Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Master Thesis
at the Interactive Systems Laboratory

Grapheme Based Speech Recognition

Mirjam Killer

Pittsburgh, PA USA
10. March 2003

Supervisors CMU: Prof. Dr. Alex Waibel, Dr. Tanja Schultz
Supervisors ETH: Prof. Dr. Lothar Thiele, Dr. Beat Pfister

Abstract

Large vocabulary speech recognition systems traditionally represent words in terms of smaller subword units. During training and recognition they require a mapping table, called the dictionary, which maps words into sequences of these subword units. The performance of the speech recognition system depends critically on the definition of the subword units and the accuracy of the dictionary. In current large vocabulary speech recognition systems these components are often designed manually with the help of a language expert. This is time consuming and costly. In this Diplomarbeit graphemes are taken as subword units and the dictionary creation becomes a triviality.

The grapheme based context dependent speech recognizers cluster or tie subsets of the subword units together to improve recognition accuracy when the system is trained with limited data. In current automatic speech recognition systems this is done by a set of linguistically motivated questions designed by an expert applied to a decision tree. In this project four different ways to generate such question sets are examined. First of all phoneme based linguistic questions are transformed into grapheme based ones serving as a baseline for the other question experiments. Second a bottom-up clustering and a hybrid clustering procedure based on the entropy distance generate two different question sets and third each grapheme is taken as a single question creating the singleton question set. It is also tested if it is reasonable to cluster trigraphemes, quintgraphemes or even septgraphemes. Recognizers in English, Spanish and German are trained and evaluated. The evaluation shows that graphemes as subword units and automatically generated question sets come close in performance to the systems using phonemes as subword units and expert knowledge.

Multilingual speech recognizers are designed with the intention to reduce work without loosing performance by sharing information between languages. Two ways of multilingual acoustic context modeling are examined for grapheme based three lingual speech recognizers, sharing data in English, German and Spanish . One system shares the same distribution and codebook models for each grapheme among all languages whereas the other system keeps the language information and models each grapheme with a language dependent distribution while sharing the codebook language independently.

It is investigated if information from the multilingual systems can improve the creation of a grapheme based speech recognizer in Swedish through bootstrapping. The Swedish bootstrapped recognizer shows that much better results can be reached with fewer trainings iterations compared to a flatstart.

Diplomarbeit

Interactive Systems Lab, Carnegie Mellon University Pittsburgh, PA
ETH Zürich

Grapheme based Speech Recognition

With the distribution of speech technology products all over the world, the portability to new target languages becomes a practical concern. As a consequence the research on rapid deployment focuses on creating methods for automatizing the design of speech systems for new languages with information gathered from already studied languages. Especially the design of pronunciation dictionaries for Large Vocabulary Continuous Speech Recognizers is a very time and cost consuming task since so far it requires partly manual design and development by human experts.
The idea of my Diplomarbeit / this project is to make the process of manually creating a pronunciation dictionary superfluous by implementing a completely data-driven method. In doing so, I will implement a speech recognizer that is not relying on phonemes as acoustic units but on graphemes. The step of transforming the transcription of a word into a string of phonemes (grapheme-to-phoneme mapping) is therefore no longer necessary. The grapheme based recognizer will function on context-dependent models, which are generated by applying a decision tree defined through a set of questions to the orthographic representation of words.
The results will be tested on a variety of languages that provide us with a wide range of close grapheme-to-phoneme relation to very loose one.

Tools:	JRTk (Janus Speech Recognition Tool Kit)
Supervisors:	Tanja Schultz (CMU) , Beat Pfister (ETH)
Professor:	Alex Waibel (CMU) , Lothar Thiele (ETH)

Acknowledgements

I would like to thank Alex Waibel for giving me the opportunity to work on my Diplomarbeit at the Interactive Systems Laboratory at the Carnegie Mellon University in Pittsburgh, USA and Beat Pfister as well as Lothar Thiele for enabling my visit.
A special Thank-you goes to Tanja Schultz and Sebastian Stücker for their support and guidance throught this project.
I enjoyed my stay (despite the freezing weather) very much which I accredit to the positive and inspiring atmosphere at CMU and its labs.

11th March 2003, Pittsburgh USA

[illegible]

Contents

List of Figures

List of Tables

Chapter 1

Introduction

One of the core components of a speech recognition system is the pronunciation dictionary. A dictionary is basically a mapping table that translates words into their subunits. Large vocabulary continuous speech recognition (*LVCSR*) systems do not usually use whole words as the basic units for classification because the vocabulary usually holds some ten thousand words. Thus there are enough training examples for every word in the vocabulary. The acoustic samples of some words might never be seen during training and can therefore not be learned by the recognizer. For these reasons the speech recognition system segments each word in the vocabulary into subunits. These subunits occur more frequently than words and for that reason are trained more robustly. It is also easier to deal with words unseen during training since they can just be decomposed into the subunits. Usually those subunits are phonemes and thus the pronunciation dictionary defines the pronunciation of a word.

The performance of a recognition system depends critically on the choice of subunits and the accuracy of the dictionary [4]. An accurate mapping of the orthographic representation of a word onto a subunit sequence is important to ensure recognition quality because otherwise the acoustic models (a HMM state for a ”poly-subunit”) is trained with the wrong data or during decoding the calculation of the scores for a hypothesis is falsified by applying the wrong models.

The dictionaries are usually created manually by language experts and depend on phonemes as subword units. But for large vocabularies this task gets very time consuming and expensive. Especially if a recognizer for a new language has to be built fast or if no experts in this language are available new methods to resolve this issue are needed. Besides manually designed subword units may generalize well, but they might not be the optimal units of classification for the specific task or environment for which the system is trained [4].

1.1 Big Picture

Several researchers have made an effort to automatically identify subword units or the optimal pronunciation for a word in terms of manually defined subword units (for more references see [4]). The problem of automatically designing subword units and dictionaries given only a set of acoustic signals and their transcripts has been addressed by [4]. Their approach does not depend on initial labels, nor makes any assumptions about the availability of any a priori knowledge or information. They look at the problem from a modeling perspective and try to identify sound classes that best fit the training data. The closeness of fit to training data is quantified by likelihood, e.g. if a set of symbols and the acoustic models for those symbols are found, such that the dictionary and the acoustic models together best fit the data, the ML solution for the problem is obtained. The results (see [4]) of their automatically generated subword units and dictionary produced models that performed worse than the manually designed subword units and dictionary. Human experts use knowledge from experience with a huge amount of hours of speech, as well as other consciously or subconsciously acquired knowledge. Their manually designed phone set is therefore expected to be generalize well.

Other approaches built a set of rules to create the pronunciation for words in a certain languages. Especially for languages with a close grapheme to phoneme relation this works well. Spanish is such an example. In other languages this is impossible (e.g. Chinese see the following Sections in this Chapter).

There are also data driven approaches (refer to [5]) that tried to create dictionaries by voting among phoneme recognizers for a new unseen language in nine different languages other than the target language. Each time frame is decoded by those nine language recognizers resulting in nine different hypotheses for a phoneme per frame. An algorithm then maps the most prospective hypothesis into a dictionary entry. The results didn't allow creation of dictionaries that were accurate enough to train large vocabulary continuous speech recognizers with them.

[6] work with spontaneous speech where word pronunciations vary more than in read speech. To cope with pronunciation variations in spontaneous Japanese speech they built pronunciation dictionaries from real speech data. An automatic method generates pronunciation dictionaries based on pronunciation networks which give the most plausible pronunciations. First they use regular phoneme recognition results to train multi-layer perceptron neural networks to predict these correct pronunciations from the symbol sequence and finally produce the dictionaries automatically with these neural networks. Their experiments show that automatically-derived pronunciation dictionaries give consistently higher recognition rates and require less computational time for recognition than the conventional dictionary.
This project tries to evaluate to what extend graphemes are suitable subunits for speech. Although it is not expected to reach equal performance to manually created subunits and dictionaries (for reasons mentioned above e.g. manually created subunits are expected to be highly generalizing and a huge amount of additional knowledge is included in the design), using graphemes is a very easy and fast approach. The idea on one hand was motivated by results from [7], who made large vocabulary speech recognition based on context-dependent acoustic modeling using graphemes for subunits, as well as by [8] who made speech recognition based on character units for Arabic. Each model for a character in Arabic (see Section 1.4.1), e.g. a consonant, can thus model the consonant alone, as well as the consonant followed by vowels. Their system which regards each Arabic character as a phoneme performed surprisingly well.
On the other hand there is an increased effort to romanize most scripts. For example logographic script systems like Chinese *Hanzi* (a grapheme stands for a whole word)

are very hard to learn. Therefore romanization is an interesting and often necessary operation when working with those languages. Romanizing a word meaning symbol conversion based on the transcription principle results per definition in the pronunciation of the word and thus making it suitable as pronunciation dictionary entry. Or languages with no existing script system are mostly transcribed using Roman letters (see Section 1.4.1).

The following Sections 1.3 and 1.4 give an overview over the languages of the world and the used script systems. It shows that the Roman alphabet is the most widely used and therefore the approach of this Diplomarbeit is applicable to numerous languages.

1.2 Objective

In this Diplomarbeit I like to show that instead of using phonemes it is possible to use graphemes as subunits. The creation of pronunciation dictionaries will therefore be a triviality and for a great deal of languages this enables us to safe a lot of time and work. Grapheme recognizers in 4 different languages, namely English, Spanish, German and Swedish, are built and it is shown that the word accuracies of the context dependent recognizers are comparable to the ones of phoneme recognizers. It is discussed how questions to create a context dependent environment can be generated automatically and how the context width influences the recognition results.
Furthermore I will investigate if grapheme based speech recognition is also qualified for building multilingual recognizers and how the models are best shared among the languages.

1.3 Languages of the World

The following pictures and information are taken from [9]. According to [10, 11, 12, 13, 9] the number of languages varies in between 5000 to 7000. [11] notes that there are probably as many different answers to this question as there are linguists. Usually

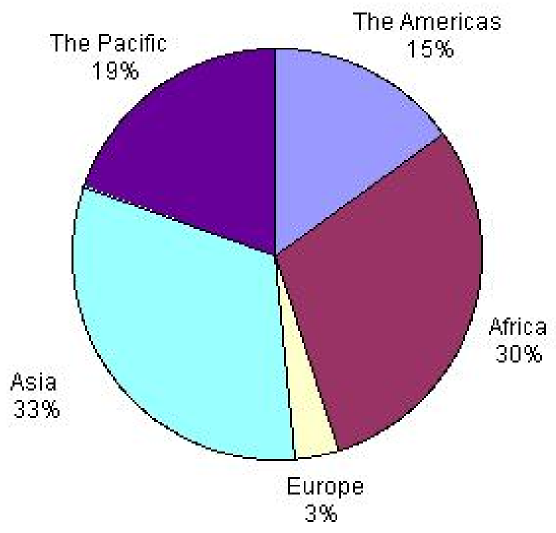

Figure 1.1: Geographic Distribution of Living Languages in 2000

languages are called different if the speakers from one language do not understand the speakers form the other language, otherwise it is mostly referred to as a dialect (a counter example to this is Norwegian and Swedish).

	Total Living Languages	Percentage
The Americas	1013	15%
Africa	2058	30%
Europe	230	3%
Asia	2197	32%
The Pacific	1311	19%
TOTAL	6809	

1.4 Writing Systems

Information and pictures are taken from [10].

العربيّة

Figure 1.2: Sample of Arabic script

There are numerous writing systems in use around the Globe. Every one of them has its own characteristics and specialities. Over time languages and their scripts might change, dissapear or be newly created. In [11] it is explained that alphabet scripts change over time because they are derived form the pronunciations of words which can change over time, whereas e.g. the Chinese script systems is manly invariant to time because they symbols not only represent sounds but have its own meanings.

1.4.1 Alphabetic writing systems

Abjads

Abjads, also called consonant alphabets, represent only consonants or consonants plus a few vowels. Most Abjads (with a few exceptions) are written from right to left.

Some scripts, such as Arabic (see Figure 1.2), are used both as an abjad and as an alphabet.

Alphabets

Alphabets, or phonemic alphabets, represent consonants and vowels. The most widely used alphabet today is the Roman alphabet (see Figure 1.3) [13]. Most European nations, some nations in Asia, almost all countries in Africa and all nations of America and Oceania use the Roman alphabet [13]. The earliest known inscriptions in the Roman alphabet date from the 6th century B.C. The Romans used just 23 letters to write Roman:
A B C D E F G H I K L M N O P Q R S T V X Y Z
which included no lower case letters. The letters J, U and W were added to the alphabet at a later stage to be able to write languages other than Roman. The modern Roman alphabet consists of 52 letters, 10 numerals, punctuation marks and a variety of of

Figure 1.3: Sample of Roman script

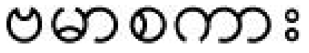

Figure 1.4: Sample of Burmese script

other symbols such as &, % and .
Many languages supplement the basic Roman alphabet with accents or extra letters used to modify pronunciation of a letter to indicate where a stress should fall in a word to put emphasis in a sentence, to indicate pitch or intonation of a word or syllable, to indicate length of a vowel or to visually distinguish homophones.

Some of the languages written with the Roman alphabet are shown in Table 1.1.

1.4.2 Syllabic writing systems

Syllabic alphabets

Syllabic alphabets, or alphasyllabaries, consist of symbols for consonants and vowels. Consonants are combined with a vowel that is changed or silenced by diacritic signs. Vowels can be written separately if they occur at the beginning of a word or by themselves. There are also special conjunctive symbols used to add essential parts to the first letter or letters after the final one when two or more consonants occur together. Burmese (see Figure 1.4) is an example of such a script.

Syllabaries

Syllabaries consist of separate symbols for each syllable of a language. Japanese Hiragana belongs to this group of writing systems (Figure 1.5).

Afrikaans	Hausa	Occitan
Ainu	Hawaiian	Polish (Polski)
Albanian (Shqip)	Hmong	Portuguese (Português)
Amo	Hopi	Prussian
Aymara	Hungarian (Magyar)	Quechua
Azeri	Ibibio	Romanian (Limba Român)
Balear	Icelandic (Íslenska)	Romany
Basque (Euskara)	Ido	Sami
Batak	Indonesian (Bahasa Indonesia)	Scottish Gaelic (Gàidhlig)
Bosnian	Ingush	Shona
Breton (Brezhoneg)	Interlingua	Slovak (Slovenský)
Catalan (Català)	Inuktitut	Slovene
Cherokee (Tsalagi)	Iñupiaq	Somali
Cornish (Kernewek)	Irish (Gaeilge)	Spanish (Español)
Corsican	Italian (Italiano)	Swahili
Cree (Nehiyaw)	Javanese	Swedish (Svenska)
Croatian (Hrvatski)	Kanuri	Tagalog
Czech	Khasi	Tagbanwa
Danish (Dansk)	Kirghiz	Tahitian
Dutch (Nederlands)	Kurdish	Tajik
Edo	Lapp	Tamazight
English	Latin	Turkish (Türkçe)
Esperanto	Latvian	Turkmen
Estonian (Eesti)	Lithuanian	Uighur
Faroese	Lushootseed	Uzbek
Fijian	Luxemburgish	Valencian
Finnish (Suomi)	Malay (Bahasa Melayu)	Vietnamese
French (Français)	Maltese (Malti)	Volapük
Frisian (Frysk)	Manx (Gailck)	Welsh (Cymraeg)
Galician (Gallego)	Naga	Yi
Gascon	Navajo (Diné)	Yoruba
German (Deutsch)	Naxi	Zulu (isiZulu)
Guarani	Norwegian (Norsk)	
Hanunóo		

Table 1.1: Some of the languages written with the Roman alphabet

ひらがな

Figure 1.5: Sample of Hiragana (Japanese) script

中文

Figure 1.6: Sample of Chinese (Zhöngwén) script

1.4.3 Logographic writing systems

In logographic writing systems each symbol not only represents a sound, but also has a meaning. Thus these writing systems usually have a numerous amount of symbols ranging from several hundred to tens of thousands. There is no theoretical upper limit to the number of symbols in some logographic scripts such as Chinese 1.6.

Logographic scripts may include the following types of symbols:

Logograms - symbol representing parts or whole words. If logograms visually resemble the things they stand for they are also known as pictograms or pictographs.

Ideograms - symbols which graphically represent abstract ideas.

Semantic-phonetic compounds - symbols which include a semantic element, which represents or hints at the meaning of the symbol, and a phonetic element, which denotes or hints at the pronunciation.

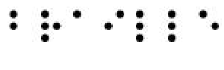

Figure 1.7: Sample of Braille script

1.4.4 Alternative writing systems

There are other additional writing systems and scripts invented for books, movies, computer games or constructed languages. Figure 1.7 shows the Braille script which consists of patterns of raised dots which enable blind and partially sighted people to read by touch.

Chapter 2

Speech Recognition Basics

This chapter gives a brief overview of the automatic speech recognition techniques used in this Diplomarbeit. It is not a general introduction to speech recognition basics but only covers the theory and models needed to understand this project. For more general and in depth explanation refer to [14, 15].

What makes *Automatic Speech Recognition* so interesting is first of all the fact that speech is the most natural and easiest way of human communication. With the increase of technology in everyday life we would like to adapt computers to human behavior, thus making it easier for us to operate machines and computers. For example *Speech Recognition* can be a big help for physically challenged people. It would eliminate the need to use your hands for certain daily challenges like turning on a TV or a radio. *Speaker identification or authentication* is another promising area which eliminates the need to remember all kinds of passwords or pins. The computer simply recognizes who is speaking and sets the access control accordingly. With the boom of cell phones and small palm tops the desire of better ways to input information arises. It is utterly annoying to type a text on a cell phone keyboard and it would be a big relief to just dictate an SMS-message or even an email.

The speech recognition process seems fairly easy for us humans. We have to be aware though that we use an enormous database of background knowledge, e.g. our "world-knowledge" to decode uttered sentences. We not only use syntactical and grammatical knowledge, but also analyse if the utterance makes sense and if it is pragmatically correct. For example *"The police man smoked my ID"* expresses a syntactical and semantical correct sentence, but (for humans easy to see) it makes no sense. The difficulties of *Automatic Speech Recognition* lie in segmenting the data (e.g. where does a word start or end), the complexity of the data *(how many different words are there and how many different combinations of all those words are possible)*, the variability of the speakers *(women compared to men have a higher basis frequency; or microphones, telephones limit the bandwidth, etc.)*, ambiguity of words *(two vs. too)*, word boundaries *(interface vs. in her face)*, semantics *(he saw the grand canyon flying to New York)* and ambiguity in pragmatism *(time flies like an arrow)*. *Automatic Speech Recognition* started with speaker-dependent single word recognizers that processed only a small amount of words in quiet surroundings. Not until the late eighties was it possible to handle continuous speaker-independent speech with a large vocabulary. Today's focus lies on system development for spontaneous or colloquial speech with a noisy background as it can be found in a car. The systems have to be able to adapt to new situations quickly and optimally.

The speech recognition problem can be formulated as follows: For a given acoustic signal $\mathbf{X}$ we'd like to find the word sequence $W*$ which was most likely produced by the event $\mathbf{X}$. This is known as the *Fundamental Equation of Speech Recognition*:

$$\begin{aligned} W* &= \underset{W}{argmax}\, P(W|\mathbf{X}) \\ &= \underset{W}{argmax}\, \frac{P(\mathbf{X}|W) \cdot P(W)}{P(\mathbf{X})} \\ &= \underset{W}{argmax}\, P(\mathbf{X}|W) \cdot P(W) \qquad (2.1) \end{aligned}$$

We can divide this equation further into three basic problems:

Acoustic Model: Calculation of the conditional probability $P(\mathbf{X}|W)$ to observe a signal $\mathbf{X}$ given a word sequence W was spoken.

Language Model: Calculation of the a priori probability $P(W)$ that word sequence W was spoken.

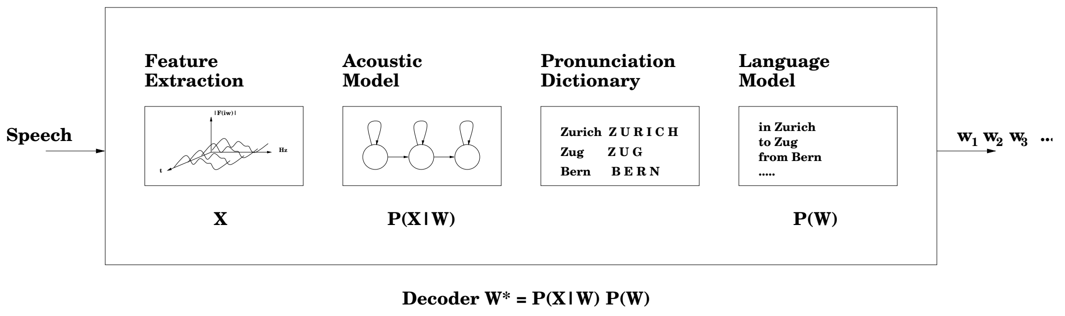

Figure 2.1: Block diagram of an automatic speech recognizer

Search: Efficient calculation of word sequence $W*$ that maximizes $P(\mathbf{X}|W) \cdot P(W)$

We like to map the spoken utterance onto a textual representation in form of a sequence of words $W* = w_1 w_2 w_3 \ldots$. First of all we need to extract proper feature vectors that sufficiently characterize the acoustic signal. To calculate the acoustic models $P(\mathbf{X}|W)$ we split the word sequences into subunits (usually phonemes) with the help of a *pronunciation dictionary* (defines how to segment each word into the subunits) and a *language model* (provides the probabilities for combining different words to a sequence of words). A *monolingual speech recognizer* (see Figure 2.1) consists of a feature extraction component, an acoustic model, a pronunciation dictionary, a language model and a decoder that searches for the optimal word sequence $W*$.

2.1 Preprocessing

The first step is to process the analog sound signal in such a way that it can be handled by a computer. The signal is digitalized and to reduce the tremendous amount of data certain relevant features have to be extracted. Those extracted features are then subject to further adaptation.

2.1.1 Digitalization

The recorded speech signal is transformed into a digitalized signal by an *analog-digital converter*. The sampling rate needs to be at least twice the bandwidth of the speech signal to be able to fully reconstruct it. This is known as the *Nyquist theorem.* Humans listening ability changes with age. In our younger years we might be able to hear anything form $20Hz$ to $20kHz$, but with age the upper bound lowers. So we can approximately say that humans hear a range of $20Hz$ to $16kHz$. Speech only covers parts of that frequency range. A sampling rate of $20kHz$ allows to represent most speech sounds, but for speech recognition it is sufficient to use a sampling rate of $16kHz$. The amplitudes are quantized with a $8-$ or $16-bit$ resolution. A $16kHz$ sampling rate and a $16-bit$ amplitude quantization results in $32kB$ data per second. The next step is therefore needed to reduce the amount of data.

This Project: The acoustic data used for this project was digitally recorded with a sampling frequency of $48kHz$ and a $16-bit$ amplitude resolution. Transmitted onto a PC it was then downsampled to 16kHz.

2.1.2 Short-term Analysis

Speech is quasi-stationary in periods of $10-30ms$. With a suitable windowing function samples are cut out and analyzed. Certain time domain or spectral features are then extracted from each time-frame of the signal. It is common to work with Cepstral- or Melscale-Spectral coefficients in combination with Log-Energy coefficients.

This Project: A *Hamming*-window cuts out 256 samples corresponding to $16ms$ time frames. The window is shifted over the data with an offset of $10ms$ thus neighboring segments overlap by $6ms$. With the *Discrete Fourier Transform* 129 spectral coefficients are calculated from the 256 samples.

2.1.3 Vocal Tract Length Normalization

VTLN is an adaptation method that tries to reduce the effects of different sized vocal tracts through transformation in the spectral domain. Men have usually a longer tract than women which causes men to have a deeper voice (e.g. a lower fundamental frequency). A gender- and speaker-independent modeling would introduce on top of other speaker-variabilities also a variation in short or long vocal tracts. The goal of *VTLN* is to reduce the impact of the variation of vocal tracts during preprocessing. The length of the vocal tract l of a speaker is estimated from the speech signal by a maximum likelihood approach. The power spectrum is then linearly transformed in order to match the vocal tract of average length.

This Project: The 129 spectral coefficients are adapted by a vocal tract length normalization. A warping parameter α is estimated in the range $0.8 - 1.2$ and the spectrum is linearly transformed. The 129 coefficients per $10ms$ are then reduced to 30 dimensions with a Mel-scale filterbank. Mel-scale filterbanks model the properties of the human ear. The so called critical bands combine frequencies together in such a way that lower frequencies have a higher resolution and higher frequencies a lower resolution. The resulting 30 Mel-Scale coefficients are then logarithmized and with an Inverse Fourier Transform transformed into 30 Cepstral-coefficients. From these 30 coefficients only 13 are used further. With a Cepstral-Mean Value subtraction the cepstren are exempt from mean value. All $10ms$ we now have a vector with 13 components. These vectors are stationary snap-shots of the speech signal therefore they are complemented with dynamic features. The first and second derivative of the 13 cepstren are approximated and the zero-crossing rate as well as the signal energy are calculated. The resulting 43 features are then combined to one feature vector.

2.1.4 Linear Discriminant Analysis

After extracting the desired features a linear discriminant analysis (*LDA*) is carried out to reduce the dimension. *LDA* tries to transform the axis in such a way that classes become separable. The LDA-Matrix is then determined such that it minimizes the average variance inside a class and at the same time maximizes the variance between classes. Feature vectors belonging to the same class rank closer together while the different class centers move away from each other. The classification process thus becomes easier. The coefficients of the *LDA-Matrix* are sorted in decreasing order according to their variances. Coefficients of higher order are less important for the classification process since they only have small variances. Therefore they can be omitted thus reducing dimensionality of the feature vector coefficients.

This Project: The LDA-Transformation reduces the 43-dimensional vectors down to 32 dimensions.

The preprocessing methods are mostly independent of languages. But the selection of the used feature vectors may well vary from one language to another. For example in *tonal languages* the progression of the basic frequency can change the meaning of a word. In such cases it makes sense to include information of the basic frequency course in the used feature vector set. In non-tonal languages this information is irrelevant.

2.2 Acoustic Model

To model the conditional probability $P(\mathbf{X}|W)$ of a signal $\mathbf{X}$ given a word sequence W we need an appropriate design that approximates the problem. Speech is a time variable, continuous and very complex phenomenon. A word for example can sound very different depending on coarticulation effects, speaker dependent pronunciation variants or characteristics of the transmission channel. Since it is infeasible to model $P(\mathbf{X}|W)$ for each word sequence (there are way too many possible word combinations), smaller units are modeled. The fragmentation of words into smaller units brings along a few other problems. First of all a pronunciation dictionary is required to split the words into the subunits. Secondly a time alignment is needed. The beginning and ending of subunits have to be found first. There are various other problems in automatic speech recognition. There are coarticulation effects at word transitions (a word be pronounced very differently depending on the context). For example American English has a lot of coarticulation effects.

Most known continuous speech recognition systems at time are based on the idea of *HMM: Hidden Markov Models*. Speech can be seen as a stochastic process. It is obvious that the same phoneme can be pronounced very differently by various people and even the same person pronounces phonemes differently at times. *HMMs* offer a way to model speech and certain of its problems.

2.2.1 Hidden Markov Models

An introduction to *HMMs* can be found in [14, 16]. A *HMM* λ is defined by the following components:

- S set of all N states $S := \{S_1, S_2, \dots, S_N\}$
- π probability distribution. Each state S_i has a certain probability $\pi_i = P(q_1 = S_i)$ to be the starting state of a state sequence.
- A $N \times N$ matrix of the transition probabilities. An entry a_{ij} of A stands for the probability $p(S_i|S_j)$ that given state S_i, state S_j follows.
- B set of emission distributions $B = b_1, b_2, \dots, b_N$. $b_i(x)$ is the probability that x is observed in state S_i.
- V set of observable features which can either be discrete or continuous.

To simplify calculations, state transitions only depend on the directly preceding states hence the name *Markov Models.* According to the start probabilities π a starting state S_1 is selected. With the probability a_{ij} the system changes from the current state S_i to S_j. In each state S_i the emission probabilities $b_i(x)$ are produced by some *hidden* (only the output of b_i is observable, not the process producing it) random process according to which the most likely observed feature is selected (see Figure 2.2).

Discrete HMMs have a discrete feature vector space. In this case the emission probabilities $b_i(x)$ are given by probability tables over the discrete observable features V. For *continuous HMMs* the feature vector space is continuous and the emission probabilities are now probability densities. Usually the emission probabilities $b(\mathbf{x})$ are approximated by a Gaussian distribution with a mean value vector μ and the covariance

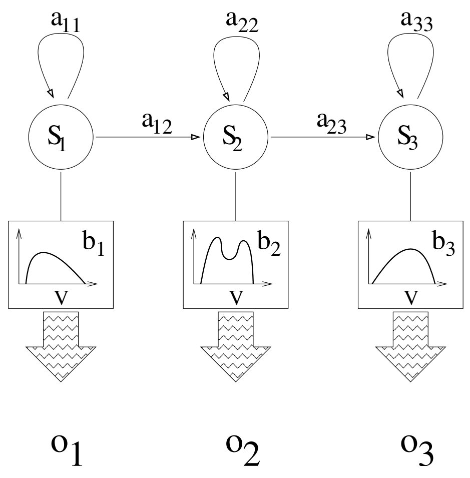

Figure 2.2: Generation of observed features using HMMs (Picture taken from [14])

matrix Σ:

$$b_i(\mathbf{x}) = \sum_{l=1}^{L_i} c_{il} \cdot Gauss(\mathbf{x}|\mu_{il}, \Sigma_{il}) \tag{2.2}$$

$$\sum_{l=1}^{L_i} c_{il} = 1 \tag{2.3}$$

L_i is the number of mixture distributions used in state S_i. The weight coefficients c_{il} are called *Mixture Components*. The Gaussian mixture distribution is defined as follows:

$$Gauss(\mathbf{x}|\mu_{il}, \Sigma_{il}) = \frac{1}{\sqrt{(2\pi)^d |\Sigma|}} \cdot e^{\frac{1}{2}(\mathbf{x}-\mu)^T \cdot \Sigma^{-1}(\mathbf{x}-\mu)} \tag{2.4}$$

$$\tag{2.5}$$

d is the dimensionality of the feature vector space (e.g. $V = \mathbb{R}^d$). See Figure 2.3 for an example of a one dimensional Gaussian mixture distribution.

The mean value vectors μ_{il} and the covariances Σ_{il} are referred to as the *Codebook* of a model. Fully continuous HMMs so called *CDHMMs* offer for each state a probability distribution with its own mixture components c_{il}, variances Σ_{il} and mean value vectors μ_{il}. Where as in semicontinuous HMMs (*SCHMM*) states share among each other the

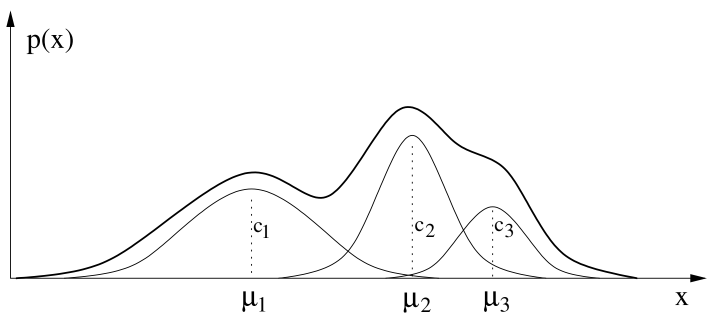

Figure 2.3: One dimensional Gaussian mixture distribution (Picture taken from [14])

variance matrix Σ and the mean value matrix μ, but have state dependent mixture weights c_i.

To solve recognition problems with the given HMM models one needs to build sentence models out of word models and those again out of sequences of subunits.

The evaluation problem Calculates the probability $P(O|\lambda)$ that a certain observed feature sequence $O = o_1 o_2 \dots o_T$ was produced by the HMM-model λ. The *Forward Algorithm* solves this problem.

The decoding problem Identify the most likely path $q*$ that produces the observed feature sequence O . The *Viterbi Algorithm* is used for this problem.

The optimization problem Given a model $\lambda = (A, B, \pi)$ from a word w and its acoustic realization O, find the parameters $\lambda*$ that maximize the probability to produce O. The *Forward-Backward Algorithm* is used.

The algorithms used are described plentifully in the referenced literature.

2.2.2 Suitable Units for Speech

Before the time of continuous speech recognition words or *morphemes* were commonly used as subunits. Morphemes are well fitted for a single word recognizer. In continuous speech there is a tremendous amount of possible words and word combinations. It got infeasible to write down all possible morphemes and it wasn't possible anymore to

find enough training data for each such unit. The simplest way to split up words is to decompose them into their *syllables*. Syllables model coarticulation effects between phonemes and capture the accentuation of a language. Although they are limited in number they are still various enough so that trainings problems might arise. The number of *phonemes and subphonemes* are well below the number of syllables. Usually there are in between 30 to 50 phonemes and subphonemes. They are well trainable and offer the advantage that a new word can be added very simply to the vocabulary. In the GlobalPhone project a phoneme is divided into 3 subphonemes to model the inter-phoneme dynamic. Especially in multilingual speech recognition it is interesting to use phonemes since it offers the perspective of an universal phoneme inventory.
In today's systems this is improved by looking at phonemes in their various contexts. If only the immediate left and right neighbor is considered we talk about *Triphones*. The term *polyphones* is used if we talk about an unspecified neighborhood. *Subpolyphones* are modeled subject to their left and right neighbor phonemes. The number of context dependent models for such polyphones can get quite large and it reaches its limit in regards to trainability. Furthermore context dependent models generalize less the wider the context neighborhood. To compromise between generalization and model accuracy clustering algorithms merge context dependent models together. Clustering algorithms are characterized by their subunits, the used distance measure and by a bottom-up or top-down search. The clustering procedure can be based on a decision tree and only clusters states of the same phoneme in different contexts. Initially the decision tree is context independent thus collecting all contexts of a subunit in its leaves. With the help of a set of questions these leaves are then split up resulting in a clustering of the contexts. All different contexts in one leave are then modeled by the same distribution and the same codebook. Usually the questions that split the branches of the decision tree are linguistically motivated and done by an expert. A question is selected if the entropy loss is maximized by a split of a node into two child nodes when applying the question. Figure 2.4 shows an example of such a splitting for quintphone units. To begin the clustering procedure a node collects all polyphones belonging to a phoneme,

thus a phoneme in its various contexts. From the previously defined question set the question maximizing entropy loss by performing a split is selected. The polyphones are then sorted into the successor nodes depending on the answer to the posed question (In the example all quintphones with a vowel as immediate left context were put into the *yes*-successor and the others into the *no* successor node. The next selected question regards the context at a neighborhood $width = 2$ from the center phoneme.
Since the entropy criterion is not useful to stop the clustering procedure automatically the number of leaves are limited by an upper bound and in addition to that a new node is only allowed to be created if enough training material can be accumulated. During the decoding process one traverses the built decision tree and uses the acoustic model of the yes-leaf. This way one always gets an acoustic model even if a context has not be seen during training.

Context dependent models have brought a big improvement in speech recognition. A reasonable context neighborhood width depends on the language and available training material.

2.2.3 Language Modeling

Language Models are used to calculate the a priori probability $P(W)$ for a given word sequence $W = w_1 w_2 w_3 \dots w_N$. The probability is independent of the acoustic signal $\mathbf{X}$. There is a linguistic and a statistical approach to calculate $P(W)$.
The linguistic technique tries to understand the syntactical and semantical structure of a language and derive the probabilities of word sequences using this knowledge .
In the statistical way huge text corpora are evaluated and word transitions are counted. It is one of the most common method used in today's state-of-the-art recognizers. The probability of a word sequence can be written as:

$$P(W) \quad = \quad \prod_{i=1}^{N} P(w_i | w_1 w_2 \dots w_{i-1}) \tag{2.6}$$

The problem lies in the tremendous amount of possible word combinations therefore a simplification is made and only the previous N words are considered to have an

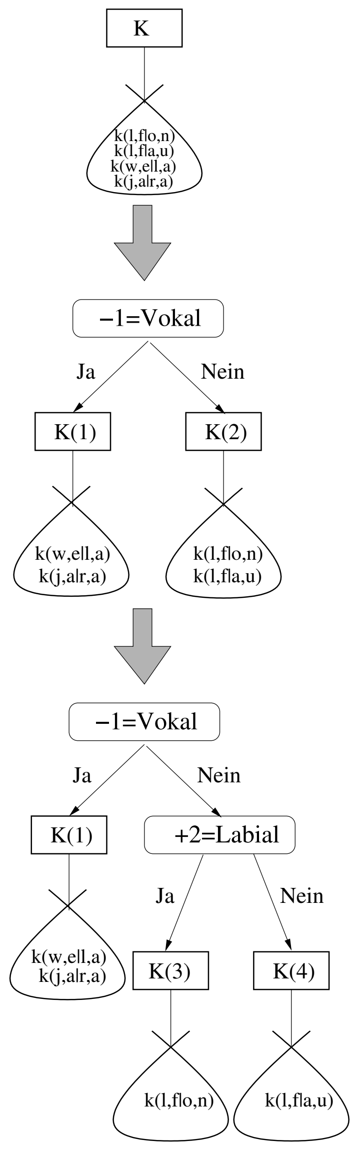

Figure 2.4: Evolution of a decision tree for quintphones (Picture taken from [14])

influence. These language models are thus called *N-gram models.* A powerful and still manageable alternative is to set $N = 3$. These *Trigram models* calculate the probabilities:

$$P(W) = \prod_{i=1}^{N} P(w_i|w_{i-2}w_{i-1}) \tag{2.7}$$

Despite the limited memory of *trigram* data deficiency can complicate accurate estimation of the probabilities. Furthermore *trigram models* might not be suited for all languages. For example in German the *trigram models* memory is too short to be able to group the rightful word forms together if parts of it enclose a subset of a sentence.

Combining Acoustic and Language Model

During the decoding process the emission probabilities of the HMMs and the word transition probabilities of the language model are combined to decide on the most likely word sequence. In reality the mean values and variances of the probabilities of the two models differ so badly that a correction is required. Otherwise the results would be dominated by one term only. This adjustment is done by a parameter z that weights the language model relative to the acoustic model. A second parameter the *word transition penalty* p normalizes the different lengths $|W|$ of the viewed word sequences W.

$$\begin{aligned} P(W|(X)) &= \frac{P(\mathbf{X}|W) \cdot P(W)}{P(\mathbf{X})} \\ &\downarrow \qquad (2.8) \\ &= \frac{P(\mathbf{X}|W) \cdot P(W)^z \cdot p^{|W|}}{P(\mathbf{X})} \qquad (2.9) \end{aligned}$$

This Project: A grapheme is split into a begin-state, a middle-state and an end-state which are marked correspondingly with -b, -m and -e. The grapheme A for example is split into its subgraphemes A-b, A-m and A-e. A grapheme is thus modeled by a $3-$state *left-right-HMM* (see Figure 2.5). Only transitions to the right neighboring state or to the same state are allowed. Each grapheme

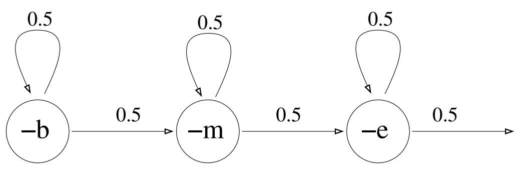

Figure 2.5: A 3-state HMM to model graphemes (Picture taken from [14])

has a minimum length of 3. Each state corresponds to a frame of $10ms$ thus the minimal grapheme-duration is $30ms$.

The silence model is a *HMM* with $4 - states$ like shown in Figure 2.6. A minimum duration of $40ms$ is required for silences. In comparison to graphemes silences are not divided in to beginning, middle and end.

The transition probabilities of the *HMMs* are uniformly set to 1.0 and will not be further trained. The emission probabilities are modeled through *Gaussian mixture distributions*. A mixture distribution consists of 32 Gaussian distributions which have a dimension of 32 after the above processing steps. Since the *LDA*-transformation decorrelates the dimensions of the feature vectors averaged over all classes, there is no need to model the complete covariance matrix. It suffices to only calculate the diagonal so that the multivariate Gaussian distributions decompose into their univariante components.

For the context independent models each subgrapheme is modeled by 32 distributions. In the context dependent case each subpolygrapheme is modeled by 32 distributions as well . The number of subpolygraphemes is limited to 3000 during the clustering procedure. Per language $32 \cdot 3000 = 96000$ distributions are approximated.

The generation of the questions for the context decision tree is described in more detail in Section 5.6.

The language models used for this work were calculated during the GlobalPhone project and the Semesterarbeit of Sebastian Stüker [14, 5]

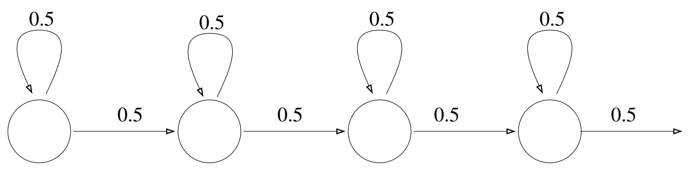

Figure 2.6: A 3-state HMM to model silence (Picture taken from [14])

2.3 Evaluation of Recognition Accuracy

A spoken utterance is decoded and the resulting word sequence is stored as a so called *hypothesis*. To evaluate the hypothesis an appropriate error measurement is necessary which allows comparison between different recognition systems.

2.3.1 Decoding Continuous Speech

To find the word sequence $W*$ that maximizes $P(\mathbf{X}|W) \cdot P(W)$ the *Viterbi algorithm* is used. Since it is too time consuming to calculate the emission probabilities for all possible word sequences one has to think of a different approach. If the segmentation of the sequence into words is know the Viterbi algorithms could just calculate the best word for each section. The *One-Stage-Dynamic-Time-Warping Algorithm* is an improved Viterbi algorithm that solves the segmentation and the single word decoding problem simultaneously. [14] explains the decoding procedure in more detail.

By back tracking the best search path a sequence of words is generated which are emitted as hypotheses by the recognizer. If the N-best search path are analyzed a N-best list of hypotheses is created and stored as word-hypotheses-graphs so called *lattices*. With additional information new hypotheses can be generated from these lattices.

2.3.2 Measuring Error Rates

To calculate the error during the recognition process one compares reference sentence with the recognized hypothesis sentence. One differs between *substitution* (word of hypothesis and reference differ), *deletion* (the word is left out in the hypothesis) and *insertions* (there is an extra word in the hypothesis) errors. The *word error rate WE* is a combination of them all:

$$WE = 100 \cdot \frac{N_{sub} + N_{ins} + N_{del}}{N} \qquad (2.10)$$

$$WA = 100 - WE \qquad (2.11)$$

Where N is the total number of words in the reference sentence and WA stands for word accuracy. The *WE* is suitable to compare recognition performance within a language but not always reasonable to compare performance between different languages since it is based on words. (Different languages might have different notions of the concept "word").

Chapter 3

Multilingual Speech Recognition

To comprehend the objective of this project it is important to understand what we mean when we talk about *multilingual speech recognition.* This chapter likes to give a brief introduction and is based on the Ph.D.-thesis of Tanja Schultz [14]. The development of recognizers for new languages used to be connected with a lot of work. Multilingual speech recognition systems try to reduce the work without loosing performance. The idea is to combine information from different languages with existing recognizers and to port it to new languages. The idea is that with the additional information a new recognizer can be build faster and needs fewer trainings data. The goal to reach is a language universality that allows a complete separation of the recognizer with all its components and its acquired knowledge and the actual language that one likes to recognize. So ideally one would like to use the same recognizer for different languages and even for new unknown ones without any readjustments.

On the one hand we have a *software sharing.* That means the trainings- and evaluation modules of a speech recognizer are independent of language specific information which will be stored separately. The software sharing works without any problems. The real challenge lies in the *data sharing.* How can we share acoustic models, pronunciation dictionaries and language models among the different languages or how can we create them without specific knowledge about the language. Sharing of training data is also an attractive offer since we would like to need as few training data as

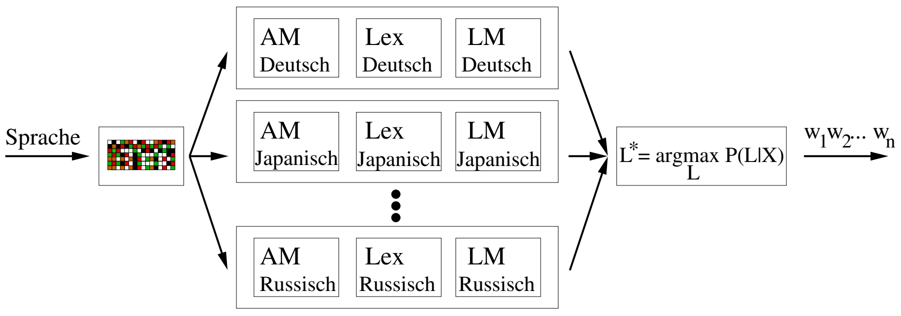

Figure 3.1: A multilingual recognizer formed through the interconnection of monolingual ones (Picture taken from [14])

possible while estimating the parameters as robustly as possible. The development of a recognizer requires collecting and preparing the speech data, defining the phoneme inventory, creation of a pronunciation dictionary, editing large text corpora for the calculation of the language models, training of the acoustic models as well as tuning of the system and its evaluation. A universal phoneme inventory allows the initialization of a recognizer in a new language as well as the adaptation of existing systems to new languages. The sharing of training's material reduces its total needed amount to estimate the parameters robustly. Another important aspect is that the financial and time costs of developing a new recognizer are reduced. There is only a limited number of human producible sounds due to our anatomy. The hope of multilingual speech researcher therefore is that unseen sounds get more rare with every new language and might be at one point fully covered.

3.1 Definition of a Multilingual Speech Recognizer

Multilinguality can be understood in two different ways. First the term refers to a system that is able to process several different languages at one time. This multilingual recognizer can be created through an interconnection of monolingual components. Each component is responsible for only one language and is trained with its own language specific data (see Figure 3.1 for an example). If the input language is not know a

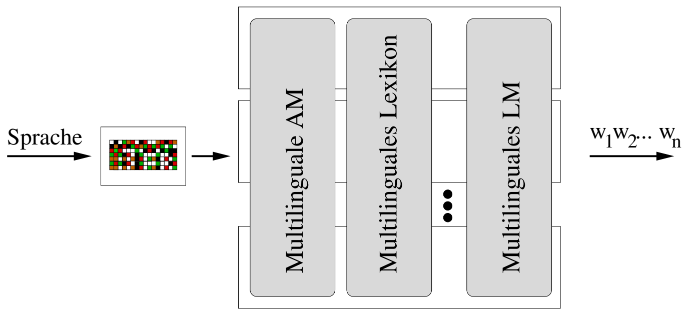

Figure 3.2: A multilingual recognizer (Picture taken from [14])

LID component (*Language Identification*) is required. It can either be consulted first to tell the multilingual recognizer which of its monolingual components to activate or after the recognition process to select the language which optimizes the output. In these *multilingual* systems *software sharing* is a common technique. For example the recognizer only needs to extract feature vectors once and the decoder is implemented once and during runtime the language specific data is linked to it.

[14] refers to *multilingual speech recognition* in a stricter sense. A multilingual system has to share at least either the acoustic model, the pronunciation dictionary or the language model. The jointly used component is trained with data from all the different languages. Therefore it is referred to as *language independent* or *multilingual.* See Figure 3.2 to see a block diagram of such a multilingual speech recognizer.

Data Sharing allows to reduce system parameters by sharing knowledge among languages. Since this enables less overhead and a slimmer structure it is an interesting approach for memory limited applications found in palm tops or cellular phones.

Multilingual Acoustic Model: If the acoustic models are shared among different languages the combined data is usable for training. The more speech data is available to train a certain state the robuster its parameters can be trained thus improving recognition rate. Ideally at one point the whole phoneme inventory of human producible sounds should be covered. Such generalizing models would enable an easy and fast portability to any new language.

Multilingual Language Model: Switching from one language to another in the middle of an utterance requires a multilingual language model. This is for example important to embed phrases that are in another language. *Language Identifying* components ask for a multilingual language model as well.

Multilingual Pronunciation Dictionary: It can stand for a pronunciation dictionary created through interleaving the monolingual dictionaries.

A multilingual speech recognition system like [14] defines it, realizes *data sharing* and *software sharing* together.

3.2 GlobalPhone Project

For her Ph.D. thesis in Multilingual speech recognition Tanja Schultz [14] needed a database in many different languages that fulfilled the following requirements:

- The most relevant languages in the sense of economic importance and number of speakers are covered
- Includes all or as many as possible human producible sounds
- The selected speakers represent their mother tongue language balanced in regard to gender, age and education
- The transcribed data material is large enough to allow robust training of acoustic models
- There are texts with millions of words at hand to extract an accurate language model
- The acoustic quality of the material is uniform to allow language specific dependencies to be identifiable.
- The same speech style was used among the different languages e.g. spontaneous, colloquial speech, monologue, dialog or read

- The data of all different languages are similar in respect to their domain and vocabulary, e.g. in their semantics.

At that time no according database existed. Therefore the *GlobalPhone* project was called to life to create a multilingual database sufficing the above conditions. Diversity was the key issue in selecting the languages. They should offer a certain variance in regard to their phonetic, phonological, prosodic and grammatical characteristics. Foreign students at the university of Karlsruhe were asked to record samples in their home country. 15 different languages were selected with the above criterion and the limitation of having such a native speaker at hand.

12 of the world's main languages are included in this selection. It covers 85% of all spoken language families. See at Table 3.2.

To keep costs and time low in producing the database the collected material is read speech since it allows to save the most costly task; the transcription process of spoken sentences into text. Newspaper articles from local newspapers were selected since they allow on one hand to have a similar domain (daily local events, world news and economic updates) over all languages and on the other hand it was easy to access them through Internet (allowing credible estimation of the language models). The data collection and training has been done in a uniform process to ensure comparability of the resulting recognition process. With the database available monolingual recognizers were constructed for various experiments and combined to form a multilingual recognizer. The results were used to analyze the possibilities to create recognizers for new languages using these resources and the knowledge of already existing systems. Results and Experiments are described in detail in [14, 17]

This project uses the *GlobalPhone* database and language model for English, German Swedish and Spanish. Certain scripts and models are copied from the *GlobalPhone* project and then changed to meet the needs of this work.

Language	Country
Arabian	Tunisia
French	France
Mandarin-Chinese	Mainland-China
Shanghai-Chinese	Mainland-China
German	Germany
Japanese	Japan
Korean	South Korea
Croatian	Croatia and Bosnia
Portuguese	Brazil
Russian	Belarus
Swedish	Sweden
Spanish	Costa Rica
Tamil	India
Turkish	Turkey
Czech	Czech Republic

Table 3.1: Languages and the country where they were collected for the GlobalPhone project

Chapter 4

Baseline Recognizers

To evaluate the performance of the grapheme based *large vocabulary continuous speech recognizers* they were compared to the baseline recognizers used and described by [14, 5]. The baseline recognizers were trained by Schultz and Stüker in the course of the GlobalPhone project [14] and a Studienarbeit [5]. The recognizers trained during this project use the database, the language model and the same preprocessing algorithms (see Section 2.1) and techniques as the baselines. The only difference lies in the generation of the dictionaries and the questions used to produce context dependent models. The baselines dictionaries are phoneme based and were created with a combination rule-based algorithms and expertise knowledge. The global phoneme set in the GlobalPhone project is defined based on the IPA chart [18] and sounds from different languages that share the same IPA symbol, share the same unit in our phoneme set. Table 4.2 shows the size of the dictionaries. Every word in our dictionaries is tagged with the language it belongs to, so that it can be distinguished from words in other languages that might share the same orthography but have a different pronunciation.

	#utterances (hours)		
Language	EN	DE	SP
Training	7,137 (15.0)	9,259 (16.9)	5,426 (17.6)
Development	144 (0.4)	199 (0.4)	250 (0.7)
Evaluation	152 (0.4)	250 (0.4)	250 (0.7)

Table 4.1: Overview of the data used from the GlobalPhone corpus

	#words		
Language	EN	DE	SP
Size Dict	9,461	24,000	18,510

Table 4.2: Size of the GlobalPhone dictionaries

4.0.1 The English Recognizer

The Database

English data was not collected by the GlobalPhone project but instead the Wall Street Journal Corpus provided the appropriate data. Table 4.1 and 4.2 give an overview of the amount of data. There were 103 speakers contributing their voices.

The Language Model

Additional material from [16] was used to appropriately estimate the trigram probabilities.

Evaluation

The English baseline recognizer showed a WA (word accuracy) of 87.3% for the development and a WA of 84.4% for the evaluation data. See Table 4.4.

4.0.2 The German Recognizer

The Database

The newspaper articles for the German recognizer are taken from the FAZ *www.faz.de* and Süddeutsche *www.sueddeutsche.de* and read by 77 speakers. See Table 4.1 and 4.2.

The Language Model

Additional material from [16] was used to appropriately estimate the trigram probabilities.

Evaluation

The German baseline offers a WA of 82.3% (Developement) and a WA of 87.0% for the evaluation data. See Table 4.4.

4.0.3 The Spanish Recognizer

The Database

The newspaper La Nacion *www.nacion.co.cr* provided the text data. See Table 4.1 and 4.2 for the amount of recorded data from 100 different speakers.

The Language Model

Additional speech data was obtained from *The European Language Resources Association* which collected data from the newspaper Expansion and from *The Linguistic Data Consortium* which did the same for the Newswire newspaper.

Evaluation

The Spanish basline offers a WA of 75.5% for the development set and a WA of 83.7for the evaluation set. See Table 4.4.

4.0.4 The Swedish Recognizer

The Database

The newspaper articles were collected from the Göteborgs-Posten Internet site *www.gp.se*. The database is described in regards to amount of hours of recordings, number of speakers (N_{spk}), number of utterances (N_{utt}), number of words (N_w) in Table 4.3.

	Rec Length	N_{spk}	N_{utt}	N_w
Training	17.4h	79	9406	144700
Cross	2.1h	9	1207	18455
Evaluation	2.2h	10	1203	17870
Total	21.7h	98	11816	181025

Table 4.3: The Swedish Database

The Language Model

Additional texts to create the language model were all downloaded from the above mentioned web site. Refer to 6.16 for the characteristics of the Swedish language model.

Evaluation

The Swedish recognizer showed a word accuracy rate of 33.2% for the context independent case and an accuracy of 47.3% for the evaluation and 51.4% for the development set in case of a context dependent recognizer. See Table 4.4.

Word Accuracy in %				
	English	Spanish	German	Swedish
Development	87.3	75.5	82.3	51.4
Evaluation	84.4	83.7	87.0	47.3

Table 4.4: Word accuracy in % of the basline systems tested on the development and evaluation set

Chapter 5

Generation of Pronunciation Dictionaries

Pronunciation dictionaries are built by splitting a word into its graphemes which are used as subunits. For example the word ZURICH would be simply followed by its fragmentation into subunits (graphemes) Z U R I C H. This is a very easy and fast approach to produce pronunciation dictionaries. The questions arise how well graphemes are suited as subunits, to what extend are they inferior to phonemes or do they perform combarably well, how do we cluster graphemes into polygraphemes, how do we generate the questions to built up the decision tree, can we build a multilingual recognizer based on graphemes and share the acoustic models among languages? It is also interesting to see for which languages this seems a reasonable way to produce dictionaries and which languages offer worse performance. Is a close grapheme to phoneme relationship helpful? References to the Tcl-scripts in this chapter might be better understood if first reading Chapter C where the general training process in *JANUS* is described.

5.1 Selected Languages

To evaluate the performance of a grapheme based system language, English was selected as a language with a fairly poor grapheme to phoneme relation. On the other side Spanish was selected because it offers a very close grapheme to phoneme relation. German lies somewhat in between those two extremes. The recognizers in these languages built during the Global Phone projects are used as a baseline. To evaluate the performance further and to test to what extend multilingual ideas are reasonable for grapheme based recognizers Swedish is chosen as a new unknown language. A Multilingual Recognizer is created using Spanish, English and German data and it is investigated how far knowledge from the multilingual recognizer can help improve the Swedish recognizer.

5.2 Dictionary Creation

A Tcl-script reads in each transcription from the database, cuts out the single words and stores them in a word list. In a further step those words are split into their graphemes and both the word and its grapheme-fragmentation are stored in dictionary files. Word beginning and ending are marked with a word tag WB. Each grapheme is also tagged with a language ID. Special characters are extracted and romanized. Ä, ö or ü in German for example need special attention. Table D.1 shows an extract from the English grapheme based dictionary.

The German and Spanish Dictionaries included numbers. In case of the German dictionary the number entries were corrected with a few script commands. Since for example 1995 can either be a year or simply an enumerator both versions were included in the dictionary (*Neunzehnhundertfünfundneunzig* as well as *Tausendneunhundertundfünfundneunzig*). German numbers offer some variability, for example 101 can be called *Hunderteins*, *Einhunderteins*, *Hundertundeins* or *Einundertundeins*. Therefore all those possibilities were added as ”pronunciation variants” in the dictionary.

The Spanish number system is straight forward and a few rules are sufficient to generate it. Two scripts (*zahl2word_SP.tcl* and *zahlpron2J3.tcl*), previously written for phoneme based dictionaries), were adapted to deal with graphemes and produced then the wanted mapping from the number to its word form representation.

5.3 Initial Labels

To start training the acoustic models and its parameters, we need a mapping between the acoustic data and the acoustic models. This mapping defines the frame-wise time alignment of the data to the according HMM state. For example in Janus labels can be thought of as something like *frame with index i belongs to HMM state with index j.* The labels are usually obtained by the Viterbi algorithm, meaning labels actually are nothing else but the stored Viterbi path for a given utterance.
Since there were no previous labels at hand for graphemes the utterances were linearly segment into its subunits the sub-graphemes (grapheme beginning, middle and end). For each utterance, which usually consist of a sentence out of a newspaper article (see 3.2), the number of graphemes $|G|$ was calculated. The total number of frames $TotFrames$ used for this utterance is then divided by the number of subunits resulting in the amount of frames per unit. As explained at the end of Section 2.2.3 a HMM for graphemes consists of three sub graphemes. A silence is modeled only by one state. So we get the following formula for the amount of frames per subunit:

$$nbrFrames = \frac{TotFrames}{3 \cdot |G| + |Silences|} \tag{5.1}$$

If the variable $nbrFrames$ is smaller than 1 for an utterance then this specific utterance was skipped for this first training cycle. A rewritten *sample.tcl* script goes through all utterances in the training data, calculates the $nbrFrames$ and extracts continuously that amount of samples and checks to which model (subgrapheme e.g. A-b, A-m, A-e or silence SIL) the data gets assigned to. After we know which data material belongs to which models we initialize the codebooks (one for each subgrapheme) with the K-means-algorithm (*kmeans.tcl*). Now everything is available to write first labels which of course are still far from being accurate (*labels.tcl*).

5.4 Context Independent Systems

After creating the initial labels the acoustic models were trained. The following training iterations are then done along the labels assuming the alignment remains unchanged.

Once in a while new labels are written under the assumption that the recognizer has improved enough to produce better alignment for the forthcoming development steps. Training is done by loading paths from the label files and then accumulating training information in so called accumulators by the Viterbi Algorithm (alternatively training could be done by the Forward-Backward algorithm). The accumulators are stored explicitly on disc because they can become very large. A training cycle looks as follows:

Calculate the LDA matrix This step is not necessary to build a recognizer, but it helps to improve the recognition accuracy. Multiplying every feature vector $\mathbf{X}$ with the LDA matrix maximizes the ratio between the diversity of all data and the average diversity of the data belonging to the same class. Finding the LDA will thus make data belonging to a class move closer together and data of different classes move a bit further apart. To calculate the LDA it needs to be defined which acoustic models belong to which LDA-class. In this Diplomarbeit one class for each Gaussian codebook was used.

Extract sample vectors for each codebook It is virtually impossible to keep all that is needed for the K-Means algorithm in memory in large enough buffers since it can easily sum up to about half a giga byte of data. Therefore the example vectors used for the K-Means algorithm below are stored on disc.

Initialize the codebooks E.g. find new reference vectors. Cluster a set of example vectors into a few classes iteratively such that the distortion measure is being minimized. The K-Means algorithm assigns each example vector to the class with the closest mean vector and updates the mean vector by replacing it with the average of all vectors.

Training Train multiple iterations along labels and update the parameters.

The context independent systems are trained with cycles of 6 training iterations followed by writing new labels. These cycles are repeated six times before changing to context dependent modeling.

5.5 Context Dependent Systems

In this Diplomarbeit we use the term polygraphemes to refer to graphemes modeled in an arbitrary wide context. Before being able to start training the context-dependent system all polygraphemes need to be collected that occur in the database in a so called *ptrees*. The ptrees are simply containers that hold all seen contexts of a grapheme. A parameter P defines the maximum context width that is considered when collecting the polygraphemes. Setting $P = 1$ means that only contexts up to one grapheme to the right and to the left are considered, thus giving "trigraphemes". Since our subunits in this Diplomarbeit are on the level of subgraphemes (beginning, middle and end) there is a decision tree for each HMM state. For each subsegment of a polygrapheme seen in the training sentence we start at the corresponding root node of the decision tree (root-b, root-m or root-e) traverse it by answering the questions of the tree (is the central grapheme an A, B ... Z?) and when reaching a leave node attach the polygrapheme to the ptree.
After processing all training data the decision tree's attached ptrees are filled. See Figure 5.1 to get an impression.

All polygraphemes belonging to a leaf node after the clustering process share the same codebook, but specific contexts as collected in the ptree are modeled by a separate distribution. This system is then trained with 2 iterations and the calculated parameters are then used for the next step in which the polygraphemes are clustered together using an entropy criteria and a set of questions. Usually the questions are generated by a person with expertise knowledge in that language. But since that would destroy this project's goal to automatically generate pronunciation dictionaries to eliminate the need of language experts a variety of ways to find such questions are examined. Let's assume for now that we have such a set of questions ready at hand. Starting at the leaf in the decision tree that holds the ptrees the benefit of every allowed question used to split the node is computed (see Subsection 2.2.2). There is a parameter Q which specifies up to which width a question can be asked regarding the context of a grapheme (a question with Q=2 for example can ask if the grapheme 2 to the right is a vowel).

This step is done until we reach a maximum amount of models (usually set to 3000 in this Diplomarbeit except where specifically noted otherwise) or if the split is not good enough, meaning there is a minimal count of training frames required per node. See Figure 5.2 to see how node "D" with its ptree is split up into two successor nodes by the question if the grapheme after the central grapheme D is a B or not. The buckets symbolize the ptrees and the numbers are indexes for the used models.
The clustering procedure results in a decision tree where all polygraphemes in a leaf are modeled by the same codebook and same distribution (the ptrees are not needed anymore after the clustering step). Thus we end up with up to 3000 models in our case. The resulting decision tree and codebooks are then saved for the generation of a context dependent recognizer.

The training cycles are similar to the ones in the context independent system. But instead of a simple training a *VLTN* is also computed.

5.6 Question Generation

Since the set of possible polygraphemes for a standard language in a context dependent speech recognizer is very large, the estimation process often runs into data-insufficiency problems. To counter these, it becomes necessary to group the polygraphemes into a statistically estimable number of clusters. It is described above in 5.5 how this is done in this Diplomarbeit. Reducing the amount of free parameters will then allow to estimate the remaining parameters more robustly. Recognizing speech is basically a pattern classification procedure and therefore it is important that the clusters are maximally separated. This is a partitioning problem and to identify the maximally separated clusters all groupings would have to be evaluated.
The question set used for the clustering procedure should be a good guess to what an optimal clustering of graphemes would look like had it been possible to exhaustively search through all possible clusters. [20] argue that the training \ recognition process is strictly a maximum likelihood statistical estimation process and therefore it is desirable to generate the questions using the same statistical criterion.

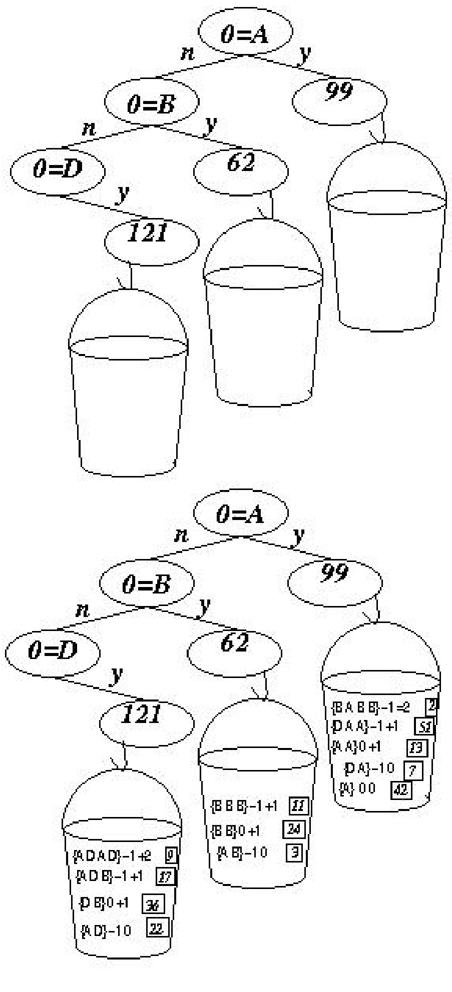

Figure 5.1: A decision tree and ptrees for a context dependent recognizer (taken from [19])

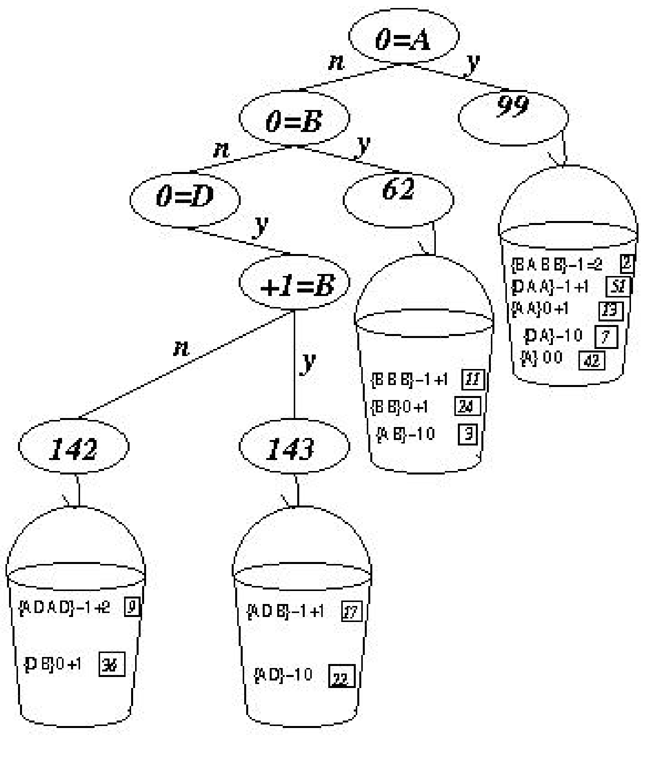

Figure 5.2: Clustering of the decision tree and its ptrees (taken from [19])

[20, 21] proposed algorithms to automatically generate question sets. They both used a log-Likelihood distance measure. [21] use a bottom up clustering while [20] used a hybrid clustering method combining a bottom up and top-down technique. They showed that it is possible to derive a set of automatically produced questions that give equivalent results to linguistically motivated questions generated by an expert. Both approaches address the issue of having contextual questions, meaning the left and right context questions are not necessarily the same. Since it was not clear how context sensitive graphemes behave this Diplomarbeit generated a question set regardless of treating the left and right context differently. The following Subsections are inspired by these two papers [20, 21].

In the next subsection two different distance measurements are introduced before explaining the implemented algorithms in this Diplomarbeit.

5.6.1 Distance Measurements

See [16] for a more detailed explanation although this Diplomarbeit uses a slightly different entropy formula. The *entropy distance*'s goal is to optimize the information in the parameter space of a recognizer while the second distance *likelihood distance* tries to maximize the probability of the training data.

Entropy Distance

To calculate the entropy distance between two sets of models for subgraphemes (grapheme-beginning, -middle and -end) a context independent system is trained where all the acoustic models share the same codebook. Let K_1 and K_2 be two sets of distribution models defined by the mixture weights of the Gaussian distributions $\gamma_{1,i}$ and $\gamma_{2,i}$ (they all share the same Gaussian, but only differ in their mixture weights):

$$\begin{aligned} K_1 &= \gamma_{1,1}, \dots, \gamma_{1,N} \\ K_2 &= \gamma_{2,1}, \dots, \gamma_{2,M} \qquad (5.2) \\ K_1 \cap K_2 &= \varnothing \qquad (5.3) \\ \gamma_{1,1} &= (\gamma_{1,1}(1), \dots, \gamma_{1,1}(k), \dots, \gamma_{1,1}(n)) \qquad (5.4) \end{aligned}$$

There are n Gaussians to form the Gaussian mixture distribution. K_1 has N and K_2 has M models in the set. The a priori probability for a set of models is calculated by summing the amount of how many samples of the training data were classified to a certain model. This is equivalent to the sum of the numbers of samples ($p_{1,i}$ and $p_{2,i}$) that got stored in the different polygrapheme models collected by the ptrees in 5.5 (for a system that shares the codebook for all graphemes of course).

$$\begin{aligned} p(K_1) &= \sum_{i=1}^{N} p_{1,i} \\ p(K_2) &= \sum_{i=1}^{M} p_{2,i} \qquad (5.5) \end{aligned}$$

Now let K_{12} be the union of K_1 and K_2 and :

$$\begin{aligned} K_{12} &= K_1 \cup K_2 \qquad (5.6) \\ \gamma_1(k) &= \frac{1}{p(K_1)} \cdot \sum_{i=1}^{N} p_{1,i} \cdot \gamma_{1,i}(k) \\ \gamma_2(k) &= \frac{1}{p(K_2)} \cdot \sum_{i=1}^{M} p_{2,i} \cdot \gamma_{2,i}(k) \\ \gamma_{12}(k) &= \frac{p(K_1) \cdot \gamma_1(k) + p(K_2) \cdot \gamma_2(k)}{p(K_1) + p(K_2)} \qquad (5.7) \end{aligned}$$

The entropy distance between the two sets of models is now calculated as follows:

$$D = (p(K_1) + p(K_1)) \cdot H_{12} - p(K_1) \cdot H_1 - p(K_2) \cdot H_2 \qquad (5.8)$$

Where H_i is the entropy of the distribution γ_i:

$$H_i = -\sum_{k=1}^{n} \gamma_i(k) \cdot \log(\gamma_i(k)) \qquad (5.9)$$

The distance between graphemes is calculated as follows:

$$D_{TOT} \quad = \quad D_b + D_m + D_e \tag{5.10}$$

For each subgrapheme class (b, m, e) the distance is calculated individually and the sum of the subgrapheme class distances forms the distance between the grapheme models.

Likelihood Distance

Again the acoustic models share the same codebook. Let $P = \{\mathbf{x_1}, \ldots, \mathbf{x_T}\}$ be a set of training vectors assigned to the codebook and $\mathcal{N}(\mathbf{x_i}, \mu_k, \Sigma_k)$ is the value of the k-th Gaussian distribution at position $\mathbf{x_i}$. γ_P marks the mixture weights estimated from all data $\in P$.
The log-likelihood, e.g. the log of the probability to observe $\mathbf{x_1}, \ldots, \mathbf{x_T}$ given the model defined by γ and the Gaussians $\mathcal{N}$ is defined as follows:

$$L_P \quad = \quad log(\prod_{i \in P} \sum_k \gamma_{P,k} \cdot \mathcal{N}(\mathbf{x_i}, \mu_k, \Sigma_k)) \tag{5.11}$$

The log likelihood distance between two clusters Q and R is defined by:

$$D \quad = \quad L_{Q+R} - L_Q - L_R \tag{5.12}$$

where L_{Q+R} is the likelihood of the set formed by merging cluster R and Q. The problem with the likelihood distance is that the parameters for the joint clusters would have to be estimated by a training step where the models in a cluster share the same distribution. This would result in a more time consuming procedure requiring a training iteration for every investigated combination of models.

5.6.2 Phoneme-Grapheme Question Generation

To get a feeling of how good those grapheme based recognizers can get compared to the phoneme based ones the original question sets from the baseline recognizers were taken. With a language dependent phoneme to grapheme mapping (described in the Tables B.1, B.2 and B.3 found in B.2.1) that was created manually and the script

PHONES	A_EN B_EN C_EN D_EN E_EN F_EN G_EN H_EN I_EN J_EN K_EN L_EN M_EN N_EN O_EN P_EN Q_EN R_EN S_EN T_EN U_EN V_EN W_EN X_EN Y_EN Z_EN @ SIL +hGH_EN +QK_EN
NOISES	+hGH_EN +QK_EN
SILENCES	SIL
CONSONANT	B_EN C_EN D_EN F_EN G_EN H_EN K_EN L_EN M_EN N_EN P_EN Q_EN R_EN S_EN T_EN V_EN W_EN X_EN Y_EN Z_EN
OBSTRUENT	B_EN C_EN D_EN F_EN G_EN J_EN K_EN N_EN P_EN Q_EN S_EN T_EN U_EN V_EN W_EN X_EN Z_EN
SONORANT	A_EN D_EN E_EN H_EN J_EN L_EN M_EN N_EN O_EN R_EN W_EN Y_EN
SYLLABIC	A_EN E_EN H_EN I_EN O_EN U_EN Y_EN
VOWEL	A_EN E_EN I_EN O_EN U_EN Y_EN H_EN
⋮	

Table 5.1: Example of a phoneme-grapheme question set for English

phonemeToGrapheme.tcl which substituted each phoneme belonging to a question by the according graphemes a new grapheme based set of questions was formed. An example of such a question set (called the phoneme-grapheme question set in this project) can be seen in Table 5.1. Explanations on why the English grapheme "H" or the German "R" are regarded as vowels can be found by looking at the mappings in the Appendix B.2.1.

5.6.3 Singleton

A very simple idea to generate questions is to ask what kind of grapheme the left or right context is. Each question consists of one grapheme, thus they are called singeltons in this Diplomarbeit. The questions look like in Table 5.2.

PHONES	A_EN B_EN C_EN D_EN E_EN F_EN G_EN H_EN I_EN J_EN K_EN L_EN M_EN N_EN O_EN P_EN Q_EN R_EN S_EN T_EN U_EN V_EN W_EN X_EN Y_EN Z_EN @ SIL +hGH_EN +QK_EN
NOISES	+hGH_EN +QK_EN
SILENCES	SIL
QUES_1	A_EN
QUES_2	B_EN
QUES_3	C_EN
⋮	
QUES_26	Z_EN

Table 5.2: Example of a singelton question set for English

5.6.4 Bottom-Up Entropy

To automatically derive a proper set of questions a context independent system sharing one codebook among all graphemes is trained for one cycle using the labels of the previously trained context independent system 5.4. The codebook and distribution

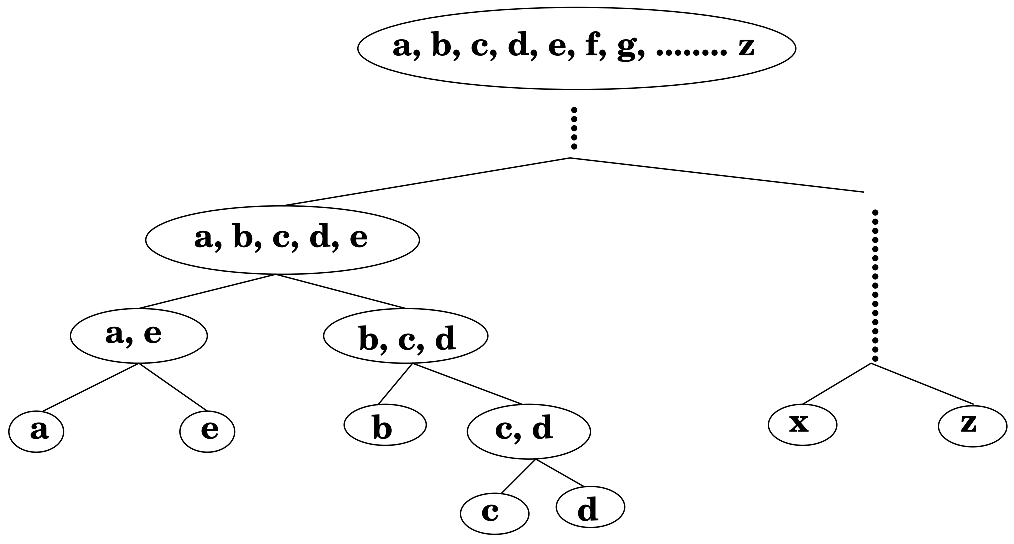

Figure 5.3: An example of how a final bottom-up clustering tree could look like

parameters are saved on disc.
The next step is to call the ptree.tcl script which will count the number of times a distribution was seen in the training data. These counts are used for the calculation of the a priori probability of a model refer to equation 5.5. Starting with a set of monographemes, the monographemes are clustered together with a bottom-up cluster algorithm using the entropy distance measure until one cluster remains. This results in a tree recording the clustering states of the set of graphemes. The intermediate clusters, the nodes in this tree, are then recorded as questions. Questions for individual graphemes and for the word boundary are added as well. This question set is then used to construct the decision tree for training and recognition of the context dependent system described in Section 5.5. The bottom-up clustering results in a tree that could look like the one in Figure 5.3.

5.6.5 Hybrid Entropy

The generation of the hybrid questions starts off like in Subsection 5.6.4. A system where all graphemes share a codebook is trained and the number of samples per ptree

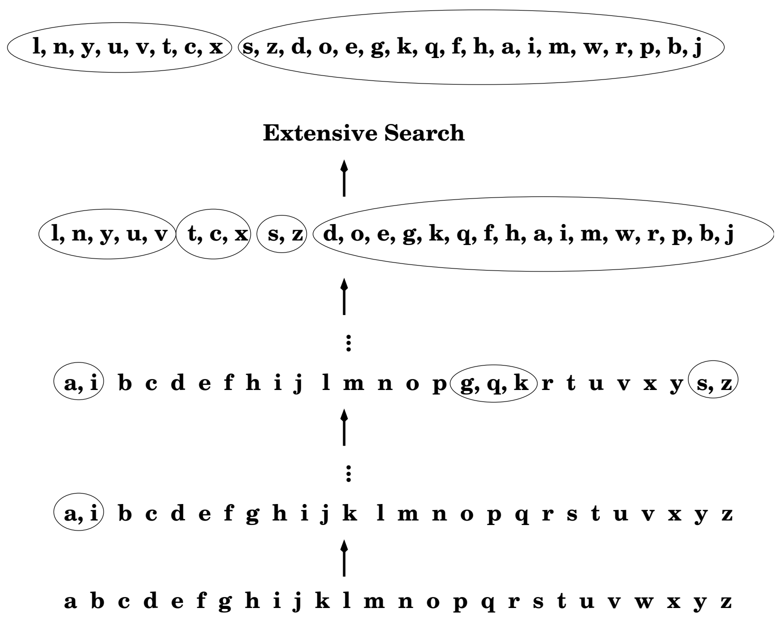
l, n, y, u, v, t, c, x
s, z, d, o, e, g, k, q, f, h, a, i, m, w, r, p, b, j
Extensive Search
l, n, y, u, v
t, c, x
s, z
d, o, e, g, k, q, f, h, a, i, m, w, r, p, b, j
a, i
b c d e f h i j l m n o p
g, q, k
r t u v x y
s, z
a, i
b c d e f g h i j k l m n o p q r s t u v x y z
a b c d e f g h i j k l m n o p q r s t u v w x y z

Figure 5.4: Sketches the hybrid clustering method: bottom-up clustering followed by an exhaustive search and recursive repetition on each subset

is evaluated. The entropy distance is used as a distance measurement. The clustering algorithm proposed by [20] is a hybrid of the top-down and bottom-up clustering technique. Starting with a set of mono graphemes the closest graphemes are clustered together in a bottom-up procedure until the number of partitions can be exhaustively evaluated. This involves the comparison of all possible groupings of clusters resulting in two maximally separated groups. The best partition is chosen as the beginning of the subsequent recursion step. See Figure 5.4 for the first step of the hybrid clustering method. On each resulting subset the bottom-up clustering step is performed again followed by an exhaustive search. If the subsets in each recursion step are stored in a top-down matter they build a tree. Again the intermediate nodes serve as questions or put in other words all final partitions resulting after the exhaustive search step are recorded as questions. Again questions concerning a single grapheme and the word boundary are added.

5.7 Three-lingual Recognizer

To study the scope of multilingual recognizers in combination with graphemes as subunits two different methods to combine the acoustic models are investigated. As already discussed in Chapter 3 multilingual recognizers could be made up of parallel monolingual once. The context modeling for this scenario is shown in the first Picture in Figure 5.5. Models are trained separately for each language and no data sharing takes place. The multilingual component in this system is the feature extraction. This is not the true multilingual idea that this project would like to examine further and is therefore only mentioned here for information purposes.

5.7.1 Language-Mixed Context Modeling: ML3-Mix

Similarly to the language-mixed context modeling of the multilingual phoneme recognizers of the GlobalPhone project [14], for each grapheme a model is provided and trained with all given data. A grapheme belonging to more than one language (e.g. a German "A", a Spanish "A" and an English "A") is then trained with data from all the languages (the model for an "A" is trained with Spanish, English and German data). This way to model context is referred to as ML3-Mix in this project. The knowledge about language affiliation of each grapheme is dismissed in this approach.
Like in the monolingual case the ML3-Mix uses a clustering based on the entropy distance to generate context dependent models. The multilingual question set is derived by combining the phoneme-grapheme questions of each underlying language. During the clustering procedure it doesn't matter to which language a context belongs. It is thus possible that a polygrapheme is modeled with contexts from different languages. The combination of all languages leads to a reduction of model parameters.
In the case of phoneme recognizers it makes sense to combine phoneme categories that then share data among languages since a phoneme as defined by the IPA [18] is pronounced in a specific way independent of the language, e.g. the IPA-notation is language independent. Therefore it can be argued that the data for a phoneme category is very similar and it makes sense to model it language independent. It the case of graphemic

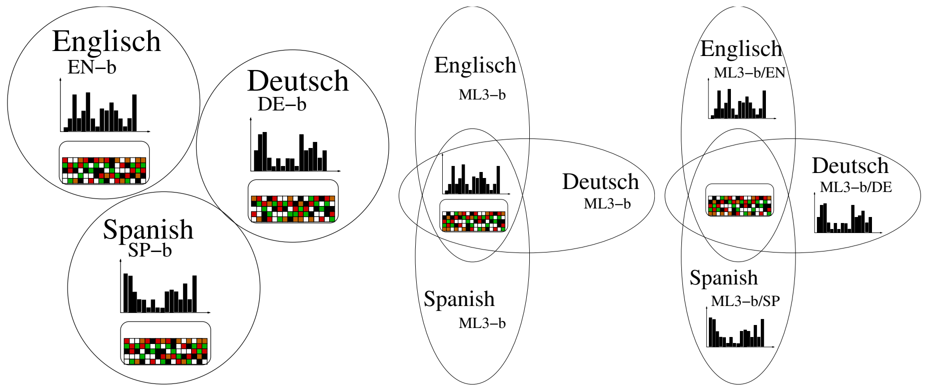

Figure 5.5: Three methods to do multilingual context modeling (ML-separate, ML3-Mix and ML3-Tag)

subunits this is a whole other issue. A Spanish ”V” for example is pronounced very differently from an English ”V” and the acoustic data is thus not necessarily expected to be similar. How much the loss due these inaccuracies is will be seen in the next Chapter 6. See the middle Picture in Figure 5.5 for language-mixed context modeling.

5.7.2 Language-Tagged Context Modeling: ML3-Tag

Each grapheme is tagged with a language ID and there is an individual distribution for each tagged grapheme. But for each grapheme (like in the case of the ML3-Mix) they all share a codebook independently of their language affiliation. The codebooks are trained with speech data across all languages, but the mixture distributions are created language specific. See the picture to the far right in Figure 5.5 for a visual example.

The phoneme-grapheme questions for the clustering procedure are derived by combining the same questions for each language to one language independent question. For example the English question *VOWELS A_EN E_EN H_EN I_EN O_EN U_EN Y_EN*, the Spanish question *VOWELS A+_SP A_SP E+_SP E_SP I+_SP I_SP O+_SP O_SP U+_SP U_SP Y_SP* and the German question *VOWELS A_DE E_DE R_DE I_DE O_DE U_DE Y_DE A_DE O_DE U_DE* are all merged into the language independent question *VOWELS A_DE E_DE I_DE O_DE U_DE Y_DE A_DE O_DE R_DE U_DE A_EN E_EN I_EN O_EN H_EN U_EN Y_EN A+_SP A_SP E+_SP E_SP I+_SP I_SP O+_SP O_SP U+_SP U_SP Y_SP*. Questions asking for the language are added. The clustering procedure then decides if questions concerning language affiliation are more important than graphemic context. Thus training data of different languages are only combined if a sufficient similarity can be stated. It can be looked at as a data driven multilingual acoustic modeling approach without giving up the advantage to save parameters.

For graphemes this way of modeling acoustics seems more promising since only those graphemes are combined into one model across languages if their acoustic representation is similar to each other.

5.8 Porting to new Languages

Porting is defined in [14] as transferring learned language knowledge of a speech recognizer to another (during training unseen) language. There are different methods to port language described in more detail in [14]. Tanja Schultz talks of *Bootstrapping* if there is a great amount of data in the target language and of *Adaptation* if there is only limited data material at hand. In a *Cross-Lingual Transfer* there is no data at hand at all. In this Diplomarbeit it is investigated if knowledge from the ML3-Mix recognizer is helpful when building a Swedish recognizer from scratch. In more detail it is tested if porting acoustic models of the ML3-Mix recognizer to initial Swedish acoustic models results in an increase of recognition accuracy.

5.8.1 Bootstrapping Swedish

Swedish training data is at hand. The problem is how to initialize the Swedish recognizer, e.g. the calculation of the initial labels. The models used for the initialization are called *seed models*. They act as a start basis. After initialization the models are trained with a considerable amount of Swedish data. Simply said the acoustic models of the

Swedish recognizer are initialized with the acoustic models of the ML3-Mix recognizer to write initial Swedish labels. Then the Swedish recognizer, its acoustic models, are trained in the usual way with Swedish data.
A mapping defines which grapheme model in Swedish is bootstrapped with which grapheme model in the ML3-Mix case. The ML3-Mix codebooks and distributions are then basically copied into the Swedish ones.
In case of graphemes as subunits the question arises if the bootstrapping process even makes sense. Because in the phoneme IPA notation the phonemes noted by the same IPA-symbol are similar to each other independent of the language, it can be argued that the acoustic data is alike and it thus makes sense to use models from the same IPA-symbol in another language to initialize it in the new target language. No similarity is guaranteed in case of graphemes, therefore it is left open until the next Chapter 6 if bootstrapping can help improve or speed up training and recognition results.

5.9 Language Model

Because the Swedish baseline system (see page 37) showed poor recognition performance an increased effort was made to collect Swedish data in order to create an improved language model. A web crawler was designed by [22] to collect Swedish HTML-pages. No special attention was given to the content of the HTML-pages, as long as they belonged to the domain .se. During text normalization only whole sentences were cut out and a minimum sentence length was required to get rid of unwanted parts.
The language model toolkit is not part of the *JANUS* distribution and was written by Klaus Ries [23].

Chapter 6

Experimental Results

To see if this project's approach is suitable to automatically produce pronunciation dictionaries, speech recognizers in English, German, Spanish and Swedish were trained using the described algorithms in the previous Chapter 5. A multilingual recognizer using the English, Spanish and German data was trained as well to see wether multilingual methods work for grapheme subunits and if they can improve or simplify the creation of a recognizer in a new language.

6.1 Grapheme-based Recognizers and Phoneme-Grapheme Questions

The context independent recognizers were trained from scratch and, after the initial sample extraction (refer to Section 5.3), trained for 6 cycles (each cycle consisting of an LDA calculation, sample extraction, Kmeans codebook initialization and 6 training iterations → Section 5.4).

Using labels from the last trained context independent system a context dependent recognizer was created using phoneme-grapheme questions in the according language. The grapheme-phoneme mapping used to transform linguistic phoneme questions into grapheme based questions is shown in the Appendix B.2.1. Different context widths were tried out during clustering: a true trigrapheme (P=1, Q=1), quintgrapheme (P=2, Q=2) and septgrapheme (P=3, Q=3) system, as well as a hybrid trigrapheme (P=2, Q=1) and quintgrapheme (P=3, Q=2) system. Hybrid meaning here in the example of a trigrapheme system (P=2, Q=1) that the collected contexts in the ptrees are quintgraphemes each modeled by an individual distribution. These quintgrapheme based models are then trained for 2 iterations. The clustering works based on these trained models, but the final clustered polygraphemes are trigraphemes. These hybrid systems allow the clustering procedure to work on finer contexts, while the output remains more generalizing.

6.1.1 English

Table 6.1 shows the results for the English context dependent phoneme-grapheme recognizers, clustered with various widths, 3000-Models and language parameters z=25 and p=2.

Context Width

What can be clearly seen is that performance decreases with increasing context width. This is due to the fact that a wider context width creates finer models, e.g. the distributions and codebooks model polygraphemes in more detail (for example a separate model for each quintgrapheme), and thus the system does not generalize well anymore. Besides there might not be enough training material available for each of these "specialized" contexts. The most promising results are reached by the English trigrapheme (one P=1, Q=1) system. Performance drops from a WA (Word Accuracy) of 80.8% to 80.5% in the fourth context dependent training cycle which can be explained by over-training.

The hybrid trigrapheme (P=2, Q=1) system is trained further for an additional three trainings cycles. It almost reaches performance of the true trigrapheme system.

3000-/ 4500- Models

For the English recognizer it was further investigated if performance of the quint-grapheme system can be increased by allowing more models during the clustering procedure (see Table 6.2). Reaching 78.0% WA after the first context dependent training cycle shows an improvement of 2.7% absolute compared to the system with only 3000 models. It has to be assumed though that the 4500-Model system generalizes worse due to more in detail modeling and the poorer performance compared to the trigrapheme system.

Language Model Parameters

Language parameter adaptation (see Section 2.2.3) results are shown in Table A.2 on page 86 for the context independent recognizer (WA-CI: 28.4%). The results are not very smooth, e.g. no clear maximum is recognizable. All z-, and p-parameters promising good results were tested again on the final context-dependent system (the most promising was taken marked by ♣ in Table 6.1). These results are listed in Table

English (CI: 28.4) Baseline (CD: 87.3)					
TC	P=1, Q=1	P=2, Q=1	P=2, Q=2	P=3, Q=2	P=3, Q=3
1	79.2	76.1	75.3	76.8	74.4
2	79.8	79.3	77.0	77.6	75.4
3	80.8 ♣	79.8*	76.9	77.0	75.1
4	80.5	80.1	78.3	77.4	76.4
5	-	80.2	-	-	-
6	-	80.2	-	-	-
7	-	79.4	-	-	-

Table 6.1: Word accuracy in % of the English recognizer using *phoneme-grapheme* questions. (TC denotes Training Cycle)

English(P=2,Q=2)		
TC	3000-Models	4500-Models
1	75.3	78.0
2	77.0	78.0
3	76.9	77.6
4	78.3	77.6

Table 6.2: Comparison of Word accuracy in % of the English recognizer using quint-graphemes as subunits and allowing 4500 models to be clustered comparing to 3000 clustered models (TC denotes Training Cycle)

A.3 on page 87. The maximum word accuracy rate of 80.9% was reached when setting z to 29 and p to 4.

Baseline Comparison

The final most promising English system reaches 80.9% word accuracy on the developement set and 77.8% on the evaluation set (z=29 and p=4). This is still significantly below the basline with 87.3% word accuracy for the development set and 84.4% word accuracy for the evaluation set. English is a language with a relatively difficult phoneme to grapheme alignment. It was expected to give the worst performance in this project. See Table 6.3 for the final results.

6.1.2 German

Table 6.4 shows the results for the German context dependent phoneme-grapheme recognizers, clustered with various widths, 3000-Models and language parameters z=25 and p=1.

Context Width

Performance of the German recognizers (compared to the English recognizer from above) does decrease with increasing context width too (for reasons mentioned above), but slower (compare the trigrapheme system (P=1, Q=1) with the septgrapheme system (P=3, Q=3)). It is interesting that the hybrid trigrapheme system (P=2, Q=1) performs up to 1.1% better than the true trigrapheme system. The author reasons that this is due the fact that for the clustering procedure more detailed models are at hand, e.g. quintgrapheme contexts are each modeled by a separate distribution, thus allowing the clustering procedure to be more precise in combining models. The finite system still generalizes well, since it goes back to only model on a trigrapheme level.

WORD RECOGNITION PERFORMANCE:		
	EVAL	DEV
Correct	81.5% (3205)	84.4% (3166)
Substitutions	15.0% (589)	13.1% (493)
Deletions	3.5% (138)	2.5% (94)
Insertions	3.7% (144)	3.5% (130)
Errors	22.2% (871)	19.1% (717)
WORD ACCURACY	77.8%	80.9%
BASELINE	84.4%	87.3%

Table 6.3: Final word accuracy rates of the English grapheme based ♣-system. Evaluated on the development and evaluation set (z=29 and p=4). Numbers enclosed in brakets denote number of words

Language Model Parameters

Language parameter adaptation can be seen on page 88 in Table A.4 for the context independent recognizer (CI: 53.0%). Again the decoder reacts very sensitive to parameter changes and there is more than one region that looks promising. Table A.5 on page 89 shows results of the language parameter adaptation on the most promising German system marked by $\diamondsuit$ in Table 6.4. The results of the language parameter adaptation in the context dependent case are smoother and it seems that 83.0% (e.g. z=25 and p=1) word accuracy is the maximum one can get out of the phoneme-grapheme recognizer for German.

German (CI: 53.0) Baseline (CD: 82.3)					
TC	P=1, Q=1	P=2, Q=1	P=2, Q=2	P=3, Q=2	P=3, Q=3
1	81.6	79.8	81.6	80.3	79.5
2	81.5	82.4	81.4	80.9	79.9
3	81.9	82.8*	81.6	81.3	80.5
4	81.7	83.0	81.5	81.2	81.3
5	-	83.0 $\diamondsuit$	-	-	-
6	-	82.8	-	-	-
7	-	82.8	-	-	-

Table 6.4: Word accuracy in % of the German recognizer using *phoneme-grapheme* questions. (TC denotes Training Cycle)

Baseline Comparison

The German system (z is set to 25 and p to 1) reaching a word accuracy of 83% for the development set shows an improved word accuracy rate of 0.7% absolute compared to the basline system (82.3% development set). But the evaluation shows an increased WE of 4.9% absolute, whereas the basline system acctually shows improved WA of

4.7% absolute. Nontheless the results show that using graphemes as subunits in case of German performs comparably well to a system based on phonemes as subunits. See Table 6.5 for the final results.

WORD RECOGNITION PERFORMANCE:		
	EVAL	DEV
Correct	82.0% (2789)	89.2% (2617)
Substitutions	15.3% (519)	10.1% (297)
Deletions	2.8% (95)	0.7% (20)
Insertions	3.8% (131)	6.2% (181)
Errors	21.9% (745)	17.0% (498)
WORD ACCURACY	78.1%	83.0%
BASELINE	87.0 %	82.3%

Table 6.5: Final word accuracy rates of the German grapheme based $\diamondsuit$-recognizer. Evaluated on the development and evaluation set (z=25 and p=1). Numbers enclosed in brakets denote number of words

6.1.3 Spanish

Table 6.6 shows the results for the Spanish context dependent phoneme-grapheme recognizers, clustered with various widths, 3000-Models and language parameters z=25 and p=0.

Spanish (CI: 44.6) Baseline (CD: 75.5)					
TC	P=1, Q=1	P=2, Q=1	P=2, Q=2	P=3, Q=2	P=3, Q=3
1	72.6	73.2	71.0	71.1	68.2
2	72.9	73.4	71.2	71.6	68.3
3	72.9	73.6*	71.0	71.8	68.6
4	73.0	73.9	71.1	70.0	68.2
5	-	73.9	-	-	
6	-	74.0 ♠	-	-	
7	-	73.9	-	-	

Table 6.6: Word accuracy in % of the Spanish recognizer using *phoneme-grapheme* questions (TC denotes Training Cycle)

Context Width

Performance of the Spanish recognizer (similar to the German recognizer) does decrease with increasing context width as well and shows an optimal performance for the hybrid trigrapheme system (P=2, Q=1). Refer to the explanations given for the German recognizer in Subsection 6.1.2.

Language Model Parameters

Language parameter adaptation can be seen on page 90 in Table A.6 for the context independent recognizer (CI: 44.6%). Again the decoder reacts very sensitive to parameter changes (ranging form around 41% to roughly 49% WA in this case). Table A.7 on page

91 shows results of the language parameter adaption on the most promising Spanish system marked by ♠ in Table 6.6. Since the author was running out of time and started to investigate the language model parameters at the wrong end first, parameters z=25 and p=5 were taken to evaluate the final system. However performance is expected to improve if searching further into the direction $24 <= z <= 26$ and $p >= 6$.

Baseline Comparison

The Spanish grapheme recognizer reaches a WA of 73.2% for the development set and comes close to the baseline's WA of 74.5%. The evaluation set performs surprisingly well with a WA of 82.5% which is almost 10% absolute decrease in WE compared to the development set. Since the results of the basline and the grapheme based speech recognizers are so close together this can be seen as confirmation of the close grapheme to phoneme relation in Spanish. It thus seems to make no difference in Spanish if the models are phoneme based or grapheme based. The Spanish recognizers (both the baseline and the ones built during this project) fall short of the expectations. A reason for this is the poor data preparation, the acoustic data was not cut very carefully. See Table 6.7 for the final results.

WORD RECOGNITION PERFORMANCE:		
	EVAL	DEV
Correct	86.7% (5217)	77.7% (5465)
Substitutions	9.8% (590)	17.3% (1217)
Deletions	3.5% (212)	5.0% (349)
Insertions	4.1% (249)	4.6% (321)
Errors	17.5% (1051)	26.8% (1887)
WORD ACCURACY	82.5%	73.2%
BASELINE	83.7%	75.5%

Table 6.7: Final word accuracy rates of the Spanish grapheme based ♠-recognizer. Evaluated on the development and evaluation set (z=25 and p=5). Numbers enclosed in brakets denote number of words

6.1.4 Summary

The grapheme-based speech recognizers show promising results. Especially for languages with a close grapheme to phoneme relation it seems to be irrelevant if the models are based on graphemes or phonemes. In case of English the phoneme recognizer is still significantly better than the grapheme based one.

It is also obvious that the recognizers are sensitive towards the context width with which they were collected and trained and the context width of their final models with which they were clustered. Table 6.8 shows another example of comparing different context widths to cluster models. New context dependent recognizers using different context widths were created from systems $P = 2$, $Q = 1$ in Table 6.1, 6.4, 6.6 marked by $\star$. This time it can be seen for all languages that the hybrid system (P=2, Q=1) looks most promising. As mentioned above the author believes that this is due to the fact that clustering can be done more precisely (distributions are based on quintgraphemes thus enabling more in detail and more accurate modeling) without loosing the advantage of good generalization of the final models (trigrapheme models).

LID	P=1, Q=1	P=2, Q=1	P=2, Q=2
English	79.7	80.1	79.1
Spanish	73.2	73.9	71.4
German	82.3	83.1	81.6

Table 6.8: Word accuracy in % for the recognizers changing context width of systems $P = 2$, $Q = 1$ in Table 6.1, 6.4, 6.6 marked by $\star$

6.2 Question Set Evaluation

It was experimented with different question sets on the English, German and Spanish recognizers. The context independent systems are the same as in Tables 6.1, 6.6 and 6.4. Generation of the individual question sets is described in Section 5.6 starting on page 44. It may seem surprising that the phoneme-grapheme question set does not perform best, but is outperformed by the singletons. Linguistic questions are derived on a phoneme basis, thus characterizing certain sounds that belong to the same sound class, e.g. are pronounced in a somewhat similar way. In the case of graphemes though, the pronunciation of a grapheme depends on its left and right context (e.g. a German "s" with a succeeding "c" and "h" is pronounced very differently than and an "s" followed by an "o"). To cluster classes together such that the acoustic material in one class is close, meaning they represent similar sounds, is in case of graphemes a question of which grapheme is to the right and left, whereas in case of sounds (phonemes) it is a question of to which sound class does the phoneme belong (is it a fricative, a plosive, etc.). This explanation is backed up by the fact that singleton questions perform less good in Spanish and best in English. Because the grapheme-phoneme mapping in Spanish is quite simple, the graphemes in Spanish can be looked at as almost phonemes. Thus the pronunciation of a grapheme (e.g. which models have similar acoustic data) is not as dependent on the neighboring graphemes, but more on the fact to which sound class it belongs to. In case of English with a loose phoneme-grapheme relation the linguistically motivated questions introduce errors, whereas the singletons are better able to characterize pronunciation and therefore acoustics.

The Hybrid Entropy questions perform generally better than the Bottom-Up questions, because the bottom-up clustering procedure imposes no closeness constraints on the set excluded from clustering (i.e. the complement of the cluster). Thus the acoustic models clustered together are similar to each other in the sense of the entropy distance criteria, but the remaining models do not have to fit together at all. The hybrid clustering procedure tries to ensure that the partitions are maximally separated. The classes defined by these questions ensure better similarity inside a class and greater distance to other classes thus making the clustering procedure more accurate.

	Language (P=1,Q=1)		
Ques Type	English	German	Spanish
Phoneme-Grapheme	76.1	79.1	73.2
Bottom-Up Entropy	74.8	79.8	72.2
Hybrid Entropy	77.5	80.7	72.5
Singleton	78.2	81.4	71.3

Table 6.9: Word accuracy in % of the recognizers using different questions

6.3 Three-lingual Recognizers

The monolingual recognizers from the above sections are used as baselines for the multilingual ones. The multilingual recognizers are trained with a third of the Spanish, a third of the English and a third of the German data. All in all the multilingual recognizers used the same amount of data across languages like the monolingual ones used for one specific language. It can thus be said that the multilingual recognizers use less data per language. The preprocessing, HMM-structure and training procedures are equivalent to the monolingual systems (refer to Chapter 5).

6.3.1 ML3-Mix

As seen in Table 6.10 the ML3-Mix system performs clearly worse than the monolingual recognizers tested on the target languages Spanish, English and German. Multilingual speech recognition based on graphemes as subunits does not reach performance of the monolingual grapheme based recognizers. To ensure that the differences in performance are not a matter of how many parameters are available to the system per language, a new context dependent recognizer was trained, allowing 3-times as many polygrapheme models (9000 Models). Results for the 9000-Models recognizer are slightly better, but they do not improve significantly to the monolingual baselines (refer to Tables 6.1, 6.6, 6.4). The differences in performance can be explained by mixing graphemes belonging to different languages which results in model inaccuracies.
Further it can be seen that the context independent ML3-Mix systems still improves significantly if trained for more than the 6 cycles of 6 training iterations as used in the monolingual trainings case (see Table 6.12). The author assumes that because the models have to stand for a wider range of acoustic data (graphemes from different languages instead of graphemes belonging to only one language), it takes more time to learn the mapping because it is more difficult. The intentions behind this was to see how good a context independent recognizer can get to see if it makes sense to use a context independent recognizer for bootstrapping Swedish later on or a context dependent ML3-Mix recognizer.

Using different question sets (as shown in Table 6.11) doesn't seem to hurt performance averaged over the languages. The results are similar to the ones reached in Section 6.2. As already mentioned before mixing phonemes among languages makes sense because the notation is based on pronunciation and thus the acoustic data for a phoneme is similar across languages. But for graphemes this is not true. A ML3-Mix recognizer based on graphemes can make significant errors by mixing data from graphemes belonging to different languages: they might be pronounced totally different, meaning the acoustic data is not similar to each other.

ML3-Mix (P=1, Q=1)						
TC	English (CI: 19.1)		Spanish (CI: 32.1)		German (CI: 27.7)	
	3000 Models	9000 Models	3000 Models	9000 Models	3000 Models	9000 Models
1	67.1	68.1	66.3	66.4	73.1	73.4
2	67.5	68.1	65.4	66.1	73.0	73.4
3	68.5	66.7	65.8	65.7	73.0	74.0
4	69.0*	68.0	65.6*	65.9	74.1*	74.4

Table 6.10: Word accuracy in % of the ML3-Mix recognizer using *phoneme-grapheme* questions and tested on English, Spanish and German data (TC denotes Training Cycle)

6.3.2 ML3-Tag

For the above mentioned reasons it makes sense to tag each grapheme with the language it belongs to. The system then can decide itself if the acoustic data of graphemes is similar to each other across languages and can thus be modeled together or if the language specific differences require a separate modeling. Evaluation of the decision tree shows that the language questions (English, German or Spanish) are used throughout the tree.

Performance of the ML3-Tag system using phoneme-grapheme questions shows a maximum word accuracy rate of 72.3% for English where as the ML3-Mix system only

ML3-Mix (P=1, Q=1)									
TC	English (CI: 19.1)			Spanish (CI: 32.1)			German (CI: 27.7)		
	P-G	Hybrid	Singleton	P-G	Hybrid	Singleton	P-G	Hybrid	Singleton
1	67.1	67.3	68.9	66.3	63.3	65.9	73.1	73.3	71.8
2	67.5	68.4	66.1	65.4	63.6	64.4	73.0	73.4	70.7
3	68.5	68.9	18.7	65.8	62.6	43.5	73.0	74.0	43.5
4	69.0	68.2	64.6	65.6	63.0	65.0	74.1	73.3	73.2

Table 6.11: Word accuracy in % of the ML3-Mix recognizer using *phoneme-grapheme* questions, *hybrid* questions and *singleton* questions. Tests are performed on English, Spanish and German data (TC denotes Training Cycle)

ML3-Mix CI Recognizer			
TC	English	Spanish	German
6	19.1**	32.1**	27.7**
7	19.6	32.4	26.2
8	24.7	37.1	28.5
9	24.0	37.3	28.6

Table 6.12: Word accuracy in % of the three-lingual context independent recognizer using *phoneme-grapheme* (P-G) questions and tested on English, Spanish and German data (TC denotes Training Cycle)

reaches 69% WA. For Spanish the ML3-Tag system reaches a WA of 67.7% compared to maximally 65.8% in the ML3-Mix case. It also looks similar for German; 75.6% WA for ML3-Tag and 74.1% WA for the ML3-Mix system. The WE decreases approximately by 2.2% averaged over the languages in the ML3-Tag system. The gain in WA has to be accounted to the language information.
Using different question sets shows again that singelton questions perform best for English or German and worst for Spanish. The best WA averaged over the languages shows that the question sets almost perform equally with WE differences less than 1%.

ML3-Tag (P=1, Q=1)									
TC	English (CI: 25.9)			Spanish (CI: 38.5)			German (CI: 40.7)		
	P-G	Hybrid	Singleton	P-G	Hybrid	Singleton	P-G	Hybrid	Singleton
1	69.5	68.1	70.5	65.7	64.9	66.1	75.1	74.9	73.4
2	72.3	69.9	72.0	66.7	65.1	64.6	75.5	75.0	73.3
3	71.4	69.8	71.6	67.7	65.1	65.6	74.8	76.0	74.8
4	71.6	70.6	71.6	67.1	65.1	65.5	75.6	75.6	75.7

Table 6.13: Word accuracy in % of the ML3-Tag recognizer using *phoneme-grapheme* (P-G) questions, *hybrid* questions and *singleton* questions. Tests are performed on English, Spanish and German data (TC denotes Training Cycle)

6.4 Swedish

6.4.1 Flatstart vs. Bootstrapping

One Swedish recognizer was built and started up in the same way as the English, German and Spanish recognizer described in Section 5.3. Two other Swedish recognizers were bootstrapped with a CI-ML3-Mix and a CD-ML3-Mix system (Bootstrapping is done with CD-system $P = 1$, $Q = 1$ in Table 6.10 marked by $\star$ and with the CI-system in Table 6.12 marked by $\star\star$). All recognizers were trained with 6 training cycles before changing to a context dependent system. The context width during clustering and for the final models was set to $P = 1$ and $Q = 1$. The hybrid trigrapheme system (P=2, Q=1) was not considered since at the start of the experiments the outcome of the context-width tests was unknown and to ensure consistency the parameters remained at these values during the remaining experiments as well. As seen by the results of Table 6.14 bootstrapping does not seem to improve performance nor does it seem to hurt recognition accuracy after enough training iterations. It is as if the recognizer forgets its initial parameters.
But looking at the situation right after the first trainings cycle the Swedish flatstart recognizer only reaches a word accuracy rate of 1.3% whereas the bootstrapped recognizer (with system in Tabel 6.10 marked by $\star$) reaches a WA of 19.0% right after a first trainings cycle (see Table 6.14). Bootstrapping can thus be seen to speed up the trainings procedure of a recognizer, thus saving development time and money.
In case of Swedish, bootstrapping does not seem to improve recognition accuracy after the same amount of trainings iterations, but it does provide better WA results sooner. It can also be looked at as an easy and alternative way (compare to Section 5.3) to initialize models to generate first labels.

6.4.2 Hybrid-Entropy vs. Singleton Questions

As already seen in the above Section 6.2 the Singleton questions perform better than the other question sets for English and German. This is also the case for the Swedish

Swedish (P=1, Q=1)			
TC	Flatstart	Bootstrap **	Bootstrap *
CI 1	1.3	-	19.0
CI 2	-	-	-
CI 3	-	-	-
CI 4	-	-	-
CI 5	-	-	-
CI 6	23.0	23.9	23.2
CD 1	52.1 ♡	52.0	50.4
CD 2	52.9	52.7	51.3
CD 3	52.1	51.9	51.1
CD 4	52.0	53.0	50.6

Table 6.14: Word accuracy in % of the Swedish recognizer using *hybrid entropy* questions. Bootstrapping is done with system $P = 1$, $Q = 1$ in Table 6.10 marked by $\star$ and with the context independent multilingual system in Table 6.12 marked by $\star\star$ (TC denotes Training Cycle)

recognizer (see Table 6.15). Since we used the Swedish language to test our speech recognition training without using any human expert knowledge, only data-driven question generation (Hybrid-Entropy questions) and singletons were considered. Singleton questions show an improved WA of 2.4% absolute compared to the best Hybrid-Entropy question set result. See Section 6.2 above for an explanation.

Swedish (P=1, Q=1)		
TC	Bootstrap * (CI: 23.2)	
	Hybrid	Singleton
1	50.4	51.7
2	51.3	53.5
3	51.1	53.2
4	50.6	53.7△

Table 6.15: Word accuracy in % of the Swedish recognizer using *hybrid entropy* questions and *singleton* questions. Bootstrapping is done with system $P = 1$, $Q = 1$ in Table 6.10 marked by ⋆ (TC denotes Training Cycle)

6.4.3 Language Model

System marked by ♡ in Table 6.14 with a word accuracy of 52.1% was evaluated using the old language model (obtained from [5]). The performance of the same system, but using the newly estimated language model reached a word accuracy rate of 53.1%. The new language model is not much better than the old one as can be seen from comparing their perplexities.

A language model calculated from the combined data makes better use of 3-grams and has a slightly lower perplexity. The performance of the recognizer is still not significantly better, but reaches a word accuracy of 53.7% on the Swedish ♡-system. See Table 6.16 for a comparision among the characteristics of the different language models.

The increased perplexity of the new language model is assumed to be compensated by an improved use of trigrams and bigrams. The language model from the combined data offers smaller perplexity and better 3-gram and 2-gram use.

6.4.4 Language Model Parameters

Initially the language parameters were set to z=25 and p=1 for all experiments. Language parameter adaptation was performed on the so far most promising Swedish system marked by △ in Table 6.15 and the old language model calculated by [5] was used to be comparable to the baseline.

Table A.1 on page 85 shows the language model parameter adaptation results. Parameters set to z=25 and p=-2 show a maximum word accuracy rate of 54.4%.

6.4.5 Baseline Comparison

The Swedish grapheme recognizer reaches a WA of 54.4% for the development set (z=25 and p=-2) and outperforms the basline's WA of 51.4%. Also on the evaluation set the performance of the grapheme based speak recognizer beats the basline's WA by 0.4% absolute. See Table 6.17 for the final results.

Language Model			
	LM-Old [5]	LM-New	Combination
Perplexity	1054	1199	973
WC Vocabulary	19661	19662	19662
WC LM	1441770	25454141	26895911
1-grams	57.22%	30.01%	27.2%
2-grams	32.24%	41.09%	41.11%
3-grams	10.53%	28.9%	31.69%

Table 6.16: Characteristics of the different Swedish Language Models (WC stands for word count)

The Swedish recognizers (both the baseline and the ones built during this project) show poor performance compared to the other recognizers. A reason for this is thought to be an inadequate language model. The perplexity is still too high even for the combined language model.

WORD RECOGNITION PERFORMANCE:		
	EVAL	DEV
Correct	57.8% (1781)	59.4% (1873)
Substitutions	35.0% (1079)	31.7% (1001)
Deletions	7.2% (222)	8.9% (281)
Insertions	10.1% (311)	4.9% (156)
Errors	52.3% (1612)	45.6% (1438)
WORD ACCURACY	47.7%	54.4%
BASELINE	47.3%	51.4%

Table 6.17: Final word accuracy rates of the Swedish grapheme based $\triangle$-system. Evaluated on the development and evaluation set (z=25 and p=-2). Numbers enclosed in brakets denote number of words

Chapter 7

Conclusion

This Diplomarbeit examined the possibilities of graheme based speech recognition. Using graphemes as subunits to model speech enables easy and fast creation of dictionaries. Different question sets were generated to produce context dependent speech recognizers without using linguistic knowledge and thus eliminating the need of a language expert.
Multilingual grapheme based speech recognizers were built and different ways were examined on how to combine the acoustic models of each language.
The multilingual recognizers built a basis to bootstrap a Swedish recognizer thus porting the multilingual knowledge over to the Swedish recognizer.

The results on the development sets for the English, Spanish, German and Swedish recognizer were promising and showed that it seems reasonable to build grapheme based speech recognizers. The data-driven question generation resulted in question sets that performed in some cases better than the baselines and in others came close to it. The results are very promising since the questions were generated with simple clustering algorithms using the entropy distance described in this Diplomarbeit. Future work will have to investigate if it enhances performance to distinguish between left context questions and right context questions or if using the likelihood distance results in better question sets.

The multilingual recognizers showed that their performance does not come close to the monolingual recognizers. This confirms that more specialized models perform better than generalizing ones provided the system requirements are known before hand. It has to be questioned if the idea of a global phoneme set introduced by [14] can be transported onto a global grapheme set. But porting the knowledge of the multilingual systems to a Swedish recognizer showed that this provides a good way to initialize the Swedish models and provides reasonable performance rates after a first training cycle. Future work should investigate if grapheme based multilingual recognizers perform equally well with limited data material or if they can be used as a language identifying component.

Bibliography

[1] K.Lenzo A.W.Black and V.Pagel. Issues in building general letter to sound rules. Workshop on Speech Synthesis, Jenolan Caves, Australia. ESCA, 1998.

[2] K.Lenzo A.W.Black and V.Pagel. Letter to sound rules for accented lexicon compression. Sydney,Australia. ICSLP, 1998.

[3] Tilo Sloboda and Alex Waibel. Dictionary learning for spontaneous speech recognition. Philadelphia, USA. Proceedings of the ICSLP 96, October 1996.

[4] B. Raj R. Singh and R. M. Stern. Automatic generation of subword units for speech recognition systems. In *IEEE Transactions on Speech and Audio Processing*, volume 10, pages 89–99, February 2002.

[5] Sebastian Stüker. Automatic generation of pronunciation dictionaries for new, unseen languages by voting among phoneme recognizers in nine different languages. Semester project, University Karlsruhe, Germany and Carnegie Mellon University, USA, 2002. Under the supervision of Professor Dr. Alex Waibel and Dr. Tanja Schultz.

[6] Toshiaki Fukada and Yoshinori Sagisaka. Automatic generation of a pronuuncia-tion dictionary based on pronunciation network. IPSJ SIGNotes Spoken Language Processsing. Information Processing Society of Japan, 2001.

[7] Stephan Kanthak and Hermann Ney. Context-dependent acoustic modeling using graphemes for large vocabulary speech recognition. In *IEEE Signal Processing Society, ICASSP in Orlando FL*, pages 845–848, 2002.

[8] J. Billa M. Noamany A. Srivastava D. Liu R. Stone J. Xu J. Makhoul and F. Kubala. Audio indexing of arabic broadcast news. In *IEEE Signal Processing Society, ICASSP in Orlando FL*, pages 5–8, 2002.

[9] SIL International. *Ethnologue*, 2003
. http://www.ethnologue.com.

[10] Simon Ager. *Omniglot a guide to writing systems*, 1998-2003
. http://www.omniglot.com.

[11] Janson Tore. *Speak a short history of languages.* Oxford University Press, New York, 2002.

[12] N.G. Jablonski and L.C.Aiello, editors. *The Orignin and Diversification of Language.* California Academy of Sciences, 1998.

[13] A. Nakanishi. *Writing Systems of the World.* Charles E. Tuttle Co., Tokyo, 1980.

[14] Tanja Schultz. *Multilinguale Spracherkennung - Kombination akustischer Modelle zur Portierung auf neue Sprachen.* Ph.D. dissertation, Univesity of Karlsruhe, Germany, 2000.

[15] B. Pfister and P.Hutter. Sprachverarbeitung 1 und 2. Lecture Notes. ETH, 2002.

[16] Ivica Rogina. *Parameterraumoptimierung für Diktiersysteme mit unbeschränktem Vokabular.* Ph.D. dissertation, Univesity of Karlsruhe, Germany, 1997.

[17] Tanja Schultz. *Tanja Schultz's homepage*, 2002
. http://www-2.cs.cmu.edu/∼tanja/.

[18] IPA. The international phonetic association (revised to 1993) - ipa chart. *Journal of the International Phonetic Association*, 1(23), 1993.

[19] Ivica Rogina. *Janus 3 Documentation*, 1998
. http://www-2.cs.cmu.edu/∼tanja/Lectures/JRTkDoc/.

[20] B. Ray R. Singh and R. M. Stern. Automatic clustering and generation of contextual questions for tied states in hidden markov models. *ICASSP in Phoenix, Arizona*, 4, May 1999.

[21] K. Beulen and H. Ney. Automatic question generation for decision tree based state tying. *Proceedings of the IEEE Conference on Acoustics, Speech and Signal Processing*, 2, May 1998.

[22] C.Bergamini J.F. Serignat L.Besacier D. Vaufreydaz and M. Akbar. A new methodology for speech corpora definition from internet documents. Proceedings of the LREC, Athens, Greece, 2000.

[23] Klaus Ries. *Language Modeling Tools*, 1997
. http://www.is.cs.cmu.edu/local/janus-lm.doku/node3.html.

[24] The Ibis-Gang. *Online JRTk Documentation*, 2002
. http://isl.ira.uka.de/∼jrtk/janus-doku.html.

[25] The Ibis-Gang. *JRTk and Ibis*, September 11, 2002
. http://isl.ira.uka.de/∼jrtk/doc.Janus5/janus-doku/janus-doku.html.

[26] Tcl Developer Xchange. *Tcl/Tk Manual Page*, 2002
. http://www.scriptics.com/man.

Appendix A

Tables

Swedish-△ System					
Word Accuracy in % (DEL,INS)					
z\p	-3	-2	-1	0	1
24	53.8(8.5—5.6)	54.0(8.7—5.3)	54.3(8.8—5.0)	54.0(9.3—4.9)	53.9(9.7—4.6)
25	54.2(8.7—5.3)	**54.4**(**8.9**|**4.9**)	54.2(9.2—4.7)	53.8(9.8—4.7)	53.7(10.2—4.4)
26	53.9(9.2—5.2)	54.0(9.5—4.9)	53.9(9.9—4.6)	54.0(10.1—4.3)	53.8(10.7—4.0)

Table A.1: Language parameter adaptation for the most promising Swedish recognizer

	Word Accuracy (DEL,INS)							
z\p	-2	-1	0	1	2	3	4	5
16	–.-(-.-—-.-)	29.0(14.8—5.7)	–.-(-.-—-.-)	29.7(16.3—5.0)	29.7(17.4—4.3)	23.8(28.6—0.0)	30.3(18.9—3.4)	24.5(20.1—4.9)
17	–.-(-.-—-.-)	–.-(-.-—-.-)	29.0(17.1—4.8)	29.9(16.1—4.8)	30.2(18.9—3.8)	22.8(21.2—6.5)	30.5(19.8—3.2)	30.4(20.4—3.1)
18	–.-(-.-—-.-)	28.5(16.7—5.1)	–.-(-.-—-.-)	31.3(17.8—3.8)	23.8(28.6—0.0)	30.3(19.5—3.5)	30.1(20.1—3.5)	23.4(22.3—6.0)
19	–.-(-.-—-.-)	–.-(-.-—-.-)	27.7(18.4—4.5)	30.6(18.8—3.7)	39.0(25.4—1.7)	23.4(19.6—7.1)	29.9(20.7—3.3)	29.3(21.8—3.1)
20	–.-(-.-—-.-)	29.9(17.5—4.4)	–.-(-.-—-.-)	20.7(27.7—6.5)	30.1(20.1—3.4)	29.9(20.3—3.1)	30.1(21.0—2.9)	29.3(22.4—2.9)
22	–.-(-.-—-.-)	–.-(-.-—-.-)	–.-(-.-—-.-)	29.1(20.5—3.3)	29.3(21.1—3.3)	29.7(22.0—2.8)	29.0(23.1—2.8)	28.6(23.9—2.3)
23	–.-(-.-—-.-)	–.-(-.-—-.-)	–.-(-.-—-.-)	28.5(20.8—3.0)	28.6(22.2—3.0)	28.7(23.1—2.6)	28.6(23.8—2.5)	28.6(24.4—2.1)
24	–.-(-.-—-.-)	–.-(-.-—-.-)	–.-(-.-—-.-)	14.3(42.9—0.0)	14.3(42.9—0.0)	28.8(23.5—2.7)	28.6(24.2—2.3)	28.5(24.6—1.9)
25	18.5(30.4—4.3)	28.6(33.3—0.0)	28.5(21.3—3.0)	29.0(21.9—2.5)	28.4(22.7—2.4)	28.2(23.9—2.4)	27.5(24.8—2.1)	27.6(25.3—1.9)
26	28.5(19.6—3.2)	32.9(18.9—4.4)	28.3(21.3—2.7)	27.4(22.8—2.5)	27.6(23.0—2.3)	–.-(-.-—-.-)	–.-(-.-—-.-)	–.-(-.-—-.-)
27	27.6(20.0—3.0)	28.6(33.3—0.0)	27.4(22.0—2.6)	26.7(23.4—2.5)	27.2(23.6—2.1)	–.-(-.-—-.-)	–.-(-.-—-.-)	–.-(-.-—-.-)
28	26.6(20.9—3.2)	26.8(21.9—2.9)	26.4(22.6—2.5)	27.5(22.1—3.7)	26.4(24.5—2.2)	–.-(-.-—-.-)	–.-(-.-—-.-)	–.-(-.-—-.-)
29	25.8(22.0—3.2)	25.6(23.0—2.8)	25.5(23.7—2.3)	25.9(24.1—2.2)	25.7(25.1—2.1)	–.-(-.-—-.-)	–.-(-.-—-.-)	–.-(-.-—-.-)
30	24.5(22.9—3.0)	24.9(23.9—2.6)	24.8(24.3—2.5)	25.0(24.7—2.1)	19.6(22.8—3.3)	–.-(-.-—-.-)	–.-(-.-—-.-)	–.-(-.-—-.-)
31	24.4(23.2—2.7)	24.0(23.7—2.6)	23.9(24.8—2.1)	24.2(26.0—1.9)	24.3(26.4—1.4)	–.-(-.-—-.-)	–.-(-.-—-.-)	–.-(-.-—-.-)

Table A.2: z and p Language model parameter adaptation for the context independent English system

English ♣-System			
Word Accuracy in % (DEL,INS)			
z\p	3	4	5
17	78.4(1.9—5.1)	–.-(-.-—-.-)	–.-(-.-—-.-)
18	78.9(1.9—4.8)	–.-(-.-—-.-)	–.-(-.-—-.-)
19	–.-(-.-—-.-)	–.-(-.-—-.-)	–.-(-.-—-.-)
20	–.-(-.-—-.-)	79.9(2.1—4.2)	–.-(-.-—-.-)
21	–.-(-.-—-.-)	80.0(2.2—4.2)	–.-(-.-—-.-)
22	80.1(2.2—4.1)	80.2(2.2—4.0)	80.3(2.3—3.9)
23	80.3(2.1—4.0)	80.3(2.2—3.9)	80.3(2.2—3.8)
24	80.3(2.2—3.9)	80.4(2.2—3.8)	80.5(2.2—3.7)
25	80.6(2.2—3.7)	80.6(2.2—3.6)	80.6(2.2—3.6)
26	80.7(2.1—3.6)	80.7(2.2—3.6)	80.6(2.4—3.6)
27	80.7(2.3—3.7)	80.8(2.3—3.5)	80.7(2.4—3.5)
28	80.8(2.3—3.6)	80.8(2.4—3.6)	80.8(2.5—3.5)
29	80.8(2.4—3.5)	**80.9(2.5|3.5)**	80.8(2.6—3.4)
30	80.7(2.4—3.5)	80.4(2.5—3.5)	80.5(2.6—3.4)
31	80.2(2.6—3.7)	80.1(2.7—3.5)	80.0(2.7—3.4)
32	80.0(2.6—3.7)	79.7(2.8—3.7)	79.8(2.7—3.6)

Table A.3: Language parameter adaptation for the most promising English recognizer

Word Accuracy (DEL,INS)									
z\p	-2	-1	0	1	2	3	4	5	6
18	–.-(-.-—-.-)	–.-(-.-—-.-)	–.-(-.-—-.-)	–.-(-.-—-.-)	48.8(10.0—5.6)	–.-(-.-—-.-)	48.0(11.0—5.0)	–.-(-.-—-.-)	48.8(12.1—4.1)
20	–.-(-.-—-.-)	–.-(-.-—-.-)	–.-(-.-—-.-)	–.-(-.-—-.-)	47.3(10.9—4.8)	47.2(11.4—4.7)	51.1(7.8—4.6)	47.3(12.5—3.9)	50.0(11.4—3.5)
21	–.-(-.-—-.-)	–.-(-.-—-.-)	–.-(-.-—-.-)	–.-(-.-—-.-)	50.0(9.5—4.6)	47.9(11.8—4.0)	52.4(7.3—4.3)	47.1(13.1—3.4)	46.4(14.0—3.3)
22	–.-(-.-—-.-)	–.-(-.-—-.-)	47.4(10.4—4.9)	–.-(-.-—-.-)	47.4(11.9—4.0)	47.6(12.0—3.6)	55.1(8.5—3.9)	47.0(13.7—3.1)	–.-(-.-—-.-)
23	–.-(-.-—-.-)	–.-(-.-—-.-)	47.0(11.1—4.6)	–.-(-.-—-.-)	47.2(12.0—3.5)	46.8(13.1—3.4)	46.5(13.9—3.3)	51.6(8.9—3.3)	–.-(-.-—-.-)
24	–.-(-.-—-.-)	–.-(-.-—-.-)	46.8(11.7—3.9)	–.-(-.-—-.-)	46.0(13.2—3.3)	45.9(13.8—3.3)	45.8(14.2—3.0)	45.5(15.3—2.9)	–.-(-.-—-.-)
25	45.9(11.2—4.3)	45.9(11.6—4.1)	46.3(12.4—3.5)	53.0(9.1—3.8)	45.8(13.7—3.3)	45.7(14.0—2.9)	45.2(14.7—2.8)	45.1(14.8—2.7)	–.-(-.-—-.-)
26	45.7(12.2—3.8)	45.6(12.5—3.6)	45.5(13.2—3.4)	45.3(13.3—3.3)	45.0(13.7—3.2)	–.-(-.-—-.-)	–.-(-.-—-.-)	–.-(-.-—-.-)	–.-(-.-—-.-)
27	45.0(12.2—3.6)	44.9(13.0—3.6)	44.8(13.4—3.2)	44.0(14.2—3.1)	43.9(14.7—2.9)	–.-(-.-—-.-)	–.-(-.-—-.-)	–.-(-.-—-.-)	–.-(-.-—-.-)
28	43.6(13.0—3.7)	43.6(13.3—3.1)	43.2(14.3—2.9)	42.8(14.8—2.9)	42.5(15.2—2.6)	–.-(-.-—-.-)	–.-(-.-—-.-)	–.-(-.-—-.-)	–.-(-.-—-.-)
29	42.2(13.3—3.0)	41.9(14.1—2.7)	41.5(14.8—2.6)	41.5(15.1—2.6)	41.2(16.1—2.5)	–.-(-.-—-.-)	–.-(-.-—-.-)	–.-(-.-—-.-)	–.-(-.-—-.-)
30	41.0(14.1—3.1)	41.1(14.7—2.9)	40.8(15.4—2.9)	40.1(16.2—2.4)	39.7(16.5—2.5)	–.-(-.-—-.-)	–.-(-.-—-.-)	–.-(-.-—-.-)	–.-(-.-—-.-)
31	40.0(14.6—3.4)	39.6(15.1—3.2)	38.9(16.2—2.9)	38.7(16.4—2.8)	38.1(17.4—2.6)	–.-(-.-—-.-)	–.-(-.-—-.-)	–.-(-.-—-.-)	–.-(-.-—-.-)

Table A.4: z and p Language model parameter adaptation for the context independent German system

German ◇-System

Word Accuracy in % (DEL,INS)

z\p	-1	0	1	2	3	4	5
20	–.-(-.-—-.-)	–.-(-.-—-.-)	–.-(-.-—-.-)	–.-(-.-—-.-)	–.-(-.-—-.-)	82.3(0.5—7.0)	–.-(-.-—-.-)
21	–.-(-.-—-.-)	–.-(-.-—-.-)	–.-(-.-—-.-)	–.-(-.-—-.-)	–.-(-.-—-.-)	82.4(0.5—6.7)	–.-(-.-—-.-)
22	–.-(-.-—-.-)	–.-(-.-—-.-)	–.-(-.-—-.-)	–.-(-.-—-.-)	–.-(-.-—-.-)	82.7(0.6—6.5)	–.-(-.-—-.-)
24	82.7(0.6—6.5)	82.7(0.6—6.4)	82.8(0.6—6.3)	82.8(0.7—6.3)	83.0(0.7—6.2)	83.0(0.7—6.2)	83.0(0.7—6.2)
25	82.9(0.6—6.3)	83.0(0.7—6.2)	**83.0(0.7|6.2)**	83.0(0.7—6.2)	83.0(0.7—6.2)	83.0(0.7—6.2)	83.0(0.7—6.1)
26	83.0(0.7—6.2)	83.0(0.7—6.2)	83.0(0.7—6.1)	82.9(0.7—6.2)	82.9(0.7—6.2)	82.9(0.7—6.1)	82.8(1.0—6.0)
27	82.9(0.7—6.2)	82.9(0.7—6.2)	82.8(1.0—6.1)	82.8(1.0—6.0)	82.8(1.0—6.0)	82.8(1.0—6.0)	82.8(1.0—6.0)

Table A.5: Language parameter adaptation for the most promising German recognizer

Word Accuracy (DEL,INS)										
z\p	-4	-3	-2	-1	0	1	2	3	4	5
16	49.2(9.9—7.7)	49.3(10.7—7.0)	49.4(11.6—6.5)	49.2(12.3—5.9)	–.-(-.-—-.-)	–.-(-.-—-.-)	–.-(-.-—-.-)	–.-(-.-—-.-)	–.-(-.-—-.-)	–.-(-.-—-.-)
17	48.9(10.4—7.2)	49.1(11.1—6.7)	48.7(11.9—6.4)	49.1(12.6—5.7)	–.-(-.-—-.-)	–.-(-.-—-.-)	–.-(-.-—-.-)	–.-(-.-—-.-)	–.-(-.-—-.-)	–.-(-.-—-.-)
18	48.1(10.9—7.2)	48.5(11.5—6.8)	48.4(12.0—6.3)	–.-(-.-—-.-)	–.-(-.-—-.-)	–.-(-.-—-.-)	–.-(-.-—-.-)	–.-(-.-—-.-)	–.-(-.-—-.-)	–.-(-.-—-.-)
19	–.-(-.-—-.-)	–.-(-.-—-.-)	47.8(12.5—5.9)	47.4(13.2—5.6)	–.-(-.-—-.-)	–.-(-.-—-.-)	–.-(-.-—-.-)	–.-(-.-—-.-)	–.-(-.-—-.-)	–.-(-.-—-.-)
20	–.-(-.-—-.-)	47.4(12.2—6.0)	47.3(12.8—5.6)	47.2(13.6—5.4)	46.4(14.4—5.1)	46.3(15.1—4.7)	46.2(15.9—4.5)	–.-(-.-—-.-)	–.-(-.-—-.-)	–.-(-.-—-.-)
21	–.-(-.-—-.-)	47.1(12.8—6.0)	47.3(13.4—5.4)	47.2(13.9—5.3)	46.7(15.0—4.9)	46.2(15.8—4.6)	46.1(16.4—4.1)	–.-(-.-—-.-)	–.-(-.-—-.-)	–.-(-.-—-.-)
22	–.-(-.-—-.-)	46.7(13.2—5.8)	47.1(13.4—5.3)	46.4(14.4—5.1)	–.-(-.-—-.-)	–.-(-.-—-.-)	45.1(17.2—3.8)	44.9(18.0—3.7)	44.6(18.8—3.5)	44.3(19.8—3.0)
23	–.-(-.-—-.-)	–.-(-.-—-.-)	–.-(-.-—-.-)	–.-(-.-—-.-)	–.-(-.-—-.-)	–.-(-.-—-.-)	44.8(17.5—4.0)	44.5(18.1—3.7)	43.8(19.5—3.5)	43.2(20.3—3.1)
24	–.-(-.-—-.-)	–.-(-.-—-.-)	–.-(-.-—-.-)	–.-(-.-—-.-)	–.-(-.-—-.-)	–.-(-.-—-.-)	44.5(17.7—3.8)	43.9(18.9—3.6)	43.2(20.1—3.2)	42.8(20.7—2.9)
25	–.-(-.-—-.-)	–.-(-.-—-.-)	–.-(-.-—-.-)	–.-(-.-—-.-)	–.-(-.-—-.-)	–.-(-.-—-.-)	–.-(-.-—-.-)	42.6(21.1—3.2)	42.2(21.9—3.0)	41.7(22.5—2.8)

Table A.6: z and p Language model parameter adaptation for the context independent Spanish system

Spanish ♠-System

Word Accuracy in % (DEL,INS)

z\p	-3	-2	-1	0	1	2	3	4	5	6
15	66.6(3.3—8.9)	67.1(3.4—8.4)	67.6(3.6—7.8)	–.-(-.-—-.-)	–.-(-.-—-.-)	–.-(-.-—-.-)	–.-(-.-—-.-)	–.-(-.-—-.-)	–.-(-.-—-.-)	–.-(-.-—-.-)
16	67.9(3.4—8.2)	68.0(3.6—8.0)	68.4(3.8—7.6)	68.9(3.8—7.1)	69.0(4.0—6.9)	–.-(-.-—-.-)	–.-(-.-—-.-)	–.-(-.-—-.-)	–.-(-.-—-.-)	–.-(-.-—-.-)
17	–.-(-.-—-.-)	68.9(3.7—7.7)	69.3(3.8—7.2)	69.8(3.8—6.7)	69.9(4.0—6.5)	–.-(-.-—-.-)	–.-(-.-—-.-)	–.-(-.-—-.-)	–.-(-.-—-.-)	–.-(-.-—-.-)
18	–.-(-.-—-.-)	69.6(3.7—7.2)	70.0(3.9—6.8)	70.3(4.0—6.5)	70.5(4.2—6.1)	–.-(-.-—-.-)	–.-(-.-—-.-)	–.-(-.-—-.-)	–.-(-.-—-.-)	–.-(-.-—-.-)
19	–.-(-.-—-.-)	–.-(-.-—-.-)	–.-(-.-—-.-)	70.7(4.1—6.3)	71.1(4.2—5.9)	71.2(4.3—5.9)	71.2(4.5—5.5)	71.4(4.6—5.2)	–.-(-.-—-.-)	–.-(-.-—-.-)
20	–.-(-.-—-.-)	–.-(-.-—-.-)	–.-(-.-—-.-)	71.2(4.2—6.1)	71.5(4.3—5.8)	71.7(4.4—5.6)	71.9(4.5—5.4)	72.1(4.6—5.0)	–.-(-.-—-.-)	–.-(-.-—-.-)
21	–.-(-.-—-.-)	–.-(-.-—-.-)	–.-(-.-—-.-)	72.1(4.2—5.7)	72.0(4.3—5.7)	72.1(4.4—5.5)	72.2(4.5—5.4)	72.5(4.5—5.0)	–.-(-.-—-.-)	–.-(-.-—-.-)
22	–.-(-.-—-.-)	–.-(-.-—-.-)	–.-(-.-—-.-)	72.2(4.2—5.7)	72.3(4.3—5.7)	72.4(4.3—5.4)	72.6(4.5—5.2)	72.7(4.7—5.0)	–.-(-.-—-.-)	–.-(-.-—-.-)
23	–.-(-.-—-.-)	–.-(-.-—-.-)	–.-(-.-—-.-)	–.-(-.-—-.-)	–.-(-.-—-.-)	–.-(-.-—-.-)	72.7(4.7—5.1)	72.9(4.8—4.8)	72.9(4.9—4.7)	–.-(-.-—-.-)
24	–.-(-.-—-.-)	–.-(-.-—-.-)	–.-(-.-—-.-)	–.-(-.-—-.-)	–.-(-.-—-.-)	–.-(-.-—-.-)	72.9(4.8—5.0)	73.0(4.8—4.8)	73.2(4.9—4.7)	73.3(5.0—4.4)
25	–.-(-.-—-.-)	–.-(-.-—-.-)	–.-(-.-—-.-)	–.-(-.-—-.-)	–.-(-.-—-.-)	–.-(-.-—-.-)	73.0(4.8—4.9)	73.0(4.8—4.8)	**73.2(5.0|4.6)**	73.3(5.2—4.3)
26	–.-(-.-—-.-)	–.-(-.-—-.-)	–.-(-.-—-.-)	–.-(-.-—-.-)	–.-(-.-—-.-)	–.-(-.-—-.-)	72.9(4.9—4.8)	73.0(4.9—4.6)	73.0(5.0—4.5)	–.-(-.-—-.-)

Table A.7: Language parameter adaptation for the most promising Spanish recognizer

Appendix B

Script Files

All Scripts marked by • will be handed in on CD at the end of this Diplomarbeit. Scripts marked by - are censored due to the JANUS license agreement.

B.1 Dictionary Creation

B.1.1 German

- makeDictDE.tcl
- makeGraphemeSetDE.tcl
- DictCleanDE.tcl
- DictCleanDE2.tcl
- DictCleanDE3.tcl

B.1.2 English

- makeDictEN.tcl

B.1.3 Spanish

- makeDictSP.tcl

- makeGraphemeSetSP.tcl
- DictCleanSP.tcl
- extractNbr.tcl
- zahl2word_SP.tcl
- zahlpron2J3.tcl

B.1.4 Swedish

- makeDictSW.tcl
- DictCleanSW.tcl
- makeGraphemeSetSW.tcl

B.1.5 Three-Lingual

- makeDictML3Mix.tcl
- insertMix.tcl
- makeDictML3.tcl

B.2 Question Generation

B.2.1 Grapheme-Phoneme Mapping

- phonemeToGraphemeDE.tcl
- phonemeToGraphemeSP.tcl
- phonemeToGraphemeEN.tcl

The phoneme notation is derived from Tanja Schultz's Ph.D. thesis [14].

English

See Table B.1

A → M_ale M_etu M_ab M_eI M_ETr M_ae M_aIp M_oVs M_aVs

B → M_b

C → M_tS M_s M_kh M_S M_z M_h

D → M_d M_dZ M_rfd

E → M_ip M_il M_i2 M_ae M_eI M_aIp M_ETr M_ab M_etu

F → M_f

G → M_g M_dZ M_Z M_kh

H → M_h M_u M_r3

I → M_ip M_i2 M_aIp M_il

J → M_dZ M_w M_Z

K → M_kh M_g

L → M_l

M → M_m

N → M_n M_g M_ng

O → M_etu M_oVs M_ocI M_aVs M_ov M_oc M_u M_ETr

P → M_ph M_f

Q → M_kh

R → M_r9

S → M_s M_S M_z

T → M_D M_th M_T M_d M_tS

U → M_etu M_ov M_u M_vst M_ip

V → M_v

W → M_w M_h M_v

X → M_z M_Z M_kh

Y → M_j M_i2 M_w M_ip M_il

Z → M_z M_Z

Table B.1: English Phoneme-Grapheme mapping for the creation of phoneme-grapheme questions

Spanish

See Table B.2.

German

See Table B.3.

B.3 Training

- samplesDE.tcl
- samplesSP.tcl
- samplesEN.tcl
- samplesSW.tcl
- samplesML3.tcl

- kmeans-lib.tcl
- kmeans.tcl
- DO.labels
- labels.tcl
- DO.ci
- lda.tcl
- samples.tcl
- train.tcl
- DO.cd
- ptree.tcl

a → M_a M_aI M_aU

a+ → M_a+

b → M_b M_V

c → M_k M_s M_T M_tS

d → M_D M_d

e → M_e M_eI M_eU

e+ → M_e+

f → M_f

g → M_G M_g M_x

h →

i → M_i M_j M_aI M_eI M_oI

i+ → M_i+

j → M_x

k → M_k

l → M_L M_l

m → M_m

n → M_n M_ng

n5 → M_nj

o → M_o M_oI

o+ → M_o+

p → M_p

q → M_k

r → M_r M_rf

s → M_s M_z

t → M_t

u → M_u M_W M_aU M_eU

u+ → M_u+

u∼ → M_W

v → M_b M_V

w → M_W

x → M_k M_s

y → M_j M_L M_aI M_eI M_oI

z → M_s M_T

Table B.2: Spanish Phoneme-Grapheme mapping for the creation of phoneme-grapheme questions

a → M_a M_ae M_aI M_al M_aU

∼a → M_ae M_e

b → M_b M_p

c → M_C M_x M_S

d → M_d M_t M_ts

e → M_a M_aI M_atu M_e M_el M_etu M_eU M_il M_oel

f → M_f M_v

g → M_g M_k M_ng M_x

h → M_h

i → M_aI M_i M_il M_j

j → M_j

k → M_k

l → M_l

m → M_m

n → M_n M_ng

o → M_o M_ol

∼o → M_oel

p → M_f M_p

q → M_k

r → M_atu M_r

s → M_S M_s M_ts M_z

t → M_t M_ts

u → M_aU M_eU M_oel M_u M_ul M_v

∼u → M_ue M_uel

v → M_f M_v

w → M_f M_v

x → M_k M_s M_x

y → M_ue M_uel

z → M_s M_ts M_z

Table B.3: German Phoneme-Grapheme mapping for the creation of phoneme-grapheme questions

- trainPT.tcl
- clusterCB.tcl
- splitCB.tcl
- trainVTLN.tcl

B.4 Questions

- searchQuestionHybrid.tcl
- searchQuestionPtree.tcl
- DO.questions

B.5 Bootstrapping

- mapping.ML3
- rewrite.tcl

B.6 Language Model

- DO.normalize
- filterHTML.tcl
- romanHTML.tcl
- finishHTML.tcl
- vocabExtract.tcl
- devExtract.tcl

Appendix C

The JANUS Recognition Toolkit

C.1 JRTk and Ibis

JANUS is a reasearch project for building multi-lingual machine translation systems for spontaneous human-human dialogs. *JRTk* stands for *Janus Recognition Toolkit* used for the development of speech recognition systems. It was and still is developed at the University of Karlsruhe in Germany and the Carnegie Mellon University in Pittsburgh, USA. It's current version is *IBIS V5/0 P011* which also includes the *Ibis* decoder which is integrated into the *JRTk*. Documentation is maninly available online [24, 25]. During this project the *JRTk* Version 4.0 was used. *JANUS* is implemented in C code with an interface in Tcl\Tk ([26]). This allows the creation of state-of-the-art speech recognizers and a platform that enables reasearchers to easely perform new experiments. Janus can be *HMM based* and combined with any kind of *Gaussian mixtures* or it can be based on *neural nets*.
A How-to built a speech recognizer with *JANUS* is attached in the Appendix D.1.

Appendix D

JANUS Howto

D.1 Starting Janus

You can either install the *JANUS* binaries and alter the Tcl-scripts or compile the source code. For this work the Sun and Linux binaries were used and most of the testing and training was performed on various dual processor machines. If *JANUS* is installed in $HOME/bin/Linux/janus and *Linux* is the operating system (Red Hat 7.1 in the case of this Diplomarbeit) the environmental variables can be set as follows in your *.cshrc* file:

```
setenv PATH "${PATH}:${HOME}/bin:${HOME}/janus/bin/Linux/janus:${HOME}/bin/Linux/align/"
setenv JanusHome         ${HOME}/janus/
setenv LD_LIBRARY_PATH   ${JanusHome}/lib

#set Site-Dependent Environment Vars
setenv JanusBin ${JanusHome}/bin
setenv JANUS_LIBRARY ${JanusHome}/library

#set architecture-depen evn vars
setenv TCL_LIBRARY   ${JanusHome}/lib/tcl8.3
setenv TK_LIBRARY    ${JanusHome}/lib/tk8.3
setenv TCLTK_INCLUDE "-I${JanusHome}/lib/include"
setenv TCLTK_LIB     "-L${JanusHome}/lib -ltcl8.3 -ltk8.3"
```

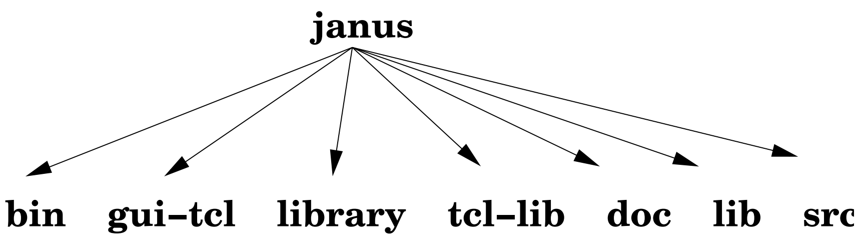

Figure D.1: How you should organize your *JANUS* directory

```
setenv CPU           linux
setenv LOCALINCLUDE  "${TCLTK_INCLUDE}"
setenv LOCALLIBS     "${TCLTK_LIB} -L/usr/X11R6/lib -lX11"
```

You also need a *.janusrc* file which you can copy into your home directory (in our example) and alter if necessary a few environmental variables descriptions. A typical *JANUS* installation uses a tree structure like in Figure D.1. In *gui-tcl* are a lot of scripts to help *JANUS* display its results. The scripts *tcl-lib* provide the user with a number of extensively used functions. After you installed everything and set the environmental variables, you can start *JANUS* by simply typing *janus* into a shell.

D.2 Getting Started

What does *JANUS* need to start training a speech recognizer? See Figure D.2 for a overview of the development of a speech recognizer in *JANUS*. First of all we need the recordings (either a wave-file or ADCs) and their transcriptions. This and a pronunciation dictionary are mandatory things to start working. Further it is of great help to have labels at hand and a language model for the recognition. The database with all the recordings and transcription as well as the dictionary have to be *JANUS*-readable. That means they have to be in a certain form and special characters that mean something to Tcl have to be dealt with. A *JANUS* readable dictionary looks like in Table D.1. The database has to be divided into a test set, a training set and a cross validation set.

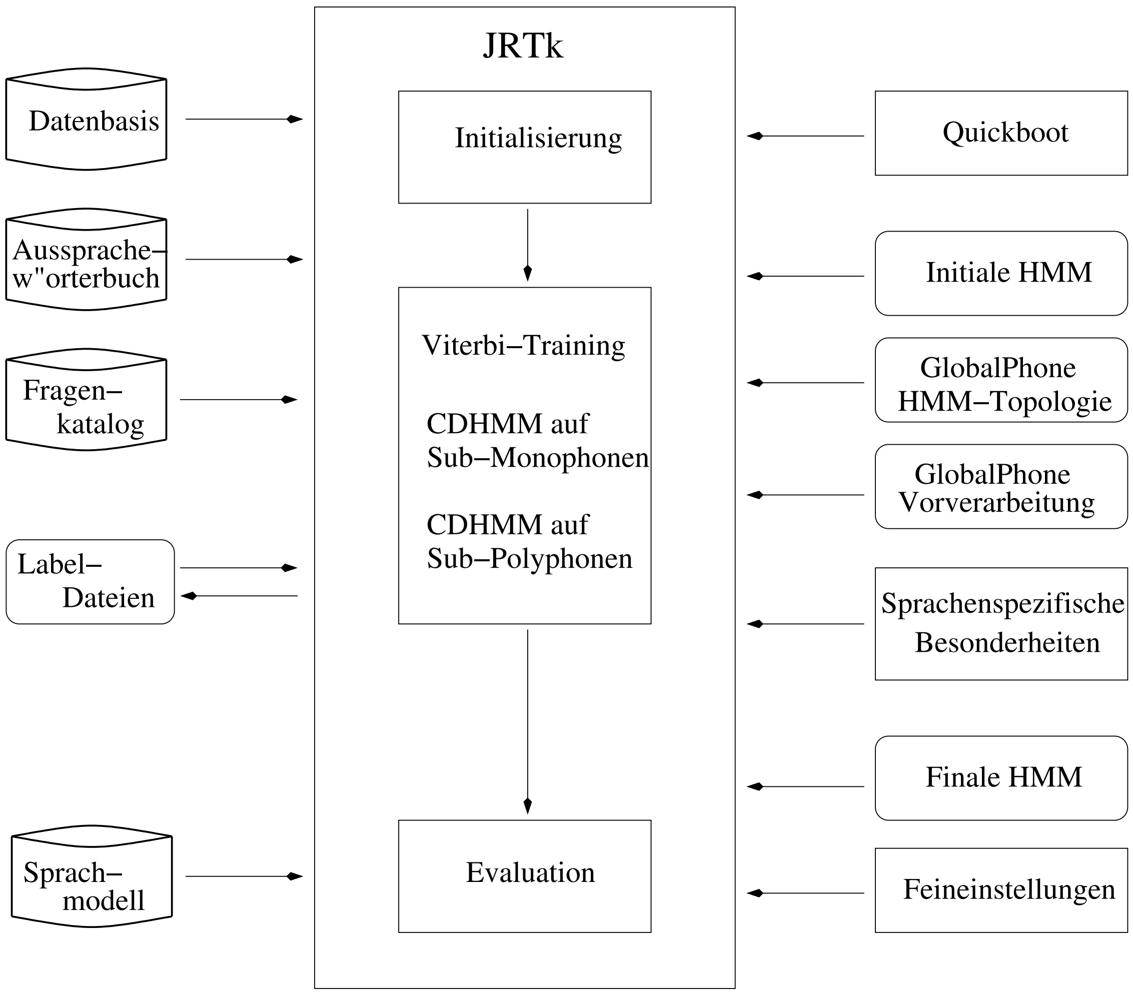

Figure D.2: System development of a speech recognizer in *JANUS* (taken from [14])

{SIL}	{{SIL WB}}
{EN=+BEEP+}	{{+QK_EN WB}}
{EN=+CHAIR_SQUEAK+}	{{+QK_EN WB}}
{EN=+TONGUE_CLICK+}	{{+hGH_EN WB}}
{EN=A}	{{A_EN WB}}
{EN=ABANDON}	{{A_EN WB} B_EN A_EN N_EN D_EN O_EN {N_EN WB}}
{EN=ABANDONED}	{{A_EN WB} B_EN A_EN N_EN D_EN O_EN N_EN E_EN {D_EN WB}}
{EN=ABANTO}	{{A_EN WB} B_EN A_EN N_EN T_EN {O_EN WB}}
{EN=ABBENHAUS}	{{A_EN WB} B_EN B_EN E_EN N_EN H_EN A_EN U_EN {S_EN WB}}
{EN=ABBOUD}	{{A_EN WB} B_EN B_EN O_EN U_EN {D_EN WB}}
{EN=ABBREVIATED}	{{A_EN WB} B_EN B_EN R_EN E_EN V_EN I_EN A_EN T_EN E_EN {D_EN WB}}
⋮	⋮
{EN=ZONES}	{{Z_EN WB} O_EN N_EN E_EN {S_EN WB}}
{EN=ZORZI}	{{Z_EN WB} O_EN R_EN Z_EN {I_EN WB}}
{EN=ZUCKERMAN+S}	{{Z_EN WB} U_EN C_EN K_EN E_EN R_EN M_EN A_EN N_EN {S_EN WB}}
{EN=ZURICH}	{{Z_EN WB} U_EN R_EN I_EN C_EN {H_EN WB}}

Table D.1: An example of how a dictionary in *JANUS* is supposed to look like. EN marks the corresponding language and WB is the word boundary tag

So far we have a database with the utterances and their transcription and an appropriate dictionary. We further need a few files that describe the architecture of the recognizer itself and provide additional information for it. The initCD.tcl script generates us some of them, namely: A codebookSet, a distributionSet, a distributionTree, a phonesSet and feature description and access files.

The CodebookSet tells the recognizer how a codebook looks like. For each codebook we thus have the description of its feature (e.g. LDA), the number of its Gaussian mixture components (e.g. 32) and the number of dimensions of the covariance and mean value matrix (e.g. 32) as well as their type (e.g. DIAGONAL).

The DistribSet file tells the recognizer which distribution uses which codebook.

The DistributionTree describes a context independent tree in the beginning which after traversing and ending in a leaf tells which distribution to use in that specific case.

The phonesSet is the list of phones that can be used. They might be grouped into classes as used for a context dependent recognizer.

The ptreeSet defines the polyphone trees. This is only used for context dependent recognizers.

The tags file lists all allowed modifiers for phones that can be used in the dictionary. WB for Word boundary is probably the most common.

The tmSet describes the transition probabilities of a *HMM* state.

In the topoSet file the *HMM* structure itself is described.

The topoTree tells which *HMM* model has to be used for which phones.

The featAccess file tells the featDesc file (look below) where to find the data.

In the featDesc file the recognizer gets to know how to process the wave files and which features should be extracted.

The *desc.tcl* script is the glue to all those files. It tells the recognizers where the created objects can find the information on how they look like.

Now we are all set to start the training step.

D.3 Training with JRTk

For each training step it is best to create a directory. The variable $SID in the *desc.tcl* scripts are usually set to the name of the directory. It is common to use e.g. EN as a language tag for English and set $SID to EN0, EN1, EN2, EN3 . . . where the number stands for the iteration step number. In this directory create another two sub directories. First *desc* which holds the *desc.tcl* file and second a sub directory called *train* which will hold a script calling all necessary training methods and eventually at the end of a training cycle all data. This so called data are the weights files produced by *JANUS* that hold e.g. the parameters calculated for the Gaussian mixtures.

There are multiple ways to build recognizers, but they all have a few steps in common:

Labels: It is possible to start with a random initialization of the parameters but it might take the recognizer a long time to learn the appropriate dependencies. It is desirable to have some precomputed labels ready at hand. Labels tell us which feature vectors are assigned to which *HMM* state. This can be calculated by the *Viterbi* or *Forward-Backward* algorithm. We usually compute new labels after every 4-6 training iterations. The scripts needed to do the job are the *labels.tcl* and the *DO.labels* script. The latter simply starts up *JANUS* and tells it to source the *labels.tcl*.

Kmeans-Initialization: With the provided labels the feature vectors from the training set are extracted and the LDA matrix is computed. This is done by the *lda.tcl* and the *samples.tcl* script. For each defined class *JANUS* initializes a codebook and a corresponding Gaussian mixture distribution. A fully continuous *HMM* is now ready. The *kmeans.tcl* script does this last part of the work.

Training: After a first initialization of all the models we are ready to optimize the *HMM* parameters with the *EM*-Algorithm and the training data. In a training iteration we fully work through all the training data. Per training step we usually perform 4-6 training iterations. A call to *train.tcl* will start the process.

So if you have everything ready and want to start training a recognizer, let's say you do have initial labels, created the directories like explained above with a $SID of EN0 in case of English. Alter the paths in the EN0/desc/desc.tcl file (copy it from somewhere into the newly created directory) and change into the EN0/train directory. You should either copy and alter or create yourself a *DO.ci* script that tells *JANUS* to start up and source the *lda.tcl*, *samples.tcl*, *kmeans.tcl* and *train.tcl* scripts in this order. There are options that allow you to tell *JANUS* how many training iterations you would like to perform. After completion you should end up with an *LDA*-matrix (look in 2.1.4 for explanation) and a codebook as well as distribution parameters. Now it's time to write new labels. Create like above another directory, in our case here that would be EN1 and its two sub directories. Copy the *desc.tcl* into the EN1/desc/ and the *DO.labels* into the EN1/train directory. Alter the path in desc.tcl. For example you need to tell *JANUS* where to store the labels and where it can find the just recently calculated parameter files from the training step. Repeat this training and label writing cycle until your recognizer produces satisfactionary results. You can test your results after each training step by calling the *test$SID.tcl*. Create another special directory to do the tests. You also need a *desc$SID.tcl* file and again a DO-file, the *DO.test.$SID.dev*. Of course $SID has to be replace by the current system ID, meaning EN0, EN2, ... When you are content with your current context independent recognizer you can start building a context dependent one.

Introduction of Subpolyphones: The initial context independent systems are build with subphone units. To get a context dependent recognizer the modeling with subpolyphones has to be prepared by allocating mixture distributions for each subpolyphone and train its parameters. All the *subpolyphones* of the same *subphones* share a codebook at this state of development. This provides us with a *SCHMM.* Set the variable $pcluster in *initCD.tcl* to 1 and recreate the codebookSet, distributionSet, distributionTree and newly create the ptreeSet file. Your phonesSet should not only consist of all possible phones, but should provide information about groups of phones. For example which phones are vowels which are not. These groups can be used as questions in the clustering process described bellow.

Clustering: With the clustering method described in 2.2.2 the subpolyphones are clustered into a specific amount of classes. The decision which subpolyphone belongs to which class is made by traversing the context decision tree. For this step you need to call *ptree.tcl*, *trainPT.tcl*, *ClusterCB.tcl* and *splitCB.tcl* in this order. An appropriate *DO.cd* will do the job. By the way *.cd* stands for context dependent now. Use the labels as well as the *LDA*-matrix computed by the best context independent system.

Kmeans-Initialization and Training of the new Classes: With the *kmeans-initialization* the codebooks and the mixture weights of the new classes are initialized and are then trained for 4 to 6 iterations with the *EM*-Algorithm. The resulting recognizer is then again a fully continuous system, but this time for subpolyphone- instead of subphone-classes.
The training of the context dependent recognizer is basically analogous to the context independent one. Labels are written with a call to *labels.tcl*, the *lda.tcl* and *samples.tcl* calculate the LDA matrix and extract the feature vectors and the *kmeans.tcl* initializes the codebooks and models. The only difference is that we use the *trainVTLN.tcl* script for the training iterations that additionally does a vocal tract length normalization.

Index