version or to another system. WER is based on the minimal editing distance between the automatically generated hypothesis and the human reference of the same speech. This can be calculated by the following formula:

$$WER = \frac{I+D+S}{R}$$

where I is the number of insertions, D is the number of deletions, S is the number of substitutions and R is the number of words in the reference.

2. TRAPs and BN: Related Work

The preprocessing at the front end can include several feature extraction techniques. The goal of these techniques is to extract important information and to eliminate unnecessary one from a raw sound file providing a proper input for a recognizer. Typically a GMM-HMM recognizer is used.
There are two categories of feature extraction techniques: the ones delivering acoustic features and the others delivering probabilistic features. Acoustic features are obtained by applying to raw speech a set of acoustic transformations without using of classifiers. Examples of acoustic features are mel-frequency cepstral coefficients (MFCC), perceptual linear prediction (PLP) features and minimum variance distortionless response (MVDR) features.
Probabilistic features are obtained by a classifier that provides posterior probabilities of phonetically motivated classes given the input utterance. The phonetic class is either a phoneme or a sub-phoneme. So, the probabilistic features are not based on acoustic transformations but consist of probabilities for phonetic units.
Both acoustic and probabilistic features should be post-processed before they can be used as an input for a GMM-HMM recognizer. Since the emission probability of an HMM state has a Gaussian distribution, the features should also have a similar distribution. In order to achieve this, logarithm of the features is taken. Then, dimensionality reduction and decorrelation techniques are applied. This allows to obtain the required size and to make the feature space better separable for the classifier. The most common techniques for post-processing are linear discriminant analysis (LDA), kernel LDA, heteroscedastic linear discriminant analysis (HLDA), principal component analysis (PCA), kernel PCA, discrete cosine transformation (DCT) and bottle-neck (BN) techniques.

In the TANDEM approach [6], [7], it was proposed to use two different classifiers consecutively. The output of the first stage classifier is used as the input to the second one. This is outlined by the figure 2.1 that shows the front-end, the first stage classifier, its post-processing and input features for the second classifier that is commonly a standard GMM-HMM recognizer. The first stage classifier is often based on an artificial neural network (ANN) and produces posterior probabilities of phonetic classes. First stage classifiers based on TRAP and on BN techniques will be discussed in the following sections.

Both TRAP and BN techniques use a multilayer perceptron (MLP) that belongs to the class of feed-forward artificial neural networks. Its network is a directed graph with several

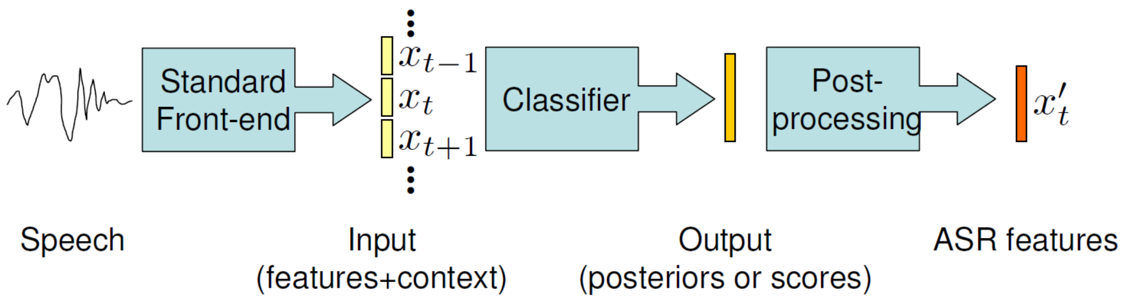

Figure 2.1.: TANDEM Approach, [8].

layers of nodes. The neighboring layers are fully interconnected. Each node, except for the input nodes, is a neuron with a nonlinear activation function. Training of the network is realized by the back-propagation technique which is a supervised learning technique. Training data should therefore have corresponding class (or target) labels.

2.1. TempoRal Patterns (TRAPs)

The idea to use long temporal context in narrow frequency sub-bands was first introduced in [9]. The figure 2.2 illustrates the difference between the classical features based on full spectrum with short time context and the ones derived from narrow spectrum band with long time context.

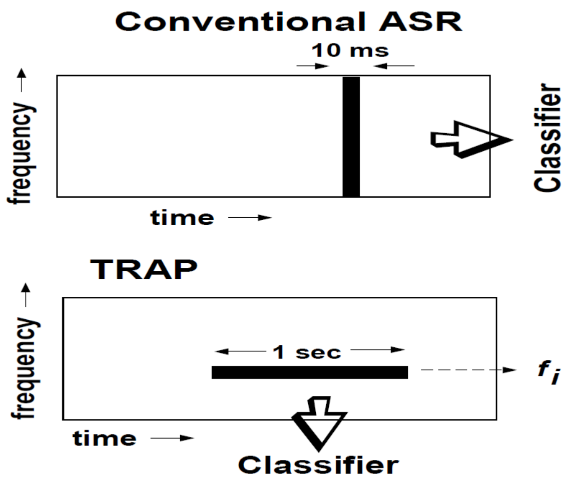

Figure 2.2.: Temporal Paradigm of ASR, [9].

TempoRal Patterns (TRAPs) are feature vectors gained by two stages of MLP classifiers on a long temporal context. The concept of TRAP processing refers to preprocessing of raw features before they are used as an input to the first stage MLPs. The term TRAP architecture implies the architecture of MLPs.
Research on TRAPs has been conducted for more than ten years. Profound theory and experiments in this field can be found in the doctoral theses [10] and [11]. Nowadays TRAPs have became a state-of-the-art component of ASR systems [2], [1].
According to [12], the basic TRAP vector can be obtained in the following way. Raw TRAP features are derived from temporal trajectory of spectral energy in frequency sub-bands (or critical bands). Critical band energies (CRBE) can be obtained by segmenting

input speech sampled at the 16 kHz into 25 ms frames shifted by 10 ms. This is followed by calculating the power spectrum, applying a filter bank, mean and variance normalization. Temporal evolution of energy in one critical band within a context of up to 1 second forms a raw TRAP vector. There is one TRAP vector per each critical band in a frame.
At the first stage there are several MLPs called band classifier MLPs. There is one MLP per critical band. Given raw TRAP vectors as an input band classifiers compute phoneme probabilities. The outputs from all the band classifiers are then concatenated into one vector and a logarithm of this resulting vector is taken. This forms the input for the MLP of the next stage which is called main merger MLP. It combines all the band-conditioned probabilities into one final TRAP vector. After the post-processing, which includes logarithmization and feature space transformation, this TRAP vector is used as an input for a GMM-HMM recognizer.

2.2. Bottle-Neck (BN) Features

The bottle-neck (BN) technique is based on the specific MLP architecture with a narrow middle layer which is called a bottle-neck. As opposed to other MLP techniques used in ASR, BN does not provide posterior probabilities for an input feature vector, but reduces its dimensionality. BN is thus an effective feature vector reduction technique.

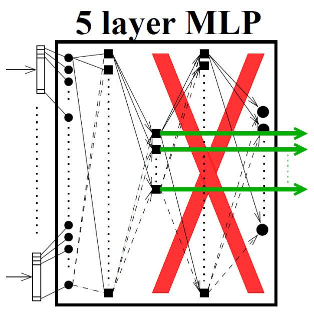

Figure 2.3.: Bottle-neck architecture, [13].

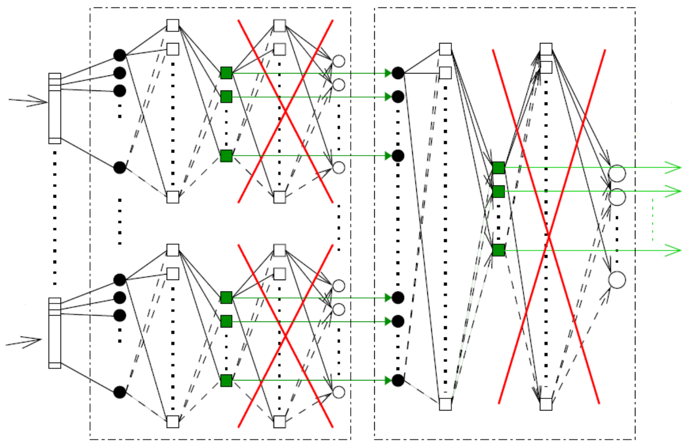

Figure 2.4.: Hierarchical Bottle-neck Architecture, [1].

As illustrated by the figure 2.3 a 5-layered MLP has a narrow 3rd layer, large 2nd and 4th layers of the same size. After the whole MLP is trained, only the first 3 layers will be used. The bottle-neck layer's output will be taken as the output of the network. Thereby

an input vector will be reduced to the size of the bottle-neck layer. This allows to choose the size of the bottle-neck layer as required for the dimensionality reduction.
The BN technique can be successfully combined with TRAPs as confirmed by the good experimental results in [14], [13], [15].
Hierarchical BN is a more advanced type of BN which has hierarchical or multi-stage networks with a bottle-neck layer in each network. An example of this network architecture is shown in the figure 2.4.

2.3. TRAP Modifications

Due to many attempts to enhance and to develop the TRAP technique many various TRAP modifications exist. Generally, TRAP modifications can be classified by the following criteria:

- Processing
  - basic TRAP processing, figure 2.5
  - dimensionality reduction, figure 2.6
  - multi-band TRAPs, figure 2.6
- Architecture
  - basic TRAP architecture, figure 2.5
  - single stage TRAP architecture, figure 2.6
  - BN-TRAP architecture, figure 2.7
- Phonetic targets used in MLP
  - phonemes
  - sub-phonemes

The processing parts are shown on the left sides of the figures 2.5, 2.6 and 2.7. The gray-scaled field is a spectrogram which represents some raw feature vector in the short time slices. A darker color corresponds to a higher energy. The x-axis represents time while the y-axis represents raw input features by the frequency or its equivalent.
As typical for TRAP processing, long time context is used. In the basic processing only one critical band is taken for the further processing while in the multi-band processing there are e.g. three neighboring bands concatenated in one vector. In the case of 3-band processing it is reasonable to reduce the large MLP input vector by applying some dimensionality reduction technique, e.g. DCT.
The basic TRAP architecture, figure 2.5, has two stages of 3-layered MLPs. Its alternatives could be 3-layered single stage MLPs, figure 2.6, and hierarchical BN MLPs, figure 2.7. This illustrates the search for the balance between a compact network with less parameters and classification performance.
Originally, phonemes were suggested as the MLP targets. It was later experimentally shown that sub-phonemes (or other units that correspond to HMM sub-states) have better performance than usual phoneme targets [16].
The most promising improvements of TRAP technique at the moment are 3-band processing, feature vector reduction with DCT, hierarchical BN architecture and sub-phonemes as classes for MLPs. Combined together they give the BN-TRAP-3band-DCT configuration that shows a good performance according to [1].

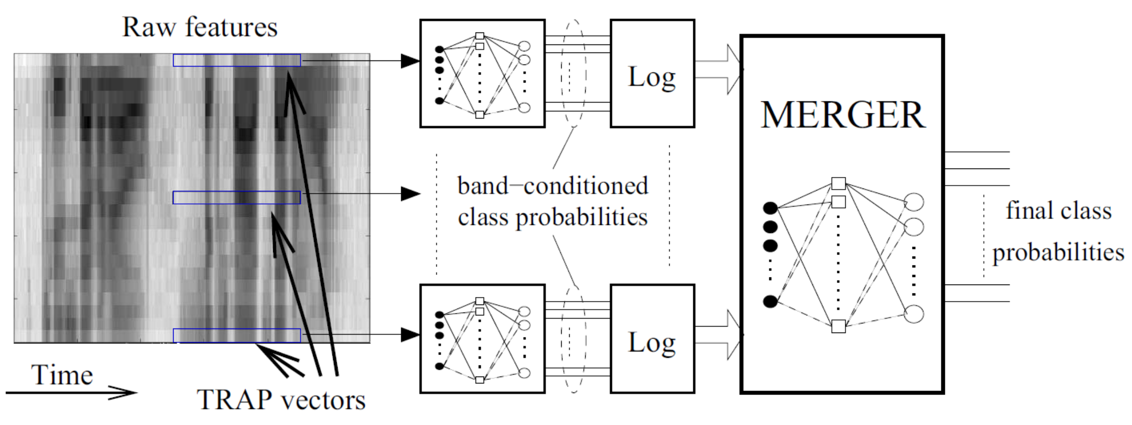
Raw features
Log
Log
MERGER
band−conditioned class probabilities
final class probabilities
Time
TRAP vectors

Figure 2.5.: Basic TRAP architecture, [1].

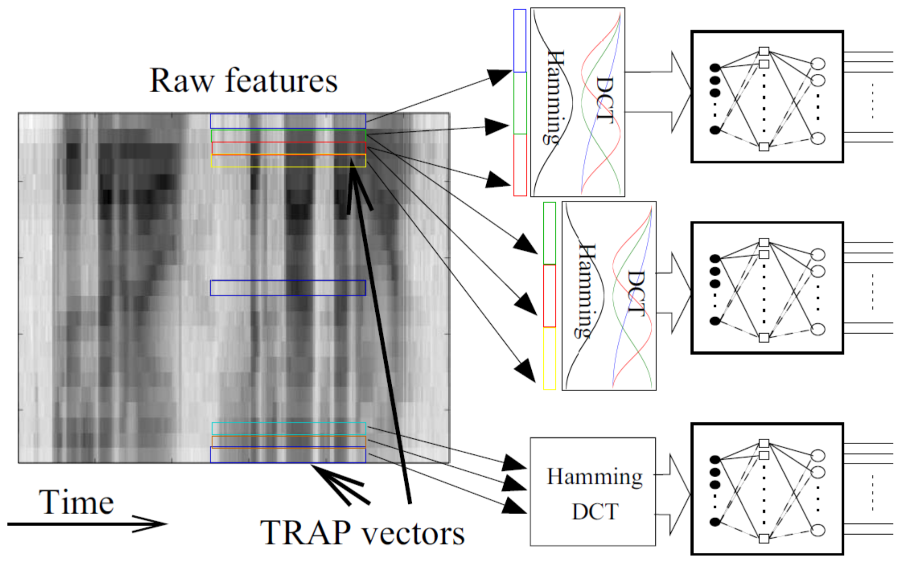

Figure 2.6.: Three band processing and DCT, [1].

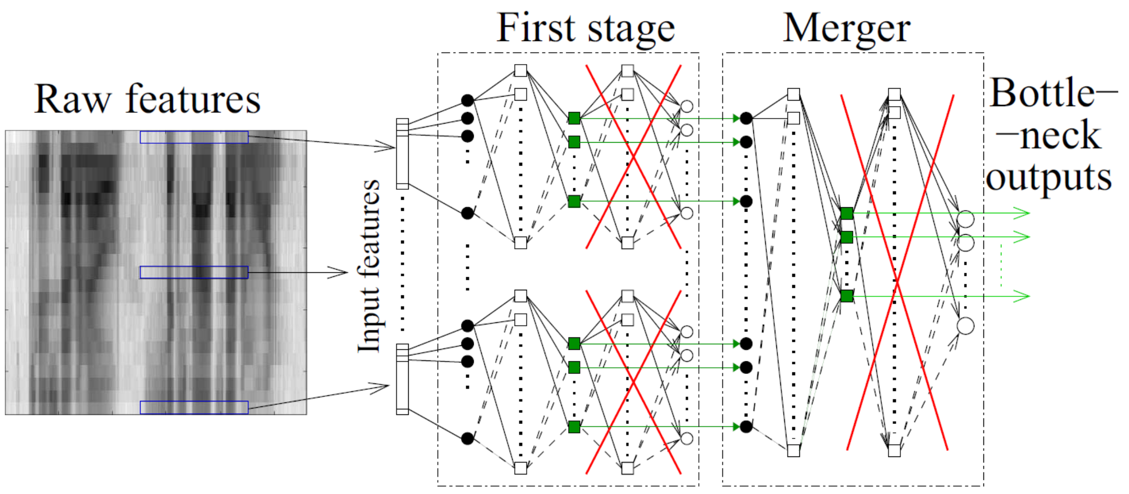

Figure 2.7.: Bottle-neck TRAP architecture, [1].

3. Experimental Setups

In this project the basic TRAP technique was reproduced. Inspired by the good results of the 3band-DCT-BN-TRAP setup in [1], an attempt was also made to reproduce a more advanced configuration. The TRAP-BN architecture with basic processing and sub-phonemes as MLP targets was chosen for this purpose.
Most publications present experiments on English data. Here, the experiments were done for another two languages - Arabic and German - to see if this technique also works well for these languages.
There are many possible parameter configurations for architecture and processing that can be combined arbitrarily with each other. Below are the examples of parameters and their values:

- raw features: MFCC, lMEL, PLP, MVDR
- MLP architecture: basic 3 layers, 4- or 5-layered BN
- MLP targets: phonemes, sub-phonemes
- band processing: 1-band, 3-band
- dimensionality reduction of the TRAP vector: PCA, DCT, no reduction
- TRAP context length: 25 frames, 50 frames
- post-processing: LDA, HLDA, PCA

The parameters chosen for the setups in this project are: basic TRAP processing with the lMEL as the raw feature, context of 25 frames; basic 3-layered MLPs and 5-layered BN MLPs; LDA as post-processing. The first two trial setups were trained on conversational Arabic data (50 hours) and the next two setups were trained on conversational German data (180 hours).
The basic TRAP setup for German was then used for AM training. The new AM was used with the already existing ASR system components to perform the decoding. A setup with the MFCC-based AM was chosen as a baseline for performance comparison.

3.1. MFCC Baseline

The workflow of MFCC-based feature extraction, used in the baseline setup, is shown in the figure 3.1. The frames here are 16 ms long, shifted by 10 ms and weighted by a Hamming window. For each frame a power spectrum is calculated, mel-scale filter-bank with 30 filters is applied, a logarithm is taken and a DCT is performed. As a result a feature vector with 13 cepstral coefficients is formed for each frame. After reducing this vector by the LDA on the context of 7 previous and 7 following frames, the input vector for the decoder is obtained.

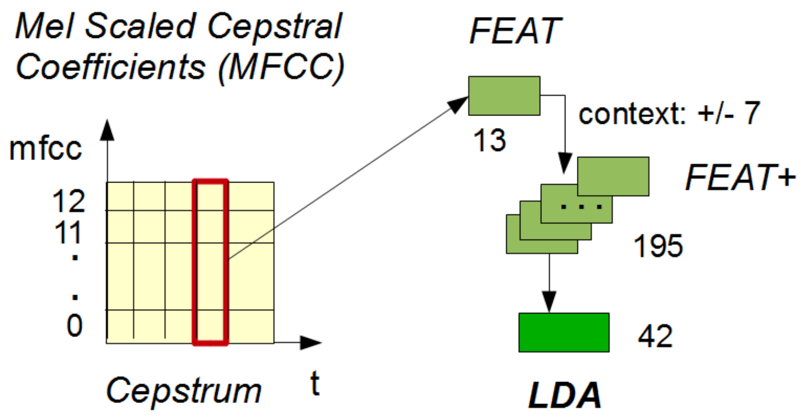

Figure 3.1.: Baseline Architecture.

3.2. Basic TRAP Architecture

The basic TRAP architecture is shown in the figure 3.2. The workflow is similar to the MFCC-based feature extraction workflow. The first steps are the same until the application of DCT, whereas the following steps differ from the baseline by the way of forming a feature vector and by using MLPs.
The filter banks are applied to get the sub-bands (or critical bands). The mel-scale filter-bank has here 30 filters that gives 30 critical bands. For each band a raw TRAP vector is formed from the 51 coefficients of the corresponding sub-band by the context of ±25 frames. The raw TRAP vectors are given as the inputs to the band classifiers. All of these 30 MLPs have 3 layers with 51 neurons in the input layer corresponding to the raw TRAP vector size and 50 neurons in the output layer corresponding to the number of target classes. The number of neurons in the hidden layers has to be calculated, see section 3.4. The MLP targets here are the 41 German phonemes:
A AEH AH AI AU B CH X D E E2 EH ER EU F G H I IE J K L M N NG O OE OEH OH P R S SCH T TS U UE UEH UH V Z.
Considering the noise classes and the MLP training tool requirements gives 50 different classes in total.
The outputs of all the band classifiers are concatenated into one vector that is given as the input for the main merger MLP. This MLP has 3 layers with 1500 neurons in the input layer and 50 neurons in the output layer corresponding to the number of targets. Again, the number of neurons in the hidden layers has to be calculated, see section 3.4. The output of this network delivers the final posterior probability for each phoneme in a given frame. According to the soft decision function used in this MLP, an output is a vector with the values in (0, 1). The coefficients of this vector should have a distribution similar to a Gaussian to be a suitable input for the GMM-HMM recognizer. To achieve this the logarithm of this vector is taken that smoothens and expands the value range. The resulting vector is a TRAP feature vector. As it is also done in the baseline, the LDA is applied on the context of ±4 frames giving the desired TRAP-LDA vector that can be fed to the recognizer.

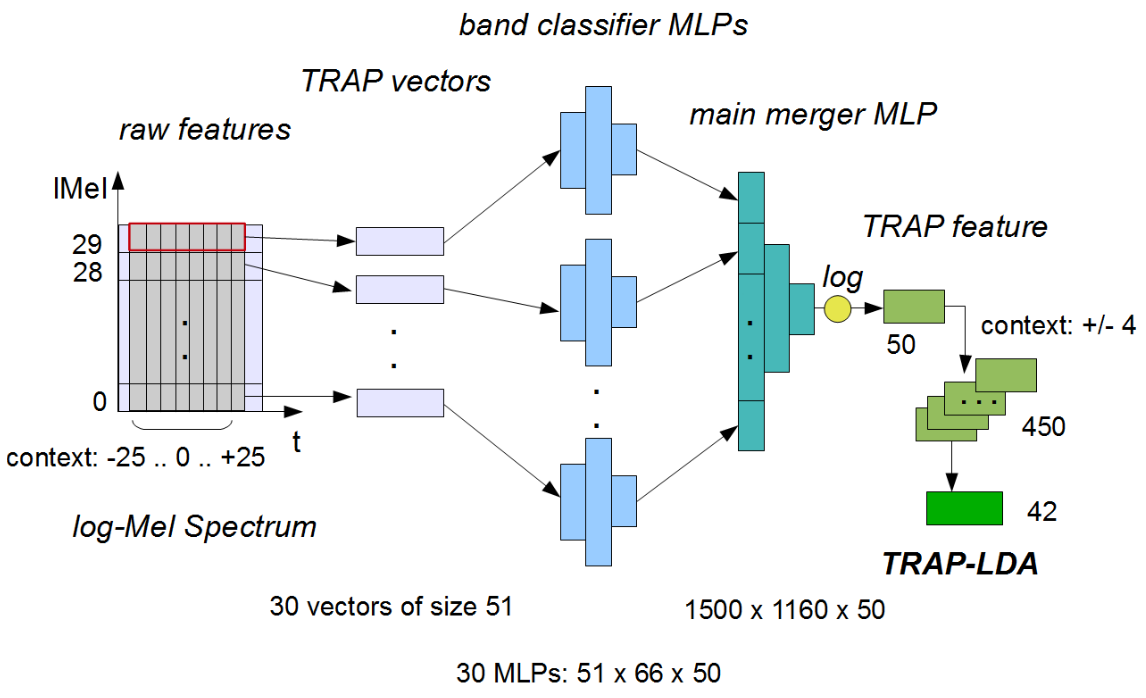

Figure 3.2.: Basic TRAP architecture.

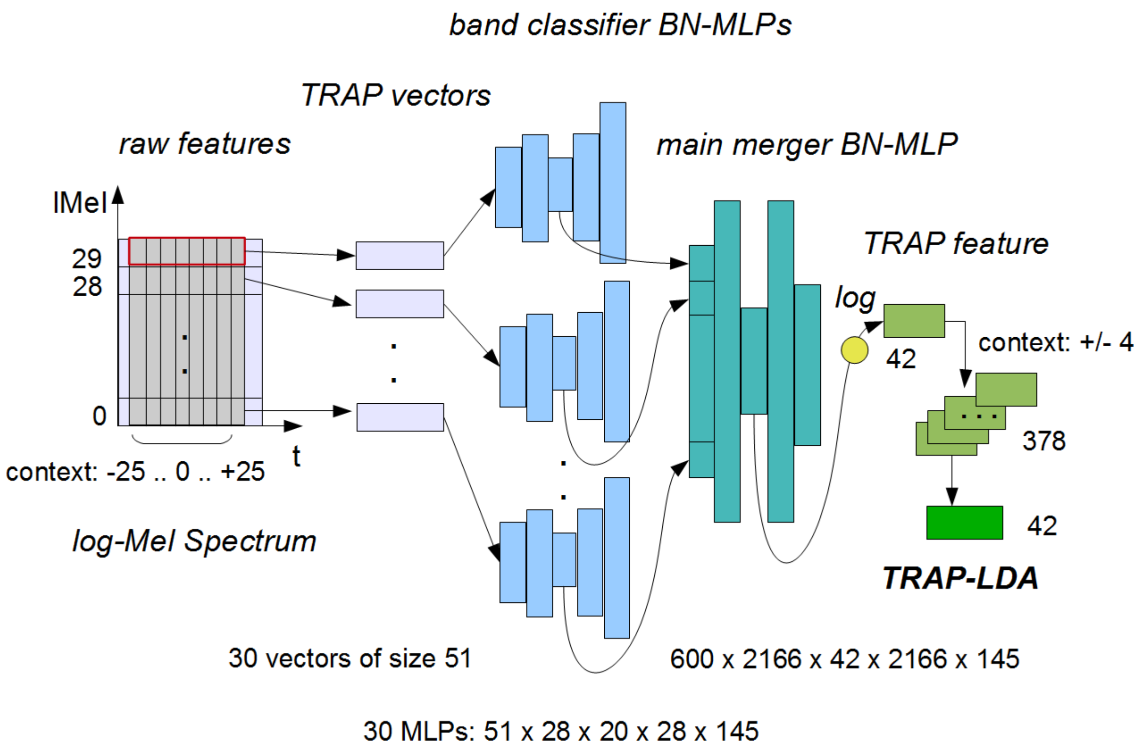

Figure 3.3.: 3band Bottle-neck feature vector extraction.

3.3. TRAP-BN Architecture

The architecture with BN-MLPs and basic preprocessing is shown in the figure 3.3. It is analogous to the basic TRAP architecture and the feature extraction workflow. The difference is only in the MLP architecture and its targets. The targets here are sub-phonemes. Using 48 different phonemes including noises and considering the beginning, middle and end HMM sub-states for every phoneme gives 144 different classes of sub-phonemes. The MLP training tool requirements increase this number to 145 target classes in total.
Every MLP has a 5-layered bottle-neck architecture. The input layers of the band classifiers and the main merger are chosen to correspond to the sizes of their input vectors. The 2nd and 4th layers are hidden layers of the same size calculated in section 3.4. The output layers have 145 neurons corresponding to the number of targets. The narrow bottle-neck layers of the band classifiers and the main merger have 20 and 42 neurons, respectively.

3.4. Implementation and Training

Development of an ASR recognition system consists of two parts: training and testing (or decoding). First, an acoustic model (AM) and a language model (LM) should be trained, then a completely trained system can be used for decoding. The quality of the recognition is normally measured by the word error rate (WER).
The experiments were done using the Janus recognition toolkit (JRTK). The standard scripts for training and testing, general setup configuration and trained LM were already available. The existing AM training was extended with scripts for two stage MLP training and with description files for TRAP-based features. These new description files and scripts were based on the existing ones for the BN features that had been successfully used for the systems in [17] and [18]. For the implementation details of the TRAP feature extraction, the descriptions given in [19] and [1] were used.

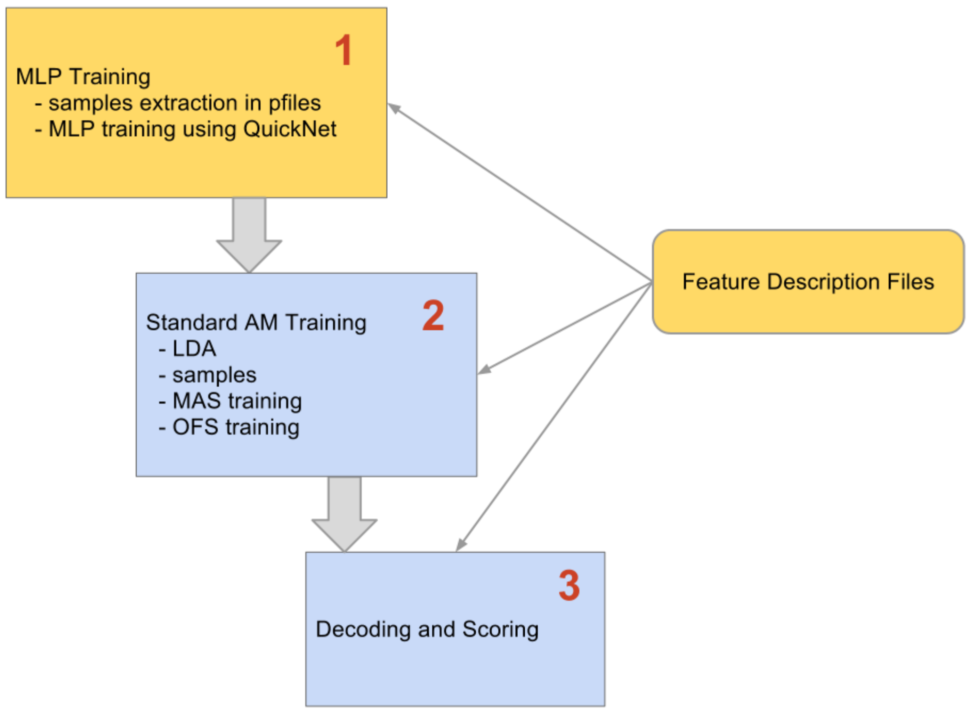

Figure 3.4.: ASR system development

In the figure 3.4 the new components are marked with orange filling. The other components marked with blue color are the ones almost unchanged. A complete setup with the TRAP-based AM consists of three parts. The first part is the two stage MLP training which can be divided into the following steps:

- define the architectures of all MLPs
- create pfliles with training samples for the band classifier MLPs
- train the band classifier MLPs with the QuickNet
- using the trained band classifiers, create pfliles with training samples for the main merger MLP
- train the main merger with the QuickNet

The second part is the standard AM training that uses the MLPs trained in the first part. Finally, the third part is the decoding with the ASR system that includes this trained AM. The produced hypotheses are used for scoring the WER.
Suitable feature description files are required for every part mentioned above. The description files for the TRAP-based features were implemented using the existing functions of Janus. The figure 3.5 shows the pre-processing workflow used in the feature description files. It schematically represents the applied steps as blue boxes and the created objects as orange circles.

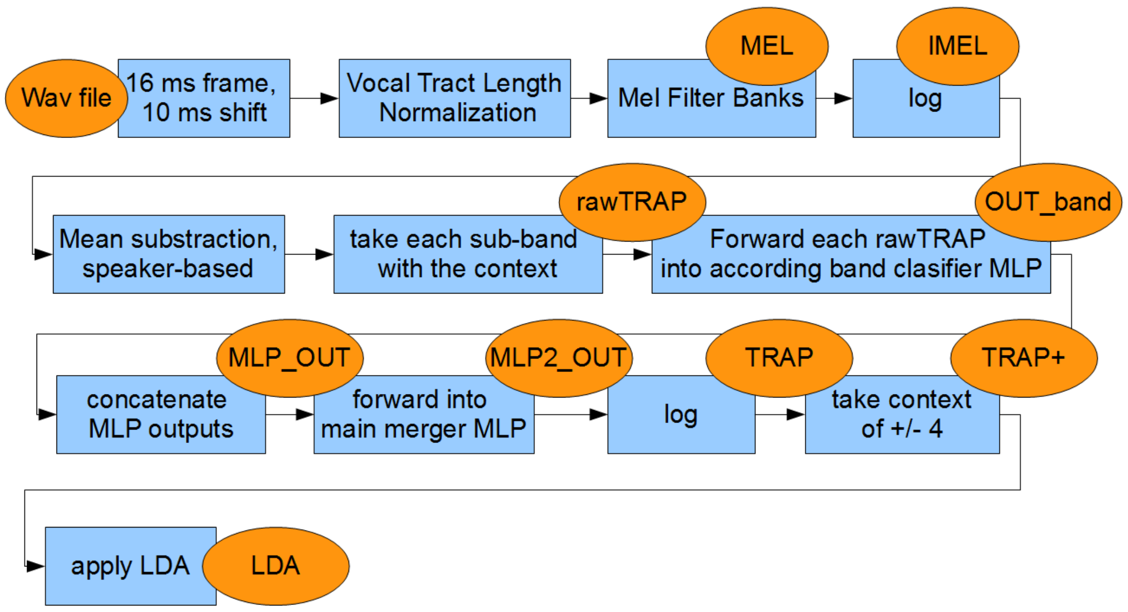
Wav file
16 ms frame, 10 ms shift
Vocal Tract Length Normalization
MEL
Mel Filter Banks
lMEL
log
Mean substraction, speaker-based
rawTRAP
take each sub-band with the context
OUT_band
Forward each rawTRAP into according band clasifier MLP
MLP_OUT
concatenate MLP outputs
MLP2_OUT
forward into main merger MLP
TRAP
log
TRAP+
take context of +/- 4
apply LDA
LDA

Figure 3.5.: Feature Extraction Steps.

The training time, convergence and recognition performance of an MLP has a direct relation with its size. That is why it is necessary to choose suitable sizes for the layers. In the experiments described in [1] the total number of weights was 2 000 000. Suggested ratio between the networks in the first and second stages was 1:9. This means, there are 200 000 weights available for all the band classifier MLPs and 1 800 000 for the main merger MLP.
All the band mergers have the same network architecture. The sizes of the input and the output layers are fixed for all the MLPs. In the BN MLPs the sizes of the bottle-neck

layers are fixed as well. Taking these constraints into consideration, the sizes of the middle layers can be computed as follows.
Basic TRAP
Given the band classifier size $51 \times N \times 50$, where N is the size of the middle layer, the following inequality can be composed. The first two summands are weights between the layers and the rest are biases for each layer and all that for each of 30 MLPs:

$$30 \cdot (51 \cdot N + N \cdot 50 + 51 + N + 50) \leq 200000$$
$$\Rightarrow N \leq 64$$
$$\Rightarrow 51 \times 64 \times 50.$$

Main merger size: $1500 \times N \times 50$

$$1500 \cdot N + N \cdot 50 + 1500 + N + 50 \leq 1800000$$
$$\Rightarrow N \leq 1159$$
$$\Rightarrow 1500 \times 1159 \times 50.$$

BN-TRAP
Normally the first and the third hidden layers have equal sizes, while the bottle-neck layer has a suitable fixed size. Given the band classifier size $51 \times N \times 20 \times N \times 145$, where N is the size of the middle layers, the following inequality can be composed:

$$30 \cdot (51 \cdot N + N \cdot 20 + 20 \cdot N + N \cdot 145 + 51 + N + 20 + N + 145) \leq 200000$$
$$\Rightarrow N \leq 27$$
$$\Rightarrow 51 \times 27 \times 20 \times 27 \times 145.$$

Main merger size: $600 \times N \times 42 \times N \times 145$

$$600 \cdot N + N \cdot 42 + 42 \cdot N + N \cdot 145 + 600 + N + 42 + N + 145 \leq 1800000$$
$$\Rightarrow N \leq 2165$$
$$\Rightarrow 600 \times 2165 \times 42 \times 2165 \times 145.$$

Note: While computing these sizes for the experiments the parameters for biases were not considered. This explains the small difference between the sizes computed above (64, 1159, 27, 2165) and those used in experiments (66, 1160, 28 and 2166). Because of high time and resource costs the networks were not retrained.

3.5. Used Tools

- The Janus recognition toolkit (JRTK) is a recognition system which is being developed by the Interactive Systems Lab at the KIT and at the CMU since 1993. General information about this toolkit is available at `http://isl.ira.uka.de/english/1406.php`. Its documentation can be found at `http://www.cs.cmu.edu/afs/cs.cmu.edu/Web/People/paisarn/papers/janus-doku.pdf`.
- The QuickNet is an artificial neural network (ANN) training and simulating tool which was developed by the speech group at the International Computer Science Institute (ICSI). It is available at `http://www1.icsi.berkeley.edu/Speech/qn.html`. This tool allows to use the training data in the pfile format and to store network weights in Matlab format files. Tools for creating, editing and reading pfiles are also included in the QuickNet package.

4. Results and Discussion

4.1. Experiments on Arabic corpus

The corpus contains 50 hours of the GALE Arabic broadcast news dataset with 9000 different speakers. The small size of this corpus allows faster training and easier debugging. It was used for the initial experiments on MLP training for TRAP-based feature extraction.

4.1.1. MLPs for Basic TRAP

The objective of this setup was to build an MLP training framework for basic TRAP feature extraction.
In this and all the following setups standard back-propagation algorithm with a "newbob" learning rate updating strategy is used for MLP training in the QuickNet tool. This learning rate strategy keeps the learning rate constant as long as the increment in cross-validation (CV) accuracy is higher than a threshold and then this rate is decreased exponentially for the subsequent epochs. The chosen relation between the CV set and training set for all the experiments is 1:10. The output neurons of all MLPs have the softmax activation function that delivers values in the interval $(0, 1)$.
The table 4.1 shows the main merger accuracy and the mean value of the band classifiers accuracies. As it was expected, the band classifier MLPs show much lower accuracy than the main merger. This can be explained by the fact that the band classifiers use only a part of input information provided by the corresponding band while the main merger uses the information from all bands. The band classifier accuracies presented in [1] are also approximately two times lower than the corresponding merger accuracies.

Bacis TRAP	band MLPs	main merger MLP
targets	40 phonemes	40 phonemes
size	$51 \times 73 \times 40$	$1200 \times 1451 \times 40$
accuracy	22%	49%

Table 4.1.: Accuracies of the AR-BasicTRAP MLPs.

4.1.2. MLPs for BN-TRAP

The objective of this setup was to build an MLP training framework for the BN-TRAP feature extraction.
Similarly to the previous setup the training accuracies are presented in the table 4.2. The band MLPs accuracies are much lower and the main merger MLP showed no convergence at all. This could be explained by the superfluous number of parameters in the MLP architecture for the given training data amount.

BN-TRAP	band MLPs	main merger MLP
targets	111 sub-phonemes	111 sub-phonemes
size	$52 \times 33 \times 20 \times 33 \times 111$	$600 \times 2264 \times 42 \times 2264 \times 111$
accuracy	11%	no convergence

Table 4.2.: Accuracies of the AR-BN-TRAP MLPs.

4.2. Experiments on German corpus

This German data base was used for the Quaero 2011 evaluation system, containing 180 hours of Quaero data with 50% of broadcast news and 50% of conversational speech. There are 6000 different speakers. The mean duration per speaker is **X** and the standard deviation is **Y**.

4.2.1. MLPs for Basic TRAP

The objective of this setup was to adapt to the German corpus the existing MLP training framework for Basic TRAP feature extraction done for the Arabic corpus.
The table 4.1 shows the mean value of the band classifiers accuracies and the main merger accuracy. While the band classifiers have the same mean accuracy compared to the AR-BasicTRAP setup, the main merger showed an improvement from 49% to 57%. This can be explained by the larger size of the used corpus.

Bacis TRAP	band MLPs	main merger MLP
targets	50 phonemes	50 phonemes
size	$51 \times 66 \times 50$	$1500 \times 1160 \times 50$
accuracy	22%	57%

Table 4.3.: Accuracies of the GER-BasicTRAP MLPs.

4.2.2. MLPs for BN-TRAP

The objective of this setup was to adapt to the German corpus the existing MLP training framework for BN-TRAP feature extraction done for the Arabic corpus.
The resulting accuracies are shown in the table 4.4. The band accuracies showed an increase compared to the AR-BN-TRAP setup. The main merger MLP could converge but only with a low accuracy. This could mean that even for the larger data amount there are still too many parameters.

BN-TRAP	band MLPs	main merger MLP
targets	145 sub-phonemes	145 sub-phonemes
size	$51 \times 28 \times 20 \times 28 \times 145$	$600 \times 2166 \times 42 \times 2166 \times 145$
accuracy	21 %	26 %

Table 4.4.: Accuracies of the GER-BN-TRAP MLPs.

4.2.3. MLPs for BN-MFCC

This is an existing setup for the single BN-MLP training based on the MFCC features. The table 4.5 shows the accuracy of this MLP. The acoustic model trained on these BN-MFCC features was used for the further comparison.

BN-MFCC	BN-MLP
targets	145 sub-phonemes
size	$450 \times 2000 \times 42 \times 145$
accuracy	62 %

Table 4.5.: Accuracies of the GER-BN-MFCC MLPs.

4.3. Comparing different AMs for the German ASR system

The objective of the following experiments was to train several acoustic models (AM) based on different input features and to compare their performance.
The existing setup for the German ASR system with the MFCC-based AM was used as the baseline. This setup was adapted to use BN-MFCC and TRAP-based features. Among the TRAP-based setups the GER-BasicTRAP setup with the best performing MLPs was chosen. Thus the MFCC-based AM, the BN-MFCC-based AM, and the basicTRAP-based AM were trained on the same data and with analogous configurations. Then the ASR system was tested with each of these three AMs.
The word error rate (WER) scores are presented in the table 4.6. The best result was shown by the baseline system with the MFCC-based AM. Close to its score was the result of the BN-MFCC. The system with the TRAP-based AM showed much higher WER compared to the others. Nevertheless the performance of the basic TRAP with 37.63 % WER is comparable to the results presented in [1] (28.6 % and 38.7 % for the RT'05 and RT'07 evaluations, respectively) while the MFCC-based systems were not considered there.
As it is stated in [20], basic TRAP features alone do not reach the performance of standard cepstral features (e.g. MFCC) nor probabilistic features from a single MLP (e.g. BN-MFCC). This could explain the obtained results for the basic TRAP setup.

Setup	WER
MFCC	26.58 %
BN-MFCC	28.06 %
Bacis TRAP	37.63 %

Table 4.6.: Word error rate (WER), German corpus, 180 hours.

4.4. Problems and Challenges

The described experiments require access to a computationally powerful cluster and a large data storage to keep large amount of temporary data for the intermediate steps of the MLP training.
Another challenge is high MLP training accuracy that is required for high recognition performance. High training accuracy can be achieved by both providing sufficient data amount divided in a good proportion into a training set and a cross-validation (CV) set and choosing a reasonable MLP architecture depending on the provided data amount.
Time required to train all the MLPs is also a challenge. The QuickNet tool supports multi-threading for single cluster node. Therefore the training time can be significantly reduced by parallelizing the MLP training between several cluster nodes.
In order to use the existing Janus scripts to train an AM on new features, several updates should be done including decoder configurations. The decoder parameters used in the basic TRAP setup and the parameters for other setups are shown in the table A.1. This table can be useful for further parameter tuning.

4.5. Future Work

According to [1], the best results are shown by the 3band-DCT-BN-TRAP setup (23.7 % and 31.7 % for the RT'05 and RT'07 evaluations, respectively). For the TRAP processing three adjacent bands are taken and their dimensionality is reduced by DCT. All the MLPs there have the BN architecture and use sub-phonemes as targets. It is therefore reasonable to extend the existing basic TRAP setup for Janus by an implementation of multi-band processing. Then with the existing framework for two stage bottle-neck MLPs training, networks should be trained on a large amount of data to reach high training accuracy. After this has been done, it will be possible to obtain the 3band-DCT-BN-TRAP features and to train an AM on them.
The other direction of research is to examine how TRAPs will perform when used for other languages. It can be tested if already trained MLP weights for one language can be used for another one.

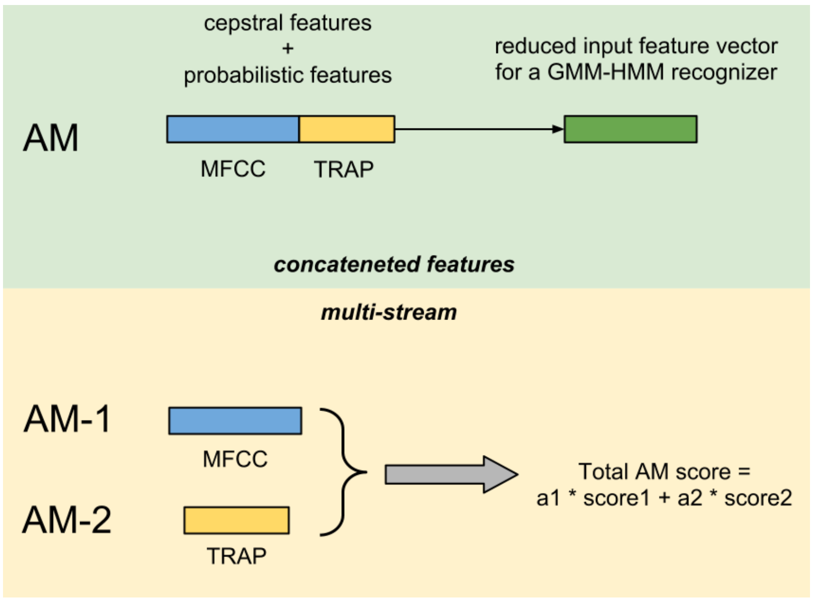

Figure 4.1.: Feature combination.

Being complementary to cepstral features, TRAP features can thereby be used as an augmentation of them [20]. In the setups for this project no feature or AM combination was used. The figure 4.1 outlines examples of possible feature combinations as suggested in [21]. One approach is to train a single AM on concatenated features with eventual dimensionality reduction. Another approach is to train two or several AMs separately and use their weighted sum in the decoding process.

A setup can also include several different systems. These systems can be combined by confusion network combination (CNC). Additionally several system outputs can be combined by recognizer output voting error reduction (ROVER). Different applications of the TRAP techniques for setups with several systems were examined in [20] obtaining up to 10 % relative WER reduction for some system configurations. Thus it is reasonable to further investigate how the TRAP technique can be applied to provide an improvement for multi-component ASR systems.

5. Conclusion

In this student project different feature extraction techniques for the acoustic modeling were investigated. In order to extend the existing in the Janus recognition toolkit mel-frequency cepstral coefficients (MFCC) features and the bottle-neck (BN) features, both the basic Temporal Pattern (TRAP) features as well as elements for other TRAP modifications were implemented.
There are still challenges which are reducing training time of multilayer perceptrons (MLPs) and boosting their recognition accuracy. It is realistic to solve this by a more advanced parallelization of the training process, by using more data and by choosing training parameters that improve MLP performance. There are already some successful applications of the TRAP technique in multicomponent systems, nevertheless further research is desirable. Both these points inspire for the further work on TRAPs technique to improve acoustic modeling for automatic speech recognition.

Appendix

A. Decoder Parameters

MFCC, BN-MFCC	Basic TRAP
base lz/lp	
lz 26	lz 38
lp 0	lp 0
sp 20	sp 40
final lz/lp	
lz2 34	lz2 62
lp2 16	lp2 -22
-	lsp 40
rescoring matrix	
llz 24 26 28 30 32 34 36 38 40 42	llz 54 58 62 66 70 74 78 82 86 90
llp 8 10 12 14 16 18 20 22 24	llp -22 -20 -18 -16 -14 -12 -10 -8 -6
decoder settings	
mb 1.3	mb 1.9
transN 40	transN 30
morphN 10	morphN 9

Table A.1.: Decoder parameters for MFCC, BN-MFCC and basic TRAP.

Bibliography

[1] F. Grezl and M. Karafiat, "Integrating recent mlp feature extraction techniques into trap architecture," *Proc. Interspeech'11*, 2011.

[2] M. Sundermeyer, M. Nussbaum-Thom, S. Wiesler, C. Plahl, A. E.-D. Mousa, S. Hahn, D. Nolden, R. Schluter, and H. Ney, "The rwth 2010 quaero asr evaluation system for english, french and german," *Proc. IEEE Int. Conf. Acoustics, Speech and Signal Processing ICASSP'11*, 2011.

[3] S. Stücker, K. Kilgour, and J. Niehues, "Quaero speech-to-text and text translation evaluation systems," *Proc. HLRS. Springer, 2010, pp. 529–542.*, 2011.

[4] C. Fügen, A. Waibel, and M. Kolss, "Simultaneous translation of lectures and speeches," *Machine Translation, vol. 21, no. 4, MTSN 2008, Springer*, 2008.

[5] S. Stücker, "Basics of automatic speech recognition." Lectures in winter semester 2009/2010.

[6] H. Hermansky, D. P. W. Ellis, and S. Sharma, "Tandem connectionist feature extraction for conventional hmm systems," *Proc. IEEE Int. Conf. Acoustics, Speech, and Signal Processing ICASSP '00*, 2000.

[7] H. Hermansky, "Trap-tandem: data-driven extraction of temporal features from speech," *Proc. IEEE Workshop Automatic Speech Recognition and Understanding ASRU '03*, 2003.

[8] A. Faria, "An investigation of tandem mlp features for asr," *ICSI Technical Report TR-07-003*, 2007.

[9] H. Hermansky and S. Sharma, "Traps - classifiers of temporal patterns," *Proc. ICSLP'98*, 1998.

[10] P. Fousek, *Extraction of features for automatic recognition of speech based on spectral dynamics*. PhD thesis, Czech Technical University in Prague, Faculty of Electrical Engineering, 2007.

[11] F. Grezl, *TRAP-based probabilistic features for automatic speech recognition*. PhD thesis, Brno University of Technology, Faculty of Information Technology, 2007.

[12] F. Grezl and J. Cernocky, "Trap-based techniques for recognition of noisy speech," *Proc. 10th International Conference on Text Speech and Dialogue TSD'07.*

[13] F. Grezl and P. Fousek, "Optimizing bottle-neck features for lvcsr," *Proc. IEEE Int. Conf. Acoustics, Speech and Signal Processing ICASSP'08*, 2008.

[14] F. Grezl, M. Karafiat, S. Kontar, and J. Cernocky, "Probabilistic and bottle-neck features for lvcsr of meetings," *Proc. IEEE Int. Conf. Acoustics, Speech and Signal Processing ICASSP'07*, 2007.

[15] C. Plahl, R. Schlüter, and H. Ney, “Hierarchical bottle-neck features for lvcsr,” *Proc. Interspeech’10*, 2010.

[16] P. Schwarz, P. Matejka, and J. Cernosky, “Towards lower error rates in phoneme recognition,” *Proc. 7th International Conference on Text Speech and Dialogue TSD’04*, 2004.

[17] U. Nallasamy, M. Garbus, F. Metze, Q. Jin, T. Schaaf, and T. Schultz, “Analysis of dialectal influence in pan-arabic asr,” *Proc. Interspeech’11*, 2011.

[18] K. Kilgour, C. Saam, C. Mohr, S. S., and A. Waibel, “The 2011 kit quaero speech-to-text system for spanish,” *Proc. International Workshop on Speech Translation (IWSLT 2011)*, 2011.

[19] J. Cernocky, “Traps in all senses,” tech. rep., Oregon Graduate Institute, School of Science and Engineering, 2001.

[20] Q. Zhu, A. Stolcke, B. Y. Chen, and N. Morgan, “Using mlp features in sri’s conversational speech recognition system,” *Proc. Interspeech’05*, 2005.

[21] F. Metze, R. Hsiao, Q. Jin, U. Nallasamy, and T. Schultz, “The 2010 cmu gale speech-to-text system,” *Proc. Interspeech’10*, 2010.