DEGREE FINAL PROJECT

Automatic Speech Recognition with Kaldi toolkit

Study branch: Degree in Science and Telecommunication Technologies Engineering

Author: Víctor Rosillo Gil

Supervisor: Bartosz Ziólko

Supervisor: José Adrián Rodríguez Fonollosa

Automatic Speech Recognition with Kaldi toolkit.

Acknowledgments

I thank to my tutors Bartosz Ziólko and José Adrián Rodríguez Fonollosa, for having helped me in all doubts I could have, and for all their support.

I also express my thanks to all Kaldi community for their disinterested assistance.

Finally I thank to my family for their continuous support not only during this stage, also during all my degree.

Abstract

The topic of this thesis is to built an accurate automatic speech recognition system to be able to recognize speech using Kaldi, an open-source toolkit for speech recognition written in C++ and with free data. First of all, the main process of automatic speech recognition is explained in details on first steps. Secondly, different approaches of training and adaptation techniques are studied in order to improve the recognition accuracy. Furthermore, as data size is a very important point in order to achieve enough recognition accuracy, the role of it, is also studied on this thesis.

Keywords: Automatic Speech Recognition(ASR), speaker adaptation, discriminative training, Kaldi, voice recognition, finite-state transducers.

Index

1 Introduction

The Automatic Speech Recognition (ASR) is a discipline of the artificial intelligence that has as main goal allow the oral communication between humans and computers, i.e. basically consist on convert the human speech into a text automatically. The main problem of the ASR is the complexity of the human language. Humans use more than their ears when they listening, they use the knowledge they have about the speaker and his environment. In ASR we only have the speech signal. But, as we will see in the following steps, there are different approaches to try to get good performance of the speech.

Both acoustic modeling and language modeling are important parts of modern statistically-based speech recognition algorithms. Modern general-purpose speech recognition systems are based on Hidden Markov Models (HMM). These are statistical models that output a sequence of symbols or quantities.

In our ASR system,we use Kaldi, a toolkit for speech recognition written in C++ and licensed under the Apache License v2.0.

1.1 Project overview and goals

The purpose of this project is learn about both ASR and Kaldi toolkit, to be able to build an Automatic Speech Recognition system.

We will study first all the process related to recognize speech automatically to understand how it works, and therefore be able to built a basic system.

In addition, we pretend to retrain and adapt the original ASR system to improve the recognition accuracy. As available data it will be limited by our resources, after evaluate different approaches, we would like to find a commitment between the quality and the data used to built a system easy to handle.

Furthermore, we pretend to create a Graphical User Interface to test our system.

1.2 Project background

The project is carried out at the electronic department of AGH University of Science and Technology.

This project is independent of any department or company research and starts with VoxForge's recipe as a base of our system.

The main project initial ideas are provided by the supervisor Bartosz Ziólko. Although the development of it is carried out by both the supervisor and I together.

1.3 Project outline

In chapter 2 we introduce the main theory about Automatic Speech Recognition and the Kaldi toolkit. At the beginning of chapter 3 we describe briefly discriminant training. After that we introduce the baseline of our training experiment with the different techniques to use. Finally, we described the acoustic models trained and the results that their present. In chapter 4, we introduce first acoustic model adaptation. Next, we present the experiments done with the different approaches of the adaption. Chapter 5, describes the graphic user interface built to integrate the ASR systems. Finally, chapter 6 summarizes the thesis.

2 Automatic Speech Recognition background

2.1 Automatic Speech Recognition

The statistical approach to automatic speech recognition aims at modeling the stochastic relation between a speech signal and the spoken word sequence with the objective of minimizing the expected error rate of a classifier. The statistical paradigm is governed by Bayes' decision rule: given a sequence of acoustic observations $x_1^T = x_1,\ldots,x_T$ as the constituent features of a spoken utterance, Bayes' decision rule decided for that word sequence $w_1^N = w_1,\ldots,w_N$ which maximizes the class posterior probability $p(w_1^N | x_1^T)$:

$$[w_1^N]opt = \underset{w_1^N}{argmax}\, P(w_1^N | x_1^T) \qquad (2.1)$$

Provided that the true probability distribution is used, Bayes' decision rule is optimal among all decision rules, that is, on average it guarantees the lowest possible classification error rate. However, for most pattern recognition tasks the true probability distribution is usually not known but has to be replaced with an appropriate model distribution. In automatic speech recognition, the generative model, which decomposes the class posterior probability into a product of two independent stochastic knowledge sources, became widely accepted:

$$P(w_1^N | x_1^T) = \frac{P(x_1^T) \cdot (P(x_1^T | w_1^N))}{P(x_1^T)} \qquad (2.2)$$

The denominator $P(x_1^T)$ in Eq. 2.2 is assumed to be independent of the word sequence w_1^N and hence, the decision rule is equivalent to:

$$P(w_1^N | x_1^T) = P(x_1^T) \cdot (P(x_1^T | w_1^N)) \qquad (2.3)$$

The word sequence $[w_1^N]_{opt}$ which maximizes the posterior probability is determined by searching for that word sequence which maximizes the product of the following two stochastic knowledge sources:

- The acoustic model $P(x_1^T | w_1^N)$ which captures the probability of observing a sequence of acoustic observations x_1^T given a word sequence w_1^N.

- The language model $P(w_1^N)$ which provides a prior probability for the word sequence w_1^N.

A statistical speech recognizer evaluates and combines both models through generating and scoring

a large number of alternative word sequences (so-called hypotheses) during a complex search process. Figure 2.1 illustrates the basic architecture of a statistical automatic speech recognition system [1].

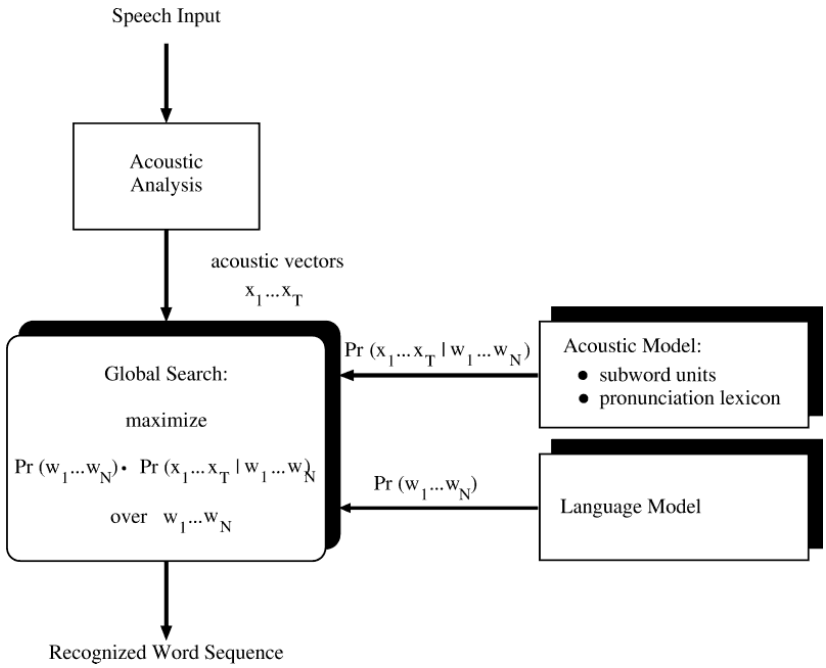

Figure 2.1: Basic architecture of a statistical automatic speech recognition system.

2.1.1 Signal analysis

The first step in any automatic speech recognition system is to extract features, i.e identify the components of the audio signal that are good for identifying the linguistic content and discarting all the other stuff which carries information. No two utterances of the same word or sentence are likely to give rise to the same digital signal. In other words, the aim of signal analysis is to derive a feature vector such that the vectors for the same phoneme are as close to each other as possible, while the vectors for different phonemes are maximally different to each other.

The main factors which could cause two random speech samples to differ from one another are:

- Phonetic identity: Differences among speakers pronunciation depends on gender, dialect, voice, etc.
- Microphone: And other properties of the transmission channel.
- Environment: Background noise, room acoustics, etc.

Common signal processing techniques used in automatic speech recognition are based on Mel Frequency Cepstral Coefficients (MFCC) and Perceptual Linear Prediction (PLP) [2]. In this project, we are going to make use of MFCC.

The main point to understand it is that the sounds generated by a human are filtered by the shape of the vocal tract. This shape determines what sound comes out and manifest itself in the envelope of the short time power spectrum. If we can determine the shape accurately, this should give us an accurate representation of the phoneme being produced. The job of the MFCC or PLP is to accurately represent this envelope.

Both MFCC and PLP transformations are applied on a sampled and quantized audio signal[1]. The overall MFCC computation that Kaldi follows is [3]:

- Work out the number of frames in the file (typically 25ms frames shifted by 10 ms each time).
- For each frame:
  1) Extract the data, do optional dithering, pre-emphasis ad dc offset removal, and multiply it by a windowing function.

  2) Work out the energy at this point (if using log-energy not CO).

  3) Do Fast Fourier Transform (FFT) and compute the power spectrum.

  4) Compute the energy in each mel bin.

  5) Compute the log of the energies and take the cosine transform, keeping as many coefficients as specified (e.g. 13).

  6) Optionally do cepstral liftering; this is just a scaling of the coefficients, which ensures they have a reasonable range.

Feature extraction is an essential first step in speech recognition applications. In addition to static features extracted from each frame of speech data, it is beneficial to use some transformations to improve the recognition.

Transforms, projections and other feature operations that are typically not speaker specific include:

- Frame splicing and Delta feature computation[4].

- Linear Discriminant Analysis (LDA) transform[5].

- Heteroscedastic Linear Discriminant Analysis (HLDA).

- Maximum Likelihood Linear Transform (MLLT) estimation[6].

[1] In our experiments we use 16KHz sampling frequency and 16 bit samples.

On the first step of the part 3, we will use and compare both Delta feature computation and LDA+MLLT.

Delta feature computation

MFCC feature only takes account of the relationship in phonetic frames without considering the relationship between them. Phonetic signals are essentially continuous, so the acquisition of the dynamic changing feature between phonetic frames will improve the performance of recognition.

Therefore, Delta feature is the Fourier Transform of the time order of the phonetic frames order. For instance: If we have 13 MFCC coefficients, with the Δ+ΔΔ transformation we also get 13+13 delta coefficients, which would combine to give a feature vector of length 39 (13+13+13). Then, the original vector is reduced to vector of 39 MFCC Δ+ΔΔ acoustic features.

LDA+MLLT

LDA: Is a linear transform that reduce dimensionality of our input features. The idea of LDA is to find a linear transformation of feature vectors from an n-dimensional space to vectors in an m-dimensional space (m<n) such that the class separability is maximum.

MLLT: Estimates the parameters of a linear transform in order to maximize the likelihood of the training data given a diagonal-covariance Gaussian mixture models; the transformed features are better represented by the model than the original features.

2.1.2 Acoustic Model

The acoustic model $P(x_1^T | w_1^N)$ provides a stochastic description for the realization of a sequence of acoustic observation vectors x_1^T given a word sequence w_1^N. Due to data sparsity, the model for individual words as well as the model for entire sentences is obtained by concatenating the acoustic models of basic sub-word units according to a pronunciation lexicon. Sub-word units smaller than words enable a speech recognizer to allow for recognizing words that do not occur in the training data. Thus, the recognition system can ensure that enough instances of each sub-word unit have been observed in training to allow for a reliable estimation of the underlying model parameters.

The type of sub-word units employed in a speech recognizer depends on the amount of available training data and the desired model complexity: while recognition systems designed for small vocabulary sizes (<100 words) typically apply whole word models, systems developed for the recognition of large vocabularies (> 5000 words) often employ smaller sub-word units which may be composed of syllables, phonemes, or phonemes in context. Context-dependent phonemes are also referred to as n-phones. Commonly used sub-word units employed in large vocabulary speech recognition systems are n-phones in the context of one or two adjacent phonemes, so-called triphones or quinphones. Context-dependent phoneme models allow for capturing the varying articulation that a phoneme is subject to when it is realized in different surrounding phonetic contexts (co-articulation)[1].

Typically, the constituent phones for various acoustic realizations of the same word are produced with different duration and varying spectral configuration, even if the utterances are produced by the same speaker. Each phone will therefore aggregate an a-priori unknown number of acoustic observations. The temporal distortion of different pronunciations as well as the spectral variation in the acoustic signal can be described via a Hidden Markov Model (HMM). A HMM is a stochastic finite state automaton that models the variation in the acoustic signal via a two-stage stochastic process. The automaton is defined through a set of states with transitions connecting the states. The probability $P(x_1^T | w_1^N)$ is extended by an unobservable (hidden) variables representing the states:

$$P(w_1^N|x_1^T)=\sum_{s_1^T} P(x_1^T, s_1^T|w_1^N) \quad (2.4)$$

2.1.3 Language Model

The language model $P(w_1^N)$provides a prior probability for the word sequence $w_1^N = w1,\ldots, w_N$. Thus, it inherently aims at capturing the syntax, semantics, and pragmatics of a language. Since language models are independent of acoustic observations, their parameters can be estimated from large text collections as, for instance, newspapers, journal articles, or web content. Due to a theoretically infinite number of possible word sequences, language models require suitable model assumptions to make the estimation problem practicable. For large vocabulary speech recognition, m-gram language models have become widely accepted. An m-gram language model is based on the assumption that a sequence of words follows an (m-1)-th order Markov process, that is, the probability of a word w n is supposed to depend only on its m-1 predecessor words[1]:

$$P(w_1^N)=\prod_{n=1}^{N} P(w_n|w_1^{n-1}) \quad (2.5)$$

$$P(w_1^N)\,model\,assumption=\prod_{n=1}^{N} P(w_n|w_{n-m-1}^{n-1}) \quad (2.6)$$

2.1.4 Global search

Given a sequence of acoustic observations x_1^T, the objective of the global search is to find that word sequence which maximizes the a-posteriori probability:

$$[w_1^N]opt=\underset{w_1^N}{argmax}\, P(w_1^N|x_1^T)=\underset{w_1^N}{argmax}\,(P(w_1^N)P(x_1^T|w_1^N)) \quad (2.7)$$

In principle, the decoder has to align the sequence of acoustic observations x_1^T with all possible state sequences s_1^T that are consistent with a word sequence w_1^N. Using m-gram language models

and an acoustic model based on HMMs, due to a complex optimization process which can be reduced by approximating the sum over all paths with the Viterbi algorithm[7]

2.1.5 Evaluation

Exist different methods to evaluate the quality of an ASR system. Word Error Rate (WER) is a common metric of the performance of a speech recognition.

The main difficulty of measuring performance lies in the fact that the recognized word sequence can have a different length from the reference word sequence. The WER is derived from the Levenshtein distance, but working at the word level. This problem is solved by first aligning the recognized word sequence with the reference word sequence using dynamic string alignment.

$$WER = \frac{100 * (S + I + D)}{N} \quad (2.8)$$

Where:

-N: Is the number of words in the reference
-S: Is the number of substitutions
-I: Is the number of insertions
-D: Is the number of deletions.

A basic alignment example:

Ref: portable	***	phone	upstairs	last	night	so	***
Hyp: preferable	form	of	stores	next	light	so	far
Eval: S	I	S	S	S	S		I

$$WER = \frac{100 * (5 + 2 + 0)}{6} \quad (2.9)$$

2.2 Kaldi

Kaldi is an open-source toolkit for speech recognition written in C++ and licensed under the Apache License v2.0. The goal of Kaldi is to have modern and flexible code that is easy to understand, modify and extend [8].

Exist severals potential choices of open-source toolkit for building a recognition system. Kaldi

specific requirements are: a finite-state transducer (FST) based framework, extensive linear algebra support, and non-restrictive license led to the development of Kaldi. Important features of Kaldi include:

- Integration with Finite State Transducers.
- Extensive linear algebra support.
- Extensible design.
- Open license.
- Complete recipes.
- Thorough testing

In figure 2.1 we give a schematic overview of the Kaldi toolkit. The toolkit depends on two external libraries that are also freely available: one is OpenFst [9] for the finite-state framework and the other is numerical algebra libraries.

Access to the library functionalities is provided through command-line tools written in C++, which are then called from a scripting language for building and running a speech recognizer. Each tool has very specific functionality with a small set of command line arguments: for example, there are separate executables for accumulating statistics, summing accumulators, and updating a GMM-based acoustic model using maximum likelihood estimation.

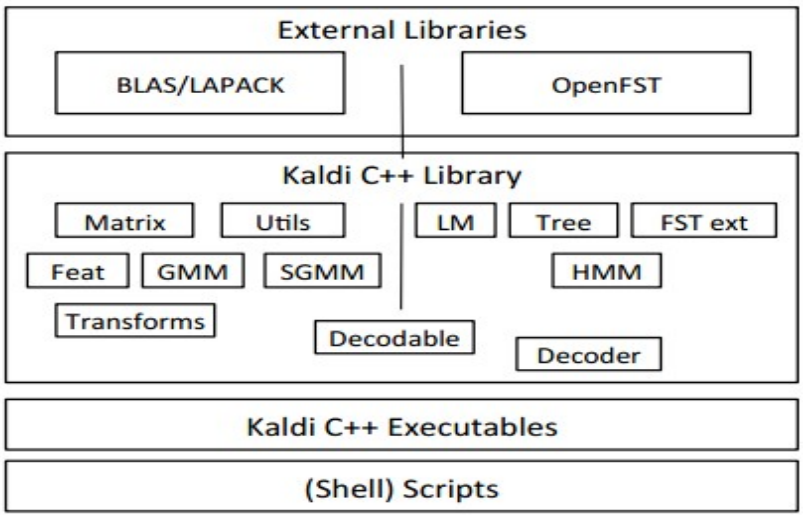

Figure 2.2: A simplified view of the different components of Kaldi.

Kaldi feature extraction: His feature extraction and waveform-reading code aims to create standard MFCC and PLP features.

Kaldi acoustic modeling: Support conventional models (i.e. diagonal Gaussian Mixture Models (GMMs)) and Subspace Gaussian Mixture Models (SGMMs), but also extensible to new kinds of model.

Kaldi phonetic decision trees: His goal is built the phonetic decision tree code were to make it efficient for arbitrary context sizes. The conventional approach is, in each HMM-state of each mono-phone, to have a decision tree that asks questions.

Kaldi language modeling: Kaldi uses an FST-based framework.

Kaldi decoding graphs: All the training and decoding algorithms use WFTs.

Kaldi decoders: It has several decoders, from simple to highly optimized. By decoder we mean a C++ class that implements the core decoding algorithm.

2.2.1 Finite-State Transducers

Much of current large-vocabulary speech recognition is based on models such as HMMs, lexicons, or n-gram statistical language models that can be represented by weighted finite-state transducers.

A FST is a finite automaton[2] whose state transitions are labeled with both input and output symbols. Therefore, a path through the transducer encodes a mapping from an input symbol sequence, or string, to an output string. A weighted transducer puts weights on transitions in addition to the input and output symbols. Weighted transducers are thus a natural choice to represent the probabilistic finite-state models prevalent in speech processing[10].

The examples of figure 2.3 is a representation of weighted FST. In figure 2.3.a, the legal word strings are specified by the words along each complete path, and their probabilities by the product of the corresponding transition probabilities. Figure 2.3.b represents a toy pronunciation lexicon as a mapping from phone strings to words in the lexicon.

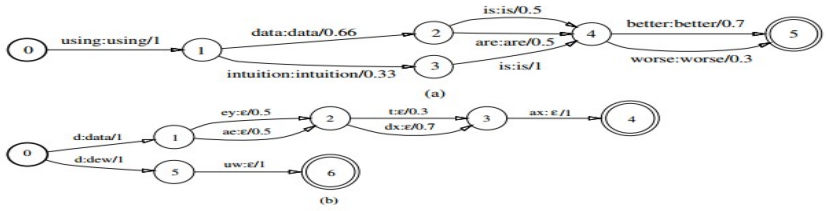

Figure 2.3: Weighted finite-state transducer

[1] In our experiments we use 16KHz sampling frequency and 16 bit samples.

The general approach that Kaldi use to decode graph construction is described briefly next. The overall picture for decoding-graph creation is that we are constructing the graph HCLG = H o C o L o H. Here

- G is an acceptor (i.e its input and output symbols are the same) that encodes the grammar or language model.
- L is the lexicon; its output symbols are words and its input symbols are phones.
- C represents the context-dependency: Its output symbols are phone and its input symbols represent context-dependent phone.
- H contains the HMM definitions; its output symbols represent context-dependent phone and its input symbols are transitions-ids, which encode the pdf-id and other information.

This is the standard recipe. However, there are a lot of details to be filled in.

3 Acoustic Model training

Estimation of HMM parameters is commonly performed according to the Maximum Likelihood Estimation(MLE) criterion, which maximizes the probability of the training samples with regard to the model. This is done by applying the Expectation-Maximization (EM) algorithm, which relies on maximizing the log-likelihood from incomplete data, by iteratively maximizing the expectation of log-likelihood from complete data[11].

The MLE criterion can be approximated by maximizing the probability of the best HMM state sequence for each training sample, given the model, which is known as Viterbi training.

This is the procedure that we will follow on the first steps of our training experiments. Then, we will try to improve the accuracy of the acoustic model training with different approaches as it is explained in section 2.1 or in 3.1

3.1 Discriminative training

As we explain in the above section, model parameters in HMM-based speech recognition systems are normally estimated using MLE. But other approach it could be carried out based on other optimization criteria. In contrast to Maximum

Likelihood, discriminative training also takes the competing classes into account to optimize the parameters. This should due as to an improvement on terms on recognition accuracy. We are going to evaluate three of the most typical discriminative training methods:

MMI

The Maximum Mutual Information (MMI) goal is the maximize the mutual information between data and their corresponding labels/symbols[12].

bMMI

Boosted MMI is a modified form of the MMI objective function. The modification consists of boosting the likelihoods of path in the denominator lattice that have a higher phone error relative to the correct transcript[13].

MPE

Basically, Minimum Phone Error, try to carried out a minimization and estimation of the training set errors.[12,14]

3.2 Acoustic data

All data used in our training experiments comes from VoxForge project[16]. It was setup to collect transcribed speech for use with Free and Open Source Speech Recognition Engines. They make available all submitted audio files that VoxForge users record under the GPL license. Therefore, the data available are recorder in different environments and under different conditions. It could be possible to download speech data of different languages. In our experiments, we will train and test our acoustic models just with English data (American, British, Australian and Zealand)[16].

In the following table 3.1 the dataset used for our experiment is described.

English dataset	**#Speakers**	**#sentences**	**Audio[sec] per sentence**
Train	358	15264	5
Test	20	399	5

Table 3.1: The data used to train and test the acoustic models consist of 358 and 20 speakers respectively. The number of sentences of each speaker differs.

3.3 Experiment

The main goal in our experiment consist on test different approaches of training with different amount of data to be able to decide which model could be better in terms of the quality. The quality will be measured by WER on the acoustic models trained by different methods.

As our data are limited, we do not fix any minimum threshold of accuracy. Instead of that, we are going to compare the results with a mono-phone acoustic model to see how each technique improve the performance of our model.

The recordings and their transcriptions from training dataset are used for acoustic modeling. The estimate dataset AMs are evaluated on the test set.

Baseline system

As a flat start, we trained a mono-phone system (mono) using the MFCC's and Δ+ΔΔ features as we described in section 2.1.1. Then, we must align the feature vectors to HMM states using utterances' transcriptions (Before any retraining, we must do forced-align) . Finally, we retrain the triphone AM (tri1a)

We are going to use different subsets of data to adjust the data necessary to train the model, since is a waste of time use all of it(Typically, mono-phone acoustic models does not need so much data to train their parameters). The amount of different data used to train the mono-phone and tri-phone models are described on the table 3.2

Dataset	**# sentences**
Train_500	500
Train_1000	1000
Train_2000	2000
Train_4000	4000
Train_8000	8000
Train_12000	12000
Train_15264	15264

Table 3.2: Different subsets of data based on the number of sentences used to train the mono-phone and tri-phone model.

3.4 Evaluation

3.4.1 Conditions of evaluation

The speech samples were recorded in different environment, sampled at 16 Khz. In each experiments, each speech signal was parameterized using 13 MFCC. The analysis windows size was 25ms with 10 ms overlap as we described in section 2.1. We use a bi-gram language model which is estimated from the training data transcription. As the test dataset is different than train dataset, may be appear unknown words, so called Out of Vocabulary Word.

The decoding of the test utterances is performed always with the same parameters, so that different Ams can be compared. Specifically the parameters set are list next:

- gmm-latgen-faster
- max-active=7000
- beam=13.0
- lattice-beam=6.0
- acoustic scale= 0.083333
- model size= #num-leaves #tot-gauss = 2000 11000

As we explain on the beginning of section 3.3, we are interested on the WER improvement in comparison with a basic mono-phone model.

3.4.2 Experiments evaluation

Mono-phone and tri-phone acoustic models

Firs of all, we are going to analyze which is the amount of data needed to train an mono-phone system (mono) and tri-phone system (tri1a). In principle as larger is the amount of data used to train acoustic models, better results on terms of recognition quality is achieve. Figure 3.1 and 3.2 shows the performance of mono-phone and tri-phone models depending on the number of utterances used to train the model respectively.

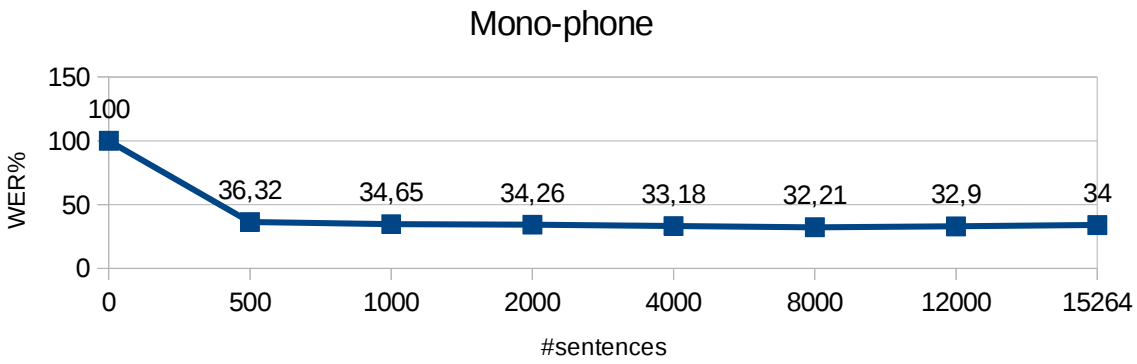

Figure 3.1: The figure displays the WER% depending the portion of the data size in terms of number of sentences on the train step.

Context-dependent tri-phones can be made by simply cloning mono-phones and then re-estimating using trip-phones transcriptions. To do that, we use the mono-phone model trained by 1000 sentences.

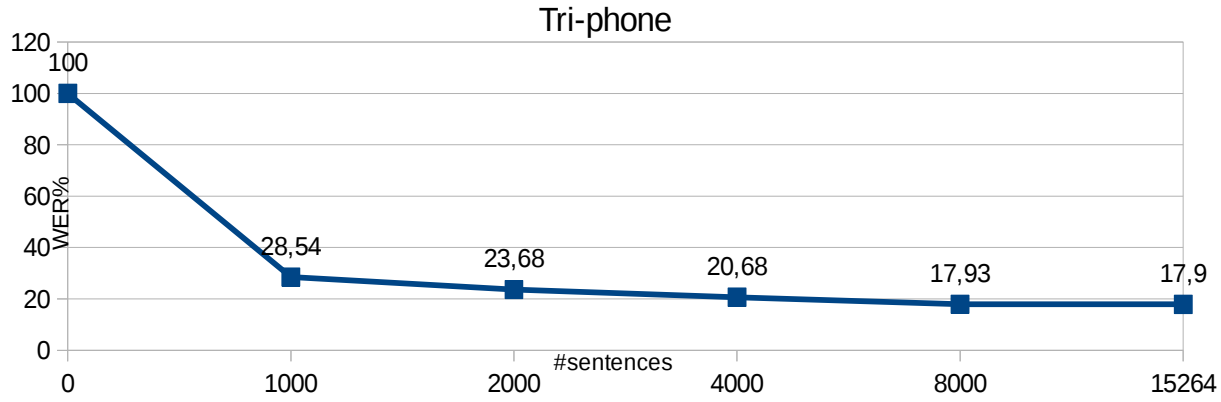

Figure 3.2: The figure displays the WER% depending the portion of the data size in terms of number of sentences on the train step.

As we can see on figure 3.1, the amount of data needed to train the mono-phone model do not increase significantly the WER since 1000 sentences are used. Because of that, it is waste of time use all data available to train the model. However, as the amount of training data is larger, best performance is achieve on trip-phone acoustic model. Next, all experiments are will be done with all available data, because as larger is the training data better results will be achieve.

Δ+ΔΔ vs LDA+MLLT

We are going to re-train the tri1a model with two different transformation described in section 2.1.1. Here, we will use all amount of data available to train the models as we conclude on section 3.4. Moreover, we are going to evaluate the performance of the model depending on the model size.

Δ+ΔΔ triples the number of 13 MFCC features by computing the first and second derivates from MFCC coefficients during 35 iterations. Therefore, 39 is the number of MFCC features per frame used to represent the shape of the speech to determines what sound comes out.

LDA+MLLT is set with typical configuration: 13-dimension input, -3..3 splicing and 40-dimensional output, so the number of features is (13*7+1) *40 [The +1 is for the bias term, which substracts the mean, IIRC]

Table 3.3 shows the WER obtained with Δ+ΔΔ transformation and table 3.4 LDA+MLLT using different model size:

#leaves-#total gauss	#1000-#9000	#1000-#11000	#1000-#13000	#1500-#9000	#1500-#11000	#1500–13000	#2000-#9000	#2000-#11000	#2000-#13000
WER%	17.91	17.79	16,67	18.16	17.85	16.72	17.77	16.78	16.77

Table 3.3: WER obtained when we decode with Δ+ΔΔ using different model sizes.

#leaves-#total gauss	#1000-#9000	#1000-#11000	#1000-#13000	#1500-#9000	#1500-#11000	#1500–13000	#2000-#9000	#2000-#11000	#2000-#13000
WER%	16.80	17.11	15.59	17.11	16.04	15.93	17.14	16.18	15.79

Table 3.4: WER obtained when we decode with LDA+MLLT using different model sizes.

We can observe that increasing the model size in average lead to better performance. Although each parameter alone improve the recognition, the most decisive parameter is the total number of gaussians.

Moreover, we can appreciate that LDA+MLLT upgrade almost 1% the WER than Δ+ΔΔ transformation.

Discriminative training

Finally, the last step, consist on evaluate MMI, bMMI and MPE as table 3.5 presents. We take as previous training stage LDA+MLLT transform with 2000-3000 as number of leaves and number of total gaussians respectively.

Model	**WER%**
tri2b_mmi	13.83
tri2b_bmmi	13.83
tri2b_mpe	13.83

Table 3.5: On this table WER of Maximum Mutual Information, boosted MMI and Minimum Phone Error is represented respectively.

We can observe that although three different approaches of discriminative training achieve better recognition than last stage, we can not distinguish between each kind of it. Probably, one of the reasons it could be that we need more training data to estimate correctly the parameters. We just achieve less than 2% of improvement with discriminative training. During the simulations when the amount of training data were almost the half data, the improvements achieved were only 0,1%.

3.4.3 Results

In this section we are going to show the result of different acoustic training methods presented on the previous sections.

Model/method	**WER%**
Mono (Δ+ΔΔ)	34.65
Tri1a (Δ+ΔΔ)	17.9
Tri2a (Δ+ΔΔ)	16.77
Tri2b (LDA+MLLT)	15.79
tri2b_mmi (LDA+MLLT+MMI)	13.83
tri2b_bmmi (LDA+MLLT+bMMI)	13.83
tri2b_mpe (LDA+MLLT+MPE)	13.83

Table 3.5: The table shows the WER of different acoustic models evaluated in two different ways: % and (substitutions+deletions+inserted)/total words in the reference transcription.

As it could be seen on table 3.5 context-dependent phoneme models (tri-phone), contrary to mono-phone models, allow for capturing the varying articulation that a phoneme is subject to when it is realized in different surrounding phonetic contexts. Therefore the WER improve significantly.

Moreover, use different linear non-dependent transforms lead to a considerably reduction of word error rate. Although it can be seen, LDA+MLLT works better than Δ+ΔΔ.

Finally, make use of an additional step of discriminant training lead also a better performance, thanks to it takes account the competing classes to optimize the main parameters.

4 Acoustic Model adaptation

In statistical speech recognition, there are usually mismatches between the conditions under which the model was trained and those of the input. Mismatches may occur because of differences between speakers, environmental noise, and differences in channels. They should be compensated in order to obtain sufficient recognition performance. Acoustic model adaptation is the process of modifying the parameters of the acoustic model used for speech recognition to fit the actual acoustic characteristics by using a few utterances from the target user[16,17,18]

In this thesis, we want to test different approaches of speaker adaptation.

4.1 Speaker adaptation

As we discussed on the previous point, adaptation it could be beneficial for our system depending on our requirements.

Speaker independent (SI) system is desirable in many applications where speaker-specific data not exist. Otherwise, if dependent data are available, the system could be trained with the specific speakers to obtain better performance. Speaker dependent systems can result in word error rate 2-3 times lower than SI systems (given the same amount of training data). But, the problem on the SD systems is that for large-vocabulary continuous speech recognition, a lot of amount data is needed to reliable estimate system parameters. Also, if a different speaker try to use the system, he will obtain very bad results[17,18].

Because of that, we would like to train the model as SI system, and then adapt the

model to a specific speakers. With the speaker adaptive (SA) system we could achieve:

- Error rates similar to SD systems.
- Building on a SI systems.
- Requiring only a small fraction of the speaker-specific training data used by an SD system.

Supervised and unsupervised adaptations

In supervised adaptation, a transcription exist for each utterance. In unsupervised adaptation, it does not.

The users in supervised adaptation should follow some steps to get the transcription of some utterances. Depending of the adaptation technique used, the amount of data required can vary. But, as the transcription is known, the HMM may be constructed.

Unsupervised adaptation is usually needed for such short-period applications since users should not have to spend time registering their voices. The problem here is in the case of the recognition accuracies of the speaker independent are not high enough the estimation could be reported big problems, because signals generated by mis-recognitions may significantly degrade adaptation performance. Although the accuracy of the speaker should not be a problem, when the speakers are non-native is it.

Therefore, we decided that to adapt our acoustic model we will use the supervised adaptation. Thus, we could control the accuracy of the adapt data, and in case it would be necessary we must record again the new data.

Batch and on-line adaptation

Batch: All adaptation data is presented to the system in a block before the final system is estimated.

On-line adaptation is used in applications where the speakers often change and change points are not given beforehand.

As batch adaptation performs better than on-line adaptation and our AM doesn't consist on a dialogue system where the speakers change very often, we will make use in our experiments of batch adaptation.

Types of speaker adaptations

Nowadays, exist several approaches to try to adapt an acoustic model. It is possible get it re-training the system, apply some transformation or combine both techniques together. We are going to present the most typical techniques used in speaker adaptation. Later we will evaluate it, and discuss how they work

depending on the data used.

MAP

The Maximum a Posteriori estimate the HMMs model parameters providing a natural way of incorporating prior information of the model in the model training process[19].

fMLLR

A set of transformation matrices for the HMM Gaussian parameters are estimated which maximize the likelihood of adaptation data. The set of transformations is relatively small compared to the total number of Gaussians in the system and so a number of Gaussians share the same transformation matrices. This means that the transformation parameters can be robustly estimated from only a limited amount of data, which allows all the Gaussians in the HMM set to be updated. For a small amount of data only a single global transformation is used. The transformation estimates the mean and variance parameters in two separates stages[20] .

fMLLR also known as Constrained MLLR is just a simplifying implementation and improving runtime performance of basic MLLR.

SAT(+fmllr)

Speaker adaptive training tries to separate speaker induced variations from phonetic ones. SAT adds speaker dependent transforms for each speaker in the training set[17,21].

4.2 Acoustic data

The acoustic data used to make different speaker adaptations are from VoxForge too. Obviously, the speakers are independent from the train dataset. Table 4.1 described the data used.

Dataset	**# speakers**	**Approx. audio[sec] per utterance**	**# utterances per speaker**
Adaptation data_1	1	5	63
Test data_1	1	5	25

Table 4.1: Number of speakers, duration of each utterance, number of utterances/sentences per speaker.

4.3 Experiment

The main goal of our experiment consist on adapt our acoustic model by different approaches to estimate which one is valid or not and to decide the best in terms of both recognition accuracy and the amount of data needed. As we described in section 1.1, our final goal is to built an acoustic model that can recognize random people speech with acceptable quality and the minimum data possible.

Baseline system

To evaluate the performance of different speak adaptation approaches explained on section 4.1, we take the tri2b acoustic model as a flat start of our adaptations. Next, we are going to evaluate it using different subsets of data from one speaker as table 4.2 shows. As the different subset of data is selected with a basic script which chose randomly a number defined of utterances from the speaker, it is logical that some sentences are most correlated with the train set than others. Because of that, we must evaluate the results evolution, even when some specific point deteriorate the performance.

Adaptation data	# utterances per speaker	Approx. audio[sec] per utterance
data_10	10	5
data_20	20	5
data_30	30	5
data_40	40	5
data_50	50	5
data_63	63	5

Table 4.2: Different amount of data based on number of utterances/sentences that are will be used on the different approaches of adaptation, with the approximate duration of each sentence in seconds.

The WER% obtained decoding our test set based on just one speaker with the tri2b acoustic model is 14.88. Therefore we will evaluate the improvement of our adapted models with this value as a reference.

4.4 Evaluation

4.4.1 Conditions of evaluation

4.4.2 Experiments evaluation

fMLLR

The fMLLR transformation adapts our speaker independent system tri2b(LDA+MLLT) to our chosen speaker, and it is performed in our case during the decoding phase.

We do not have to use any adaptation data to adapt our model, because it is generated during the decoding. Basically, it performs a first decoding on the speaker independent system and use the decoded transcriptions as the adaptation data for a second pass decoding.

The WER% achieve it with the fMLLR adaptation is 12.68.

MAP

As the MAP transform needs an adaptation data to adapt the speaker independent system, we are going to evaluate the perform of it with different amount of adaptive data described in table 4.2

The adaptation proceeding used in our experiment does not retrain the tree, it just does one iteration of MAP adaptation to the model.

First of all,we are going to evaluate it with different set of the smoothing constant as table 4.3 shows which corresponds to the number of the “fake counts” that we add for the old model. As larger is the value of the smoothing constant, less aggressive is the re-estimation and more smoothing. 20 is the typical value. The amount of adapt data used to test it consist just on 63 sentences.

Smoothing constant value	WER%
10	12.84
15	13.30
20	13.32

Table 4.3 WER% obtained on the MAP estimation based on different values of smoothing constant.

Finally, we are going to evaluate the improvement of MAP adaptation based on the amount of adaptation data used fixing the smoothing constant as 10 (it aims to

little better performance as Table 4.3 shows) as the table 4.4 shows.

Adapted data 10– WER%	Adapted data 20 – WER%	Adapted data 30 – WER%	Adapted data 40– WER%	Adapted data 50 – WER%	Adapted data 63 – WER%
13.30	13.76	12.84	13.76	12.90	12.84

Table 4.4: WER% obtained on the MAP adaptation depending the different amount of data used in terms of number of sentences per speaker. For instance: 25-WER% corresponds to the WER% obtained using as an adapted data a subset of 25 sentences of the specific speaker.

We can observe that generally, word error rate is reducing when most adapted data is available. As it is logical, when most data available, main parameters of the specific features can be better estimated.

MAP+fMLLR

Since the adaptation by MAP or MLLR alone carries to a significant improvement we want to evaluate how this two adaptation techniques work together. When the adaptation data is available, MAP adaptation is executed first and then MLLR adaptation is followed using the adapted parameters.

As we made with MAP adaptation alone, here we also want to evaluate the accuracy of our system depending the amount of adaptation data used as table 4.5 shows.

Adapted data 10 – WER%	Adapted data 20 – WER%	Adapted data 30 – WER%	Adapted data 40– WER%	Adapted data 50 – WER%	Adapted data 63 – WER%
11.52	11.47	11.46	11.93	11.32	11.01

Table 4.5: WER% obtained on the fMLLR+MAP adaptation depending the different amount of data used in terms of number of sentences per speaker.

As we can observe, combine both approaches together lead to a significant improvement on the recognition accuracy. It achieves almost 2% of WER reduction.

SAT

Finally, we evaluate the performance of Speaker Adapted Training, i.e. the acoustic model is trained with the objective of obtain better estimation of the speaker MLLR transforms.

The WER% achieve it with SAT is 12.84.

4.4.3 Results

In this section we are going to show the result of different acoustic training methods presented on the previous sections.

Model/method	WER%	#Clarification
tri2b(LDA+MLLT)	14.88	-
MAP	12.84	63 sentences used as adapted data
fMLLR	12.68	-
MAP+fMLLR	11.01	63 sentences used as adapted data
SAT	12.84	-

Table 3.6: The table shows the WER% of different acoustic models. Furthermore, an additional section of clarification is include to define the amount of data used in approaches based on the requirement of adaptation data.

We can observe that all different adaptation approaches improver the word error rate of our reference model. Maximum a Posteriori and Speaker Adaptive Training obtain same improvement. A little bit more than 2%, but it seems that if we will use more adapted data on the first approach, better results could be achieve.

Moreover, we observe that fMLLR achieve better performance than SAT and MAP adaptation using just 63 sentences of adapted data.

Therefore, as a conclusion, depending of the sistem approach and user requirements, fMLLR and MAP+fMLLR would be selected when it will be necessary speaker adaptation.

If the system will be used in a environment when the speakers often change and they do not have time to register their voices, fMLLR will be ideal.

However, if the system is often used by the same speakers, or people should have time to register their voices, MAP+fMLLR speaker adaptation it would be selected.

5 API

Once we studied and create different acoustic models we create an User Graphic Interface (GUI) written in Python where we can make use of them. Basically, the API allows off-line recognition speech in a comfortable environment using our acoustic models described in previous sections and shows the estimate transcriptions. Moreover, the API allows supervised adaptation in case adaptation data is required.

5.1 Acoustic models

Accordingly with the results obtained in the previous sections, I decided that the acoustic model used to recognize speech automatically it will be ….

Furthermore, it is possible to adapt the system in two different approaches according to speaker requirements:

a) In case speaker does not want to record adaptation data, it should be use fMLLR adaptation

b) In case speaker does want to record adaptation data to improve the performance accuracy, it should be use MAP+MLLR adaptation.

5.2 Procedure

In this section, we describe the main procedure that recognize speech follows.

1) Record speech

To record the speech we make use of the PyAudio in python. The set parameters are the followings:

- Format = pyaudio.paInt16
- Channels = 1
- Rate = 16000
- Chunk = 1024
- Record_seconds = 5

2) Prepare data

Data preparation is an important step in order to recognize speech in Kaldi. As we want to prepare data which we will decode with an already exist system an already existing language model (created in the previous chapters) only the follow documents must be prepared:

- text: This file contains mappings between utterances and utterance ids which will be used by Kaldi

- spk2utt: This is a mapping between the speaker identifiers and all the utterance identifiers associated with the speaker.

- utt2spk - This is a one-to-one mapping between utterance ids and the corresponding speaker identifiers.

- wav.scp - This file is actually read directly by Kaldi programs when doing feature extraction.

All procedure is done automatically using Perl scripts after the first steps.

3) Adaptation or not

If the speaker is not in the database of the training set, It could be possible that the user wants to make use of the speaker supervised adaptation technique.

Depending of the requirements of the user in terms of time, adaptation data could be needed or not. In case adapted data is needed the user must follow some instructions.

4) Result display

Finally, the user will obtain the transcription of the speech.

5.3 API interface

In this section, we explain how is distributed the graphic user interface.

The main windows it is formed by two top level menus. File menu contains all options to performance the speech recognition. Whereas, Edit menu contains different options to configure a few parameters of the acoustic models like number of jobs used to decode, or smoothing factor in MAP adaptation among others. Figure 5.1 shows how it looks like the main interface.

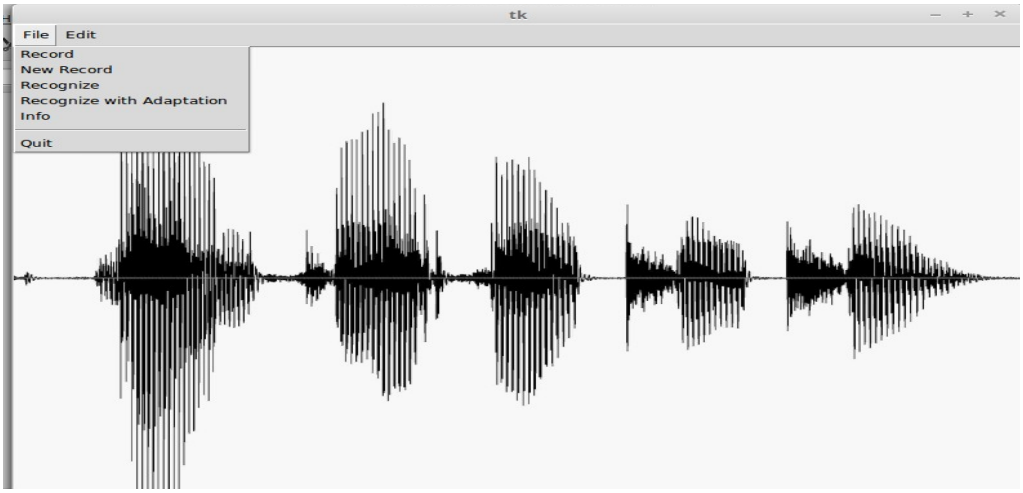

Figure 5.1: Main windows of the ASR API.

In the File menu, there are different options available. Record a new audio file to include on the audio folder to recognize or start a new record removing all existing audio files to make a new recognition.

Moreover, there are two different options of recognition as we explain in the previous sections: It is possible to recognize speech with our selected acoustic model tri2b explained in capitol 3 or it is possible to adapt the system to the speech characteristics of an specific speaker. Figure 5.2 and 5.3 show how is the process of adapting with fMLLR technique.

Figure 5.2: Selection of the required adaptation on the GUI.

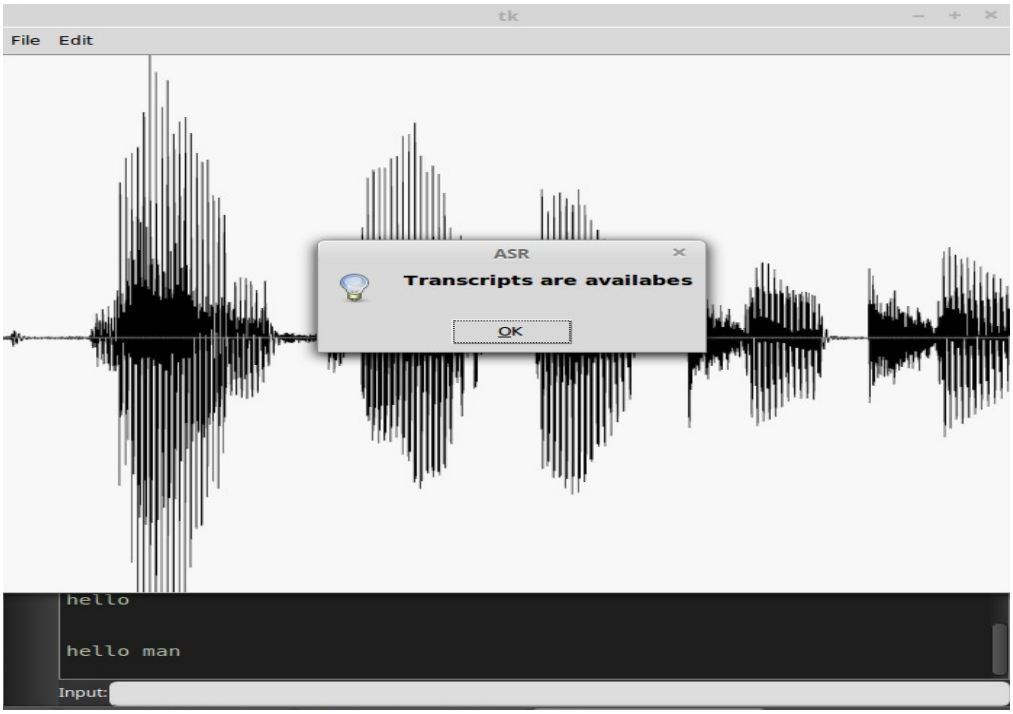

Figure 5.3: Estimated transcriptions of the recorded speech.

Furthermore, information option allows to read about which kind of adaptation is more appropriate depending on speakers requirements.

6 Conclusion

Different approaches to train and adapt acoustic models have been studied to be able to built an accurate automatic speech recognition system. On one hand, the training part is defined as the most important step since it will be which determine mainly the accuracy of our system. In our experiments, we observed during the training phase a reduction of 20,82% in terms of word error rate from the initial mono-phone acoustic model to the final system based on a discriminative training on top of a trip-phone acoustic model with LDA+MLLT feature transformations. Moreover, the amount of training data available is a decisive parameter in order to get good results in term of recognition accuracy. As more data obtainable, better results could be achieve.

On second hand, we observe that depending the requirements of the user, and the amount of adaptation data available different approaches of adaptation could be used. In this experiments, the adaptation step lead us to a reduction of almost 3% in the word error rate, a quality measure of speech recognition accuracy.

After the realization of the thesis, we can express that we achieved the goals defined on the introduction section. We start with almost any knowledge about what automatic speech recognition was and we end with a remarkable learning of it, and with an accurate model which allows to recognize speech.

ASR is a complex part of signal processing that has a lot of fields to study. Future plans include the incorporation of an On-line Latgen Recognizer as well as the use Subspace Gaussian Mixture Models to try to performance a multilingual acoustic modeling, amount others.

As all in life, once the first step is done, in this case an ASR system is built, a lot of different options, that you have not noticed at the beginning, emerge.

7 References

[1] Wolfgang Macherey, *Discriminative Training and Acoustic Modeling for Automatic Speech Recognition*, 2010.

[2] Suba,P. and Bharathi,B. *Analysing the performance of speaker identification task using different short term and long term features.* 2014 IEEE International Conference on Advanced Communication Control and Computing Technologies (ICACCCT).

[3] http://kaldi.sourceforge.net

[4] Yu Hongzhi, *A Research on Recognition of Tibetan Speakers Based on MFCC and Delta Features.* Proceedings IFCSTA. IEEE, 2009, volume 2: pp, 234-238.

[5] R. Haeb-Umbach and H. Ney, *Linear discriminant analysis for improved large vocabulary continuous speech recognition,* Proceedings ICASSP. IEEE, 1992, pp, 13-16.

[6] Omar, M.K. and Hasegawa-Johnson, M. *Model enforcement: a unified feature transformation framework for classification and recognition,* Signal Processing, IEEE, 2004, volume 52: pp,2701-2710.

[7] Steve Renals, *Automatic Speech Recognition. Search and decoding.* 2012 ASR Lecture 10.

[8] D.Povey and A. Ghoshal, *The Kaldi Speech Recognition Toolkit*, ASRU 2011.

[9] C. Allauzen, M. Riley, J. Schalkwyk, W. Skut, and M. Mohri, “OpenFst: a general and efficient weighted finite-state transducer library,” in Proc. CIAA, 2007.

[10] Mohri, Pereira and Riley, *Speech Recognition with Weighted Finite-State Transducers*, in Springer Handbook on SpeechProcessing and Speech Communication, 2008.

[11] Haofeng Kou and Weijia Shang, *Parallelized Feature Extraction and Acoustic Model Training*, Digital Signal Processing. Proceedings ICDSP, IEEE, 2014.

[12] D. Povey, *Discriminative Training for Large Vocabulary Speech Recognition*, PhD thesis, Cambridge University Engineering Dept, 2003

[13] D. Povey, D. Kanevsky, B. Kingsbury, B. Ramabhadran, G.Saon and K. Visweswariah, Boosted MMI for model and feature-space discriminative training, ICASSP, 2008.

[14] D. Povey and P.C Woodland, *Minimum Phone Error and I-Smoothing for improved discriminative training*, Cambridge University Engineering Dept, 2002

[15] http://www.voxforge.org/home

[16] K. Shinoda, *Speaker Adaptation Techniques for Automatic Speech Recognition*, Tokyo Institute of Technology, Japan.

[17] J. Ganitkevitch, S*peaker Adaptation using Maximum Likelihood Linear Regression*, Seminar ASR,2005.

[18] Nguyen, P. , Gelin, P. , Junqua, J.C and Chien, J.-T. , *N-best based supervised and unsupervised adaptation for native and non-native speakers in cars*. Acoustics, Speech and Signal Processing, IEEE International Conference, 1999, volume 1: pp, 173-176.

[19] R. Chengalvarayan and L. Deng, *A Maximum A Posteriori Approach to Speak Adaptation Using the Trended Hidden Markov Model*, IEEE Transactions on Speech and Audio Processing, Vol 9: NO 5, 2011.

[20] Gales, M.J.F and Pye,D. *Variance compensation within the MLLR framework for robust speech recognition and speaker adaptation.* Proceedings Spoken Language, IEEE, 1996. ICSLP 96.

[21] D.Povey, H-K J. Kuo and H. Soltau, F*ast Speaker Adaptive Training for Speech Recognition*, Interspeech 2008.

8 Acronyms

ASR: Automatic Speech Recognition
HMM: Hidden Markov Models
AM: Acoustic Model
LM:Language Model
MFCC: Mel Frequency Cepstral Coefficients
PLP: Perceptual Linear Prediction
MLLT: Maximum Likelihood Linear Transform
FFT: Fast Fourier Spectrum
LDA: Linear Discriminant Analysis
HLDA: Heteroscedastic Linear Discriminant Analysis
MLLR: Maximum Likelihood Linear Transform
FST: Finite-State Transducers
GMM: Gaussian Mixture Model
SGMM: Subspace Gaussian Mixture Model
WER: Word Error Rate
MLE: Maximum Likelihood Estimation
EM: Expectation-Maximization
MMI: Maximum Mutual Information
bMMI: Bosted Maximum Mutual Information
MPE: Minimum Phone Error