Triphone clustering in Finnish continuous speech recognition

Markku Ursin

July 1, 2002

Master's Thesis
Helsinki University of Technology
Department of Computer Science
Laboratory of Computer and Information Science

HELSINKI UNIVERSITY OF TECHNOLOGY — **ABSTRACT OF THE MASTER'S THESIS**

Author: Markku Ursin

Name of the Thesis: Triphone clustering in Finnish continuous speech recognition

Name in Finnish: Trifoniklusterointi suomenkielisessä jatkuvassa puheentunnistuksessa

Date: July 1, 2002	**Pages:** 125

Department: Computer Science
Professorship: Tik-61 Computer and Information Science

Supervisor: Professor Mikko Kurimo	**Instructor:** Professor Unto K. Laine

Abstract:

In this master's thesis the use of context-dependent phoneme models (triphones) in continuous Finnish speaker-dependent speech recognition is examined. In the first part of the thesis, a review of general concepts relating to the human speech production and hearing systems is covered, as well as the properties of the Finnish language. A summary of the structure of automatic speech recognition systems is also presented. Context-dependency of phonemes and coarticulatory effects are emphasized.

In the second part of the thesis, a speaker-dependent recognizer is built. The recognizer is based on hidden Markov models and it is developed using the Hidden Markov Model Toolkit (HTK). Different triphone clustering methods – a data-driven approach based on binary decision trees, and two approaches utilizing a priori knowledge on the place of articulation of phonemes and the type of phonemes – are studied. The best results are produced by the tree-based clustering algorithm, which also results in the highest number of models.

An extensive error examination is compiled based on the recognition tests. For each phoneme typical errors and contexts causing errors are analyzed.

Keywords:

Speech recognition, phonemic speech recognition, context-dependency, triphones, triphone clustering, hidden Markov models, HMM

TEKNILLINEN KORKEAKOULU **DIPLOMITYÖN TIIVISTELMÄ**

Tekijä: Markku Ursin
Työn nimi: Trifoniklusterointi suomenkielisessä jatkuvassa puheentunnistuksessa
Suomenkielinen nimi: Trifoniklusterointi suomenkielisessä jatkuvassa puheentunnistuksessa

Päivämäärä: 1. heinäkuuta 2002	**Sivumäärä:** 125

Osasto: Tietotekniikka
Professuuri: Tik-61 Informaatiotekniikka

Valvoja: Professori Mikko Kurimo	**Ohjaaja:** Professori Unto K. Laine

Tiivistelmä:

Tässä diplomityössä tutkitaan kontekstiriippuvien foneemimallien (trifonien) käyttöä suomenkielisen puhujariippuvan jatkuvan puheen tunnistimessa. Työn ensimmäisessä osassa tarkastellaan ihmisen puheentuotto- ja kuulojärjestelmiä, suomen kielen ominaisuuksia puheentunnistuksen kannalta sekä esitellään puheentunnistusjärjestelmien yleinen rakenne ja toiminta. Selostuksessa painotetaan foneemien kontekstiriippuvuutta sekä koartikulatorisia efektejä.

Työn toisessa osassa opetetaan puhujariippuva tunnistin käyttäen kätkettyjä Markov-malleja (HMM) sekä Hidden Markov Model Toolkit (HTK) -ohjelmistoa. Trifoniklusteroinnissa kokeillaan datalähtöistä binääriseen päätöspuuhun perustuvaa menetelmää sekä menetelmiä, jotka käyttävät hyväkseen tietoa foneemien äännetyypeistä sekä ääntämispaikoista. Parhaat tunnistustulokset saavutetaan puuklusterointimenetelmällä, jolloin myös malleja on suurin määrä.

Tunnistuskokeiden virheitä tarkastellaan laajasti. Foneemikohtaiset tyypilliset virheet ja eniten virheitä tuottaneet kontekstit analysoidaan.

Avainsanat:

Puheentunnistus, foneeminen puheentunnistus, kontekstiriippuvuus, trifonit, trifonien klusterointi, kätketyt Markov-mallit

Thanks

This thesis has been written as a part of the USIX Speech-To-Text project in the laboratory of Acoustics and audio signal processing.

I would like to thank my instructor professor Unto K. Laine for his valuable advice especially in articulatory and perceptual issues. The expertise of my supervisor professor Mikko Kurimo in the field of speech recognition was also precious. I am grateful to my co-worker D. Sc. Toomas Altosaar who, despite the tight schedule, agreed to read this thesis through and eliminate numerous insertion errors involving commas and articles and otherwise greatly improve my English. Without the financiers of the project, TEKES and the industrial partners, this thesis would never have been written.

I thank my parents for their support during my studies. Finally, my thanks go to my beloved Virve for her support, care, and love.

Otaniemi, July 1, 2002

Markku Ursin

Contents

Abbreviations and symbols

ASR	Automatic Speech Recognition
BFCC	Bark Frequency Cepstral Coefficients
BM	Basilar Membrane
DARPA	Defense Advanced Research Projects Agency
DCT	Discrete Cosine Transform
DTFT	Discrete Time Fourier Transform
ERB	Equivalent Rectangular Bandwidth
FFT	Fast Fourier Transform
HMM	Hidden Markov Model
HTK	Hidden Markov Model Toolkit
IDFT	Inverse Discrete time Fourier Transform
IPA	International Phonetic Alphabet
MFCC	Mel Frequency Cepstral Coefficients
MLE	Maximum Likelihood Estimation
MLF	Multiple Label Files
MLLR	Maximum Likelihood Linear Regression
MMIE	Maximum Mutual Information Estimation
PLP	Perceptual Linear Prediction
SLF	Standard Lattice Format
STT	Speech-to-text
TTS	Text-to-speech
VTLN	Vocal Tract Length Normalization
WLP	Warped Linear Prediction

/a/	Phoneme “a”
[a]	Speech sound “a”
Mon	Monophone Model Set
PoA	Triphone models clustered according to the Place of Articulation
ToP	Triphone models clustered according to the Type of Articulation
TLo	Aggressively tree-clustered triphone models
THi	Moderately tree-clustered triphone models

Part I

Theory

Chapter 1

Introduction

The goal of automatic speech recognition systems is to transcribe human speech into text, which can be further processed by machines or displayed for humans for reading in various applications. Despite the vast amount of effort put into research in automatic speech recognition, there still are problems to overcome.

Depending on the application, different kinds of recognizers are used. Isolated word recognition is sufficient for simple user interface systems. A harder task is to transcribe continuous speech, and even more difficult is to recognize spontaneous speech. Continuous speech recognition is needed for example in aids meant for the hearing or visually impaired.

Finnish speech recognition has been studied in several thesis before. The first [1, 2] described the overall structure of hidden Markov model (HMM) recognizers, while some later ones have concentrated on speaker adaptation [3, 4]. This thesis' emphasis is on context-dependency, and especially triphones, which have been succesfully used in other languages, but have not been studied for Finnish.

Large vocabulary recognition requires the use of subword units. The most commonly used unit is the phoneme, although other possibilities also exist. Straightforward monophone models produce hardly satisfactory results for difficult recognition tasks. This is due to the differences in phoneme realizations induced by coarticulatory effects in different contexts.

To overcome this, context-dependent models have been introduced. They are usually diphones, triphones, or quinphones, depending on how much of the phoneme context they take into account. The most intuitive choice is the triphone – a structure that covers immediately preceding and following phonemes. For diphones, only either the preceding or the following phoneme is included, so one has to make a decision about which is considered more important – both of them naturally affect the pronunciation of the central phoneme. On the other hand, it is questionable whether phoneme units further away from the examined phoneme have significant coarticulatory effects. This makes the use of quinphones less beneficial. Furthermore, when using larger units, the need for training data increases.

The number of triphones in a language is large. In a book used as the training and test material for this thesis, the number of different triphones was almost 10000. For

42 % of these units there were less than five occurrences in the data, which is far too few to find estimates for the model parameters. Clearly, there is a need for reducing the parameter space.

There are three solutions for this problem: parameter tying, clustering of states, and clustering of models. Parameter tying is the most lightweight operation where some parameters across the models (or states) are forced to be equal. The mechanisms of state and model clustering are similar. Three different approaches are presented in this thesis: data-driven clustering, decision tree based clustering, and classification based on articulatory facts. The first two are based on similarities found in the models or states after initial training, and the third is based on decisions made beforehand based on human knowledge.

This thesis discusses speech recognition from the Finnish point of view. There are many aspects about the language that make speech processing easier and many that make it more difficult than for other major European languages. The Finnish writing system is close to phonetic, but the huge number of different words due to Finnish morphology makes language processing very challenging.

Often, when speech recognition tests are run, the results are reported as raw numbers describing the error rates. In this thesis, a more thorough analysis is carried out. Errors typical of each phoneme are presented as well as the effect of the phoneme's position in the sentence and word.

This thesis is divided into eight chapters. Chapters 1 to 4 form the theoretical part of this thesis, and chapters 5 to 8 the experimental part.

The theoretical part aims to provide the reader with sufficient understanding of the human auditory and speech production system as well as the methods used in automatic speech recognition. Special attention is paid to issues involving context-dependency.

In the practical part designing and implementing recognizers using different methods for parameter space reduction is described, and rhese results are analyzed.

Chapter 2 presents an introduction to the human speech and auditory systems and describes the special features of the Finnish language.

Chapter 3 describes briefly the structure of modern speech recognition systems and the methods used in them. The HMM concept is shortly presented.

Chapter 4 focuses on context-dependency and the issues raised by it. Different triphone clustering algorithms are presented, and a couple of large vocabulary recognizers are described.

Chapter 5 describes the environment where the recognizer is built as well as the data used for recognizer training and testing.

Chapter 6 gives insight into the recognizer building process itself, and provides the reader with the needed HTK commands for accomplishing the different phases.

In Chapter 7 the results and different error types for different phonemes are presented. Each phoneme is checked for substitution, deletion, and insertion errors, and which are the most common and in which contexts. Some example sentences with recognized transcriptions are presented, as well.

Chapter 8 concludes the results of the recognition tests. Also, suggestions for improvement are given.

Appendices A, B, and C include many graphs that describe the error analysis visually.

Chapter 2

Speech, articulation, and phonetics

2.1 Speech events, phones, and phonemes

A *speech event* is any kind of sound produced by the speech organs of a person. Two speech events are considered to be different if any difference can be found between them, for example, in the spectral structure or timing. Therefore, any two speech events are most likely different – even if produced by the same person. [5]

Speech events that are phonetically equivalent between speakers can be referred to as *phones*. Phonetic equivalence is a complex concept that relies on the idealizing assumption that the differences between a set of speakers' phone organs can be ignored when evaluating the speech event's phonetic quality. [5]

Phones are regarded as realizations of the same abstract *phoneme* – they are its *allophones* or variants [6]. It should be emphasized that a phoneme is indeed an abstract concept. It is not possible to divide a word into phones with clear, unambiguous borders – the speech signal is changing continuously. The phoneme model is simply an idealizing abstraction of the complex phenomenon called speech.

Quite often the word "phone" is used when one actually means a phoneme. For example, one could speak about phone recognition, when the ASR system in question is actually based on phonemes. Even more often, there is confusion between the corresponding adjectives – phonetic and phonemic.

2.1.1 Notation

It is customary that speech sounds are marked in brackets (e.g. [a]) and phonemes between slashes (/a/). In both cases quantity, i.e., wheteher a phoneme or a phone is short or long, is marked with a colon ([aː] or /aː/).

There exists an alphabet that includes symbols for phonemes in different languages, and is called the International Phonetic Alphabet (IPA). Instead of IPA symbols, this thesis uses Finnish graphemes for describing the corresponding phonemes. An exception is the /ŋ/ phoneme that has no corresponding grapheme. In this case, the IPA symbol is used.

For most of the Finnish phonemes the IPA character is identical to the grapheme. Table 2.1 presents the exceptions.

Grapheme notation	IPA notation
/a/	/ɑ/
/ä/	/æ/
/ö/	/ø/

Table 2.1: Finnish phonemes that have a different character in the IPA notation than their corresponding grapheme in Finnish writing notation.

2.2 Speech production

According to the source-filter model of phonation there are three main aspects needed for speech production: a source of energy, a source of sound, and a filter. None of these can be directly mapped to specific organs – for example, the location of the sound source may vary for different speech sounds. [6]

The main part of the energy source is provided by the respiratory muscles. When the volume of the lungs is decreased by relaxing the diaphragm and intercostal muscles – and thus increasing the pressure inside the lungs, the pressure difference between the lungs and the parts of the vocal tract above the glottis (supraglottal vocal tract) begins to equalize, and air starts flowing through the glottis and the vocal tract – unless there is something blocking the tract. This kind of air flow from the lungs to the environment – called egressive air flow – is the most important mechanism for human speech production. In some rare cases, ingressive air flow to the lungs is utilized as well.

As such, air flowing through the vocal tract does not produce much sound. The glottis, a gap between the vocal chords in the larynx, works as a valve between the lungs and supraglottal vocal tract. For in- and exhaling, the valve should be open, allowing air flowing to and from the lungs as freely as possible.

In phonation of *voiced* speech sounds the vocal folds are moved closer to each other and the glottal orifice is narrowed. The air flow forced through the glottal slit makes the vocal folds vibrate. The glottis opens and closes quickly at a frequency around 100 Hz for males, 200 Hz for females and 300 Hz for children, causing constant changes in the air pressure above the glottis. This opening and closing acts as the source of sound. This frequency can be heard in voiced sounds and is refered to as pitch. An example of the speech sound [a] is presented in figure 2.1. The speaker can, to some extent, regulate the frequency of these *glottal pulses*. The pulses generated by the glottis could be heard as such but in reality the sound wave passes through the vocal tract filter, which causes significant changes to the produced sound.

For *unvoiced* speech sounds the source of sound is a constriction somewhere along the vocal tract that is so narrow that air flow through it produces turbulence and

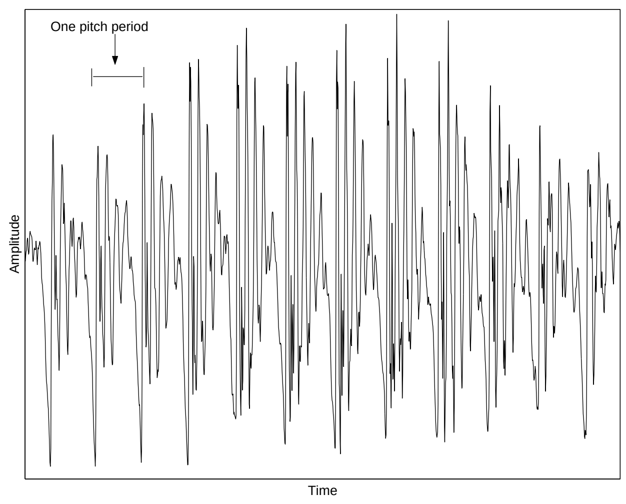

Figure 2.1: Waveform of the vowel [a]. The pitch period is indicated in the figure.

noise-like sound. For the speech sound [s], for example, the constriction is between the tip of the tongue and the alveolar ridge.

The part of the vocal tract that is located in front of the sound source acts as the main acoustic filter – for voiced sounds it consists of the part from the glottis all the way up to the lips and for labial sound sources, e.g., [f], only of the small gap between the bottom lip and the upper teeth. As the word filter suggests, this part of the speech system amplifies some and attenuates other frequencies of the signal produced by the sound source. [7]

2.3 Production of the different speech sounds

The classification of phonemes to vowels and consonants is a universal feature of all languages. For the Finnish language, vowels are sometimes defined as the set of phonemes that can alone form a syllable. Unfortunately, this definition does not hold for all other languages. Therefore, a more general definition is required.

Vowels are voiced sounds during which the air flow through the vocal tract is relatively free. The state of the vocal tract organs remains relatively stable during their phonation. Additionally, in order to exclude nasals ([m, n, ŋ]) and laterals ([l]) from the vowel class we require that air has to flow freely out of the middle of the mouth.

All the speech sounds that do not match the definition of a vowel are called consonants. The different types of consonants in the Finnish language are stops, fricatives, nasals, tremulants, laterals, and semi-vowels. [8]

2.3.1 Vowels

Vowels can be classified by the positions of the articulators (jaw, tongue, lips). The position of the tongue during the phonation of six Finnish vowels, [i, e, ä, u, o], and

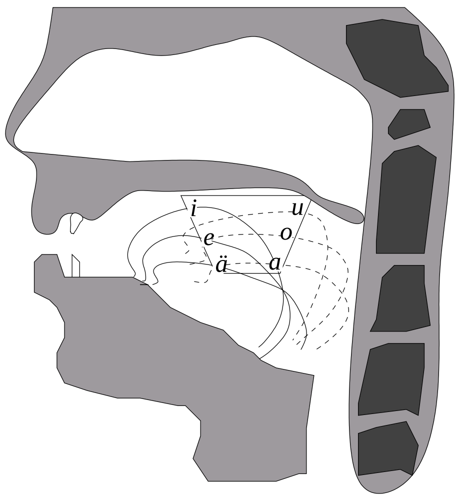

Figure 2.2: Position of the tongue during the production of different vowels. [8]

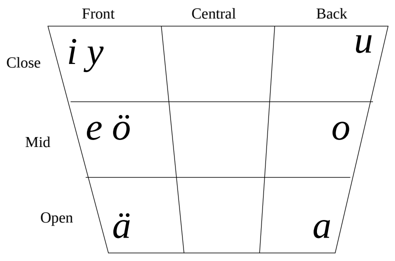

Figure 2.3: Vowel chart showing the basis for vowel classification. [8]

[a] is shown in figure 2.2. [y] and [ö] are not shown, since the tongue is approximately in the same position as for the vowels [i] and [e], respectively. [8]

The difference in the constrictions of the vocal tract is to a good part defined by the location of the highest point of the tongue. A schematic drawing describing the location of the maximum tongue height and position for the vowels mentioned above is shown in figure 2.3. All Finnish vowels are included in the picture, and all vowels in any language fall inside the chart.

As depicted in the figure, vowels can be classified in different ways. Whether they are front, central, or back vowels is decided by the location of the constriction caused by the tongue. Two other ways to classify them is by the narrowness of the strait and the roundness of the vowel. For example, [i] and [y] are both closed front vowels, but [i] is broad while [y] is round.

2.3.2 Consonants

The production mechanism of different vowels is quite similar, but the differences between the consonant classes are larger. While producing [l] or [j], for example, the air flows in a fashion that resembles greatly the air flow while producing vowels. On the other hand, for the stops [k, p, t], the air flow stops completely for a moment and the actual sound of these phonemes is produced by the burst-like explosion noise when the vocal tract finally opens. In this subsection, the production of the native Finnish consonants [d, h, j, k, l, m, n, ŋ, p, s, t, v] as well as [b, f, g] included in Finnish due to other languages are considered.

Stop consonants [k, p, t, g, b, d]

When producing stops the vocal tract is at first completely, or almost completely, closed at some point causing a high difference in pressure between the parts of the vocal tract above and below the closure. At this point, there is little or no sound present. The actual stop consonant is produced when the vocal tract suddenly opens and a pulse-like sound is released. [k, p] and [t] are voiceless stops while [g, b], and [d] are voiced. For [k] and [g] the closure during the closed phase of the stops is located in the velum, [t] and [d] in the dental area, and [p] and [b] in the labial part of the vocal tract.

Fricatives [f, h, s]

In the production of fricatives there is a constriction somewhere along the vocal tract that is narrow enough to produce noisy turbulent air flow. This noisy sound is then altered by the vocal filter. Finnish fricatives are always classified as voiceless. However, for example, the [h]-sound between two vowels may be voiced.

Nasals [n, m, ŋ]

Nasals are voiced consonants in which air flows through the nasal cavity while the primary vocal tract is closed. The location of the closure creates the main distinction between the three nasals. The sound is acoustically filtered by both the part behind the closure of the vocal tract and the nasal tract.

Tremulants [r]

The Finnish [r] is produced by letting the tip of the tongue vibrate against the alveolar ridge. The frequency of this vibration is typically 20 – 25 Hz. In Indo-European languages such as English or German, the place of the articulation is farther back in the vocal tract and no strong tip vibration is produced.

Laterals [l]

The tip of the tongue blocks free air flow by pressing against the alveolar ridge. However, on both sides of the tongue tip there is a passage for sound waves and air flow.

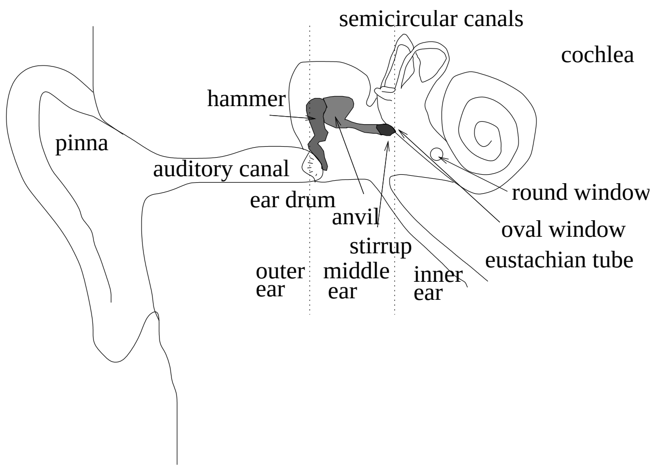

Figure 2.4: Different parts of the human ear. [7]

Semi-vowels [j, v]

The consonants [j] and [v] resemble vowels in general, but the constriction in the vocal tract is more powerful and the state of the tract is more context-dependent.

2.4 Speech perception

2.4.1 Anatomy of the ear

The ear is composed of three sections: the outer, middle, and inner ear, as depicted in figure 2.4. The outer ear is responsible for directing the sound waves to the eardrum, where the middle ear transforms air pressure variations into mechanical motion. In turn, the inner ear transforms the movements into electrical signals in the auditory nerve. [7, 9]

The outer ear consists of the pinna and the ear canal. The pinna is essential for sound localization as it attenuates sound coming from behind the head. The ear canal, or the *meatus*, can be regarded as a hard walled acoustical tube which boosts frequencies between 2 and 8 kHz. The first and second formants of speech are located within this frequency range.

The middle ear begins at the eardrum. Together with the three ossicular bones (hammer or *malleus*, anvil or *incus*, and stirrup or *stapes*) it linearly converts air pressure variations in the ear canal into the oval window membrane at the start of the inner ear. The acoustic impedance of the inner ear fluid is about 4000 times that of air. Therefore, most of the pressure waves hitting the inner ear would be reflected back if there was no impedance transformation mechanism. The middle ear also includes a mechanism for protecting the delicate inner ear against loud and low frequency sounds.

The part of the inner ear which performs in hearing is the *cochlea*, a snail-shaped fluid-filled tube. It is connected to the middle ear through the oval and round

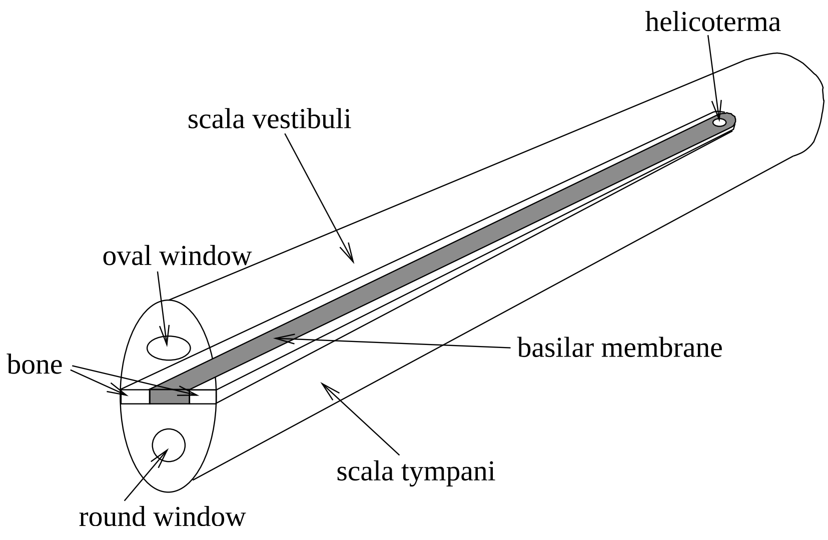

Figure 2.5: Cross section of the cochlea. [9]

windows. The cochlea is a sensitive and complex organ. It is divided into several compartments by membranes (see cross-section in figure 2.5), and each of these compartments contains fluid with different chemical makeup, causing potential differences between the compartments.

On the basilar membrane (BM) lies the organ of Corti, which contains *hair cells* organized in several rows along the length of the cochlea. The hair cells' hairs bend when the basilar and tectorial membranes vibrate, causing neural firings in the auditory nerve.

The BM varies gradually in tautness and shape. It is stiff and thin in the beginning (close to the round and oval windows), but flexible and thick at its end. Thus, its frequency response also varies: each location on the cochlea has a *characteristic frequency* at which it vibrates maximally. The characteristic frequency is high in the beginning of the BM and low at the end.

Sound entering the cochlea causes vibrations in the fluid, which creates traveling wave on the BM, which progresses from the beginning of the cochlea towards the end. This wave reaches a maximum amplitude at the point on the BM whose characteristic frequency matches the frequency of the input sound.

2.4.2 Psychoacoustics and the ear

There are two main ways to study the properties of the ear: 1) direct physiological measurements and experiments, and 2) psychophysical experiments. The former is very difficult to perform due to the delicate and complex structure of the ear. In the latter, the response to sounds are studied indirectly, and rely on psychic response. This approach is called *psychoacoustics* and studies *sensations*, subjective responses to different stimulus.

There are several results of psychoacoustical research that are essential in speech processing. The most important aspects are the different psychoacoustical pitch scales which are used to imitate the human hearing system. Before describing the different scales, a few psychoacoustic concepts need to be introduced.

Critical band

Critical band is an essential concept when studying the frequency resolution of hearing. "A critical band defines a frequency range in psychoacoustic experiments for which perception abruptly changes as a narrowband sound stimulus is modified to have frequency components beyond the band. When two competing sound signals pass energy through such a critical-band filter, the sound with the higher energy within the critical band dominates the perception and masks the other sound." [7] A critical band is approximately 100 Hz wide for center frequencies below 500 Hz, and 20 % of the center frequency for higher frequencies.

ERB band

Equivalent Rectangular Bandwidth (ERB) bands are another way to measure the bandwidth of analysis of the hearing system. The ERB of the auditory filter is assumed to be closely related to the critical bandwidth, but it is measured using a more sophisticated method rather than on classical masking experiments involving a narrowband masker and probe tone. As a result, the ERB is thought to be unaffected by activity in frequencies outside the studied band. [7]

Pitch

The *pitch* describes the subjective sensation of sound on the frequency scale ranging from "low" to "high". The closest match to pitch in physically measurable quantities is frequency, but pitch is also dependent on other quantities. There are two concurrent theories that try to explain the sensation of pitch: the place theory and the timing theory. The former is based on the fact that that the functioning of the basilar membrane (see section 2.4.1) is responsible for the sensations, while the latter is based on observations of the ear performing some sort of time domain periodicity analysis. Currently it is known that neither of the theories alone explain pitch sensations and that the hearing system includes more than one method.

Several pitch scales have been developed based on slightly different aspects of psychoacoustics. The mel, Bark, and ERB scales are presented in the following subsections. [7]

Mel scale

The mel scale is formed as follows: a starting frequency is selected and played to a subject. Then, a sound is searched for that in the subject's opinion is half (or double) in pitch. This procedure is repeated as necessary with the frequencies of new sounds as starting frequencies, and the results are averaged over many subjects.

Finally, an anchor point is selected where the frequency and pitch scales are set to be equal.

The dependency between frequency and pitch is approximately linear up to 500–1000 Hz but above that it is close to logarithmic.

This kind of scale is called the mel scale[1], and the unit of the scale is mel. An approximation usable on low frequencies is given by the equation:

$$m = 2595 \log_{10}(1 + f/700) \tag{2.1}$$

where m denotes the number of mels and f is the frequency in hertz. The anchor point for this scale is 1000 Hz which corresponds to 1000 mels. Other approximations exist, as well, and they are usable on different frequency bands. [9]

Bark scale

The Bark scale (named after the German acoustician Barkhausen) is based on critical bands. On this scale one critical band corresponds to one *Bark*. The Bark and mel scales are closely related and 1 Bark approximately equals 100 mels. [9]

ERB-rate scale

At different times the different pitch scales have been thought of mapping sounds linearly to the basilar membrane so that a constant difference on the pitch scale corresponds to a constant shift in the resonance point on the BM. According to latest research, the ERB-rate scale is the best match for this relation. As the name suggests, the ERB-rate scale is based on the ERB bands in a similar way that the Bark scale is based on critical bands. [9]

Stevens' power law

According to the Power law by Stevens [10], introduced in 1957, the relationship between stimulus intensity and sensation magnitude is described by the equation:

$$Y = kI^n \tag{2.2}$$

where Y is the sensation magnitude, k is a constant factor, I is the magnitude of the stimulus, and n is the exponent that has a different value for different sensations. If the exponent is less than one, the function is compressive, if n is equal to one, the function is linear, and otherwise it is expansive. Stevens measured values for n for different sensations, for example for brightness ($n \approx 0.33$) and for electrical shocks to teeth ($n \approx 7.00$). For the loudness of a 3000 Hz tone a value of $n \approx 0.67$ was obtained. However, on moderate sound levels and lower frequencies the value is significantly lower [10, 11].

[1]Mel is short for melody and thus written in lower case.

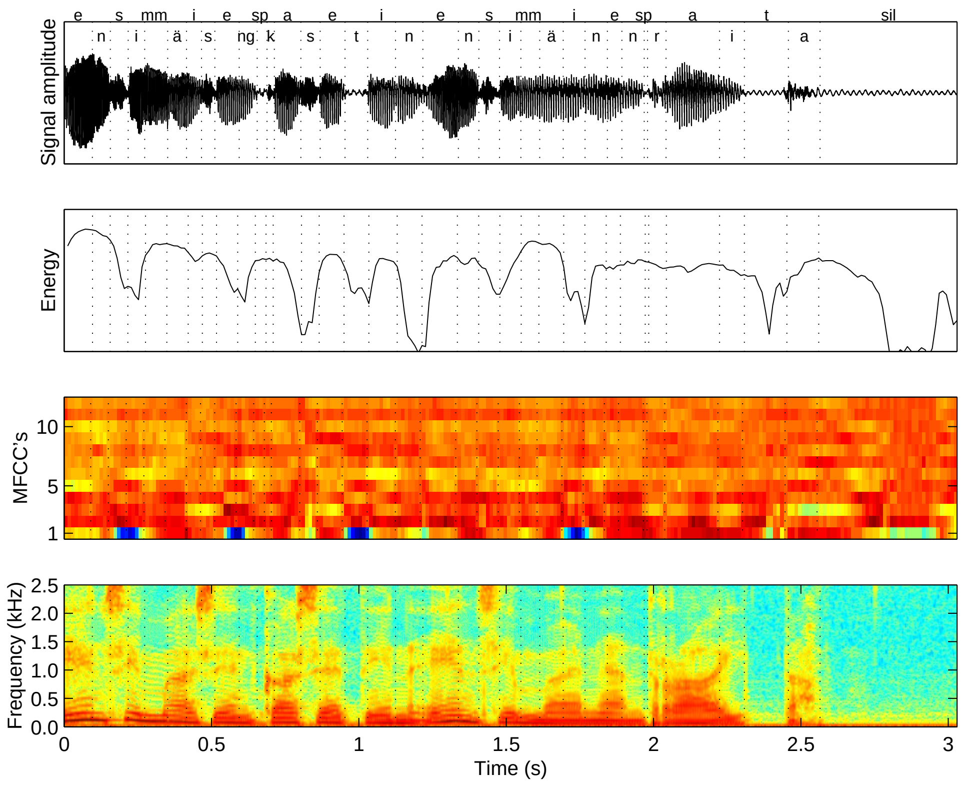

Figure 2.6: Waveform, spectrogram, energy, and mel-cepstral representation of the same speech signal along with phoneme annotations.

The power law is in contradiction with previous results by Fechner who argued that the relation between stimulus and sensation intensities would be logarithmic ($Y = k \log I$). Even today, many algorithms use the logarithmic function when approximating human sensation intensities.

2.5 Speech as seen by the computer

Humans use both audible and visual cues when recognizing speech with the audible information being more important. In the early days of automatic speech recognition only audible information was utilized since computers were not powerful enough for image processing. currently, there are projects aiming at combining audible and visual information in ASR [12], but for many common applications the acoustical stimulus is the only one that can be used. An example of this are telephone applications.

The analog speech signal is converted into an analog electrical signal within a microphone and into discrete, digital samples by an analog-to-digital converter (ADC). The amount of digital data depends on the sampling frequency as well as the amplitude resolution of the samples. For telephone speech, the frequency is usually 8 kHz and for CD-quality 44 kHz. A reasonable compromise between sound quality and storage space required dictates a 16 – 22 kHz sampling frequency. In the experiments of this thesis a frequency of 16 kHz was used.

In figure 2.6 four different views of the same speech signal are shown. The topmost is the speech waveform, the second is the overall energy of the signal, the third the standard 12th order mel-cepstral representation, and finally the short-time FFT representation, shown as a spectrogram.

The utterance presented is part of the training data used for the experiments of this thesis and it is uttered by a female speaker.

The individual sounds are visible in the spectrogram as well as the different formants which are the resonances produced by the vocal tract. Different phonemes produce different patterns of formants, and a human being can, for example, identify clearly pronounced vowels from each other quite simply by looking at their spectrograms.

For all the /s/'s, there is clearly more energy in the higher frequencies than for other phones. This shows the difference between noise-like fricatives and voiced sounds.

2.6 Special features of the Finnish language

The Finnish language, along with other Finno-Ugric languages such as Estonian and Hungarian, is fundamentally different from all other major European languages. In this section, the most important features of the Finnish languages from the speech recognition point of view are discussed.

1. Finnish is close to a "phonetic" language
   As a rule of thumb, the Finnish writing system follows the phonemic principle: there exists a different grapheme corresponding to each phoneme that is not used for any other purpose. This fact makes many aspects easier in speech and language research. For example, the conversion of written text to phoneme strings in speech synthesis is straightforward.

   Still, there are exceptions, with the most obvious being the phonemes /ŋ/ and /ŋː/ that are transcribed as graphemes *"n"* (usually, in front of a *"k"*) and *"ng"*. Additionally, the difference between spoken and written language causes irregularities between the graphemes and phonemes; phonemes are often dropped and added at certain locations of a word. In loanwords, on the other hand, a less educated speaker might replace the foreign consonants [b, f] and [g] with the unvoiced Finnish counterparts [p, v] and [k], respectively.

   Coarticulatory facts do cause phonemes to appear as different allophones in different contexts, but the number of different phonemes perceived by a native Finnish speaker is close to the number of different graphemes in the written language.

   As a result of Finnish being a phonetic language, native Finnish speakers can easily interpret a string of phonemes produced by a phoneme recognizer.

2. Number of words

   The morphology of the Finnish language is complex, which makes the number of different words immense. This is due to several factors: [13]

   (a) Finnish is a *synthetic/agglutinative* language
   As opposed to *analytic* languages, the meaning of a word is typically

changed by adding a suffix to the word and not by using additional words. There are sixteen different cases and additionally several other grammatical meanings which are expressed through suffixes, e.g., possessive relation, questioning and hesitation. These kind of suffixes are used with nouns, pronouns, adjectives, numerals, and even verbs. Suffixes often replace prepositions used in other languages.

As an example, the word *"puissaanhan"*, *"indeed in his/her/their trees"* can be split into morphemes (*/pu/i/ssa/an/han/*) in the following way:

- *pu* is the root of the word. The basic form of the word tree is *puu*, here the second */u/* has disappeared. This shows that Finnish is not cleanly agglutinative.
- *i* denotes a plural form.
- *ssa* is the suffix for the inessive case, which is roughly equivalent to the English preposition *in.*
- *an* represents the possessive form of the third person, singular or plural.
- *han* makes the word have an insisting tone.
- Verbs are conjugated in person, tense, and mood.

The word *'juoksentelisinkohan'* is as such a proper sentence in Finnish. When translated accurately into English, one needs seven words: *Would I really run a little around.* Again, this word can be split into different morphemes in the following way:

- *juoks* is the root of the word, derived from the verb *juosta*, *to run*
- *entel* means that the running is done in an aimless way
- *isi* indicates the conditional mood
- *n* indicates the first person, singular
- *ko* makes a question of the word
- *han* the same suffix as in the previous example, but, in this case, it makes the tone hesitating, rather than insisting.

(b) New words are formed by *composing* other words.
This is true for some Indo-European languages, e.g. German, as well. To some degree, it also applies to English.

The great number of different words makes it inefficient to make a simple list of Finnish words that would be usable for a large vocabulary speech recognizer; one would need to add knowledge about morphology to the language model. This is totally different for Indo-European languages and the large vocabulary recognizers for those languages usually rely highly on word lists, i.e., dictionaries. For Finnish some other approach is required. In this work, as well as previous works, phoneme recognition has been chosen as the solution. [14, 15]

3. Quantity of phonemes
Phoneme duration, or quantity, is a distinctive feature in Finnish – the meaning of a word can be changed by varying the temporal durations of the phonemes. For example, the words *taka, takka, takaa, taakka, takkaa,* and *taakkaa* all have different meanings, and even the non-existent words *taaka* and *taakaa* sound like proper Finnish words, but do not happen to have a meaning.

4. Vowel harmony
   In a Finnish non-compound and non-loan word, front vowels [y, ä], and [ö] and back vowels [a, o], and [u] are never present at the same time. The front vowels [i] and [e] are neutral in this respect: they can be combined with both front and back vowels. This fact can be utilized in speech recognition applications especially at the language model level.

5. Few consonant sequences
   Consonant sequences are rare and never appear in the beginning or end of a native Finnish word. In contrast, it is possible that a word consists solely of vowels (e.g., *aie*).

2.6.1 Phonemes present in the Finnish language

The Finnish phoneme set is smaller than the English one. In native words, there are eight vowels and thirteen consonants, and three additional consonants appear in loanwords. Both long and short versions exist for most phonemes.

Vowels

The Finnish set of vowels consists of five front vowels and three back vowels. The front vowels are /e, i, y, ä/, and /ö/, corresponding to the graphemes *e,i,y,ä*, and *ö*, respectively. The back vowels are /a, o/, and /u/ with the respective graphemes *a, o,* and *u*.

Consonants

There are eleven different consonants in native Finnish words: /d, h, j, k, l, m, n, p, r, s, t, v/, and /ŋ/. Not all of them appear at all word positions; for example, the consonants /d, h, j, k, m, p, v/ and /ŋ/ do not appear at the end of a word. Additionally, /d/ and /ŋ/, which are usually present only due to word inflection, have more restrictive appearance constraints.

The appearance of the phoneme /h/ is fundamentally different, depending on its location in the syllable. In the beginning of a syllable it is unvoiced and could be classified as a semi-vowel. In the end of a syllable it is often voiced and clearly a fricative. Both of these variants undergo a heavy allophonic variation. [16]

Phoneme statistics

Table 2.3 shows the statistics of Finnish phonemes from four different studies. Also, the statistics of the book used for the experiments of this thesis are shown (Paloheimo, 1996). The statistics made by Setälä [17] and Pääkkönen [18] are rather letter statistics than phoneme statistics, and the phoneme ŋ is not taken into account. Setälä based his study on the Finnish translation of the New Testament from the year 1914, Pääkkönen on randomly selected Finnish newspapers, magazines, literature, radio programs, and recorded spontaneous conversational speech, all from

Place of articulation	Bilab.	Labio	Alveol.	Post.	Palat.	Velar
Type of articulation		dental		alveol.		
Stops	p (b)		t d			k (g)
Nasals	m		n			ŋ
Laterals			l			
Tremulants			r			
Fricatives		(f)	s	(ʃ)		h
Semi-vowels		v			j	

Table 2.2: Finnish consonants classified according to articulatory type and place. The foreign consonants are presented in parenthesis.

the 1960's. Vainio's [19] and Iivonen's [20] studies are based on dictionary material. The table is ordered according to the statistics gained from Paloheimo's book.

The table shows well how important vowels are in Finnish: almost fifty percent of the phonemes are vowels. It also clearly shows that phonemes /a, i, t, e/, and /n/ are most common while /ö, ŋ, g, b/, and /f/ the most infrequent.

	Setälä 1972		Pääkkönen 1990		Vainio 1996		Iivonen 2000		Paloheimo 1996	
a	91168	11,90%	457350	11,63%	74255	11,35%	110368	11,40%	43672	11,96%
i	76738	10,01%	421366	10,71%	74908	11,45%	112231	11,59%	38758	10,61%
t	73142	9,54%	388711	9,88%	66977	10,23%	83479	8,62%	37758	10,34%
e	69677	9,09%	323087	8,21%	46044	7,04%	65330	6,75%	32500	8,90%
n	76303	9,96%	341181	8,67%	41426	6,33%	62200	6,43%	29602	8,10%
s	53167	6,94%	309350	7,86%	48956	7,48%	71349	7,37%	27868	7,63%
o	36208	4,72%	208923	5,31%	31270	4,78%	52057	5,38%	19062	5,22%
k	38750	5,06%	207520	5,28%	35402	5,41%	63959	6,61%	19824	5,43%
u	35310	4,61%	196678	5,00%	38781	5,93%	64326	6,65%	17859	4,89%
ä	45855	5,98%	189134	4,81%	24975	3,82%	31206	3,22%	15884	4,35%
l	44175	5,76%	226627	5,76%	40841	6,24%	56451	5,83%	18644	5,10%
m	25579	3,34%	137972	3,51%	20219	3,09%	28852	2,98%	11906	3,26%
v	17498	2,28%	96316	2,45%	13097	2,00%	23583	2,44%	9353	2,56%
r	11207	1,46%	85116	2,16%	25393	3,88%	38044	3,93%	8724	2,39%
j	20395	2,66%	75961	1,93%	6556	1,00%	12653	1,31%	6544	1,79%
y	9231	1,20%	71316	1,81%	15504	2,37%	20911	2,16%	7242	1,98%
p	11577	1,51%	65358	1,66%	15995	2,44%	29270	3,02%	6154	1,68%
h	20775	2,71%	71733	1,82%	15441	2,36%	21072	2,18%	5701	1,56%
d	6250	0,82%	33148	0,84%	5972	0,91%	7035	0,73%	3932	1,08%
ö	2347	0,31%	18655	0,47%	5285	0,81%	7123	0,74%	1978	0,54%
ŋ	-	-	-	-	2314	0,35%	1729	0,18%	1321	0,36%
g	646	0,08%	4151	0,11%	1912	0,29%	2017	0,21%	600	0,16%
b	80	0,01%	2068	0,05%	1287	0,20%	1317	0,14%	219	0,06%
f	305	0,04%	1934	0,05%	1644	0,25%	1465	0,15%	179	0,05%
N	766383		3933655		654454		968027		365284	

Table 2.3: Phoneme statistics from the material of the experiments of this thesis (Paloheimo 1996) compared to four different studies. The phonemes are ordered according to their frequency in Paloheimo 1996.

Chapter 3

Automatic speech recognition systems

3.1 Historical review

This section is based on [21], unless otherwise stated.

The first pioneer in speech technology design was Wolfgang von Kempelen. He designed a mechanical device mimicking the human vocal tract in 1791 [22]. Forty years later, based on the design by Kempelen, Charles Wheatstone built a machine capable of producing a good selection of both voiced and unvoiced sounds. This machine had a leather tube that replicated the vocal tract and a vibrating reed as the vocal cords, and bellows representing the lung.

The trend of imitating the human vocal tract using mechanical means continued into the 20th century. Gradually, when new algorithms, electronics, and finally computers with increasing computational power were developed, the scope of speech research broadened to speech coding and recognition applications. In 1928, Homer Dudley invented the vocoder that made it possible to transfer speech on the phone line by using only a fraction of the bandwidth. [23]

The first application of automatic speech recognition was a toy dog called Radio Rex produced in the 1920's. An electromagnet kept the dog in its house until its name was called. Then the simple electrical recognizer would react to the acoustical energy contained in the /e/, cut the current from the electromagnet, and let the dog jump out of its house. Of course, the recognition was not accurate, the dog would react to a host of other words as well. [24]

An early but more serious application of speech recognition was the single isolated speaker digit recognizer designed by Davis et al. at Bell Laboratories in 1952 [25]. Olson and Belar tried to build a recognizer for ten monosyllable words at RCA Laboratories in 1956. These two recognizers relied heavily on spectral measurements and direct pattern matching [26].

In 1959, Fry and Denes tried to build a phoneme recognizer to recognize four vowels and nine consonants using a spectrum analyzer and a pattern matcher to make the recognition decision. This recognizer contained simple language model that relied in statistical information about allowable phoneme sequences.

During the 1960's and 1970's new methods for speech recognition were introduced: silence detection methods, time warping, and dynamic programming. In the 1970's the first large vocabulary and speaker-independent systems were developed at IBM and AT&T. These recognizers mostly concentrated on isolated word recognition.

The most fundamental change in the 1980's in the field of speech recognition was the shift from the spectral pattern matching techniques to statistical modeling methods, for example neural networks and more importantly hidden Markov models or HMM's. HMM's were first introduced in several papers by Baum et al. in the late 1960's and early 1970's [27, 28, 29, 30, 31] and shortly after independently extended to automatic speech recognition by Baker and Jelinek [32, 33]. However, they did not become popular in speech recognition until the 1980's.

In 1975, Jelinek et al. presented the idea of language modeling in the form it is used today [34]. The goal of language modeling is to find and represent the relationship among words in sentences, just as the goal of acoustic modeling is to find regularities inside models. The so-called N-gram model is a set of conditional probabilities of vocabulary words followed by other words, estimated from large text material. A well trained model gives lower probabilities to ungrammatical than grammatical sentences.

Few fundamental new innovations have been discovered in speech recognition since the discovery of HMM's. Fine details have been tuned and growing computational capacity has allowed the use of more complex models giving some performance boost. However, at the same time the need for training data has also grown.

Finnish speech recognition has been studied at the Helsinki University of Technology since 1975. Different approaches have been used: the Learning Subspace Method [35, 36], neural network based methods [37], but for the latest decade, HMM's have been the main point of interest [38]. The shortage of available speech data has forced the research to focus on recognition limited in terms either of the number of speaker or the vocabulary size. Commercial speech recognition systmems also exist for Finnish [39, 40].

There is evidence that the performance of HMM's is not going to be greatly improved from what the best systems achieve currently. Since there is still room for hope that the recognition quality will improve, new methods are still being searched for. For example, in the laboratory of Acoustics and signal processing, new kinds of methods are being sought out, but at this stage no results are ready for publication. Furthermore, it is very difficult for new ideas to surpass the HMM concept which has been studied and developed for decades worldwide.

3.2 Structure of an ASR system

This section explains in short the basic structure of a current state of the art standard speech recognition system. In the following subsections the different parts of the system are discussed in more detail.

A block diagram of a speech recognition system is shown in figure 3.1. The basic structure of all modern recognizers, independent of recognizer type, is presented in this figure. First, discrete, equally spaced feature vectors are formed from a continuous speech signal. These vectors are assumed to carry a compact presentation suitable for classification of parts of speech. The actual recognizer makes use of the acoustic models, the lexicon, and the language model. Using these information sources the most likely transcription of the sentence is produced. [41]

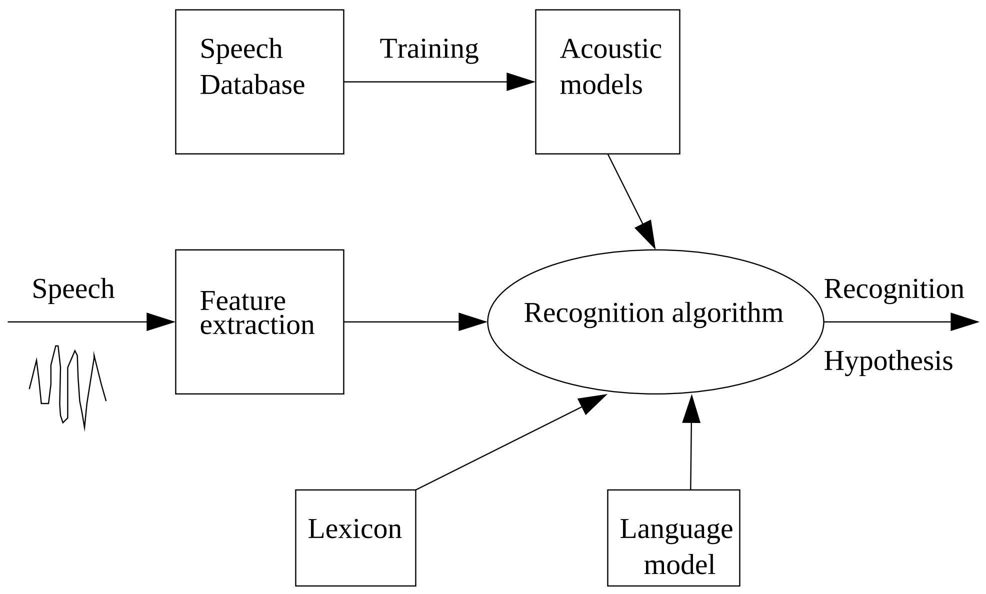

Figure 3.1: Block diagram of a speech recognizer. [41]

First of all, the speech signal has to be digitized so that it can be processed by the computer. The signal is then fed to the *preprocessor*, which is the feature extraction unit. The signal is split into frames consisting of about 200–800 samples, and the frames are fed to the feature extractor itself which reduces the amount of data by calculating a vector of about 10–60 features, depending on the front end.

The feature vectors of the training data are used for parameter estimation for the acoustic models which characterize the acoustic properties of the basic recognition unit, e.g. a phoneme or a word.

The lexicon is used to map the acoustic models to vocabulary words. In the case of Finnish phoneme recognition without a vocabulary, the lexicon is simply a mapping from phonemes to corresponding graphemes. The language model restricts the number of acceptable word combinations based on statistical information gathered from large text materials.

3.3 Acoustical front ends

Acoustical front ends are used for feature extraction purposes to transform the raw speech of the training and test files into more usable information. Features are numerical vectors that are designed to contain as much information relevant to recognizing phonemes as possible. On the other hand, they are a compressed representation of the speech data: much irrelevant information is discarded by the front ends.

Naturally, features should be discriminative – they should differ sufficiently for the different phonemes so that building and training models is possible. Additionally, they should be robust – features computed from separate utterances under different conditions should be similar. Another factor is computational efficiency: how much processor time is needed to extract the features from a speech utterance.

In this thesis, three different acoustical front ends were used. The first, Mel Frequency Cepstral Coefficients (MFCC) is the most commonly used front end in the field of speech recognition. Similar to it but somewhat more light-weight computationally is Bark Frequency Cepstral Coefficients (BFCC) which is based on Warped Linear Prediction and has been developed in the laboratory of Acoustics and Audio Signal Processing at the Helsinki University of Technology. The third front end is Perceptual Linear Prediction (PLP), which takes features of the human ear into account more specifically than the two other methods. Before going into detail about these front ends, some operations commonly applied to all front ends are presented.

The reason for including three front ends was to study if there was any difference in their performance. Especially BFCC, being rather newly introduced and more lightweight than MFCC, was of special interest in this respect.

3.3.1 Common front end operations

- Sampling. The acoustic speech signal has to be converted into a form that a computer can handle, a sequence of numerical values. This is done by taking discrete samples, or discrete values of the signal evenly spaced in time. A reasonable sampling rate for speech processing is 16000–22000 times a second, corresponding to sampling rates of 16 kHz–22 kHz

- Pre-emphasis. Since there is more energy in lower than in higher frequencies in speech, speech data is high-pass filtered according to equation 3.1. The value of the constant a is typically around 0.97.

$$x'(n) = x(n) - ax(n-1) \tag{3.1}$$

- Windowing. Single sample values of raw digitized speech are hardly of any use for feature extraction. On the other hand, the whole utterance is usually too long and contains many different phonemes so it cannot be used for feature extraction, either. Instead, the utterance is divided into equally sized pieces, called frames.

  Some further operations that are done for the framed data consider the signal as a periodic signal that is formed by concatenating the contents of the frame several times after each other, with the Discrete Time Fourier Transform (DTFT) being the most important of these. If the frames are formed simply by cutting pieces of the long utterance discontinuities arise wherever the beginning and the end of the original frame meet, which introduces anomalies to the spectrum produced by the DTFT. To prevent this from happening, a windowing function is used, with the most common being the Hamming window.

Additionally, in order to prevent problems occurring due to the selection of the frame locations, the signal is windowed in such a way that the frames overlap.

- Frequency warping. It seems naturally desirable to have the front end simulate the function of the human ear, which is more sensitive at low than high frequencies. The frequency scale of the front end is altered similarly which can be achieved through filter banks or Warped Linear Prediction, for example.

- Δ- and ΔΔ-coefficients. The performance of a speech recognition system can be enhanced by adding derivate coefficients to the static feature coefficients. While the static features describe the static state of the speech signal, the first and second order derivative coefficients, called Δ- and ΔΔ-coefficients, describe its dynamic properties, which are essential for modeling the transitions from one phoneme to another. The Δ-coefficients are calculated according to the following equation:

$$d_t = \frac{\sum_{\theta=1}^{\Theta} \theta (c_{t+\theta} - c_{t-\theta})}{2 \sum_{\theta=1}^{\Theta} \theta^2} \tag{3.2}$$

  where d_t is the Δ-coefficient, Θ is the width of the window used for calculating the Δ-coefficients, normally equal to two. c_t is the value of the static coefficient at time t.

  ΔΔ-coefficients are calculated similarly, except that instead of c_t-coefficients, the d_t-formulas are used in the equation above.

- Cepstral liftering. The high order cepstral coefficients are often numerically quite small, and thus the variances of the different order coefficients would be from a wide range. This is not a recognition issue, but when displaying purposes it is convenient to rescale the cepstral coefficients to have similar magnitudes. This is done by cepstral liftering according to the equation 3.3, where c_n and c_n' are the original and the liftered coefficient, n is the cepstral order, and L is a liftering constant.

$$c_n' = (1 + \frac{L}{2} \sin \frac{\pi n}{L}) c_n \tag{3.3}$$

3.3.2 Mel Frequency Cepstral Coefficients

Mel Frequency Cepstral Coefficients (MFCC's) is the standard front end used in speech recognition. It provides good performance under clean conditions but is not very robust to noise.

When calculating MFCC's, the spectrum of the speech signal is divided into channels by applying a set of band-pass filters to the signal. A computationally efficient way to do this is to operate directly on the DTFT of the signal with a triangular filter bank directly in the frequency domain (see figure 3.2). The triangles of the filter bank overlap partially, and they are broader at high than at low frequencies. This provides better resolution at low frequencies where the most important first and

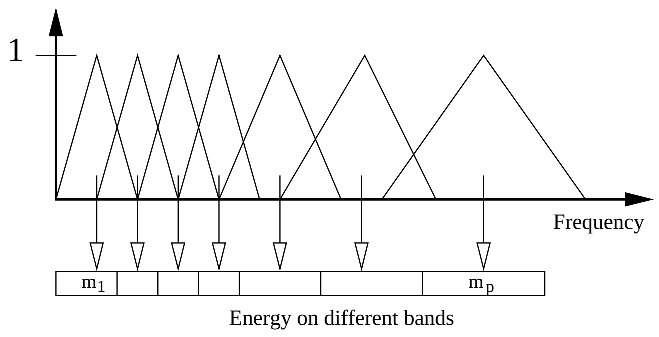

Figure 3.2: The structure of the mel filter bank. The separate m-terms are components of the mel spectrum.

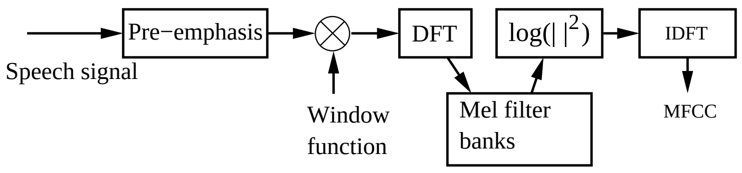

Figure 3.3: Computation of the mel cepstral coefficients.

second formants are located. The triangles need to be tuned carefully to perform mel-warping. If they are located differently, the front end might still produce reasonably good results, even though the resulting scale would be different from the mel scale.

The outputs of the filter bank form the mel-spectrum. The final cepstrum coefficients are obtained by applying the inverse Fourier transform on the logarithm of the absolute values of the mel-spectrum squared. In this case, the inverse Fourier transform is equivalent to the discrete cosine transform (DCT), which is a computationally lightweight operation. Additionally, the DCT produces values that are highly uncorrelated, making the production of speech models easy [42]. A summary of the calculation of the MFCC is depicted in figure 3.3.

3.3.3 Bark Frequency Cepstral Coefficients

The Bark Frequency Cepstral Coefficients (BFCC) method is based on frequency warped linear prediction (WLP) [43, 44]. WLP has previously proved suitable for many applications in speech processing, and it was introduced as a front end for speech recognition in [45]. It has also been compared to the MFCC and other front ends in terms of both performance and computational load[46].

WLP is identical to normal linear prediction except for frequency warping, which is achieved by a chain of all-pass filters as seen in figure 3.4 where the unit delays of a regular LP filter are replaced with second order all-pass filters with a warping parameter λ. With a proper choice of λ, the frequency warping closely approximates the Bark scale. The value of λ producing this warping depends on the sampling frequency.

A more detailed description of WLP is presented in [47] and more information about

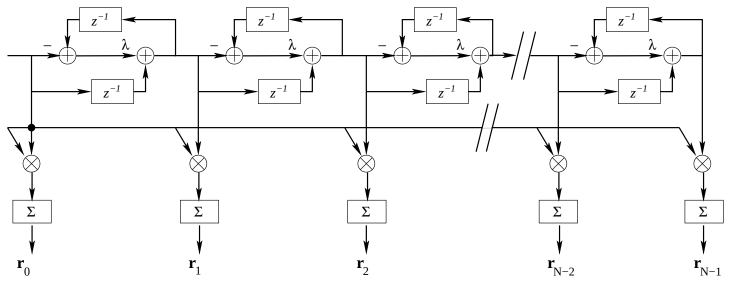

Figure 3.4: The all-pass chain used for calculating the warped autocorrelation sequence.

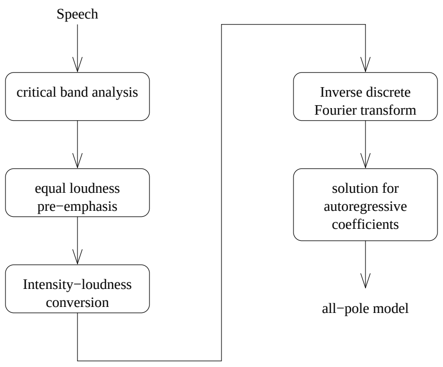
Speech
critical band analysis
equal loudness pre−emphasis
Intensity−loudness conversion
Inverse discrete Fourier transform
solution for autoregressive coefficients
all−pole model

Figure 3.5: Calculation of the perceptual linear prediction.

its application to speech recognition is available in [46].

3.3.4 Perceptual Linear Prediction Coefficients

PLP was introduced by Hermansky in 1989 [48]. A picture about the calculation of PLP is presented in figure 3.5. It is somewhat more sophisticated than the front ends described above. It uses the Bark scale and approximates three individual perceptual aspects: the critical-band resolution curves, the equal-loudness curve, and the intensity-loudness power-law relation. Additionally, autoregressive modeling is applied to smooth out a certain amount of detail from the auditory spectrum.

PLP coefficients are calculated as follows:

- After windowing, a short term power spectrum is calculated for each frame.
- The power spectrum is warped into the Bark-scale.
- The warped power spectrum is convolved with the power spectra of the critical band filter. This simulates the frequency resolution of the ear which is approximately constant on the Bark scale.
- The non-equal perception of loudness at different frequencies is compensated by preemphasis by an equal-loudness curve.
- The perceived loudness is approxmiated by the cube root of the intensity.

- The IDFT is applied to get the equivalent of the autocorrelation function, and a LP model is fitted to it.
- The LP coefficients are transformed into cepstral coefficients.

HTK supports a variation of PLP coefficients, based on mel filter banks and called MF-PLP (mel frequency PLP). They are formed as follows:

1. The mel filter bank coefficients are calculated as in the MFCC front end.
2. They are weighted by an equal-loudness curve and then compressed by taking the cubic root.
3. From the resulting auditory spectrum LP coefficients are estimated which are then converted to cepstral coefficients.

3.4 Recognition units and types

When building a speech recognizer, an important decision one has to initially make is to choose the recognition unit. This unit can be of different lengths, and each choice has its own benefits and disadvantages. The longer the unit is the more accurately it will models the effects of context-dependency, but more training data will be required. The unit should be consistent and trainable: different instances of the same unit should have similarities, and there should be enough training examples to reliably estimate the model parameters. The following discussion is based on [49].

3.4.1 Words

Words are the actual unit we eventually want to recognize. They are thus a natural choice for the speech recognition unit. They inherently take context-dependency issues into account, since phonemes inside a word are very likely the same in each utterance. The obvious problem with word models is the large number of different words and the problem of too little training data. This makes word models unusable for large vocabulary recognition.

Sometimes word models are used together with other kinds of models. For example, short function words may have their own models implemented within a phoneme recognizer.

3.4.2 Phonemes

To facilitate larger vocabularies than what are supported by word models one has to find a way to share information between different models. In other words, one has to take subword units into use. The most obvious choice is the phoneme.

The number of phonemes is moderate, ranging between 15 to 50 in most languages. Therefore, it is very easy to get enough training examples for each phoneme. However, the obvious problem is that the context of the phonemes is ignored. Thus, while word models lack generality, phoneme models overgeneralize.

3.4.3 Multiphone units

Larger units of speech can be used as recognition units to model the coarticulatory effects, for example, syllables or demisyllables. They solve most of the problems involving context-dependency leaving the middle phonemes of the unit out of the effect of the neighboring units. However, the edges of these units undergo some coarticulatory variation. Furthermore, there may not be enough data to train rare units.

3.4.4 Explicit transition modeling

Biphones model pairs of phonemes without the use of stationary phonemes. They are used for explicit transition modeling. Another approach is to use stationary phoneme models and create explicit models for the transitions. These approaches suffer from the problem of the large number of different models: with N phonemes there may be up to N^2 transitions.

3.4.5 Word-dependent phonemes

Word-dependent phonemes are a compromise between word modeling and phone modeling. Phoneme models are trained for each word separately, but if a specific word phoneme is poorly trained, the parameters can be interpolated from the phoneme models in other words. Therefore, not all words have to be present in training, and new words may be added later.

3.4.6 Diphones, triphones and quinphones

As mentioned earlier, context-dependent phoneme models are specialized instances of the corresponding context-independent phoneme models. They are often poorly trained because of their large number, and therefore some generalization (clustering) is necessary. With sufficient training data, context-dependent phoneme models seem to be the most successful unit in speech recognition. They are discussed in more detail in chapter 4.

3.5 Basic structure of Hidden Markov Models

Despite their many weaknesses, Hidden Markov Models (HMM's) are the most widely used technique in modern speech recognition systems. This is due to the fact that a great deal of effort has been devoted in research during the 1980's and 1990's, making it very challenging for alternative methods to get even close to their performance level with moderate investments.

Markov models were introduced by Andrei A. Markov and were initially used for a linguistic purpose, namely modeling letter sequences in Russian literature. Later on, they became a general statistical tool.

Markov models are finite state automatons with probabilities attached to the transitions. The following state is only dependent on the previous state.

Traditional Markov models can be considered as 'visible', as one always knows the state of the machine. For example, in the case of modeling letter strings, each state would always represent a single letter.

However, in hidden Markov models the exact state sequence that the model passes through is not known, but rather a probabilistic function of it.

Hidden Markov models can be built for parts-of-speech differing in size. Usually, simple, small-vocabulary recognizers use word models and larger-vocabulary recognizers use phoneme models.

3.5.1 Definition of HMM's

The Hidden Markov Model is a finite set of states, each of which is associated with a probability distribution. Transitions among the states are governed by a set of probabilities called transition probabilities. In a particular state an outcome or observation can be generated, according to the associated probability distribution.

For defining HMM's, the following elements are needed:

- The number of states, N.
- The number of elements in the observation alphabet, M. The alphabet can also be infinite (continuous).
- Transition probabilities a_{ij} between the states. These are usually presented in a transition matrix.
- A probability distribution associated with each state: $B = b_j(k)$. For continuous probabilities, the probability density function can be approximated by a sum of Gaussian mixtures ($\mathcal{N}$), each having their own weighting coefficients c_{jm}, mean vector $\boldsymbol{\mu_{jm}}$, and variance vector $\mathbf{\Sigma_{jm}}$. With these defined, the probability of an observation vector $\mathbf{o_t}$ can be calculated according to the equation 3.4.

$$b_j(k) = \sum_{m=1}^{M} c_{jm}\mathcal{N}(\boldsymbol{\mu_{jm}}, \mathbf{\Sigma_{jm}}, \mathbf{o_t}) \tag{3.4}$$

An example of a three-emitting-state left-to-right model is shown in figure 3.6. In addition to the emitting states, this model has non-emitting enter and exit states. This kind of structure is used by the HTK Hidden Markov Model Toolkit to facilitate the construction of composite models [50].

In this example, each emitting state has transitions to itself and the next state. This simple left-to-right structure is quite common in speech recognition. Sometimes transitions for skipping a state are used and sometimes even backward transitions may exist.

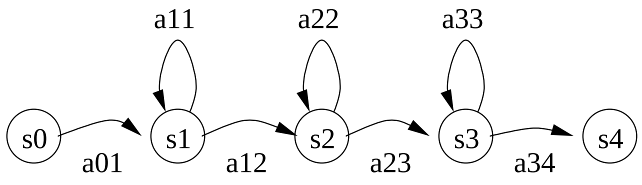

Figure 3.6: An example of a hidden Markov model.

The mathematical problems of HMM's are quite complicated and have been covered in several good papers before, e.g. [51, 52], so they will not be covered in detail here. In short, the three main problems that have to be solved for utilizing the models in speech recognition are:

1. *The evaluation problem*
   Given a model and a sequence of observations, one needs to compute the probability that the sequence was produced by the model. This can be also seen as the problem to score the match between the observations and the model. For this problem an exact solution exists and can be efficiently calculated by using the *forward-backward algorithm* [28, 29].

2. *The estimation problem*
   Given an observation sequence or a set of sequences, this problem involves finding the parameter values that specify a model most likely to produce the given sequence. This problem is involved in speech recognition in the training phase, and is solved iteratively using the *Baum-Welch algorithm* [27, 28, 29, 30, 31].

   Two approaches can be used in training, maximum likelihood estimation (MLE) and maximum mutual information estimation (MMIE) [53]. The goal in MLE is to maximize the probability of the model of generating the training sequences, while MMIE-training tries to maximize the ability of the model to discriminate between models of different speech classes (e.g. phonemes).

3. *The decoding problem*
   The third problem involves finding the most likely state sequence for a given observation sequence. There are different search algorithms for this, for example, the *beam search algorithm.*

The HMM concept has several weaknesses. As its name suggests, the Markov assumption, i.e., the probability of being in a given state at time t depends only on the state at time $t-1$, is used. This is definitely not true for speech signals. Another assumption not true for speech signals is that successive observations should be independent. [51]

On a more practical level, HMM's are poor at modeling the temporal aspects of speech. Additionally, the number of parameters needed for recognition can grow large especially when context-dependent models are to be used. This in turn results in the need for massive amounts of training data so that it is hard to collect this material even for major languages like English, not to mention smaller languages like Finnish.

3.6 Language models and lexicons

An ASR system that only utilizes acoustic features ignores a lot of linguistical information involved in the spoken text. Even though perfect modeling of linguistical information is not possible (e.g. speakers do not strictly follow the grammatics of a language), using statistical *language models* often radically improves the performance of an ASR system.

Lexicon is a list of allowed words in a task. The larger the lexicon is and the more there are words that sound similar the harder the recognition task is.

Language models are used to restrict the possible word combinations based on statistical information collected from large text material. The most common approach is using stochastic descriptions of text usually involving the likelihoods of the local sequences of one to three consecutive words in training texts. Given a history of prior words in a sentence and based on the statistical model the number of words that need to be considered is lower than the size of the vocabulary.

For successful language modeling the text on which the language model is based on should be about similar topic as the data that is going to be recognized. Language features, such as word inflection, also affects the success of language modeling.

Chapter 4

Context-dependency in phonemic speech recognition

Monophone models, i.e. context-independent models, assume that a phoneme in any context is equivalent to the same phoneme in any other context. This assumption is fundamentally wrong since the articulators do not move from one position to another immediately in most phoneme transitions. The transition duration is variable and depends on the phonemes. For example, for /v/ and /j/ the transitions are very long but from stop consonants to vowels the transition is significantly shorter but by no means discrete. Thus, neighboring phonemes are bound to have an effect on the examined phoneme. It seems only natural that these *coarticulatory effects* caused by context-dependency should be taken into account.

In this work, the terms *precontext* and *postcontext* are used to indicate the preceding and following neighbor of a phoneme. Figure 4.1 illustrates these terms for the triphone `i-t+aa`[1].

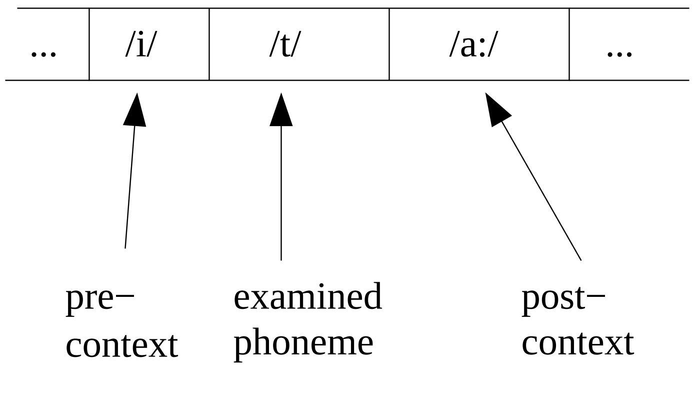

Figure 4.1: The definition of pre- and postcontext for the triphone `i-t+aa`.

One way of dealing with the dependency is using word models. They are perfectly suitable for small vocabulary tasks and have been shown to be more accurate than phoneme models [54]. However, for large or infinite vocabulary recognizers they are not feasible. Instead, subword models are likely to produce better results.

[1]In HTK, triphones are named as `a-b+c`, where `b` is the phoneme itself and `a` and `c` are the left and right contexts. This notation is also used in this thesis.

In the past, a natural shift occured from word models to syllable models. They were proposed already in the mid 70's by Fujimura [55]. It should be noted, that the early syllable recognizers were non-HMM based. Also, demi-syllable and biphone models (a unit consisting of two consecutive phonemes) have been used.

The number of legal syllables in a language is limited, for example, in native Finnish words there are about 3300 different syllables [56], while in English, for example, there are about 10000 syllables [57]. This is true even though there are many more different words in Finnish than in English due to the Finnish morphology. Since phonemes are abstract superclasses of phones, classification can be performed with different precision. This causes some variation in the number of phonemes. Syllable models have been popular especially in Mandarin, Cantonese, and Japanese recognizers but they are suitable for other languages as well.

Diphones (phoneme units dependent on either pre- or postcontext) were introduced in the late 1970's and early 1980's [58, 59] for vocoder and acoustic processor use. A few years after that, Schwartz et al. proposed the use of triphones in speech recognition [57].

This work concentrates on modeling triphones. This selection was made based on the fact that the most important coarticulatory effects on a phoneme are due to its immediate neighbors on either side. Furthermore, triphones are commonly used in large vocabulary recognition in other languages, but not so much in Finnish.

4.1 Fundamental mechanism of triphones

Schwartz [57] noted that all the subword units longer than phonemes (biphones, syllables, etc.) applied as units to speech recognition were merely trying to model the coarticulatory effects between phonemes and that there was nothing special about the units themselves. This motivated him to return to modeling phonemes. Only this time they were made context-dependent, which led to the introduction of the concept of triphones – a model for a single phoneme conditioned on its preceding and following neighbor phoneme. This is the very idea of triphones used in any modern recognizer, and in [60] the actual results of the first recognizer utilizing triphones are presented.

Despite its name, a triphone is simply a model of a single phoneme conditioned on its immediate neighbors, and not a structure of three phonemes. Similarly, a diphone is a model of a phoneme conditioned on either its left or right phoneme and a quinphone is conditioned on two neighboring phonemes on either side.

Context-dependent models can be constructed in two ways: they can either be word-internal or cross-word. When constructing word-internal models, context beyond the word borders are not considered. On the other hand, for cross-word triphones the phonemes at the end or beginning of neighboring words are considered to affect the phoneme. Naturally, the number of cross-word triphones is considerably higher than the number of word-internal triphones. For example, in the book used as data for this work, there were about 7100 different word-internal and 9600 different cross-word triphones.

The cross-word triphones are a natural choice for continuous speech recognition, since there are seldomly clear pauses between words in fluent speech. Actually, a stop consonant might introduce a longer pause than a word break, see figure 2.6 for an example. The problem, again, is the increasing number of models and the shortage of data for training them.

Out of the 9600 different cross-word triphones 4065 (42 percent) had less than five occurrences in the book. That is far less than training models requires: the triphone models used in this work consisted of three states, each having a 39-element mean and variance vector. Additionally, in the transition matrices, there are six free parameters in the allowed transitions to the same and the following states. This adds up in $3 \times 2 \times 39 + 6 = 240$ free parameters per model.

4.2 Clustering mechanisms for context-dependent HMM's

A set of full triphones is excessive in two respects. First, in all practical cases there is not enough training material for many of the triphones. Second, despite coarticulatory effects some triphones are quite similar, and had better be covered by the same model.

The training data problem can be solved by reducing the number of parameters. This can be done in several ways. The number of models, or the number of states can be reduced by state or model clustering. Another approach is tying parameters inside the states or models, that is, forcing them to be equal for two different states (means and variances) or models (transition matrices).

A straightforward way of reducing the number of parameters in a triphone model set could be to tie all the parameters of all models' center states. The assumption that the center of each triphone (for the same phoneme) is similar could lead to this kind of an approach. However, clustering mechanisms lead to better results than this kind of direct tying. In this section some clustering algorithms are presented.

4.2.1 Data-driven (bottom-up) clustering

In the data-driven clustering algorithm each state is initially placed in a cluster of its own. Then two clusters forming the smallest cluster are merged together. This merging is repeated until the smallest cluster that would be possible to form by merging any two clusters would be larger than some predefined limit.

The size of the cluster is defined as the longest distance between any two members of the cluster. The metric is defined as the Euclidean distance between the means of the two states.

A limitation with this approach is its inability to deal with unseen triphones (triphones not present in the training data), which are bound to occur in large vocabulary recognition with cross-word triphones. However, diphone and monophone models are normally used to deal with this problem.

This algorithm is bottom-up since it starts with individual states and ends with clusters. An illustration of the algorithm is depicted in figure 4.2. This clustering algorithm was introduced in [49].

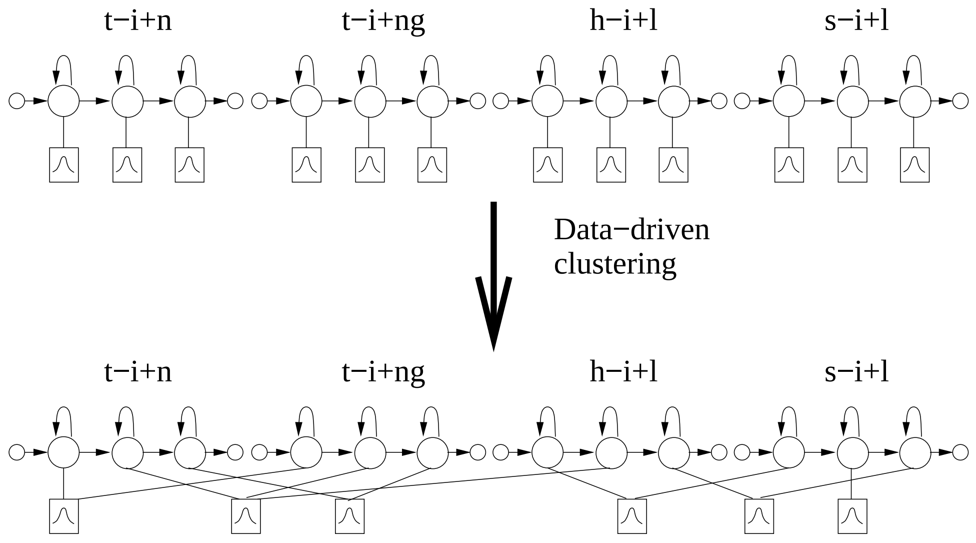

Figure 4.2: Data-driven state clustering for some triphones of the phoneme /i/.

4.2.2 Decision-tree (top-down) clustering

Another approach for clustering states is based on binary decision trees [61]. In addition to states, and unlike data-driven clustering described above, this algorithm can be used to cluster entire models as well.

A set of questions regarding phonemes context is needed for splitting the clusters during the process. A typical question could be: "Is the left context of this phoneme either an /a/ or an /o/?" There is no actual limit for the number of questions (the number of possible phoneme subsets gives a theoretical limit), and too many questions do not usually cause any harm.

A short description of the algorithm follows: initially all states/models in a given list are placed in the root node of a tree. The nodes are iteratively split by selecting a question. Depending on the answer, states/models in the state are placed either in the right or left child node of the current node. This is done iteratively until the log likelihood increase of the states/models in the tree node obtained by the best question is below a predefined limit. At this point, all the parameters in the state/model are tied. For an illustration, see figure 4.3.

The question used is chosen to maximize the likelihood of the training data given the final set of model/state tyings. When the node is split, the likelihood of its child nodes is bound to increase since the number of parameters to describe the same data increases.

The log likelihood can be calculated based on the statistics (means, variances, and state occupation counts) gathered from the training data, and based on that information the best question for each node can be chosen.

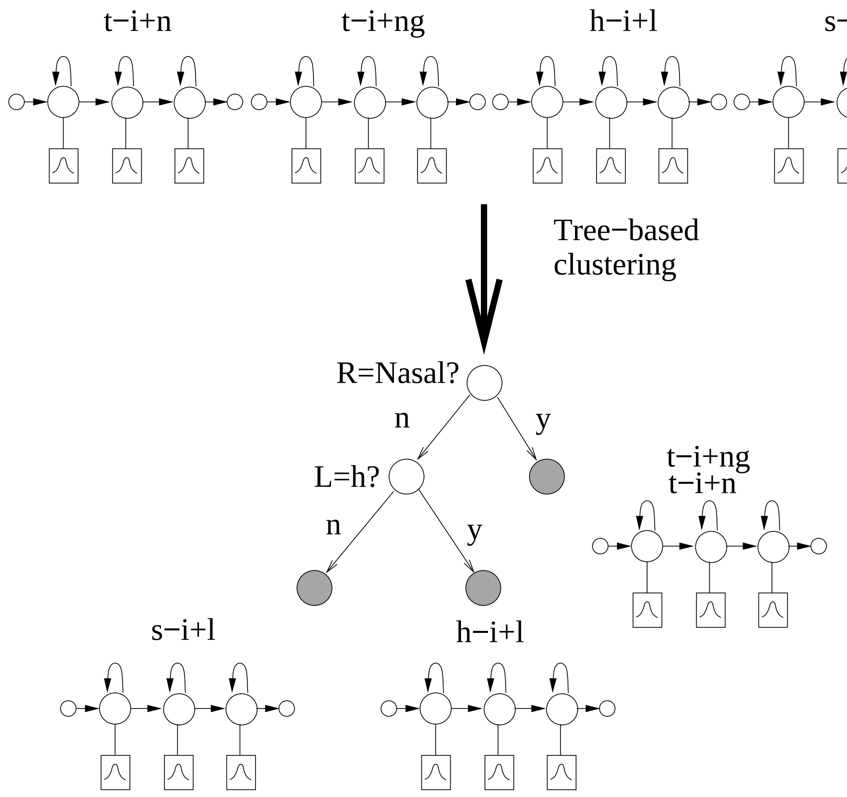

Figure 4.3: Part of the tree-based model clustering process of /i/-triphones. Leaf nodes are gray, and they form the final clusters.

Models for unseen triphones can be formed by use of the trees generated in the clustering processes. One simply follows the questions from the root of the tree, and once a leaf node is reached, that state/model is used for the unseen triphone in question.

4.2.3 Classification based on articulatory facts

A simple way to classify triphone models is to use decisions made a priori about the context as criteria for classifications. Basically, one decides classifications for phonemes and then classifies those triphones with contexts from the same phoneme classes to belong to the same broad class triphone (or cluster). In principle, the phoneme classes could be formed arbitrarily but intuitively it would be beneficial if there was some similarity between the members of a phoneme class. Therefore, natural choices are based on articulatory facts. This kind of approach has been suggested in [62] and [63].

In this work two different classifications were used: one based on the type of the phoneme (short: **ToP**) and the other on the place of articulation (short: **PoA**). The ToP classification for Finnish includes six classes and the PoA classification eight classes. The phonemes for each class are presented in tables 4.1 and 4.2. Long variant of each phoneme always belongs to the same class as the short one.

For example, using the ToP classification, triphones `e-i+k, ö-i+t`, and `e-i+d` would all belong to the same broad class of triphones, namely `FW-i-ST`.

Class name	Short	Phonemes
Back vowels	BV	/a, o, u/
Frontal vowels	FV	/e, i, y, ä, ö/
Stops	ST	/b, d, g, k, t, p/
Fricatives	FR	/f, h, s/
Semi-vowels	SV	/j, v/
Nasals	NA	/m, n, ŋ/
Laterals	LA	/l/
Tremulants	TR	/r/

Table 4.1: Classes for the ToP classification.

Class name	Short	Phonemes
Back vowels	BV	/a, o, u/
Frontal vowels	FV	/e, i, y, ä, ö/
Labials	LA	/b, f, m, p, v/
Dentals	DE	/d, l, n, r, s, t/
Palatals and velars	PV	/g, j, k, ŋ/
H	H	/h/

Table 4.2: Classes for the PoA classification.

Even when this kind of classification is used it is probable that for some broad classes there would be too few training examples. In this case, tree-based clustering similar to the one described in the previous subsection could be used.

This classification method is less flexible than the two described above, but it produces predefined clusters that could potentially be utilized in the higher levels of the recognizer.

4.3 Examples of large vocabulary ASR systems utilizing context-dependency

This section describes two high-end large vocabulary recognizers. Both of them are English recognizers and rely on the HMM paradigm. Sphinx was developed at Carnegie Mellon University (CMU) in Pittsburgh, USA, and the CU-HTK transcription system at Cambridge University in the UK.

Please note that the *word error rate* used in this section is not comparable with the *phoneme error rate* for the experiments of this thesis discussed in chapter 7. When calculating word error rates the most probable words are calculated based on the phoneme string, lexicon, and language model. This tends to hide the performance of the acoustical models and is therefore not used in the tests reported in this work. Furthermore, the data in the recognition tests described here was different, and the recognizers are speaker-independent so the recognition problem is of a completely different nature.

4.3.1 SPHINX

One of the most famous speaker independent continuous speech large vocabulary recognizers is SPHINX. When the project started, all of the recognizers developed had constraints either in terms of speaker dependency, speech continuousness, or vocabulary size. The goal of the SPHINX project was to overcome all of these obstacles.

The first version, SPHINX-1 was developed in the end of the 1980's [64]. Its front end was based on LPC coefficients. Function-word dependent phone models and generalized triphone models were used for speech modeling. The triphone generalizing algorithm is essentially the same as the one described in section 4.2.1. For decoding, SPHINX-1 used a single pass beam search, and the results for the 997-word vocabulary task with a bigram language model were about 96 per cent.

The SPHINX-2 recognizer from 1994 [65] differs from SPHINX-1 mainly in the decoding algorithm. Instead of a one pass beam search, a four pass approach is used[66]: word lattices are formed by a forward (1) and a backward search (2). Then, a best-first search is carried out to produce an N-best list of words (3), and finally, in the rescoring phase, the N-best list is processed.

The most recent version of the CMU recognicer is SPHINX-3 from 1996 [67]. It was included in the 1996 DARPA (Hub-4) evaluation for broadcast news recognition.

The data used in the evaluation was quite varied, including both spontaneous and read speech, speech over wide- and narrow-band channels, and non-native speakers.

The system uses senonically clustered, fully-continuous, five-state triphone models. Senonically clustered means that the observation distributions are shared between states in similar models [68]. The models were trained on the Wall Street Journal corpus and then adapted with the Broadcast News corpus. There were two sets of models, one for wide-bandwidth data and one for narrow-bandwidth data. There were altogether 6000 states in both sets, each consisting of sixteen Gaussian mixtures. There were also models for several types of filled pauses (noise).

The lexical model was based on the Broadcast News corpus and included the 51000 most common words and some 350 multi-word phrases or acronyms. The vocabulary size was a trade-off between having out-of-vocabulary words and having new confusable words in the vocabulary.

Two different language models were used, one to guide the decoder in the recognition process, and a larger one for rescoring the N-best output hypotheses. The first language model was a quite standard trigram model which included support for consecutive utterances being in the same acoustical segment. The second language model was a 7-gram model.

The recognition system was composed of seven stages:

1. *Segmentation, classification, and clustering.* This stage included detection of the bandwidth (for model set selection), sub-segmenting long utterances into shorter ones to produce speech segments that the decoder could handle, and silence detection.

2. *Initial-pass recognition.* Here, the models selected in the previous phase were utilized with a straight-forward continuous-density Viterbi beam search. In addition to a hypothesis containing words and their times, this recognition produced a word lattice for each sub-segment.

3. *Initial-pass best-path search.* The lattices generated above were searched for the global best path according to the trigram grammar.

4. *Acoustic adaptation.* The HMM means were adapted using Maximum Likelihood Linear Regression (MLLR) based on the best path found in the previous stage.

5. *Second-pass recognition.* The adapted models were used to recognize the sub-segments again, producing new word lattices.

6. *Second-pass best-path search.* The best path through the lattices generated at the previous stage was searched. At this point the N-best lists were also generated.

7. *N-best rescoring.* The N-best lists were rescored, and the final recognition result was produced.

The final word error rate for the Hub 4 evaluation was 34.9 percent for data, where speech condition changes were marked and 35.9 for data where they were not.

4.3.2 CU-HTK March 2000 Hub 5E transcription system

The CU-HTK recognizer [69, 70], developed over several years [71, 72], was used in the NIST Hub 5E evaluation for conversational telephone speech recognition systems. This task is particularly hard due to limited bandwidth, distorted audio channels, cross-talk, and highly variable speaking styles. Due to these facts, the word error rate is 25.4 %, which is a very good result for this difficult speaker-independent recognition problems.

The system uses thirteen perceptual linear prediction cepstral coefficients derived from a mel-scale filter bank (MF-PLP) covering the range from 125 Hz to 3.8 kHz, including the C_0, and their Δ- and $\Delta\Delta$-coefficients (see section 3.3.4). Cepstral mean subtraction and variance normalization are performed for each conversation side, and vocal tract length normalization (VTLN) was applied.

The training and testing data is from two corpora: Switchboard 1 and Call Home English. There were a total of 265 hours of training data, but smaller subsets of 68 and 18 hours were used on certain stages of training.

Traditionally, HMM's are trained using Maximum Likelihood Estimation (MLE). This means that the probability of each training observation is maximized by adjusting the model parameters regardless of the other training observations. CU-HTK applies another approach: Maximum Mutual Information Estimation (MMIE). MMIE takes into account possible competing words and while maximizing the probability of the correct transcription the probabilities of the incorrect transcriptions are minimized.

The main issue with MMIE training is generalization to test data. For this reason, it is very important that the training data is representative. In the CU-HTK recognizer, the generalization is enhanced by broadening the training data posterior distribution by the use of acoustic scaling and a weakened language model [70]. The MMIE approach leads to a decrease of 2.6–2.7 % in word error rate compared to MLE training.

In soft tying [73], Gaussians from a particular state can be also used in other mixture distributions with similar acoustics. The recognizer uses a simplified form of soft tying: For each state, a single Gaussian version is created, and with its help the two nearest states are found. Then all the mixtures in the three states are used in the state in question. This allows the use of three times as many Gaussians per state without an increase in parameter space. With this, reduction of 0.3 % for triphones and 0.5 % for quiphones in word error rate is achieved.

A further 1.4–1.7 % improvement is achieved by the use of pronunciation probabilities: three different pronunciations were included in the dictionary with different kinds of silence in the end of the words. The probability for each pronunciation is estimated from the training data.

Another solution for increasing performance is the use of a side-dependent block-full variance transformation [74]. It is essentially the same as having a speaker-dependent global semi-tied block-full covariance matrix. Furthermore, confusion networks [75] are used to estimate word posterior probabilities. They are also used for the combination of results gained by the triphone and quinphone recognition systems.

The decoding (recognition) phase takes place in several phases. In phase 1, the basic gender independent, non-VTLN MLE trained triphones with a trigram language model are used to generate an initial transcription. The output of this phase is used for gender detection and VTLN warp factor computation. For all further stages the training and test data is warped using the VTLN parameters.

In phase two, gender independent MMIE trained triphones were used to generate transcriptions for unsupervised test-set MLLR adaptation with a 4-gram language model. In the third stage word lattices were generated using the adapted gender independent MMIE triphones and a bigram language model. These lattices were then expanded to contain language model probabilities of the full 4-gram language model.

Further phases rescored these lattices in two branches: a) gender independent MMIE trained models and b) gender dependent soft-tied MLE models. The fourth phase used triphones and the fifth quinphones. For the MMIE branch two MLLR transform were run (phase 6a). Finally, the system word output was used by using confusion network combination based on confusion networks from phases 4a, 4b, 6a, and 5b.

By this approach, the CU-HTK system achieves a word error rate of 25.4 % for the test data, while the best single model set gained from the phase 6a, had an error rate of 26.5 %.

Part II

Experiments

Chapter 5

Specification of the recognition environment

A speaker-dependent phoneme recognition system was built in order to compare the different clustering approaches. The implementation of both the front end and the HMM's in this thesis are quite standard.

5.1 Data

For the training and test data this thesis used a talking book, *Syntymättömien sukupolvien Eurooppa* by *Eero Paloheimo*. The book was created for the visually impaired and it had been read to tape by a female speaker. The reader had tried to be as neutral as possible avoiding any emotional emphasis since that is what the target group of the book desires – the visually impaired want to experience the sentiments of the book themselves, just as any sighted reader would. The words are pronounced quite clearly and much more accurately than in normal conversational speech. Still, not all the phonemes in all words are audible, especially at the end of words.

The book is about the future of Europe and the text is very fact-based. It is written completely in the standard Finnish language and has no dialects involved whatsoever, making it favorable for speech recognition. On the other hand, the book contains quite a few foreign names and terms.

Originally, the book was recorded on compact cassettes. It was then digitized with a sampling frequency of 22 kHz. The sound files were segmented using the Hidden Markov Models from an earlier recognizer at the Neural Networks Research Center at the Helsinki University of Technology [3], and at this point the sound files were sampled down to 16 kHz. Based on the segmentation information, the large sound files containing altogether 11 hours 10 minutes speech were divided into 4283 smaller files each containing one sentence. A histogram of sentence lengths is shown in figure 5.1. The lengths vary from one to sixty-six words. The shortest ones are section and chapter headings.

The phoneme set of the segmented data was the same as used in the recognition

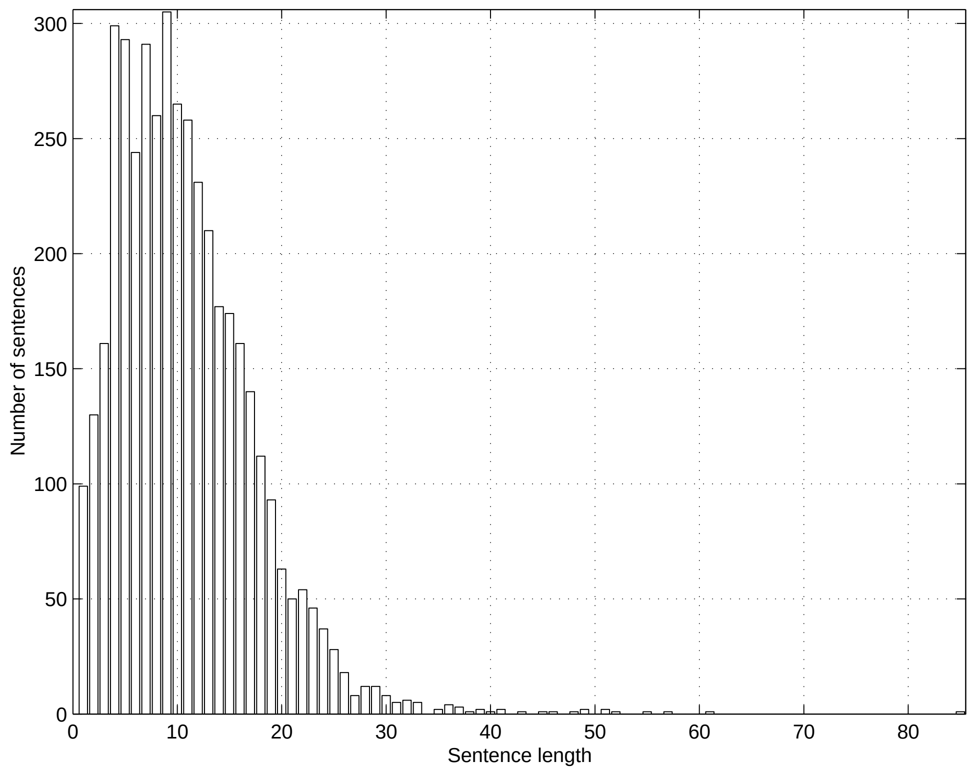

Figure 5.1: Histogram of sentence lengths in the data.

test, except that it included few instances of double consonants that are very rare in Finnish (/f:, v:/, and /h:/). These double phonemes were converted to their single counterparts because of the low number of training examples.

During the segmentation process the text, including abbreviations and numerals, was converted into phoneme strings using Finnish pronunciation rules and a program for expanding abbreviations, numerals and other structures [76]. This resulted in transcription errors in foreign proper names, dates and other complicated structures involving numerals. Additionally, the spoken and written material did not match perfectly and there were some inaccuracies caused by the segmentation process itself. As a result, some of the files, but only about one percent of the material, were found unusable for training. Seventy percent of the sentences were used for training and the rest for testing.

5.2 Hardware and software

The recognition system is based on the Hidden Markov Model Toolkit (HTK) version 3.1 [50]. Some modifications were made to the source code (e.g., BFCC calculations, transcription processing for error analysis), but mainly the HTK features were used in their standard form. Numerous helpful Perl and shell scripts were written to automate the different phases of training and recognition as well as to assist analyzing the results.

The recognition tests were run on a modern standard PC (1 GHz AMD Athlon processor, 256 MB of RAM) running RedHat Linux 7.2, kernel version 2.4.12.

Chapter 6

Building the recognizer

This chapter describes the steps used in feature extraction, HMM building and training as well as testing. The steps presented here are those required by the HTK software package, and for all stages, corresponding HTK commands are shown. For clarity reasons these steps are showd as simplified forms of the actual commands used in this work. In the work, debugging options were used for increased verbosity of the tools, and the files resided in different directories making the actual commands very long. However, the functionality of the commands presented here is preserved.

All of the commands have many command line options, some of which are the same for all the tools. These are explained in table 6.1 and the options specific to each tool are explained when the tool is described. However, not all of the options are described here and further information may be found by consulting the HTK manual [77].

6.1 Feature extraction

The first task in building the recognition system was to calculate the cepstral coefficient files from the sound files. This process involves many different parameters and are summarized in table 6.2. Refer to section 3.3 for information about the configuration options.

Feature extraction is accomplished by using the `HCopy` tool as follows:

```
HCopy -C hcopy.conf -S script.scp
```

Here, `hcopy.conf` defines the parameters to be used and script.scp contains simply a list of waveform files to process, one file per line. An example configuration file, used for standard MFCC calculation, is given below:

```
TARGETRATE = 100000.0
SAVECOMPRESSED = F
SAVEWITHCRC = F
```

Command line option	Usage
-I <*file.mlf*>	<*file.mlf*> contains the (possibly segmented) labeling information for the files used in training/testing.
-i <*file.mlf*>	Output the recognized/edited labeling to <*file.mlf*>.
-C <*file.conf*>	Read configuration information from <*file.conf*>.
-M <*dir*>	Write the trained models to directory <*dir*>.
-H <*macrofile*>	Read HMM information from file <*macrofile*>.
-d <*dir*>	Read HMM information from separate files in directory <*dir*>.
-S <*file.scp*>	Instead of specifying the speech files used in training/recognition, read a list of them from the file <*file.scp*>.
-B	Save the models in binary instead of text files.

Table 6.1: Command line options common for all HTK tools.

```
WINDOWSIZE = 250000.0
USEHAMMING = T
PREEMCOEF = 0.97
ZMEANSOURCE = T

NUMCEPS = 12
NUMCHANS = 26
CEPLIFTER = 22
TARGETKIND = MFCC_E_Z
SOURCEFORMAT = WAV
```

6.2 Phoneme models

As the first stage of recognizer building, simple monophone models were built. Their creation process is explained in the following subsections.

6.2.1 Creating prototype models

At first, prototype models have to be created. They describe the structure of the models, while the actual values of the coefficients are unimportant. At this stage, the number of states and allowed transitions need to be determined and set. In this work, it was decided to use standard three state left-to-right HMM models. At an early stage of the tests, models with more states for long vowels and less for

	Parameter	value
Common Parameters	Number of Cepstral coeff.	12
	Window size	25 ms
	Window rate	10 ms
	Hamming windowing	True
	Pre-emphasis coeff.	0.97
	Cepstral lifter	22
	Zero mean	True
BFCC	λ	0.646
	WLPC order	18

Table 6.2: Parameters used in feature extraction.

short vowels were tried, when trying to solve a problem where short vowels were intermingled with each other. However, this approach did not have any positive effect on recognition accuracy.

The prototype models are equal for all phonemes and part of the prototype used in this work is shown below.

```
~o <VecSize> 39 <MFCC_E_D_A_Z>
~h "a"
<BeginHMM>
  <NumStates> 5
  <State> 2 <NumMixes> 1
  <Mixture> 1 1.0000
    <Mean> 39
      0.0 0.0 ... (repeated 39 times)
    <Variance> 39
      1.0 1.0 ... (repeated 39 times)
  <State> 3 <NumMixes> 1
  <Mixture> 1 1.0000
    <Mean> 39
      0.0 0.0 ...
    <Variance> 39
      1.0 1.0 ...
  <State> 4 <NumMixes> 1
  <Mixture> 1 1.0000
    <Mean> 39
      0.0 0.0 ...
    <Variance> 39
      1.0 1.0 ...
  <TransP> 5
   0.000e+0   1.000e+0   0.000e+0   0.000e+0   0.000e+0
   0.000e+0   6.000e-1   4.000e-1   0.000e+0   0.000e+0
   0.000e+0   0.000e+0   6.000e-1   4.000e-1   0.000e+0
   0.000e+0   0.000e+0   0.000e+0   6.000e-1   4.000e-1
   0.000e+0   0.000e+0   0.000e+0   0.000e+0   0.000e+0
```

```
<EndHMM>
```

On the first line, the vector length is defined to be 39, and the used parameter kind as MFC-coefficients (MFCC) with the energy term (_E), delta- (_D), and delta delta coefficients. The _Z flag instructs HTK to extract the mean of each frame from the samples.

The line beginning with "`~h`" states the name of the model. After that follows the actual HMM definition. Initially, the number of states is defined (five means three emitting, that is operative states, since HTK uses a structure with an initial and final non-emitting state), and then the structure of each state is defined. The number of mixtures can be defined for each state individually but in this work the number of mixtures has been one for all models.

The essential parts of the state definition are the mean and variance vectors. In the prototype model they are simply initialized with any value. The important thing is that there is a correct number – the number stated on the first line of the prototype – of values.

Finally, the `<TrasP>` section defines the transition matrix. In this example the initial probabilities are set so that the probability of staying in the same state is 60 %, the probability of moving on to the immediately following state is 40 %, and the probability of moving to any other state is 0 %. At this point the actual values are not important – they are estimated during the training phase. Only if a transition is given a zero probability at the prototype phase then the transition will never be allowed during training unless the models are altered manually.

There is no tool in HTK for creating the prototypes – they need to be created by hand using a text editor or a script.

6.2.2 Initializing the prototype models

After the prototype models for each phoneme are created it is time to estimate more reasonable values for the prototype models. Two possible approaches are available in HTK for this. One is to use the tool `HCompV`, which calculates the global mean and variance of a set of training files and then uses these values for all models. This approach is called the *flat start* training scheme, since all the models receive the same values for the parameters.

The other approach used in this work is the use of the tool `HInit`. Its principle depends on the concept of a HMM as a generator of speech vectors – it tries to find out which training data vectors were emitted by each state in the models and then calculates the means and variances for the corresponding states. This is done by repeatedly applying the Viterbi algorithm on the data and updating the parameters until the likelihood of each sentence falls below a defined value or a number of iterations has been reached.

The `HInit` approach was chosen since it utilizes the segmentation information that was available and proved to provide slightly better results than the `HCompv` approach in preliminary tests. The number of iterations was chosen to be ten.

The initialization is made for each phoneme model separately with a command like

```
HInit -i 10 -I labels.mlf -l a -C hinit.conf -M hmm.0 -S train.scp\
   proto/a.
```

Here, the command line option `-i` specifies the maximum number of iterations, the option `-l` specifies the phoneme in question, and the final parameter `proto/a` specifies where the prototype file resides.

6.2.3 Model training

The following step describes model training itself. This is done by using the tools `HRest` and `HERest`. `HRest` works in very similar way to `HInit`, except that `HRest` expects the models to be initialized properly and uses the Baum-Welch algorithm instead of Viterbi training. This involves finding the probability of being in each state at each time frame using the forward-backward algorithm. This probability is then used to form weighted averages for the HMM parameters. Thus, whereas Viterbi training makes a hard decision as to which state each training vector was 'generated' by, Baum-Welch takes a soft decision. This can be helpful when estimating phone-based HMM's since there are no hard boundaries between phones in real speech and using a soft decision may give better results." [77]

The tool for *embedded training*, `HERest`, is different from `HInit` and `HRest` in that it simultaneously updates all of the HMM's in a system using all of the training data. After loading the complete HMM definitions into memory, `HERest` processes each training file in turn. At this stage, the segmentation information in the files is ignored and only the sequence of phonemes is significant.

For each file, `HERest` constructs a composite HMM consisting of all the models corresponding to the phoneme sequence in the labeling of the utterance. The Forward-Backward algorithm is then applied as normal. When all of the training files have been processed, the new parameter estimates are formed from the weighted sums and the updated HMM set is output.

In this work, `HRest` was called first for each phoneme:

```
HRest -I labels.mlf -v 0.001 -l a -C hrest.conf -M hmm.1 -S train.scp\
   hmm.0/a
```

The parameters are as in `HInit` except for `-v`, which sets the minimum variance to 0.001. This is necessary to avoid overtraining.

After running `HRest`, `HERest` was called iteratively seven times:

```
HERest -C herest.conf -v 0.001 -d hmm.1 -M hmm.2 -I labels.mlf \
   -t 250 150 1000 -S train.scp hmmlist
```

The option `-t` specifies the beam width in the backward phase of the Forward-Backward algorithm to 250. If pruning errors occur, the beam width is increased by 150, and the file in question is reprocessed using this new beam. This is repeated until no pruning error occurs or the upper limit for the beam – in this case 1000 – is reached. In the latter case the file is rejected and probably contains labeling errors.

6.2.4 Fixing the silence model

Speech utterances may contain pauses of different lengths and nature. There are longer periods of silence at sentence borders, and shorter inside a sentence, mainly at punctuation marks. Between most words the pause is very short or even nonexistent. In stop consonants there is a short silent section, and sometimes there are glottal stops in different locations of speech. Additionally, some background noise is present and should be handled by some models. Therefore, several kinds of silence models are needed.

In this work, two different silence models are used. The silence model `sil` was until now handled as any other model. Some modifications are made to it so that it will better absorb silences of different lengths, and another short pause model (`sp`) model is added for the word break silences.

For the `sil` model, transitions from the state 2 to state 4 and from state 4 to state 2 (from the first emitting state to the last and back) are added to allow the different states absorb the background noise without creating too many consecutive `sil` observations in the recognition results.

Additionally, a short pause model `sp` is created. It has only one emitting state (state number 2), which is tied (set equal) to the center state of the `sil` model. There is also a transition from the beginning state directly to the end state (from state 1 to state 3).

The `sp` model is useful in that it allows very short silence sections. For example, the period of silence between words in a sentence is very short and sometimes even nonexistent. At this point, the labeling was also changed so that in most places the `sil` labels were replaced by `sp` labels. Only at locations that suggest that there really is silence in the sentence, e.g., at punctuation marks, the `sil` label was preserved.

The `sp` model was created by copying the center state of the `sil` model and adding appropriate definitions for the rest of the model to it. Then, new transitions to the `sil` model were added and state 2 of the `sp` model was tied to state 3 of `sil` by the following `HHEd` script:

```
AT 2 4 0.2 {sil.transP}
AT 4 2 0.2 {sil.transP}
AT 1 3 0.3 {sp.transP}
TI silst {sil.state[3],sp.state[2]}
```

The needed `HHEd` command was

```
HHEd -C hhed.conf -d hmm.3 -M hmm.4 fixsil.hed hmmlist,
```

where `fixsil.hed` contains the lines above.

This silence model fixing was made between the second and third iteration of `HERest`. Therefore, for iterations three to seven the `hmmlist` should contain the short pause model as well.

At this point the monophone models are ready. There are still many possibilities to improve their performance – adding mixtures being the most obvious – but since this thesis studies the use of triphones, no further attention was paid to developing the monophone models.

The monophone models were used for comparison reasons for test data recognition. The recognition procedure is almost identical to the triphone case described in section 6.4.

6.3 Triphone models

In theory, triphone models are clones of monophone models that have been renamed to include the left and right context of the phoneme and retrained with the occurrences of the triphone in question.

As mentioned before, training a full triphone set is not feasible due to the large number of possible triphones and the small number of training examples for many of these in any realistic training data set. Therefore, the triphones need to be clustered.

This section describes the steps required to create a triphone set from the monophones built above as well as different ways of clustering.

6.3.1 Making triphone labels from monophone labels

Initially, monophone label files need to be transformed into triphone form. That is, in the case of decision tree based triphone clustering, we want a phoneme string

```
    sil h ae n sp s a n o i sil e tt ae sp h ...
```

to be transformed into

```
    sil+h sil-h+ae h-ae+n ae-n+s sp n-s+a s-a+n a-n+o n-o+i o-i+sil \
      i-sil+e sil-e+tt e-tt+ae tt-ae+h sp ae-h+...
```

If we want to cluster based on articulatory facts then the context of the triphones will be the broad class of the context phoneme. For example, instead of the triphone `n-s+a` we would have `NA-s+BV`, where `NA` denotes nasal and `BV` denotes a back vowel. See tables 4.1 and 4.2 for the acronyms of different phoneme classes.

As one can see from the example, the two kinds of silence are treated differently. The short pause is ignored when forming triphones. The `sil` label is considered as a normal context to other phonemes. Both of the silence models are in practice context-free, even though the contexts are included in the `sil` labels – all `sil` models are tied later on to one model.

This approach was chosen because the short pauses, being short or even non-existent, do not usually affect the pronunciation of other phonemes, while the longer silence is usually realized in a longer pause in speech. Some recognizers consider the silence models as context-dependent while others do not, and the context-free approach was chosen for this work.

The labels are transformed into triphone format by applying the following `HLEd` script to the label files.

```
NB sp
TC
```

Assuming that this script is saved into a file called `mktri.led`, it is applied to a label file with the command:

```
HLEd -C hled.conf -i new.mlf mktri.led new.mlf
```

This script seems to leave the silences in the very beginning and end without context. These need to be added to the labels by hand or a script so that recognition would work as described in section 6.4.

6.3.2 Making triphone models from monophone models

The next step is to make clones of the phoneme models for each triphone in the training data. The script driven HMM editor `HHEd`does this when given a script file containing only the line “`CL traintrilist`”.

At this point the number of models grows so large that it is no longer convenient to have the models in separate text files in a directory. Instead, HTK supports saving several models in one file and using a binary file format, which saves storage space.

A command for creating one large binary file (which defaults to `newMacros`) is:

```
HHEd -d hmm.9 -B -M hmm.10 mktri.hed hmmlist
```

The models produced this way were trained two times with `HERest`, and at the second pass, an extra `-s stats` option was added – it generates a file of state occupation statistics, that is used in the clustering process as described below.

At this point there exists a model for all triphones present in the training data. Many of them have only a few training examples (42 percent of them have less than five training examples). In order to provide sufficient training material for all models, clustering is required.

6.3.3 Triphone clustering – binary decision tree

In order to perform binary decision tree based triphone clustering (as described in section 4.2.2), the questions have to be generated. In HTK, clustering is accomplished by `HHEd`. A script for clustering is:

```
RO 100 stats

QS "L_Sil" {sil-*}
QS "R_Sil" {*+sil}
QS "L_PallatalOrVelar" {j-*,jj-*,k-*,kk-*,g-*,gg-*,ng-*,ngng-*}
QS "R_PallatalOrVelar" {*+j,*+jj,*+k,*+kk,*+g,*+gg,*+ng,*+ngng}
QS "L_Labial" {p-*,pp-*,b-*,bb-*,m-*,mm-*,f-*,ff-*,v-*,vv-*}
QS "R_Labial" {*+p,*+pp,*+b,*+bb,*+m,*+mm,*+f,*+ff,*+v,*+vv}
...
QS "L_yy" {yy-*}
QS "R_yy" {*+yy}

TB 250 "a" {(a, *-a, *-a+*, a+*)}
TB 250 "aa" {(aa, *-aa, *-aa+*, aa+*)}
...
AU "fulltrilist"
CO "tiedlist2"
```

First, the `stats` file created above is read in, and the outlier threshold is set to 100. Then, the questions created manually or by a script are stated in the `QS` lines. The `TB` lines tell `HHEd` to actually cluster all mono-, di-, and triphone models of the phoneme in question. Clustering goes on until the log likelihood achievable by any node is less than the threshold specified by the argument (in this case: 250).

The binary decision tree clustering mechanism allows building models for unseen triphones based on the clustering trees of seen triphones. To make the recognition process simple (see section 6.4), it is useful to have models for all possible triphones available. The list of all phonemes is given in the file defined by the `AU` (add unseen triphones) command. Finally, the `CO` command defines that a compacted list of models is to be generated into the file `tiedlist2`.

The command needed to actually run the script is as follows:

```
HHEd -H hmm.2/newMacros -M hmm.3 tree.hed triphonelist
```

6.3.4 Triphone clustering based on articulatory facts

When clustering based on articulatory facts, most of the work is done simply in the phase when the monophone labels are transformed into triphone labels. Some of the clustered triphones still have too few training examples and they need to be clustered using the same tree based clustering mechanism described in the previous section. Here, the set of questions consists of the different subsets of the selected broad classes for both left and right contexts, for example, the following question would define the context where on the left there is a fricative, stop or a labial:

```
QS "L_FRKLLA-ques" {FR-*,KL-*,LA-*}
```

Additionally, we need to define which real non-clustered triphone, e.g. `a-n+e`, corresponds to a clustered triphone. Even though initial clustering based on articulatory facts would rename this triphone to `BV-n+FV`, in the second clustering process it might be tied with other triphone models of `n`, and the final physical model could, for example, have the name `ST-n+sil`.

Therefore, we need a new list of triphones for recognition. This is a text file consisting of all logical triphones followed with their physical counterparts on each line. This kind of file was created using a Perl script based on the `tiedlist2` created in tree based clustering.

6.4 Recognizing the test data

HTK needs some kind of rules regarding allowed phoneme sequences in order to be able to recognize test utterances. Normally, these rules are provided by the lexicon and the language model, but since none were used in this work, the simplest possible rule set, a phoneme loop, was used. It states that any phoneme may follow any other without a limit for the phoneme string length.

This is defined by a network file such as:

```
$phn = a|aa|ae|aeae|b|d|e|ee|f|g|h|i|ii|j|k|kk|
        l|ll|m|mm|n|ng|ngng|nn|o|oe|oeoe|oo|p|pp|
        r|rr|s|sil|ss|t|tt|u|uu|v|y|yy;
(sil <$phn> sil)
```

Given appropriate configuration parameters, HTK expands the phonemes to match their context. This kind of approach requires that all possible logical triphones have a physical counterpart. That is, a line exists in the triphone list file for them.

The network file above also defines that a sentence always has to begin and end with silence. This approach reduces the need of diphone models to only silence models. Furthermore, in this work, all `sil`-models are tied to a single physical model.

The network file has to be transformed into a *Standard Lattice Format* (or SLF) file using the `HParse` tool:

```
HParse network lattice
```

After this, recognition is accomplished using the `HVite` tool:

```
HVite -C hvite.conf -i recout_test.mlf -w lattice -H \
 hmm.5/newMacros -t 150 -p -30 -S test.scp vocab tiedlist_rec
```

The “vocabulary” file vocab contains the “description pronunciation” for all phonemes. All phonemes, but not the silence models, are given two different ways of pronunciation: one with a following short pause and one without a pause. This looks simply like:

```
   a a sp
   a a
   aa aa sp
   aa aa
...
   sil sil
```

The `-t 150` option enables beam searching such that any model whose maximum log probability token falls more than 300 below the maximum for all models is ignored in decoding. The `-p -30` specifies the phoneme insertion log probability to be -30.

If the value given with the `-t` option is too small, the most probable path through the states may not be found, resulting in all kinds of recognition errors. If the value is too large – or beam searching is disabled altogether by setting the option to 0 – recognition takes much time.

A positive value for the option `-p` results in many falsely inserted phonemes while too small a value causes many phonemes to be deleted.

Iterative methods were used to find suitable values for these options, and appropriate values depend on the model type. For monophone models the phoneme insertion penalty was -10 and the beam width was 300, while for triphone models the corresponding values were -30 and 120. The `-p` values were selected so that the number of insertion and deletion errors was roughly equal. A good value for the `-t` option was searched for by gradually decrementing it and looking for the smallest value with which the recognition performance still remained good.

6.5 Processing the output

Following the steps described in this chapter, the recognized transcription is produced into the file `recout_test.mlf` with one phoneme on each line. In addition to the phoneme name, its beginning and ending times are included as well as the score (total likelihood of the segment):

```
#!MLF!#
"/data_dir/synt.3000.rec"
0 8100000 sil -665.910339
8100000 8600000 p -169.781860
8600000 9700000 oe -168.355362
9700000 10700000 l -162.563538
...
99700000 100100000 b -228.922165
100100000 101200000 a -313.813904
```

```
101200000 112200000 sil -1044.941772
.
"/data_dir/synt.3001.rec"
0 11200000 sil -1191.084839
11200000 11700000 p -153.512268
```

6.5.1 Alignment by HResults

HTK includes a performance analysis tool, called `HResults`. It compares the recognized label files to the correct reference files and calculates the number of insertion, deletion and substitution errors as well as accuracy and correctness figures. Furthermore, it can produce confusion matrices between phonemes and output the test and reference sentences time-aligned.

By default, `HResults` does not print the time-aligned transcription for sentences with no errors. This was changed in order to simplify the further analysis process. Also, if the `-e` or `-c` flags are used to make two phonemes equivalent, the original phonemes are replaced in the output transcription. Both of these cases were undesired, and the behavior was modified by making some minor changes in the `HResults` source code.

For utilizing the time-aligned transcription produced by `HResults`, a set of Perl scripts were written. They gather information about:

- Overall correctness and accuracy
- Sentence based results
- Confusion matrices
- Error chain lengths
- Overall results for the individual phonemes
- Results for the individual phonemes in different contexts
- Errors for all phonemes in different contexts
- Overall phoneme statistics

The scripts produce Matlab code, which is used to visualize the vast amount of information. The recognition results are discussed in the next chapter.

Chapter 7

Analysis of the recognition tests

In order to achieve further insight into the performance of different HMM types the results were analyzed thoroughly. First, the overall insertion, deletion, and substitution errors were counted and the accuracy and correctness figures derived from these. The same was done to phonemes appearing in the beginning and end of words and sentences to test whether word stress would have an effect on the performance. Furthermore, it was analyzed what kind of errors were most common for each phoneme, which phonemes are confused with each other, and in what context.

As noted in the previous chapter, five different kinds of model sets were built, one monophone set and four triphone sets, which differ from each other in terms of triphone clustering and to some extent number of models. The model sets and acronyms used for them throughout this chapter are presented in table 7.1.

Model set	Short	Number of models
Monophones	**Mon**	41
Place of Articulation clustered triphones	**PoA**	623
Type of Phoneme clustered triphones	**ToP**	991
Aggressively tree-based clustered triphones	**TLo**	1259
Moderately tree-based clustered triphones	**THi**	3785

Table 7.1: The different model sets used in the experiments and the number of models for the MFCC front end.

7.1 Performance figures

As explained earlier, there are three different types of recognition errors: substitution (S), deletion (D), and insertion errors (I). The total number of phonemes is marked with N, and the number of correctly recognized phonemes with H. There are two commonly used figures that can be calculated from these numbers. The correctness (C) figure is calculated according to equation 7.1 and it describes the portion of phonemes recognized correctly from all phonemes.

$$C = \frac{H}{N} = \frac{N - S - D}{N} \tag{7.1}$$

Accuracy (equation 7.2) takes the number of inserted phonemes into account, and is therefore a better measure for comparing the performance of recognizers producing phonemes as output. It would be possible to create phoneme strings achieving a correctness figure of 100 % for any utterance simply by repeating all phonemes at every time unit.

$$A = \frac{H - I}{N} = \frac{N - S - D - I}{N} \tag{7.2}$$

The same figures could be used to calculate word error rates and in that case the insertion error would be much smaller. However, this was not done since neither a language model nor a lexicon was used in this work. This choice was made since the phoneme errors were of major interest in this work and converting phoneme strings into words tends to mask those errors. It would be simple to produce a list of words from the data and then use a unigram language model for the recognition but this kind of simplistic approach would lead to large word error rates (probably above 80 %). Furthermore, that kind of an approach would have little in common with any real application.

Correctness is also calculated for separate phonemes. Additionally, another error measure, global error proportion (E) is calculated for them as defined in equation 7.3. S, D, and I are the values of different errors made for this particular phoneme, and N_{phon} is the number of all phonemes. This figure describes the number of errors for this particular phoneme proportionally to the global number of phonemes, and it does not take into account the frequency of the phoneme in the data. Therefore, this error figure is usually higher for more common phonemes.

The motivation for the use of E for phonemes is that it shows in which phonemes the most errors occur. In contrast to the C figure calculated for phonemes, E measures the absolute error rate, (not proportional).

$$E = \frac{S + D + I}{N_{phon}} \tag{7.3}$$

A hypothesis was made that the phonemes in the beginning of words – and especially sentences – would be recognized with better performance than those in the middle or end of words, and that those ending a sentence would produce the worst performance of all. This was justified by the fact that in Finnish the beginnings of words are usually stressed, and towards the end of the word or sentence the pronunciation becomes unclear, and some phonemes might be dropped out altogether. In order to test the hypothesis the performance figures were calculated for the first and last two phonemes of words and sentences.

In order to gain further insight into the nature of the most common error types the contexts of errors made for each phoneme were analyzed.

Furthermore, the recognition accuracy of individual sentences was analyzed in order to find out what kinds of sentences produce the best and worst performance. Related to this, the lengths of error chains were also studied.

7.2 Significance of difference and accuracy of results

When examining the results of a recognition test one should always make sure that the performance difference between two algorithms is not random. Similarly, the real precision of the results should be estimated and be presented with only those decimals that are significant. Methods for these tasks are described in this section.

7.2.1 Matched-pairs test

It is often ignored in speech recognition research papers, as noted in [78], whether an apparent difference in recognition results is statistically significant. [78] presents two tests for this, one of which, the matched-pairs test, is suitable for continuous speech recognition. The test is described as follows.

Assume that there are two algorithms A_1 and A_2 that are tested using the same material. The test material is divided into n segments so that any errors made inside a segment are independent of any errors made in any other segment. A natural choice for segments are sentences, after which the speaker pauses.

Let N_1^i and N_2^i be the numbers of errors made in a segment by the two algorithms, and let $Z^i = N_1^i - N_2^i, i = 1, 2, \ldots, n$. Let μ_z be the unknown average difference in the number of errors in a segment made by the two algorithms. We would like to test whether $\mu_Z = 0$. μ_Z is estimated by $\hat{\mu} = \sum_{i=1}^{n} Z_i/n$. The estimate of the variance of the Z_i's is:

$$\hat{\sigma_Z}^2 = \frac{1}{n-1}\sum_{i=1}^{n}(Z_i - \hat{\mu})^2 \tag{7.4}$$

The estimate of the variance of $\hat{\mu_Z}$ is:

$$\hat{\sigma_\mu}^2 = \frac{\hat{\sigma_Z}^2}{n} \tag{7.5}$$

Another random variable W is defined as:

$$W = \frac{\hat{\mu}_Z}{(\hat{\sigma}_Z/\sqrt{n})} \tag{7.6}$$

If n is large enough, W will approximately have a normal distribution with unit variance. We make the null-hypothesis, that $\mu_Z = 0$, that the difference between the recognition results of A_1 and A_2 is not significant. This hypothesis is tested by computing the probability P of observing the value W under the null-hypothesis:

$$P = 2 \int_{|W|}^{\infty} \exp^{-x^2/2} \tag{7.7}$$

If P is below a certain threshold the null-hypothesis is rejected and the difference between the two algorithms is statistically significant. Typical threshold values are 0.05, 0.01, and 0.001. Where this significance examination is applied in this thesis, the value of 0.01 is used.

7.2.2 Precision of the results

Following similar principles to those used in the matched-pairs tests one can also estimate the precision of the results so that the results can be presented with meaningful precision.

Independent observations can be estimated to be from the normal distribution provided that there are enough of them. Sentences can be thought of as independent observations, and then one can calculate the interval where the true correctness is according to a selected level of confidence.

7.3 Phoneme duration

It is known that the HMM scheme does not deal very well with phoneme durations. After a preliminary analysis the substitution errors between long and short variants of the same phoneme are ignored in the error analysis. This decreases the overall differences in both correctness and accuracy in the models by 12–13 % in Mon, 8–9 % in PoA, 7–8 % in ToP, 5–5.5 % in TLo, and 4.5–4.8 % in THi.

Short phonemes are more often recognized as long phonemes than vice versa. For monophone models the difference is as large as 500 %, for ToP and PoA models 160–230 %, and for TLo and THi models 20–60 %. Context-dependency and the addition of more models clearly improves the discrimination between short and long phonemes. There were about nine times as many short as long phonemes in the data.

7.4 Overall recognition results

Front end	Correctness of THi	Accuracy of THi
PLP	95.6 ± 0.3	92.3 ± 0.3
MFCC	95.3 ± 0.3	91.8 ± 0.4
BFCC	94.5 ± 0.3	90.6 ± 0.4

Table 7.2: Overall recognition results obtained by the best-performing model set, THi, for all tested front ends.

Table 7.2 summarizes the best results achieved for each front end. Figures 7.1 and 7.2 show the overall correctness and accuracy figures for all phonemes and all tested models. The error intervals are calculated by estimating the fitting of results of the separate sentence to the normal distribution with a confidence level of 0.01.

The highest achieved performance figures were for the THi models and the PLP front end. Those using MFCC were not far behind, but according to the matched-pairs test (section 7.2.1), the difference is statistically significant with a threshold of 0.01. BFCC performed clearly worse. One possible explanation is that the λ-coefficient was not optimal for the female speaker. As can be seen, the differences between the front ends are not very noticeable.

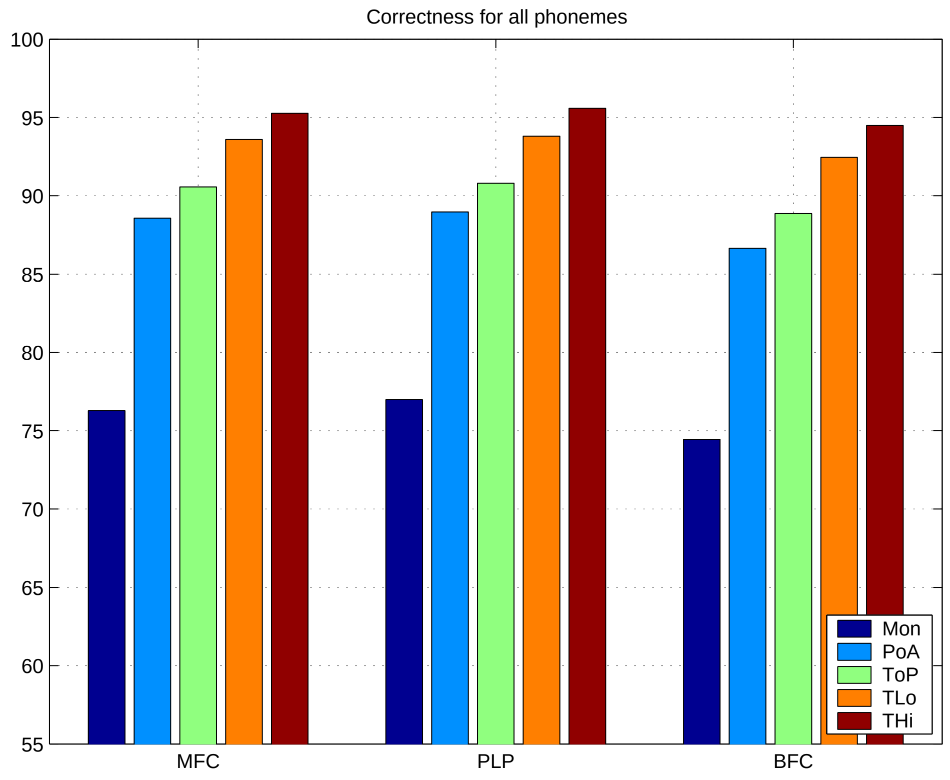

Figure 7.1: Correctness of all phonemes, models, and front ends.

The recognition performance increases with the number of models regardless of the front end. This result was as expected. Furthermore, the number of models produced by the different clustering strategies seems to be equal regardless of the front end (see figure 7.3).

The results from all front ends were closely similar. If there had been more noise in the data the results would probably have been different and PLP would have benefited from its sophisticated structure. Since there was no great difference between the front ends only the results of MFCC are discussed in further detail.

Figure 7.4 shows the correctness of word and sentence beginnings and endings, and figure 7.5 shows the accuracy figures for the same. Furthermore, figure 7.3 displays all the results in one figure.

In the following the recognition results for sentence and word beginnings and endings

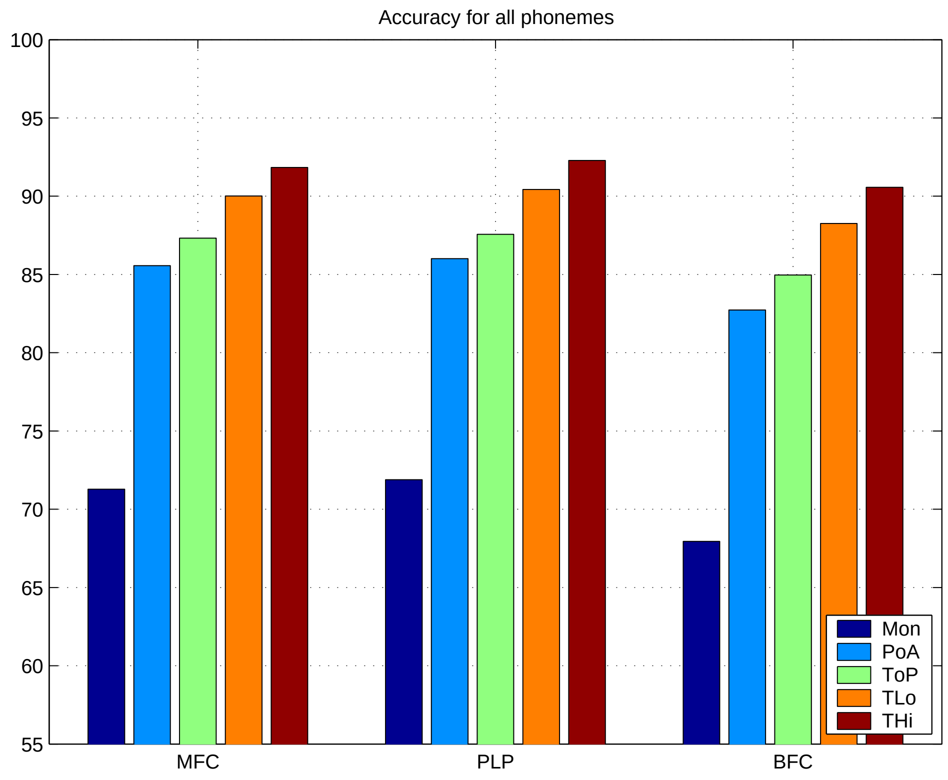

Figure 7.2: Accuracy of all phonemes, models, and front ends.

(abbreviated SB, WB, SE, and WE) are discussed.

The hypothesis of sentence beginnings producing better recognition results proved to be false. With closer examination of the recognized labels it can be seen that in many sentences a stop consonant is inserted into the front of the transcription of an utterance. Also, other kinds of errors are more common in the beginning of the sentence than in the rest of the sentence. This is partly explained by the properties of a HMM recognizer and partly by the fact that the utterances contain a variable amount of silence and often some noise caused by inhalation before the actual sentence begins. Since the beginning of the utterance is not accurately marked there is a rather large amount of variation producing errors.

For all models the results are better in SE than SB. The reasons for this are explained in the previous paragraph. The differences are smaller for THi models than the others, so having more models helps in SB.

Word beginnings perform better than average for all models in terms of correctness. This suggests that word stress still has positive effects on the recognition results: there recognizers find the correct phonemes more often from the WB parts than from the rest of the material. However, insertion errors are more common in WB which lowers the accuracy figures for WB to below average.

The accuracy of ToP and PoA models is clearly better in SE than in SB, and they even outperform the TLo models in the endings. In all models except THi the results were better in SE than the overall results.

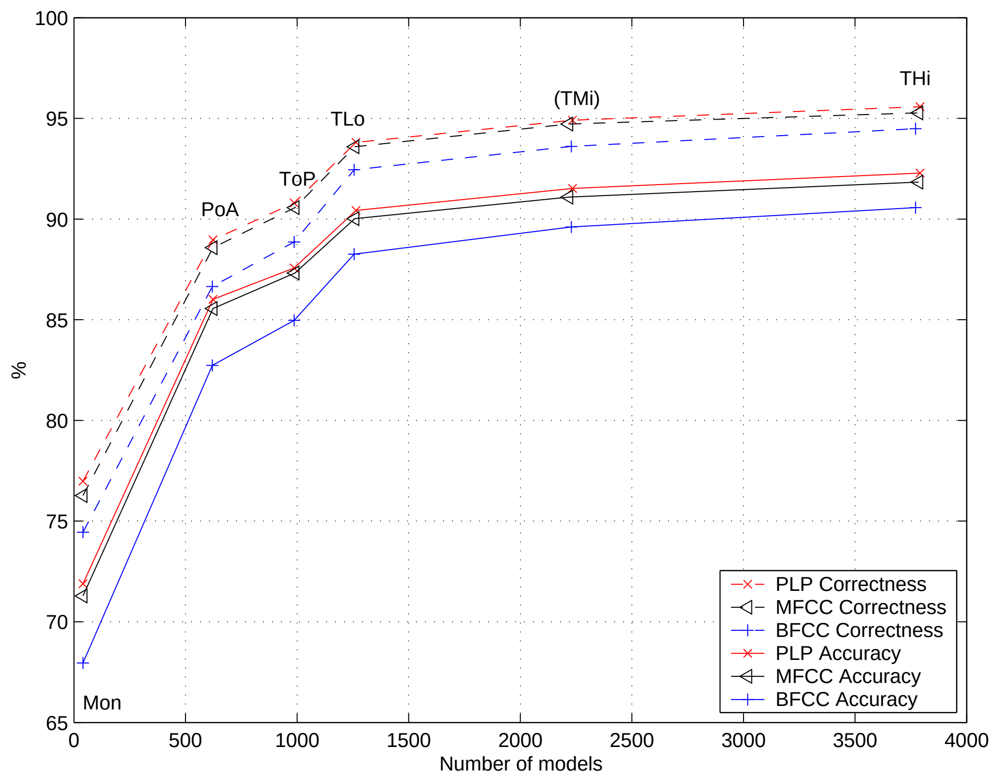

Figure 7.3: Results for all model sets and front ends relative to the number of models. The TMi model set, which is not covered elsewhere in the document, corresponds to a model set gained from tree-based clustering with the number of models between that of THi and TLo.

However, for testing the effect of word stress there would have been better methods. The differences shown here are probably mainly due to HMM properties.

One of the original goals of this thesis was to compare different clustering algorithms. Unfortunately, it proved hard to force the number of models to be the same, so a direct comparison cannot be made. However, the deviation in the lines of the figure 7.3 at the ToP models suggest that the performance difference is not only due to the number of models, but the clustering strategies utilizing a priori knowledge perform worse.

7.5 Error distributions of sentences

Since the phoneme recognition performance increases greatly when introducing context-dependent phoneme models, so do the sentence results as well. This is illustrated in figure 7.6 that shows the histograms of sentence correctness and accuracy for both Mon and THi models in the same figure.

The text contains some sentences with foreign proper names. When they are transcribed automatically to phoneme strings using Finnish pronunciation rules, some transcription errors occur. Thus, even if the recognizer correctly recognizes what has been said, an error is found due to the erroneous transcription. Dates, numerals,

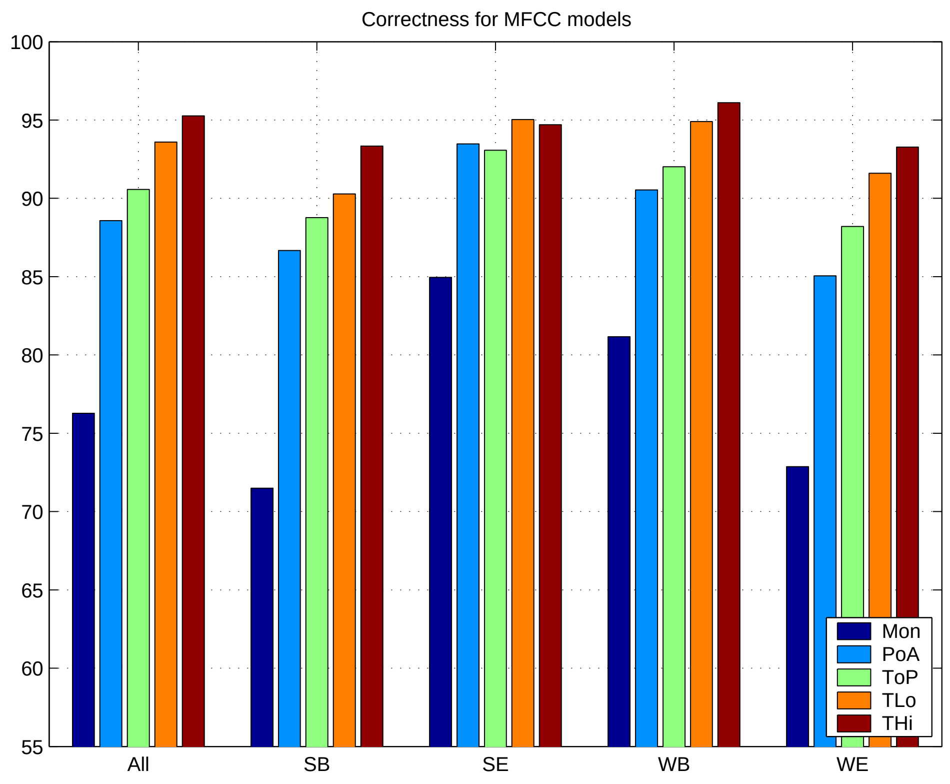

Figure 7.4: Correctness of all phonemes, sentence beginnings (SB), sentence endings (SE), word beginnings (WB), and word endings (WE) for the MFCC front end.

and abbreviations also cause similar problems.

A set of sentences consisted of chapter and section titles and were very short consisting of only one or two words. On one hand, these kind of sentences formed the group of sentences that were recognized perfectly. On the other hand, if there was an error in these short sentences the performance figures would drop immediately to a very low value.

A couple of example sentences of both well and poorly performing sentences that do not fit into the special cases mentioned above are now presented. Both monophone and context-dependent models are considered. For all sentences, the sentences recognizer outputs of each model set is presented in figure 7.7.

Some errors that one might think of as being easy to remove afterwords are visible in this small set of examples. Inside a word the same phoneme may occur several times in a row. These kinds of repetitions would be easily removed from the output phoneme stream, but as long as there is no information about the word boundaries, one cannot do this. The word boundary may be in between those two, and the next word might begin with the same phoneme as the previous one ends in.

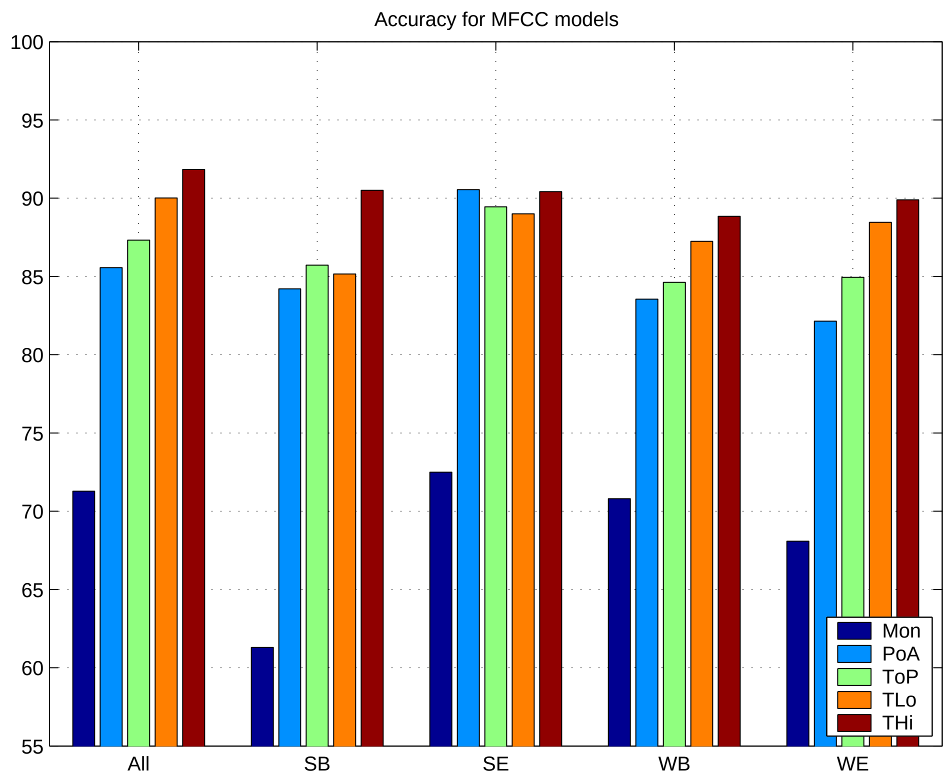

Figure 7.5: Accuracy of all phonemes, sentence beginnings (SB), sentence endings (SE), word beginnings (WB), and word endings (WE) for the MFCC front end.

7.5.1 Example sentence A – well recognized

The first example sentence (see figure 7.7) performed relatively well with monophone models and the improvement gained with context-dependent models was not high. The differences in recognition precision are not great between different models. No deletion errors are produced but a couple of insertion errors occur for each model set. This could imply that a larger penalty for inserting phonemes in recognition might improve the performance. However, as we will see in examples C and D, the optimum penalty value is sentence dependent and thus cannot be adjusted for each separate sentence.

The substitution errors that are present for this sentence are quite typical and are mainly due to short vowels getting mixed with each other.

7.5.2 Example sentence B – well recognized

The second example sentence (see figure 7.7) performed well in terms of accuracy in the monophone system, compared to other sentences with the same model set. The performance of the different context-dependent systems was quite equal and clearly better than that of the monophone system.

There were no insertion errors in the output of the monophone system. The short functional word *ja* (meaning *and*) was completely misrecognized by the monophone

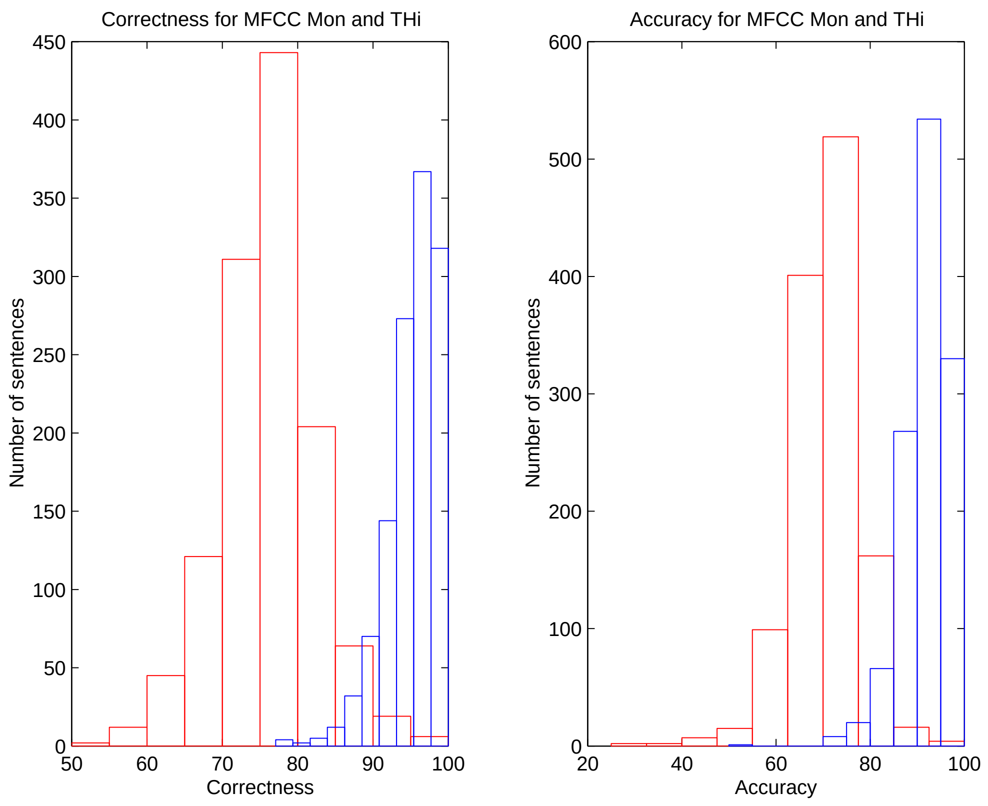

Figure 7.6: Correctness and accuracy histograms of sentences for Mon (red) and THi models (blue). This figure gives an overview on how the sentence distribution of errors differs between the best and the worst model set.

system, but all context-dependent systems recognized it with 100 % correctness. This is a typical place where triphones models are most effective.

Most of the context-dependent models suggested an inserted stop consonant at the beginning of a word. As mentioned earlier, this is a typical problem that is due to the properties of HMM recognizers.

7.5.3 Example sentence C – poorly recognized

The third example sentence is the longest of the four examples presented. It was selected from the worst sentences in terms of correctness produced by the monophone system.

The output of the monophone system is unitelligible and the phoneme recognition rate is low, below 50 %, and basically all kinds of errors are present. The improvement when using context-dependent models is large. A speaker of Finnish would understand the output of any of them without great difficulty.

There are, however, a few errors common to most of the models: the /p/ in *läpikulku-juna* has been deleted by all models. Another phoneme posing problems is the /l/ in the last word of the sentence. The word *Köln* is not Finnish, it is the name of the town Cologne in Germany, and the reader pronounces the word like it is pronounced

```
Example sentence A:
Cor:  ! rakeNuksen%hajoteTavUs% nä  kY  %ja%sA %näky ä%kaiKiaLa%!
Mon:  !trakeNuksen hajyteTavUs yNä  kY   jä SA  Näkyöh KaiKjaLa !
PoA:  ! rakeNoksen hajetetavUs  nä  kYn  ja SA  Mäky ä KaiKiaLa !
ToP:  ! rakeNuksön hajetetadUs tnä  kyY  jä SA  Mäky ä KaiKiaLa !
TLo:  ! rakeNoksen haiyteTavUs  nyNäkyYn ja SA  näkyöä kaiKiaLa !
THi:  ! rakeNuksen hajeteTavUs  nä  kyYy ja sAa näky ä kaiKiaLa !

Example sentence B:
Cor:  !se%mUTA%puh En%kirjoitetuksi% tekstiksi% ja%päinva stoin%!
Mon:  !se mUpA puh En ki joytetuksi  Kekstiksi  pe pÄ eva Stoin !
PoA:  !se mUTA pUh En kirjoitetuksi Kpekstiksi  ja päinva stoin !
ToP:  !se mUTA puhjEn kirioitetuksi  tekstiksi  ja päi vaSstoin !
TLo:  !se mUTA puh En kirjoitetuksi  Kekstiksi tja päinva stoin !
THi:  !se mUtA puh En kirjoitetuksi Ktekstiksi  ja päinva stoin !

Example sentence C:
Cor:  !kun%läpikulkujuna%on%noin%kyMenen%kilometrin%pÄSä%bo Nista!  lähtE% sieltä%l Ity nt  äjuna%kIh dyTämÄn%kohti%köl  ni ä%!
Mon:  !KoN lä iku hodlNa o  No ■ kyMäNE■ k jam ähi■ tÄSä bo NIsta hTbähTE  S jltä L Iky■ke  idlNA ki  dytäMÄ■ kuhti kÖr  NIeä !
PoA:  !kuN läpikulkujoNa o  Noin kyMene■ k joMätrim pÄSä bO NIsta   lähtE  SielTä l ItY nt  öjuNa kij dytämÄn kohti köLyrni ä !
ToP:  !koN läpikolhhyona o  Noin kyMenE■ kil Mätpim pÄSä bo Nista   lähke iSi ätä l Ity nteihyoNA kIhidyTömÄn kohti kÖr  NI ä !
TLo:  !kun läpikolhojuna o  noin kyMene■ kilomntrin pÄSa bOnNista  plähte iSieltä liity nt  äjuna kI  dyTämÄn kohti kÖy  ni ä !
THi:  !kun lä ekolkojuna on noin kyMene■ kilometrim pÄSä bO nistA tTlähtE  Sieltä lIity nt  ejuna kii dyTämÄn kohti kÖr  ni ä !

Example sentence D:
Cor:  ! ensiNäkÄn%hiSi en%  a jo väyliä%  e i%  a id  a ta ! ne%vain%merkitÄn%selkeästi%!
Mon:  !te syNäkän  ISI eN akaij0 vÄydjÄ   EKI  kA id  ä TA  pye va   MErkitÄA sEltE sti !
PoA:  ! ensiNäkÄn  ISi äN  Aa jo väyliä   EpI kaaiidäpAnt    ne vai  mErkitÄn selkeasti !
ToP:  !te s näkÄn hiSi äN  aaido väyriä   EKI kAa ii  d ätA Tne vai  mErKitAn sElkeasti !
TLo:  ! ensinäkÄn  ISi en Maa johväy iä Ete i  aa id  Ä TA  tne va   MErkitÄn selkeasti !
THi:  ! insinäkÄn hiSijäN  aa jo väyliä  ee i  Aa id  a ta  hne vai  mErkitÄn selkeasti !
```

Figure 7.7: Example sentences recognized using the different models. The topmost row of each sentence shows the correct labeling, and the following lines contain the recognition results for each model set. The transcriptions are aligned so that any recognition errors are easy to compare. Short (lower case letters) and long (upper case letters) are marked in the figure, even though they are considered equivalent in error analysis. The sign § is used for /ŋ/; the percent sign (%) stands for the short pause; an exclamation mark denotes silence. Any whitespace is added merely for alignment purposes.

in German. Furthermore, the combination "ln" is illegal in Finnish non-compound words, and there were no training examples with this combination. These kind of errors are hard to avoid but they are not very common.

Two other "errors" are also present but they are merely errors in the original transcription and not the recognizer output: the words *on noin* are pronounced together, merging the two short /n/'s together forming one long /n:/ sound. Similarly, the /n/ sound at the end of the word *kymmenen* is transformed to a /ŋ/ sound since the following word begins with a /k/. By going through the label file manually these kinds of errors could be avoided but this would be very tedious since these transformations do not always take place.

7.5.4 Example sentence D – poorly recognized

The fourth example sentence was uttered a little oddly: there is some mumbling between the words *hissien* and *ajoväyliä*, there is a glottal stop after *ajoväyliä*, the /e/ in the following *ei* is long, and the fundamental frequency raises clearly within the /ai/ diphthong of the word *aidata*. All of these seem to produce mainly insertion errors in the data. Additionally, the /h/ is hard to recognize in the beginning of *hissien*.

7.6 Results for separate phonemes

The overall per-phoneme correctness (equation 7.1) of all models is shown in figure 7.8. The difference from the monophone baseline system is shown in 7.9. Common consonants /t, n, h/, and /v/ seem to gain the most from the context-dependent phonemes. Having clearer steady states, vowels are better modeled even with monophone models. Some rare phonemes, /b, f/, and /ŋ/, even degrade in correctness when context-dependency is introduced.

The global error proportion (equation 7.3) for all phonemes is shown in 7.10. This error measure is the number of errors made in each phoneme divided by the number of all phonemes, and it describes the absolute number of errors, instead of a relative error rate that could be calculated from the correctness figure by $E_{rel} = 1 - C$.

For confusion matrices for the different models refer to Appendix A. Table 7.3 summarizes the phoneme correctness rates and the numbers of models for each phoneme.

Figures such as 7.11 were generated for all models and phonemes, and those for the TLo models are presented in Appendix C.

In 7.11a, the number of errors concerning /h/ in different precontexts (labels on the x-axis; the asterix stands for the beginning of a sentence) are plotted.

Figure 7.11b gives further insight into what *kind* of errors are occuring in each context. The error type is shown on top of the bars: an X represents deletion, any other character represents substitution with the phoneme printed. Still, errors concerning /h/ are treated, and the precontexts are shown in the x-axis labels.

			MP models				PoA models				ToP models				TLo models				THi models			
	Tr. ex.	Tst ex.	S m	L m	C %	E %	S m	L m	C %	E %	S m	L m	C %	E %	S m	L m	C %	E %	S m	L m	C %	E %
a	21947	11499	1	1	78.3	2.2	43	29	88.8	1.1	76	56	89.0	1.1	132	40	91.2	0.9	355	131	91.8	0.8
b	129	90	1	0	77.8	0.0	5	0	84.4	0.0	6	0	63.3	0.0	1	0	41.1	0.1	6	0	53.3	0.0
d	2821	1111	1	0	86.3	0.1	7	0	90.2	0.1	8	0	86.7	0.1	11	0	92.0	0.1	33	0	95.7	0.0
e	19136	8452	1	1	78.3	1.6	42	14	85.3	1.1	71	33	91.8	0.6	93	15	93.7	0.5	269	59	95.5	0.3
f	120	59	1	0	84.8	0.0	6	0	66.1	0.0	8	0	71.2	0.0	1	0	72.9	0.0	6	0	81.4	0.0
g	487	113	1	0	47.8	0.1	5	0	63.7	0.0	6	0	66.4	0.0	2	0	60.2	0.0	7	0	68.1	0.0
h	3957	1744	1	0	59.1	0.6	14	0	82.7	0.3	21	0	72.0	0.4	24	0	85.0	0.2	79	0	91.2	0.1
i	22765	10647	1	1	71.5	2.7	38	13	90.9	0.8	69	27	92.3	0.7	101	16	94.8	0.5	294	51	95.6	0.4
j	4379	2165	1	0	75.8	0.5	10	0	58.4	0.8	16	0	72.4	0.5	22	0	79.7	0.4	59	0	87.8	0.2
k	12059	5365	1	1	80.6	0.9	16	7	97.3	0.1	24	9	92.6	0.3	62	10	96.3	0.2	142	35	98.2	0.1
l	6415	4373	1	1	80.6	0.7	15	4	92.8	0.3	19	4	96.3	0.1	34	15	96.2	0.1	112	44	98.0	0.1
m	6703	3025	1	1	82.2	0.5	11	4	96.5	0.1	18	4	95.3	0.1	37	5	96.2	0.1	118	18	96.8	0.1
n	17862	8080	1	1	62.7	2.6	19	4	79.7	1.4	26	4	90.5	0.7	84	10	92.7	0.5	227	29	95.2	0.3
ŋ	667	282	1	1	93.3	0.0	2	2	92.5	0.0	2	2	49.3	0.1	4	1	85.5	0.0	9	3	90.8	0.0
o	12708	5308	1	1	81.7	0.8	40	6	89.1	0.5	66	10	91.6	0.4	77	4	94.4	0.3	225	13	96.2	0.2
p	3689	1777	1	1	78.2	0.3	14	6	83.3	0.3	20	6	84.5	0.2	25	3	90.1	0.1	83	13	93.4	0.1
r	5674	2290	1	1	70.9	0.6	20	2	85.8	0.3	20	2	94.8	0.1	26	2	94.6	0.1	101	6	97.6	0.1
s	16423	6963	1	1	99.7	0.0	22	6	99.8	0.0	30	6	99.7	0.0	75	9	99.7	0.0	197	24	99.7	0.0
t	18199	9563	1	1	60.9	3.3	25	6	88.4	1.0	32	8	88.3	1.0	82	27	95.8	0.3	215	70	97.0	0.2
u	8380	4293	1	1	82.6	0.7	31	19	90.9	0.3	54	30	90.5	0.4	49	20	92.8	0.3	160	68	94.1	0.2
v	6493	2860	1	0	67.1	0.8	8	0	90.4	0.2	16	0	92.8	0.2	26	0	93.6	0.2	79	0	94.3	0.1
y	4471	1821	1	1	81.9	0.3	29	8	90.2	0.2	43	18	92.6	0.1	28	4	94.8	0.1	109	25	96.4	0.1
ä	7656	4086	1	1	83.5	0.6	31	18	84.5	0.6	59	35	85.9	0.5	54	17	90.9	0.3	184	72	92.8	0.3
ö	1305	535	1	1	77.4	0.1	21	1	80.8	0.1	26	1	84.5	0.1	10	1	81.5	0.1	53	2	89.0	0.1
Tot. no. of models			41				623				991				1259				3785			
Summ. of results			Corr: 76.7 Acc: 72.1				Corr: 88.3 Acc: 85.5				Corr: 90.8 Acc: 87.2				Corr: 93.8 Acc: 90.1				Corr: 95.3 Acc: 91.8			

Table 7.3: Summary of data, models, and phoneme recognition results. The column Tr. ex. and Test ex. state the number of training and test examples for each phoneme, correspondingly. Sm and Lm are the numbers of models for short and long phonemes in each recognizer, and C % and E % the correctness and global error proportion figures.

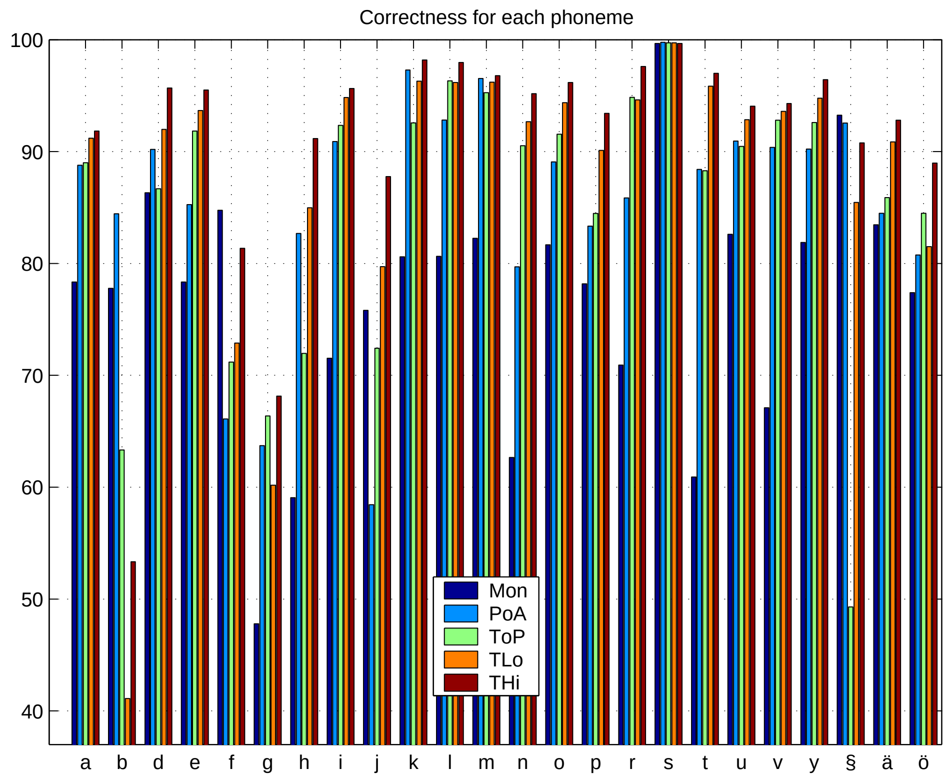

Figure 7.8: Correctness for each phoneme in each model set.

Figures 7.11c and Figure 7.11d are similar to figures 7.11a and 7.11b, except that on the x-axis is the postcontext, not precontext.

Each error type had to occur at least five times before it was included in figures 7.11a to 7.11d. All error rates are proportional: for example, the first bar of figure 7.11b shows that in about fourty percent of the cases where /h/ is preceded by an /s/, it has been recognized as an /f/.

As a summary, the most common errors (substitution and deletion) are presented in figures like 7.11e. Here, on the x-axis label there is either an 'X' meaning deletion or some other phoneme label meaning substitution by the presented phoneme. The height of the bar is again proportional: for example, about eight percent of /h/'s have been deleted (first bar in 7.11e). Information provided by this figure is present also in a confusion matrice.

The following discussion is based on these figures (see Appendix A) as well as the confusion matrices presented in Appendix A.

7.6.1 Vowels

/a/

The phoneme /a/ behaved similarly for all models. By far, the most common error was mixing with the phoneme /ä/, the second most common error was deletion. Rarely, this vowel was also confused with other vowels.

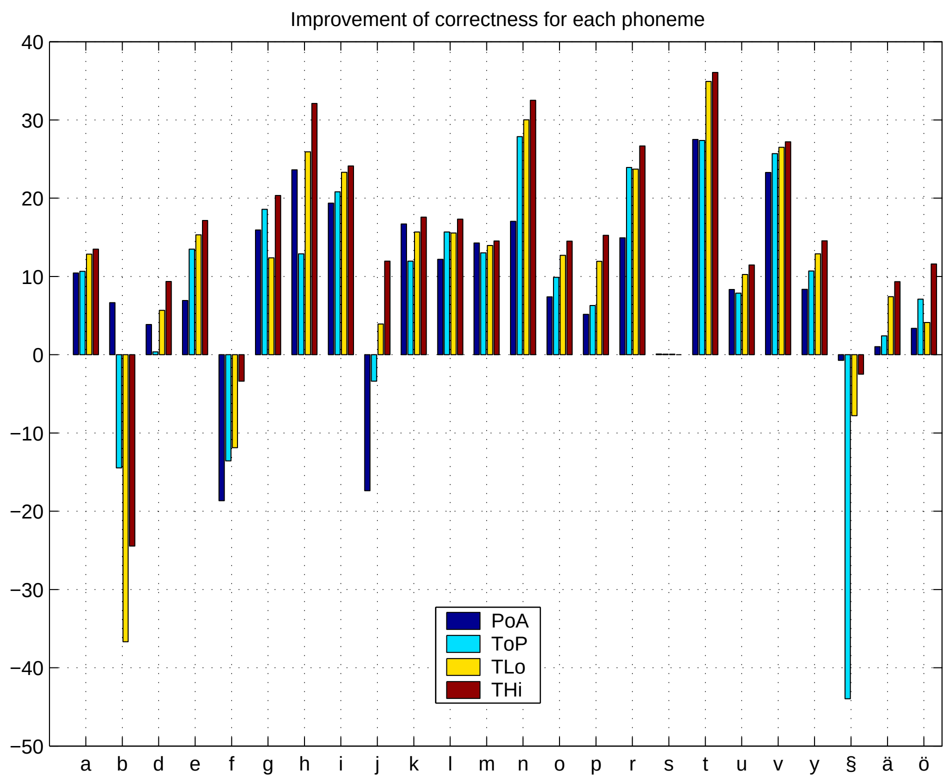

Figure 7.9: Difference in correctness for each phoneme from the monophone baseline system.

By far the most error-prone precontext for the phoneme was /j/. This is due to the word *ja*, which means *and*, and is thus very common. Another error-prone context is the phoneme /y/. This is explained by the fact that an /a/ followed by a /y/ (also the other way around, but there were no cases in the data) collides with vowel harmony – they cannot be close to each other in a Finnish word. However, they can be after each other in two different words, in which case the pronunciation of /a/ tends to move towards the vowel /ä/.

/e/

The phonation of /e/ varies greatly depending on the context. This is due to the fact that vowel harmony does not limit its occurrence, and different surrounding vowels yield very different characteristics for each instance, often rendering the phoneme quite similar to some other vowels.

In monophone models the most common errors are deletion followed by substitution to /ö/, /i/ and /ä/. The same is true for the context-dependent models but with much lower error rates. The most tricky precontext is /y/, which renders the following /e/ very close to /ö/. Furthermore, this combination is not very common – there are ten times as many /yö/ combinations than /ye/ combinations in the data – which leads to a high proportional error rate.

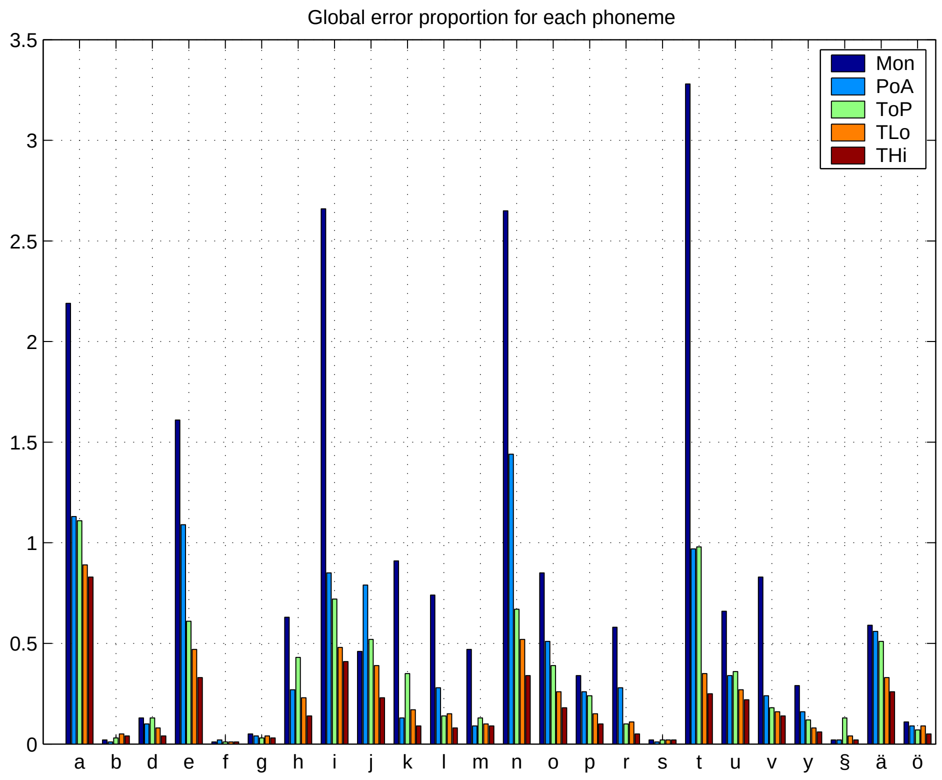

Figure 7.10: The global error proportion for each phoneme. Due to the nature of the error measure the error rates are high for common phonemes and low for rare ones.

/i/

There are several phonemes with which /i/ is mixed: /e/, /j/, and /y/. In some words an /i/ removed from its context actually sounds like /e/, even though a human being hears a clear /i/ when the whole word is listened to. With a following vowel, /i/ is misinterpreted as /j/. The phoneme is often deleted when there is a /j/ on either side.

/o/

The pronunciation of /o/ is similar to /a/ or /u/ in some contexts. Therefore, it is not surprising that there is a fair amount of substitution errors with these phonemes. An /a/ as a neighbor makes the /o/ to be recognized as a /u/.

/u/

The phoneme /u/ is either recognized correctly, deleted, or replaced by /o/. /j/, either as the left or right neighbor, causes a mix-up with /o/ in 15–20 % of the cases where this combination appears (more for monophone models).

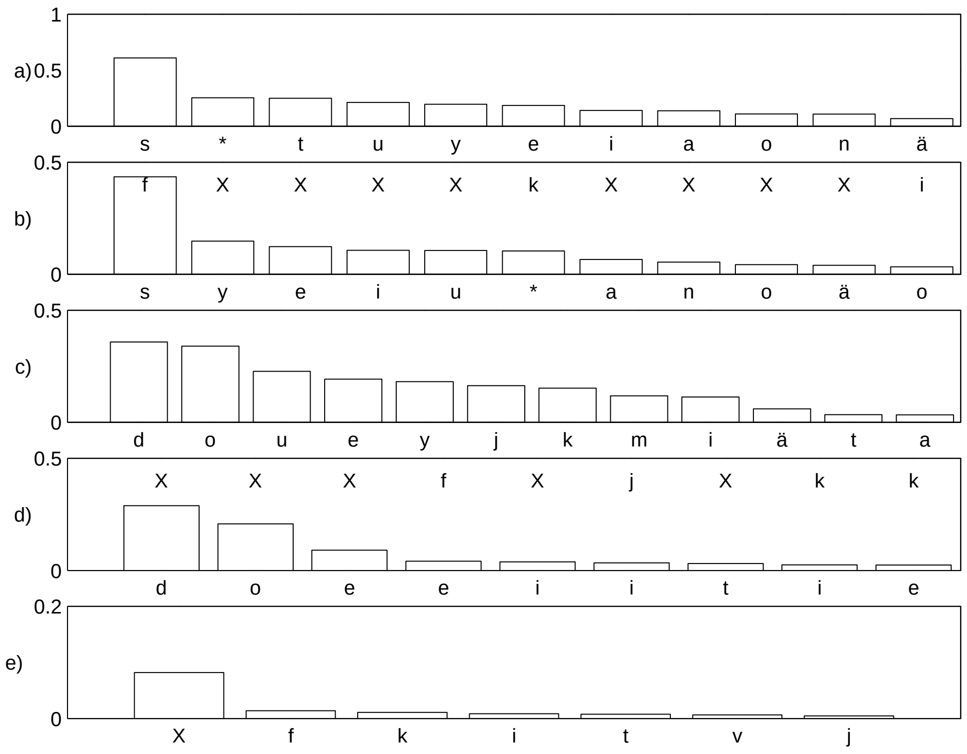

Figure 7.11: Error context analysis for the phoneme /h/ (TLo models). In figures a)–d) the letters on the x axis denote the contexts in which /h/ appears. An asterix refers to the beginning of sentence. When present, the letters above the bars show what kind of an error /h/ has undergone. A capital X means deletion, all other characters substitution errors with the phoneme that is printed on top of the bar. **a)** The most error-prone precontexts. **b)** Errors of /h/ made in different precontexts. **c)** As a), but for the postcontext. **d)** As b), but for the postcontext. **e)** The most common errors for the phoneme /h/, regardless of the context.

/y/

/y/ experiences an amount of deletion errors in vowel contexts with monophone models, but otherwise, errors are quite rare. Some random mixing with /i/ takes place.

/ä/

With monophone models, /ä/ is sometimes misrecognized as /a/ or /ö/, depending on the context. Context-dependency helps here as well diminishing the number of errors.

/ö/

A small amount of confusion with other vowels is also present with /ö/. However, this phoneme is relatively rare so the absolute number of errors is not very high.

7.6.2 Stops

/b/

Being a foreign phoneme there is not much training data for the phoneme /b/. This is the reason for its poor rate of success. Almost all of the errors are substitutions with the phoneme /v/, a close relative of /b/.

There is one odd thing about this phoneme – the performance is dramatically low when using the tree clustered models. The introduced errors are mix ups with /v/. No logical reason for this behavior was found. One might think that the reason is insufficient training data for the several models created. However, the performance is better in THi (six /b/ models) than TLo (one /b/ model). Similarly, the performance of ToP, which contains five /b/ models, is significantly better.

/d/

The phoneme /d/ is rather well recognized and its performance increases logically when more context-dependent models are available. The most common error is deletion and the second most common error is mixing with the phoneme /g/.

Vowels in preceding and following contexts seem to be the most error-prone context.

/g/

The phoneme /g/ seems to be the most difficult phoneme to recognize – the correctness rate does not climb above 70 % for any of the model sets. The phoneme is rare (third after /f/ and /b/), and the errors are usually deletion or substitution with /d/.

/k/

As with the other stop consonants, /k/ is often inserted in the beginning of words. It is sometimes substituted by /t/ or /p/, that is by another stop especially when the following phoneme is a vowel. Overall, with all context-dependent models, /k/ is recognized very accurately.

/p/

/p/ is mixed with the other stops, especially /t/. Difficult contexts for its recognition are the beginning of sentences and places where the following phoneme is a /s/.

/t/

The stop /t/ gains the most in performance when introducing context-dependent models. In the statistics the most common error is deletion when the following phoneme is a /t/. This is the case when two consecutive words both have a /t/ at the border of the words. These are very often pronounced together so this is not always a recognition error.

In addition to this, /t/ seems to be rather randomly mixed with /k/ and /p/ and is added at the beginning of sentences.

7.6.3 Fricatives

/f/

This fricative is another example of a rare phoneme. It suffers from a similar performance drop when using context-dependent models as /b/, but not as severely. In the monophone recognizer the phoneme is mixed with /ŋ/ and the other fricative /h/. For context-dependent systems there are several equally common errors.

/h/

The phoneme /h/ is a difficult to recognize at the beginning of words and especially sentences. It is very often deleted (interpreted as silence) even within words. Hardly any other errors exist for it. Preceding fricatives and following /d/'s are fatal to /h/. The errors are similar in all models, but the performance receives an improvement when more models are added. This is due to the fact that /h/ has a voiced and an unvoiced variant, which are not indicated in the labeling.

/s/

The spectral structure of /s/ is such that it is very easily and accurately recognized even with monophone models. The few errors are of marginal importance.

7.6.4 Nasals

/m/

A very surprising result with /m/ is that it performs very poorly at the beginning of sentences. Monophone models make a mistake in 64 % and the context-dependent models in 16–20 % of the cases when /m/ occurs in that position. The most common error was substitution with /p/. Phonemes /m/ and /p/ are articulatorily related since they impose similar kinds of formant transitions in the following vowel.

In a /p/ postcontext the phoneme is mixed with the other nasals.

/n/

With monophone models, the phoneme /n/ is very often mixed with /ŋ/, especially in a context with stop consonants. This is understandable since /n/ is very common and its pronunciation is very context-dependent. This is a typical case, where triphone models are performing well, and /n/ is one of the three phonemes gaining the most performance from these types of models.

/ŋ/

The most surprising aspect about the results of nasal /ŋ/ is that using triphones clustered by the place of articulation causes a performance loss of 40 %. Also, the other context-dependent systems suffer from a performance loss, but only by a moderate 1–8 %.

The main detrimental factor of /ŋ/ is /n/, which in many contexts is very similar to /ŋ/. In monophone systems many /n/'s are recognized as /ŋ/'s, while in the context-dependent models it is the other way around due to the existence of /n/ models.

7.6.5 Tremulants, laterals, and semi-vowels

/j/

The phoneme /j/ is quite difficult to recognize. There is no steady state section in the phoneme and the formant shifts differ depending on the context. The phoneme is either deleted or mixed with /i/.

/l/

/l/ is recognized well with context-dependent models. There are hardly any substitution errors even when using the simple monophone system. There are a few contexts where in almost 50 percent of the cases the phoneme is deleted when using monophone systems: after a /u/ and before a /k, t/ or /j/. Context-dependency works here very effectively, but still deletion errors occur when /t/ is the following phoneme.

/r/

The phoneme /r/ encounters hardly any substitution errors. It is subject to deletions, when a vowel precedes it and when it is followed by /j, t/, or /k/.

/v/

The text contains many instances of the word *kivi* in different forms. In most cases, the /v/ is recognized as /d/ in monophone models. With context-dependency, these errors are almost totally absent.

A peculiar detail was discovered: all models were recognizing /v/ before a /l/ incorrectly in 90–100 % of the cases. A reason for this was a set of sentences containing the proper name *Vladimir* in the test data. This name, and no other words having /v/ and /l/ after each other was present in the training set. Thus, this is similar to the case in the example sentence C in section 7.5.3.

About 25 % of the time /v/ is recognized incorrectly in the beginning of the sentence, usually as /p/.

7.7 Difficult contexts

An analysis was performed to see which phonemes produce the most errors as pre- and postcontexts. This analysis was also done for the different error types (substitution, deletion, and insertion).

Example graphs for the overall context analysis of THi models are shown in figure 7.12. The complete graphs for all models are presented in Appendix B. The error percentages for a few couple of example phonemes (some of the most difficult contexts) are presented in table 7.4.

As a preceding phoneme for all models, /j/ was by far the most problematic. As postcontexts phonemes /j/ and /f/ were the most error-inducing. The rare stops /b, g/, and /d/ also caused many errors in the preceding phoneme. These are the worst in this sense for all models with slightly differing orders.

The phoneme /d/ is clearly the most probable to cause deletion errors as the postcontext. For monophone models, the deletion percentage is 25 %, for the context-

	Mon		ToP		PoA		TLo		Thi	
Phn	Pre	Post	Pre	Post	Pre	Post	Pre	Post	Pre	Post
/j/	57.4	52.1	32.2	35.7	42.9	34.8	25.2	29.0	21.3	26.7
/f/	22.0	55.1	11.8	27.5	11.8	29.3	16.9	39.6	10.1	24.1
/g/	40.7	30.0	15.0	16.8	14.1	16.8	11.5	23.8	15.0	19.4
/b/	6.6	31.2	6.6	18.7	3.3	17.5	15.5	15.0	5.5	22.5
/ö/	34.1	38.7	14.3	21.4	17.9	16.3	19.4	14.3	13.3	8.4
/ŋ/	21.9	24.8	9.2	14.1	2.8	11.3	6.0	11.7	6.3	8.1

Table 7.4: Percentages of errors for all phonemes in selected pre- and postcontexts.

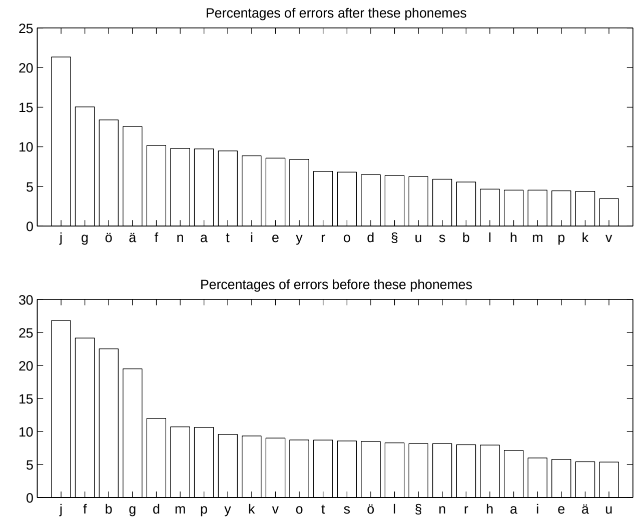

Figure 7.12: Difficult contexts for the THi models. The bars show the percentage of misrecognized preceding or following phonemes in the context of the mentioned phoneme.

dependent models it varies between 3.75 % (thi) and 15 % (PoA). Most phonemes are equally poor in this sense as the left context – the differences are not large.

Insertion errors are most probable before /g, b, j/, and /f/, and after /n, a, ä, ö/, and /g/.

7.8 Error chain length analysis

Statistics were formed from the occurrences of successive errors. This was done to check if most of the errors would be random – meaning single errors or short error chains – or caused by the recognizer becoming "unsynchronized".

The results are clear. For all model sets there are more than twice (for the context-dependent models more than three times) as many single errors than double-error chains, and for longer chains, the number decreases even more rapidly. As an example, the histogram plot of error chains for the TLo models is shown in figure 7.13. Apparently, long chains of errors are not the main concern for the recognizers. Instead, single random-like errors are. However, if a language model would have been used, the results would be totally different. In that case, the restricted vocabulary would force some phonemes to be output incorrectly even if the acoustical models would have recognized them correctly.

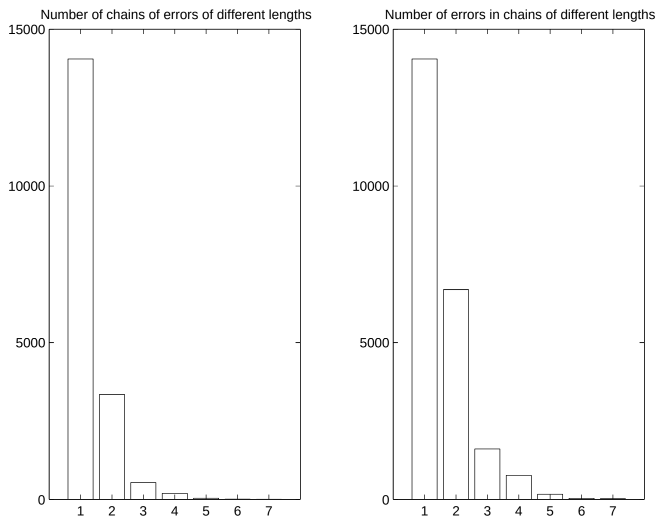

Figure 7.13: Error chain analysis for the TLo models. Over half of the errors are single errors.

7.9 Computational load

Even though computational load was not a major issue in this work, and there was no goal such as reaching real time speed, the time consumed by the recognition process was measured. This was done by selecting a set of sentences with a total length of 11:54 minutes. The recognition time of the different models is shown in table 7.5. The Mon1 and Mon2 model sets refer to the same model set while just a different beam width is used in `HVite` (300 for Mon1 and 120 for Mon2). The beam width was 300 in the recognition tests but when measuring the computational load it proved that it can be significantly lowered: the recognition results for Mon1 and Mon2 were identical, even though Mon2 spent 23 percent less time in recognition. This was not optimized earlier since even with too wide a beam recognition using the monophone models was very fast.

As can be seen, the monophone models easily reached real time speed. This is not the case for context-dependent models, but the difference is not that great – with a computers doubling in speed every eighteen months [79], real time should soon be reached.

There are no large differences between different triphone models which is somewhat surprising since the difference in the number of models is large between them. Evidently, the beam search algorithm succesfully avoids searching through incorrect models, when high probabilities for the context-specialized models are found.

Model	Time (min:sec)	x real time	N. of models
Mon1	0:30	0.042	41
Mon2	0:23	0.033	41
PoA	22:13	1.9	623
ToP	26:06	2.2	991
TLo	26:16	2.2	1259
THi	25:16	2.1	3785

Table 7.5: Recognition time of the different models for a test set of 11 minutes 54 seconds.

Chapter 8

Conclusions

The effect of introducing context-dependent phoneme models (triphones) in Finnish continuous speech recognition was studied in this work. This was done by training single speaker recognizers using:

- monophone models: **Mon**
- decision-tree clustered triphone models: aggressively clustered **TLo** and moderately clustered **THi**
- triphone models clustered based on a priori information about the type of the surrounding phonemes: **ToP**
- triphone models clustered based on the place of articulation of the surrounding phonemes: **PoA**

Three different front ends were tested for all model sets: Mel Frequency Cepstral Coefficients (MFCC), Bark Frequency Cepstral Coefficients, and Perceptual Linear Prediction (PLP). Analysis of errors in different phonemes and in different contexts was an important part of this thesis.

The most important results of this work were:

- Tree-based clustering produced the best results, but they also had the highest number of models.
- PLP performed the best as a front end by a statistically significant margin, but the differences to MFCC and BFCC were not large.
- Common consonants gained the most from context-dependency. Some rare consonants even had lower performance figures for triphone than monophone models. This was perhaps due to the lack of training data.
- The contexts producing most errors in neighboring phonemes were /j/ and /f/.
- When recognizing without a language model, most errors appear as single errors, without forming long error chains.

The best overall result, 92.3±0.3% accuracy, was achieved by the THi models and the PLP front end. Generally speaking, the performance was better for those clustering methods that allowed more models.

In error figure calculation, errors involving silence and substitution errors between the long and short variants of the same phoneme were ignored (the corresponding figures with short and long phoneme mixups taken into account were 90.8 ± 0.3 % and 87.3 ± 0.3 %).

Due to the difficulties in making the number of models equal for all clustering strategies, a direct comparison between them was not possible. However, it seems that the data driven tree-clustering approach provides better results than those methods utilizing phonetic or articulatory a priori knowledge. The performance seems to rise almost systematically with the number of models.

This work has concentrated on the Finnish language, which has both important pros and cons from the speech processing point of view. Having almost a one-to-one relation between phonemes and the orthography, transformation between written Finnish text and phoneme strings are easier than in many other language. On the other hand, the huge number of different words, often differing only slightly, poses serious problems especially for language modeling. Furthermore, some simpler aspects that are not present in major languages, such as information related to phoneme duration, have been somewhat neglected in research concentrating on languages like English.

A major part of this thesis concerned the analysis of the recognition results. In addition to the usual overall error analysis, several different aspects were studied:

- The error-producing contexts were analyzed both globally and for each phoneme separately.
- The most common errors for each phoneme were recorded.
- Results for sentences were analyzed separately, trying to find out what kind of sentences perform well or poorly.

The most problematic contexts were phonemes /j/ and /d/, as well as the rare phonemes /b, f, g/, and /ŋ/. Deletion was a common error for vowels as well as /h/ and /l/ especially in the monophone systems, and stops and /h/ were often inserted. Substitution errors were common inside phoneme type classes (stops, nasals, and vowels).

The recognizer achieved in this work is by no means perfect. The main weakness of it is the lack of a proper language model. As such, the recognizer merely produces a phoneme stream, instead of words and sentences. Adding a simple unigram model would have been straightforward using HTK tools. However, the construction of a proper language model, especially for the Finnish language (studied e.g., in [80]) is a very challenging task for the reasons mentioned above – there would easily be enough work for another master's thesis. Furthermore, a language model (especially a poor one) would increase the phoneme error rate and mask the phoneme errors uncovered in this work.

Finally, there is much improvement required in utilizing acoustical information, especially regarding modeling of phoneme transitions. This can be seen, for example, from the high percentage of errors that occur in a /j/-context. This particular phoneme has no real steady state, and it also causes long transitions in its neighboring phonemes.

Some relatively easy improvements could be made to the recognizer. These include applying the following techniques:

- Adding Gaussian mixtures to the states. This would be a major improvement for monophone models, but probably not so significant for triphone models. This is since the single-Gaussian monophone models try to model the realizations of phonemes in different contexts with just one Gaussian, while the triphone models are already specialized in realizations from one kind of context. In comparison, it would have been "fair" to allow monophone models the same number of parameters as the triphone models have by adding Gaussians. However, this would have been outside of the scope of this work.
- State-clustering instead of model clustering. This would lead the clustering process even more towards the data-driven direction, which seems to be a good method. Additionally, one could imagine that the clustering results would be more fine-grained.
- Parameter tuning. The recognition process includes many parameters that are probably not at their optimal value. Much experimentation would be needed for fine-tuning them.
- More carefully selected training data and/or more training data. Currently, the training data was selected quite randomly from the data set. This may leave many of the less common triphones under-trained.

For many practical applications speaker independency is a major criterion. The methods presented in this thesis could also be applied to such a recognizer, but speaker independency poses many new problems, as well. The need for training data increases dramatically when the variation between speakers, in addition to the variation between phoneme realizations, needs to be modeled. There is still very much to be performed before a computer recognizes spontaneous speech of any person.

Bibliography

[1] M. Rämä. Markov-mallit puheentunnistuksessa. Master's thesis, Helsinki University of Technology, 1988.

[2] P. Utela. Foneemitunnistus diskreetin jakauman Markov-malleilla. Master's thesis, Helsinki University of Technology, 1992.

[3] V. Siivola. An adaptive method to achieve speaker independence in a speech recognition system. Master's thesis, Helsinki University of Technology, 1999.

[4] M. Creutz. Adaptering i automatisk taligenkänning. Master's thesis, Helsinki University of Technology, 2000.

[5] J. Laver. *Principles of Phonetics.* Cambridge University Press, 1994.

[6] K. Suomi. *Johdatusta Fonologiaan.* University of Oulu, 1988.

[7] D. O'Shaughnessy. *Speech Communications – Human and Machine.* IEEE press, second edition edition, 2000.

[8] K. Wiik. *Fonetiikan perusteet.* Werner Söderström Osakeyhtiö, second renewed edition edition, 1998.

[9] M. Karjalainen. *Kommunikaatioakustiikka.* Raportti 51 / Teknillinen korkeakoulu, sähkö- ja tietoliikennetekniikan osasto, akustiikan ja äänenkäsittelytekniikan laboratorio, 1999.

[10] S. S. Stevens. On the psychophysical law. *Psychological review*, 64:153–181, 1957.

[11] S. Buus and M. Florentine. Modifications to the power function for loudness. In *Fechner Day 2001. Proc. of the 17th Annual Meeting of the International Society of Psychophysics*, pages 236–241, 2001.

[12] Tsuhan Chen. Audiovisual speech processing. *IEEE Signal Processing Magazine*, 18(1):9–21, 2001.

[13] J. Korpela. A short introduction to the finnish language, 2002. `http://www.cs.tut.fi/~jkorpela/finnish-intro.html`.

[14] M. Karjalainen, P. Boda, P. Somervuo, and T. Altosaar. Applications for the hearing-impaired: Evaluation of finnish phoneme recognition methods. In *Proc. of EUROSPEECH'97*, volume 4, pages 1811–1814, 1997.

[15] K. Torkkola, J. Kangas, P. Utela, S. Kaski, M. Kokkonen, M. Kurimo, and T. Kohonen. Status report of the Finnish phonetic typewriter project. In T. Kohonen, K. Mäkisara, O. Simula, and J. Kangas, editors, *Artificial neural networks*, volume 1, pages 771–776. North-Holland, 1991.

[16] A. Iivonen. /h/:n aseman dynaamisuudesta suomen fonologisessa järjestelmässä. *Virittäjä*, pages 125–136, 214–230, 1981.

[17] V. Setälä. *Suomen kielen dynamiikkaa.* Suomalaisen kirjallisuuden seura, 1972.

[18] M. Pääkkönen. *Grafeemit ja kontekstit.* Suomalaisen kirjallisuuden seura, 1990.

[19] M. Vainio. Phoneme frequencies in finnish text and speech. In *Studies in Logopedics and Phonetics 5. Publications of the Department of Phonetics, Series B: Phonetics, Logopedics and Speech Communication 6*, pages 181–192. University of Helsinki, 1996.

[20] A. Iivonen and M. Lennes. Suomen fonetiikkaa. `http://www.helsinki.fi/hum/hyfl/Finnish_Phonetics/`, 2000.

[21] D. Childers, R. Cox, R. DeMori, S. Furui, B. Juang, J. Mariani, P. Price, S. Sagayama, M. Sondhi, and R. Weischedel. The past, present, and future of speech processing. *IEEE Signal Processing*, 15(3):24–48, 1998.

[22] W. V. Kempelen. *La mechanisme de la parole, suivi de la description d'une machine parlante.* J. V. Degen, 1791.

[23] H. Dudley. Synthesis speech. *Bell Labs. Record*, 15:98–102, 1936.

[24] B. Christensen, J. Maurer, M. Nash, and E. Vanlandingham. Accessing the internet via the human voice. `http://www.stanford.edu/~jmaurer/homepage.htm`.

[25] K. H. Davis, R. Biddulph, and S. Balashek. Automatic recognition of spoken digits. *Journal of the Acoustical Society of America*, 24:637–642, 1952.

[26] H. F. Olson and H. Belar. Phonetic typewriter. *Journal of the Acoustical Society of America*, 28(6):1072–1081, 1956.

[27] L. E. Baum and T. Petrie. Statistical inference for probabilistic functions of finite state Markov chains. *Ann. Math. Stat.*, 37:1554–1563, 1966.

[28] L. E. Baum and J. A. Egon. An inequality with applications to statistical estimation for probabilistic functions of a Markov process and to a model for ecology. *Bull. Amer. Meteorol. Soc.*, 73:360–363, 1967.

[29] L. E. Baum and G. R. Sell. Growth functions for transformations on manifolds. *Pac. J. Math.*, 27(2):211–227, 1968.

[30] L. E. Baum, T. Petrie, G. Soules, and N. Weiss. A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains. *Ann. Math. Stat.*, 41(1):164–171, 1970.

[31] L. E. Baum. An inequality and associated maximization technique in statistical estimation for probabilistic functions of Markov processes. *Inequalities*, 3:1–8, 1972.

[32] J. K. Baker. The dragon system–an overview. *IEEE Trans. Acoust. Speech Signal Processing*, ASSP-23(1):24–29, Feb 1975.

[33] F. Jelinek. A fast sequential decoding algorithm using a stack. *IBM J. Res. Develop.*, 13:675–685, 1969.

[34] F. Jelinek, L. R. Bahl, and R. L. Mercer. Design of a linguistic statistical decoder for the recognition of continuous speech. *IEEE Trans. Information Theory*, IT-21:250–256, 1975.

[35] T. Kohonen, G. Nèmeth, K.-J. Bry, M. Jalanko, and H. Riittinen. Classification of phonemes by learning subspaces. Technical Report TKK-F-A348, Helsinki University of Technology, 1978.

[36] M. Jalanko. *Studies of Learning Projective Methods in Automatic Speech Recognition.* PhD thesis, Helsinki University of Technology, 1980.

[37] T. Altosaar and M. Karjalainen. Diphone-based speech recognition using time-event neural networks. In *Proc. of ICSLP'92*, pages 979–982, 1992.

[38] M. Kurimo. *Using self-organizing maps and learning vector quantization for mixture density Hidden Markov Models.* PhD thesis, Helsinki University of Technology, 1997.

[39] Lingsoft speech recognizer. `http://speech.lingsoft.fi/en/lssr.html`.

[40] Philips speech recognition software. `http://www.speech.philips.com/wr/srs.asp`.

[41] O. Viikki. *Adaptive Methods for Robust Speech Recognition.* PhD thesis, Tampere University of Technology, 1999.

[42] C. Becchetti and L. P. Ricotti. *Speech Recognition, Theory and C++ Implementation.* John Wiley & Sons, 1999.

[43] H. W. Strube. Linear prediction on a warped frequency scale. *J. Acoust. Soc. Am.*, 68(4):1071–1076, 1980.

[44] U. K. Laine, M. Karjalainen, and T. Altosaar. Warped linear prediction (WLP) in speech and audio processing. In *Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing 1994*, volume 3, pages 349–352, 1994.

[45] P. Boda. *Psychoacoustical Considerations in Speech Analysis and Recognition.* Licentiate's thesis, Helsinki University of Technology, 1995.

[46] M. Ursin. WLP-pohjaisten esikäsittelymenetelmien vertailu perinteisiin mel-kepstri -pohjaisiin puheentunnistuksessa. Informaatiotekniikan erikoistyö, Helsinki University of Technology, 2001. `http://www.hut.fi/~mtursin/et.ps.gz`.

[47] A. Härmä, M. Karjalainen, L. Savioja, V. Välimäki, U. K. Laine, and J. Huopaniemi. Frequency-warped signal processing for audio applications. *Journal of the Audio Engineering Society*, 48(11):1011–1044, 2000.

[48] H. Hermansky. Perceptual linear predictive (PLP) analysis of speech. *J. Acoust. Soc. Am.*, 87(4):1738–1752, April 1990.

[49] K.-F. Lee. Context-dependent phonetic hidden Markov models for speaker-independent continuous speech recognition. *IEEE Trans. Acoust. Speech Signal Processing*, 38(4):599–609, April 1990.

[50] HTK hidden Markov model toolkit – speech recognition research toolkit. `http://htk.eng.cam.ac.uk/`.

[51] L. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

[52] B. H. Juang and L. Rabiner. Hidden Markov models for speech recognition. *Technometrics*, 33(3):251–272, 1991.

[53] R. L. Bahl, Brown P. F., P. V. de Souza, and R. L. Mercer. Maximum mutual information estimation of hidden Markov model parameters for speech recognition. In *Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing 1986*, pages 49–52, 1986.

[54] L. R. Bahl, P. F. Brown, P. V. De Souza, and R. L. Mercer. Acoustic markov models used in the Tangora speech recognition system. In *Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing 1988*, volume 1, pages 497–500, 1988.

[55] O. Fujimura. Syllable as a unit of speech recognition. *IEEE Trans. Acoust. Speech Signal Processing*, ASSP-23(1):82–87, Feb 1975.

[56] K. Wiik. Suomen tavuista. *Virittäjä*, pages 265–278, 1977.

[57] R. M. Schwartz, Y. Chow, S. Roucos, M. Krasner, and J. Makhoul. Improved hidden Markov modeling of phonemes for continuous speech recognition. In *Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing 1984*, pages 35.6.1–35.6.4, 1984.

[58] N. R. Dixon and H. F. Silverman. The 1976 modular acoustic processor (map). *IEEE Trans. Acoustics, Speech and Signal Processing*, pages 367–379, Oct 1977.

[59] R. M. Schwartz, J. Klovstad, J. Makhoul, and J Sorensen. A preliminary design of a phonetic vocoder based on a diphone model. In *Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing 1980*, pages 32–35, 1980.

[60] R. M. Schwartz, Y. Chow, O. Kimball, S. Roucos, M. Krasner, and J. Makhoul. Context-dependent modeling for acoustic-phonetic recognition of continuous speech. In *Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing 1985*, pages 31.3.1–31.3.4, 1985.

[61] S. J. Young, J. J. Odell, and P. C. Woodland. Tree-based state tying for high accuracy acoustic modelling. In *Proc. of ARPA Workshop on Human Language Technology*, pages 286–291, 1994.

[62] A.-M. Derouault. Context-dependent phonetic Markov models for large vocabulary speech recognition. In *Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing 1987*, pages 360–363, 1987.

[63] L. Deng, M. Lennig, V. N. Gupta, and P. Mermelstein. Modeling acoustic-phonetic detail in an HMM-based large vocabulary speech recognizer. In *Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing 1988*, pages 509–512, 1988.

[64] K.-F. Lee, H.-W. Hon, and R. Reddy. An overview of the SPHINX speech recognition system. *IEEE Trans. Acoust. Speech Signal Processing*, 38(1):35–45, April 1990.

[65] W. Ward and S. Issar. Integrating semantic constraints into the SPHINX-II recognition search. In *Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing 1994*, volume 2, pages II/17–II/19, 1994.

[66] F. Alleva, X. Huang, and M.-Y. Hwang. An improved search algorithm using incremental knowledge for continuous speech recognition. In *Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing 1993*, volume 2, pages 307–310, 1993.

[67] P. Placepway, S. Chen, M. Eskenazi, U. Jain, V. Parikh, B. Raj, M. Ravishankar, R. Rosenfeld, K. Seymore, M. Siegler, R. Stern, and E. Thayer. The 1996 Hub-4 Sphinx-3 system. In *Proc. DARPA Speech Recognition Workshop '97*, 1997. `http://www-2.cs.cmu.edu/~pwp/papers/h496_system/H496CMU.HTM`.

[68] M.-Y. Hwang. *Subphonetic Acoustic Modeling for Speaker-Independent Continuous Speech Recognition*. PhD thesis, Carnegie Mellon University, 1993.

[69] T. Hain, P.C. Woodland, G. Evermann, and D. Povey. New features in the CU-HTK system for transcription of conversational telephone speech. In *Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing 2001*, volume 1, pages 57–60, 2001.

[70] T. Hain, P.C. Woodland, G. Evermann, and D. Povey. The CU-HTK march 2000 HUB5E transcription system. In *Proc. Speech Transcription Workshop 2000*, volume 1, 2000.

[71] P. C. Woodland, S. Kapadia, H. J. Nock, and S. J. Young. The HTK system for the march 1997 Hub 5 evaluations. In *Presented at the DARPA Hub5E Conversational Speech Recognition Workshop*, May 13–15, 1997.

[72] T. Hain, P. C. Woodland, T. R. Niesler, and E. W. D. Whittaker. The 1998 HTK system for transcription of conversational telephone speech. In *Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing 1999*, volume 1, pages 57–60, 1999.

[73] X. Luo and F. Jelinek. Probabilistic classification of HMM states for large vocabulary continuous speech recognition. In *Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing 1999*, pages 2044–2047, 1999.

[74] M. J. F. Gales. Maximum likelihood linear transformations for HMM-based speech recognition. *Computer Speech & Language*, 12:75–98, 1998.

[75] L. Mangu, E. Brill, and A. Stolcke. Finding consensus among words: Lattice-based word error minimization. In *Proc. of EUROSPEECH'99*, pages 495–498, 1999.

[76] N. Volk. Lavennin. `http://www.ling.helsinki.fi/suopuhe/lavennin/index.shtml`.

[77] S. Young, G. Evermann, D. Kershaw, G. Moore, J. Odell, D. Ollason, V. Valtchev, and P. Woodland. The HTK book (for HTK version 3.1). `http://htk.eng.cam.ac.uk/docs/docs.shtml`.

[78] L. Gillick and S. Cox. Some statistical issues in the comparison of speech recognition algorithms. In *Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing 1989*, pages 532–535, 1989.

[79] G. Moore. Cramming more components onto integrated circuits, 1965. `http://www.intel.com/research/silicon/moorespaper.pdf`.

[80] V. Siivola, M. Kurimo, and K. Lagus. Large vocabulary statistical language modeling for continuous speech recognition in finnish. In *Proc. of EUROSPEECH'01*, pages 737–740, 2001.

Appendix A

Confusion matrices for the different models

In this appendix confusion matrices between phonemes are presented for the MFCC Mon, ToP, PoA, TLo, and THi model sets. The D and I on the axis denote deletion and insertion errors. The /ŋ/ is denoted by the sign §.

As its name suggests, confusion matrices describe the confusion between phonemes. The correct phonemes are presented in the rows and the mistaken phonemes in columns. The more often a phoneme is misrecognized as another, the darker the rectangle in the crossing of the corresponding row and column is (except along the diagonal where correct results are expected).

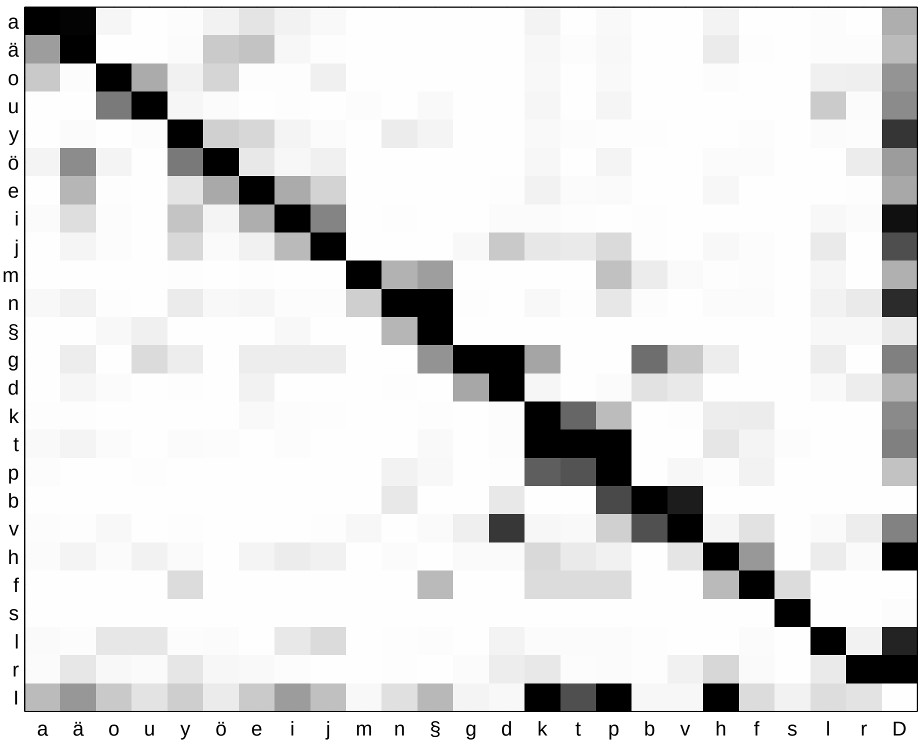

Figure A.1: Confusion matrix for MFCC monophone models – all phonemes.

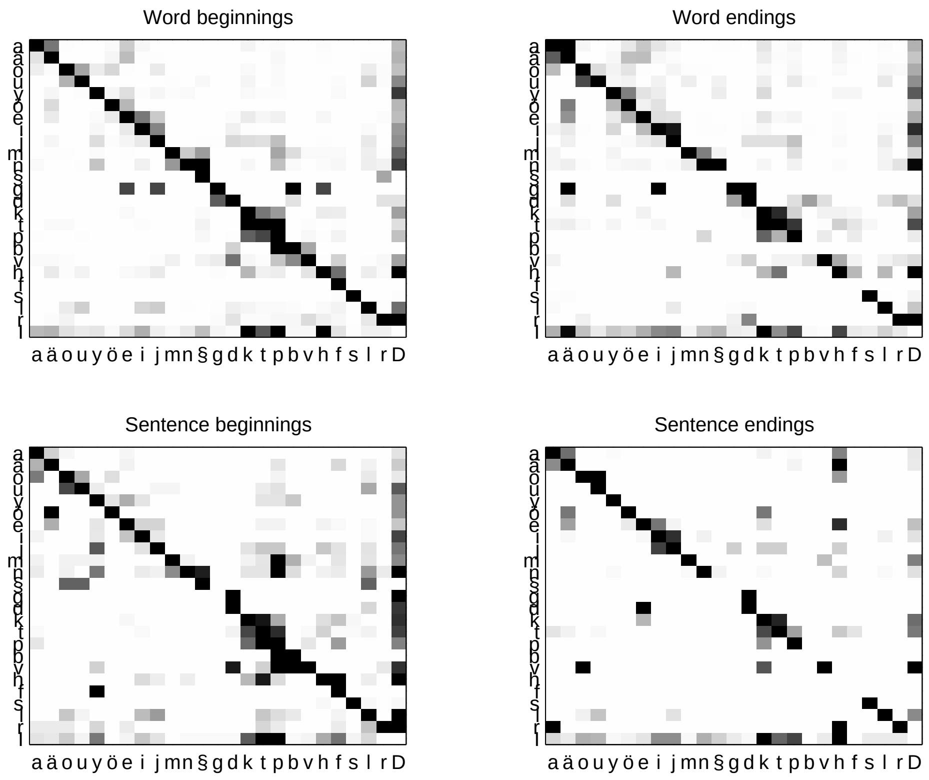

Figure A.2: Word and sentence beginning and ending confusion matrices for MFCC monophone models.

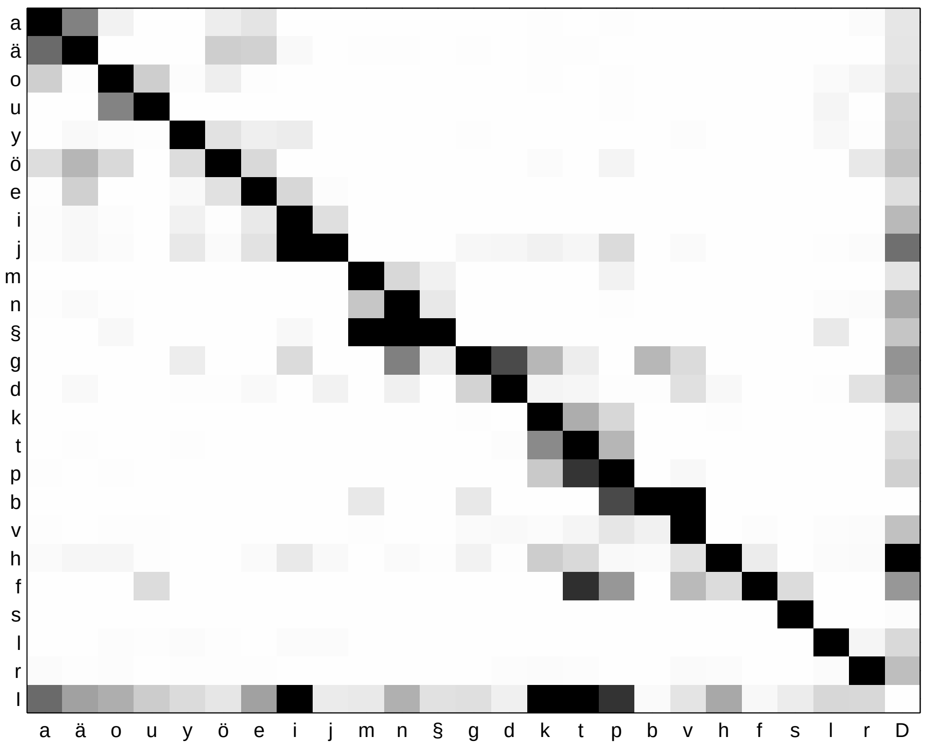

Figure A.3: Confusion matrix for MFCC ToP models – all phonemes.

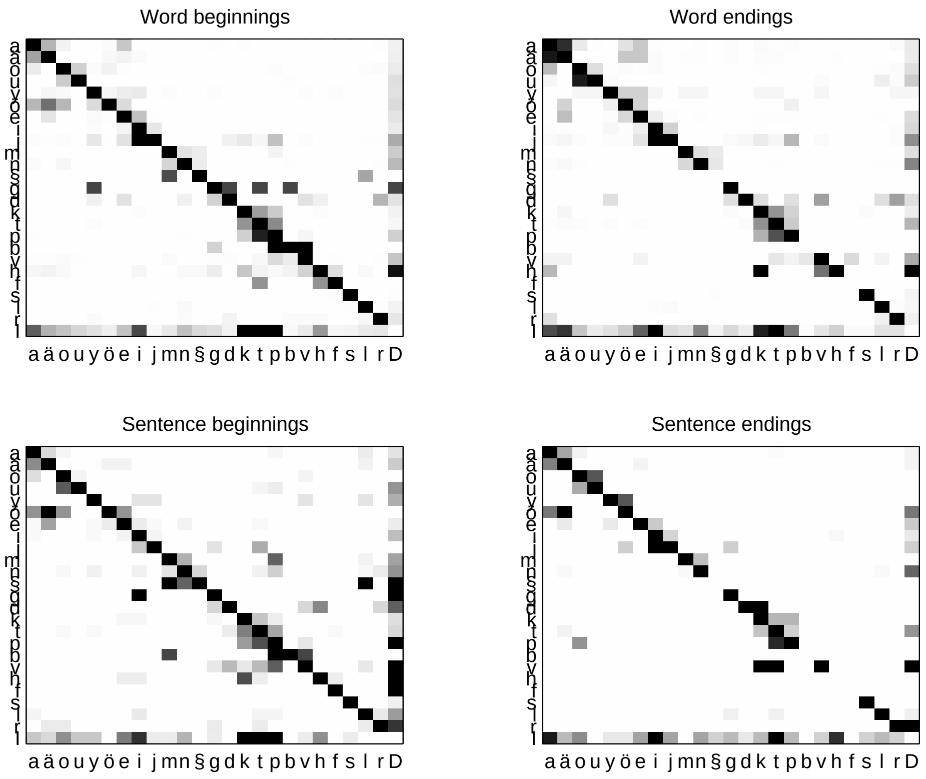

Figure A.4: Word and sentence beginning and ending confusion matrices for MFCC ToP models.

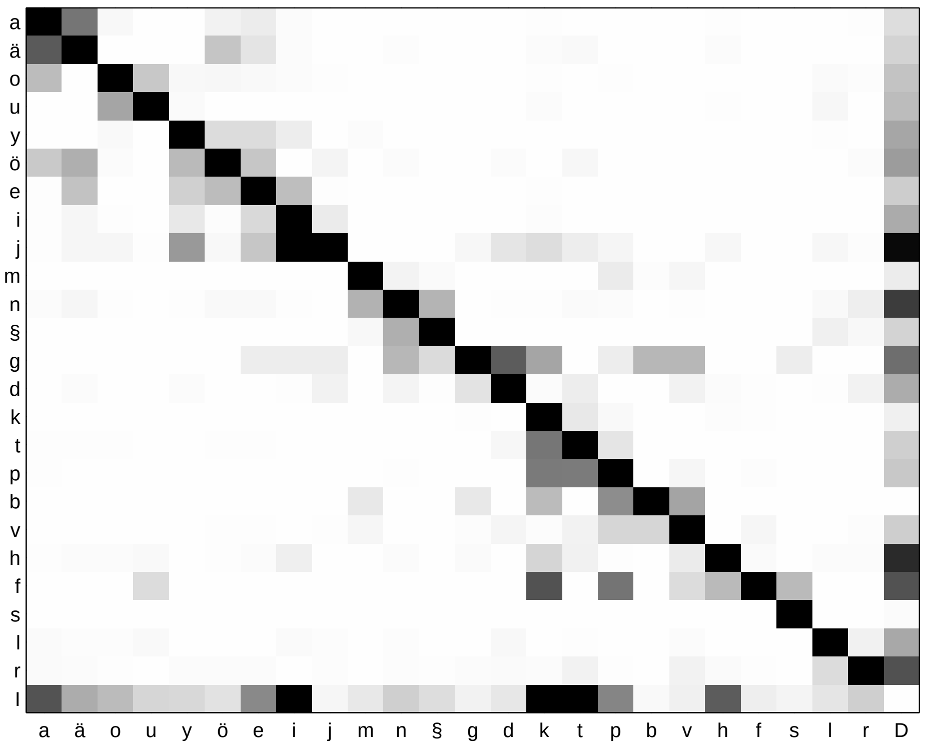

Figure A.5: Confusion matrix for MFCC PoA models – all phonemes.

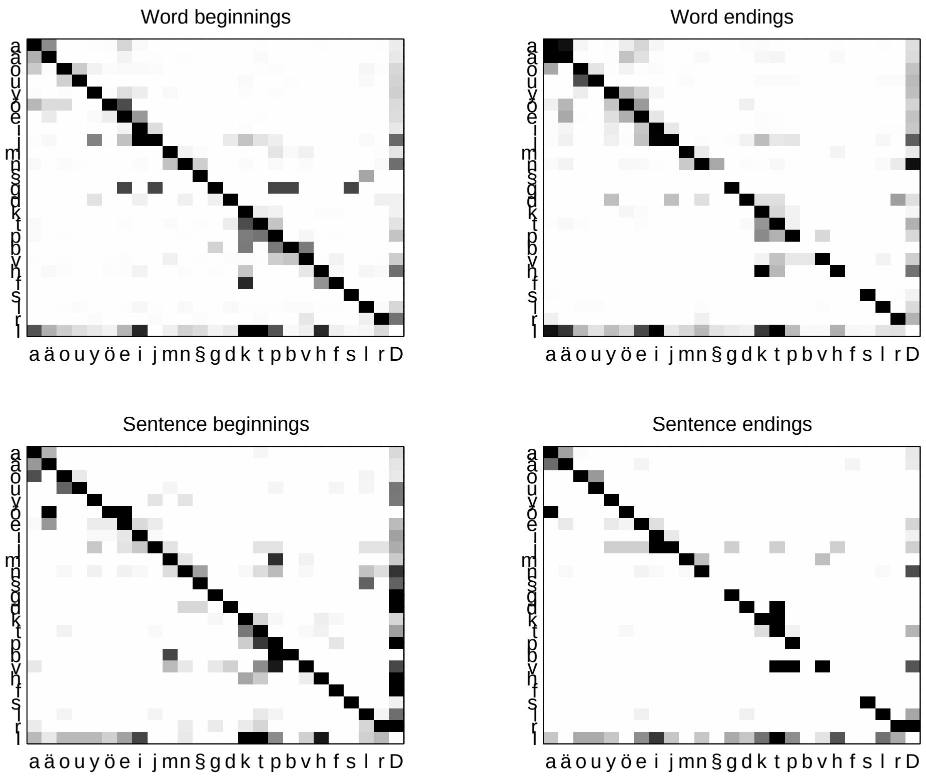

Figure A.6: Word and sentence beginning and ending confusion matrices for MFCC PoA models.

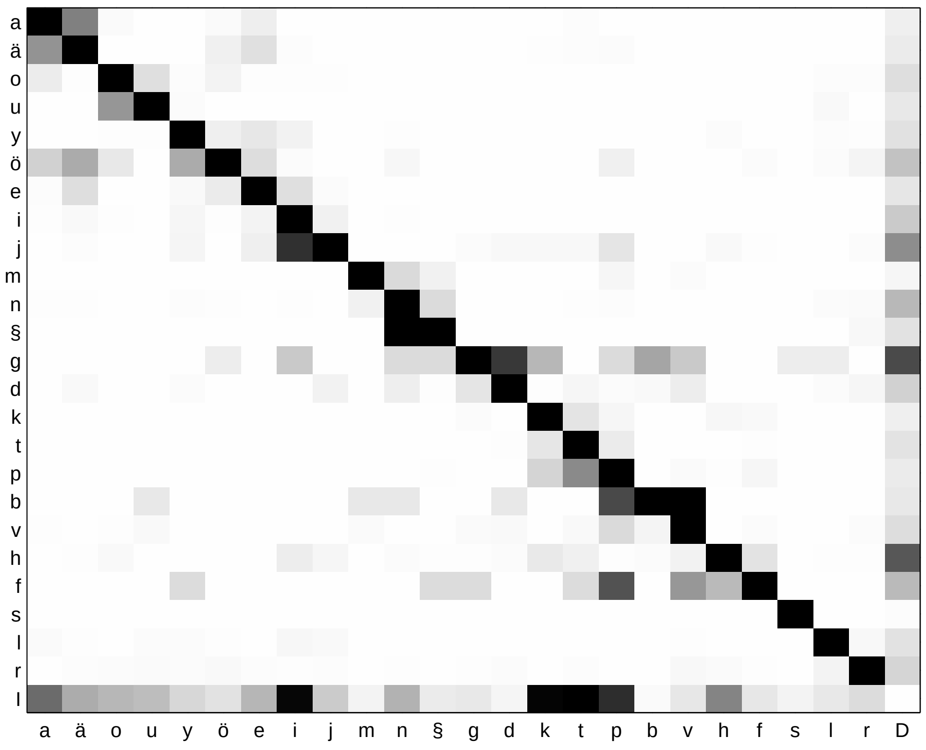

Figure A.7: Confusion matrix for MFCC TLo models – all phonemes.

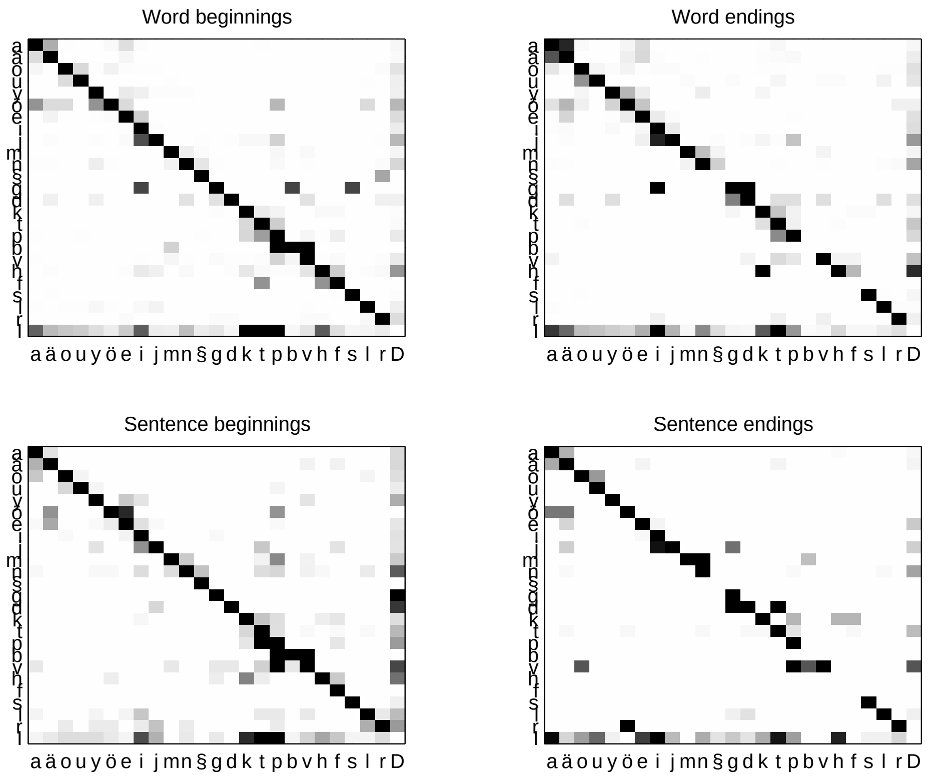

Figure A.8: Word and sentence beginning and ending confusion matrices for MFCC TLo models.

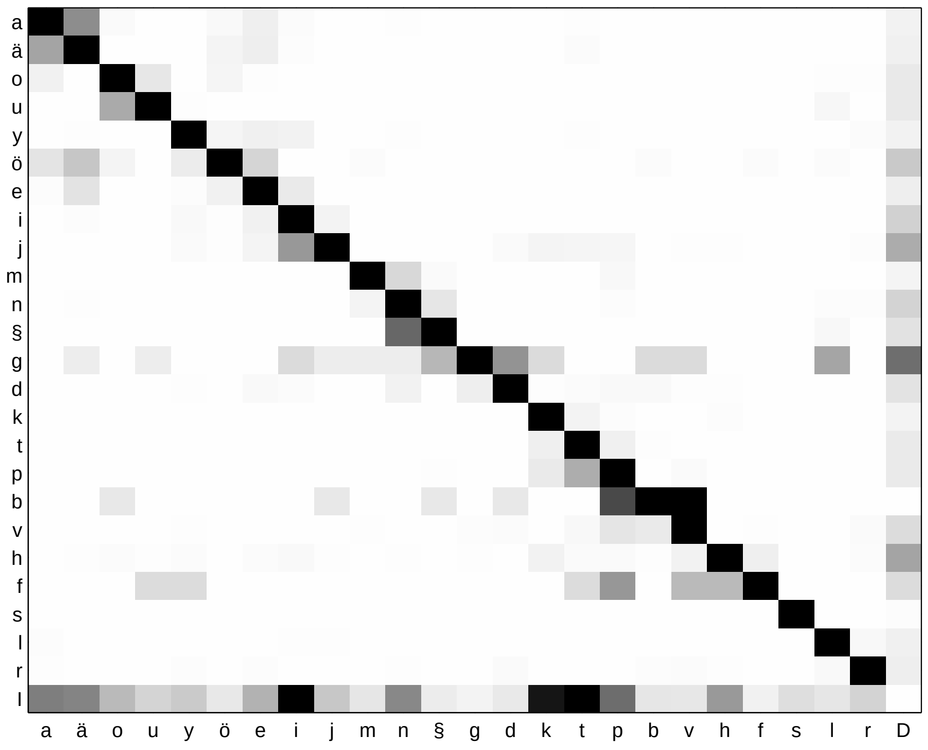

Figure A.9: Confusion matrix for MFCC THi models – all phonemes.

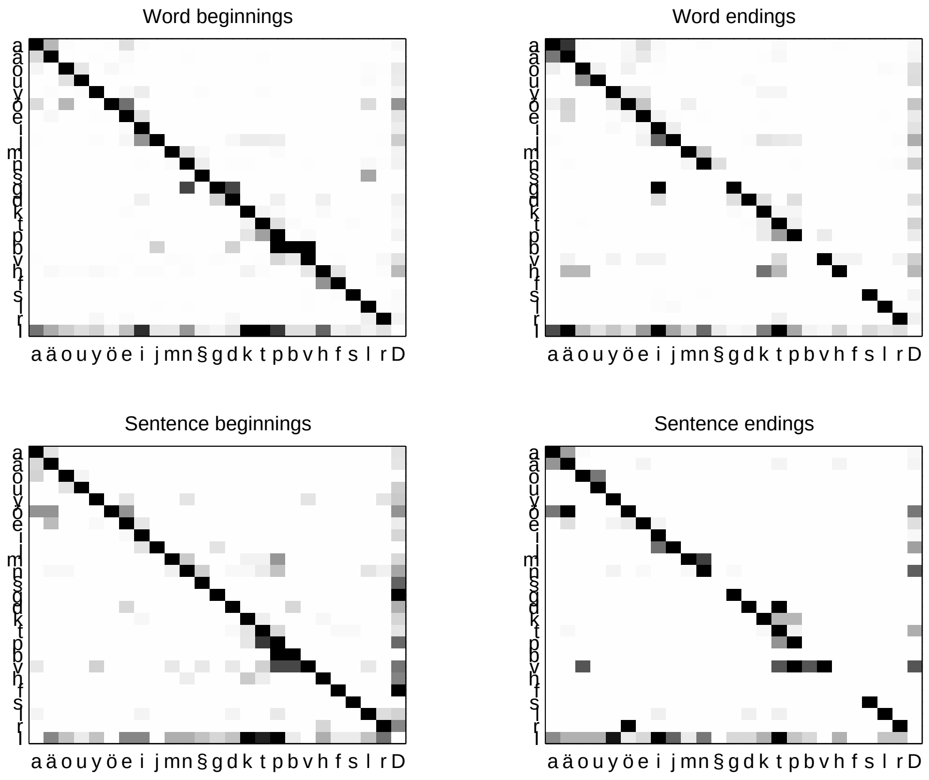

Figure A.10: Word and sentence beginning and ending confusion matrices for MFCC THi models.

Appendix B

Context analysis

In this appendix the context analysis graphs of the different model sets are presented. The figures of each model are divided into eight sections:

a) All kinds of errors with the examined phoneme as the left context
b) All kinds of errors with the examined phoneme as the right context
c) Substitution errors with the examined phoneme as the left context
d) Substitution errors with the examined phoneme as the right context
e) Deletion errors with the examined phoneme as the left context
f) Deletion errors with the examined phoneme as the right context
g) Insertion errors with the examined phoneme as the left context
h) Insertion errors with the examined phoneme as the right context

In the following, the graphs for the different models are presented.

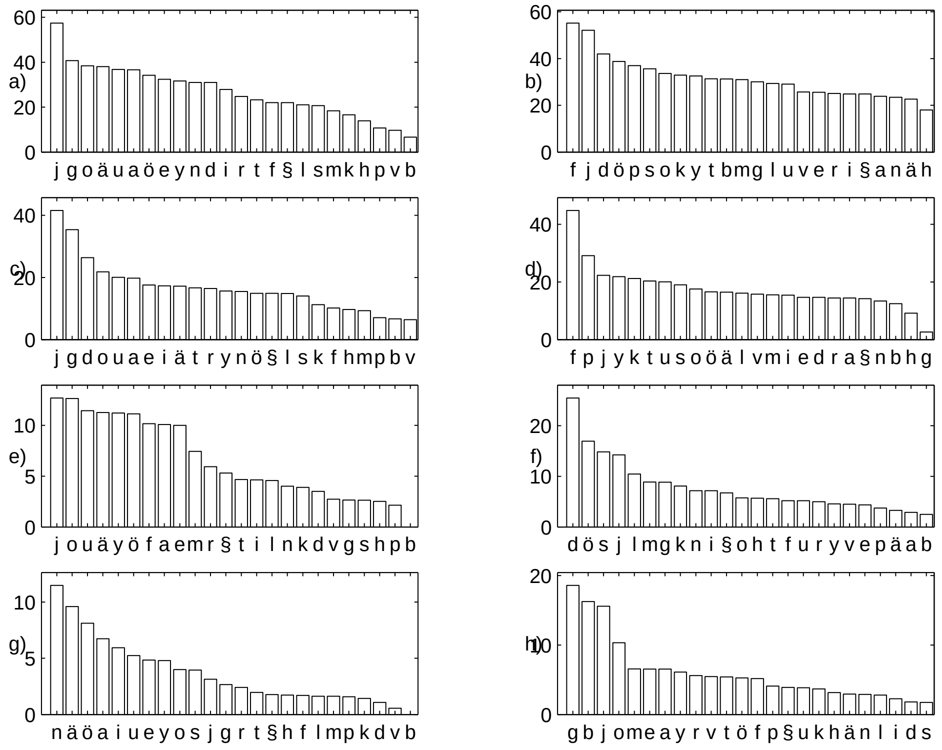

Figure B.1: Error-prone context analysis for the monophone models.

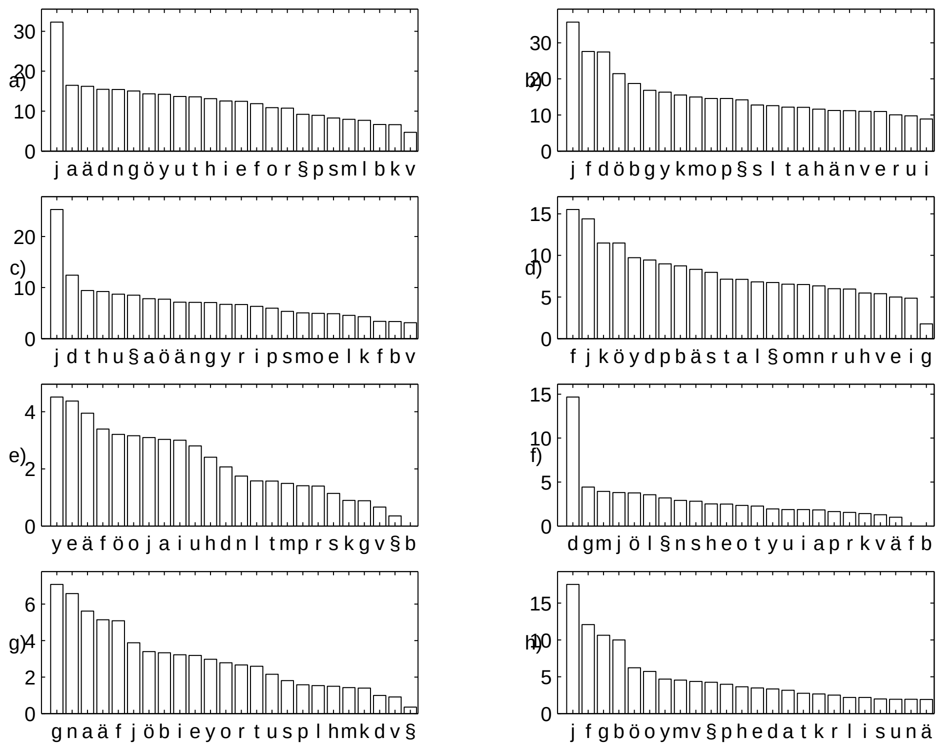

Figure B.2: Error-prone context analysis for the ToP models.

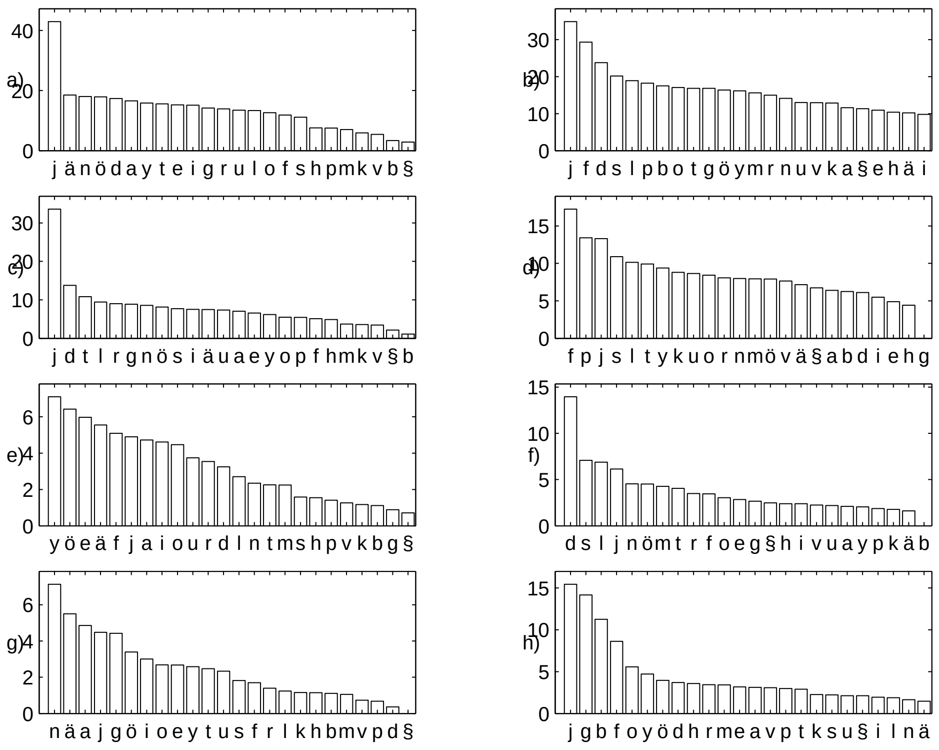

Figure B.3: Error-prone context analysis for the PoA models.

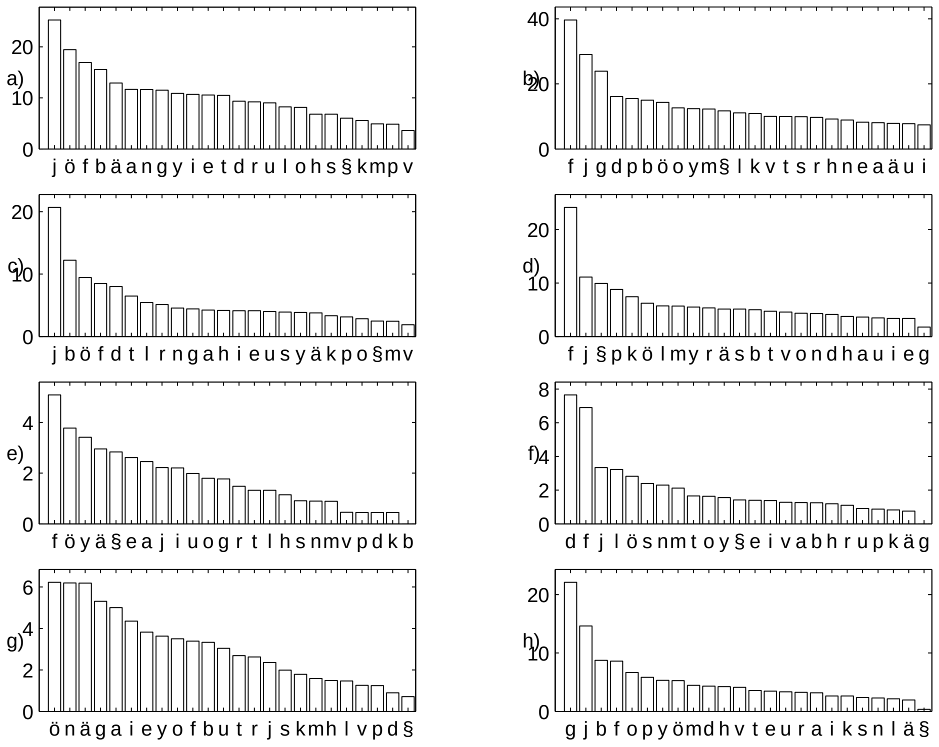

Figure B.4: Error-prone context analysis for the TLo models.

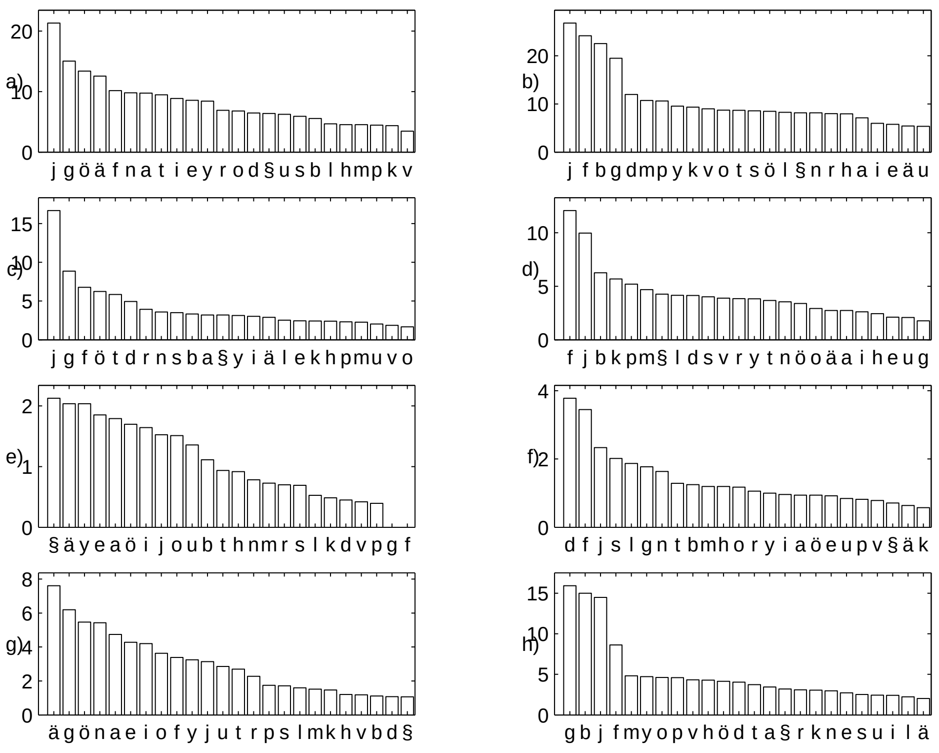

Figure B.5: Error-prone context analysis for the THi models.

Appendix C

Context analysis of errors

This Appendix presents the figures for TLo models used in per-phoneme error analysis. For an explanation of the figures, see section 7.6.

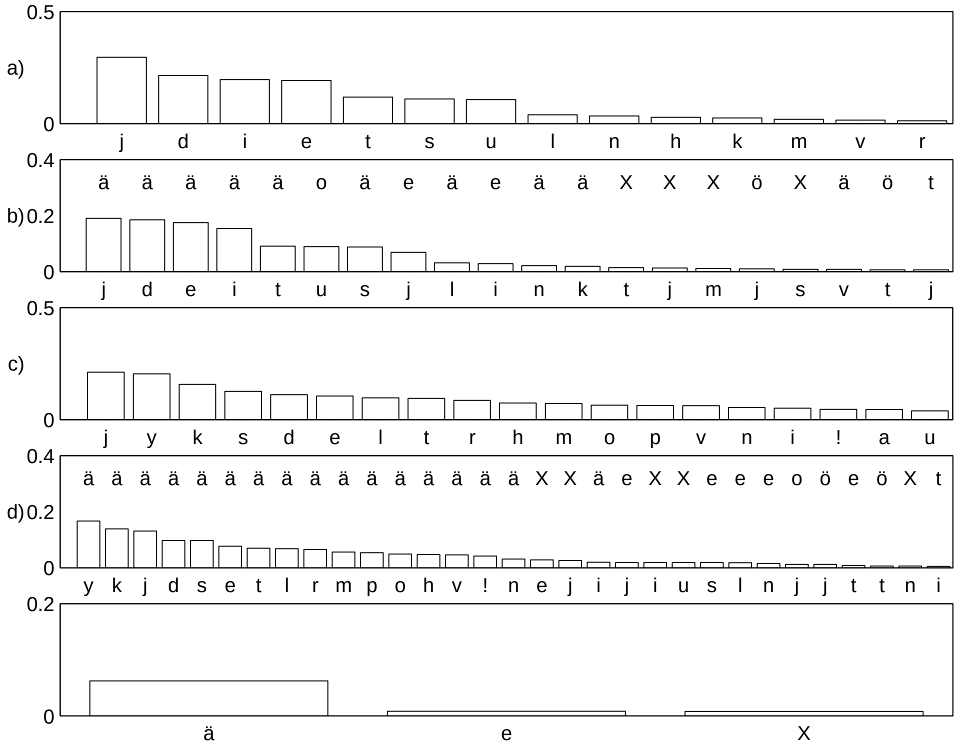

Figure C.1: Context analysis for the phoneme /a/.

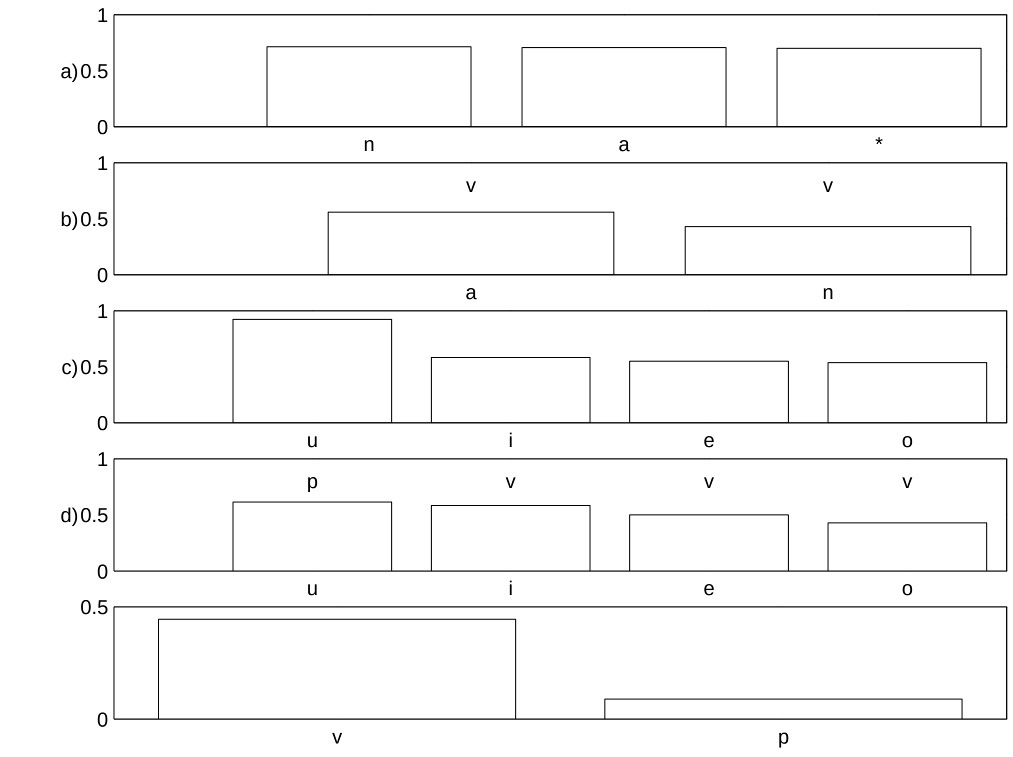

Figure C.2: Context analysis for the phoneme /b/.

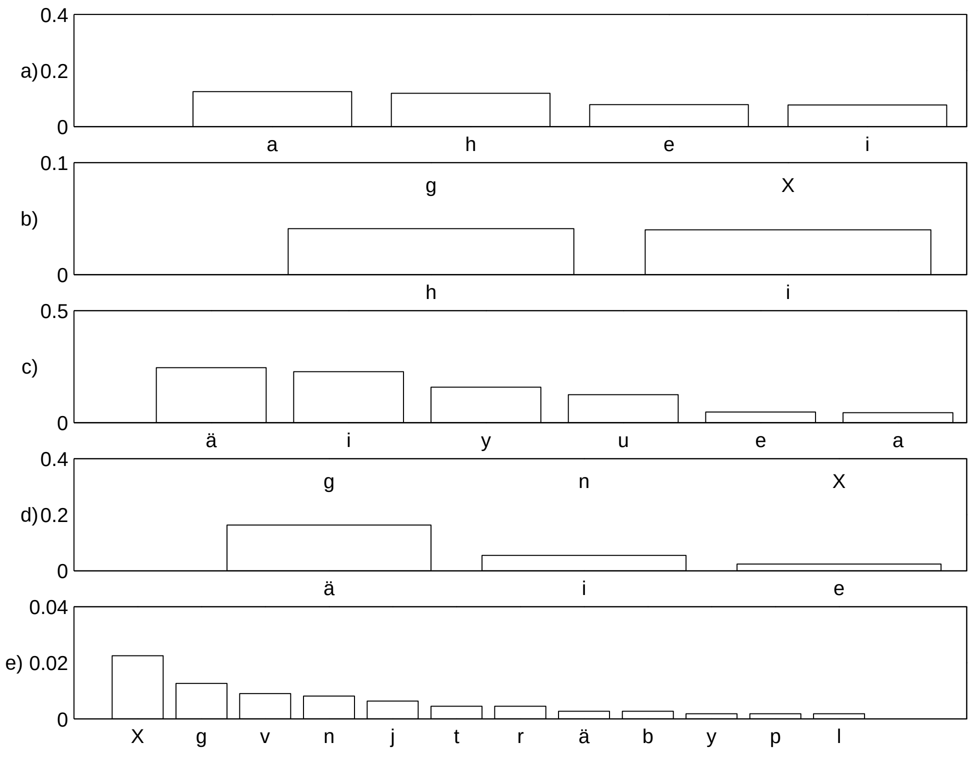

Figure C.3: Context analysis for the phoneme /d/.

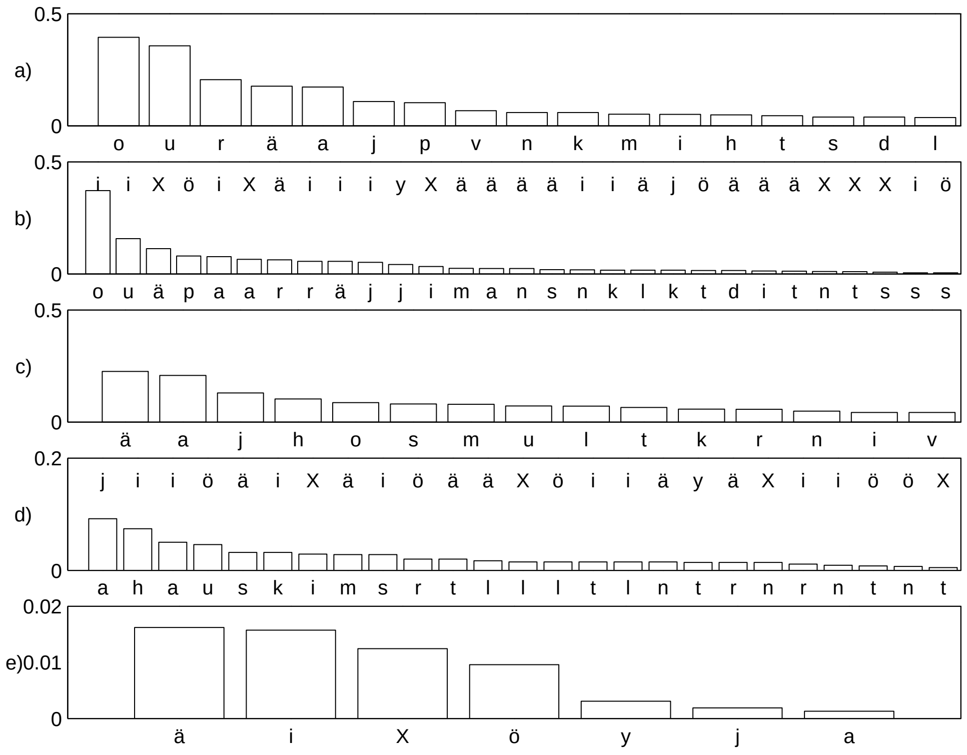

Figure C.4: Context analysis for the phoneme /e/.

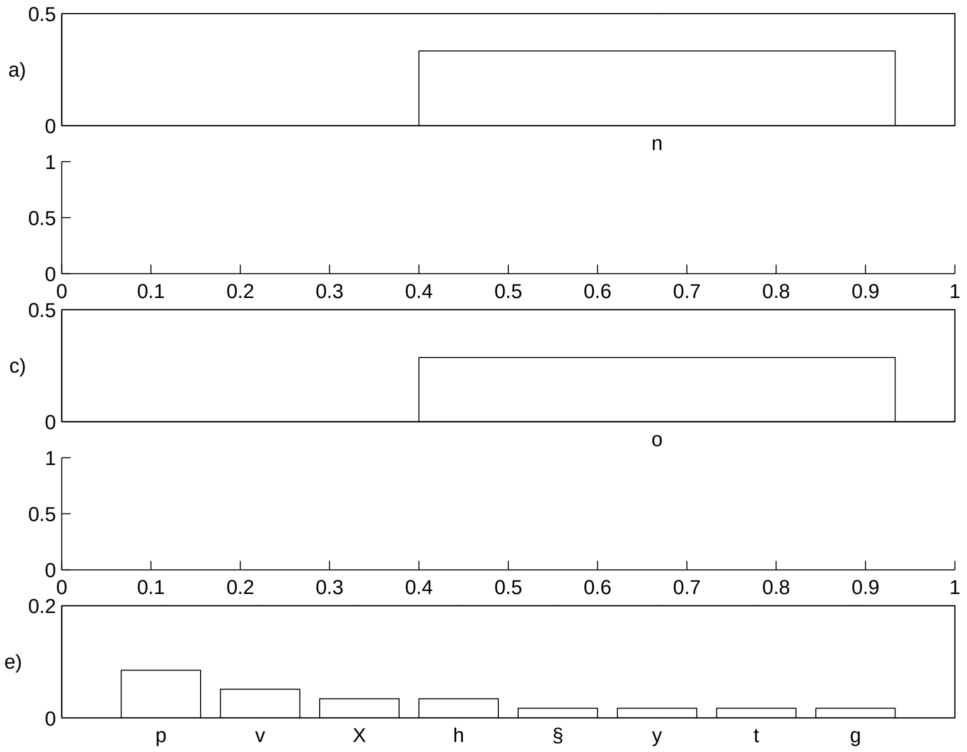

Figure C.5: Context analysis for the phoneme /f/.

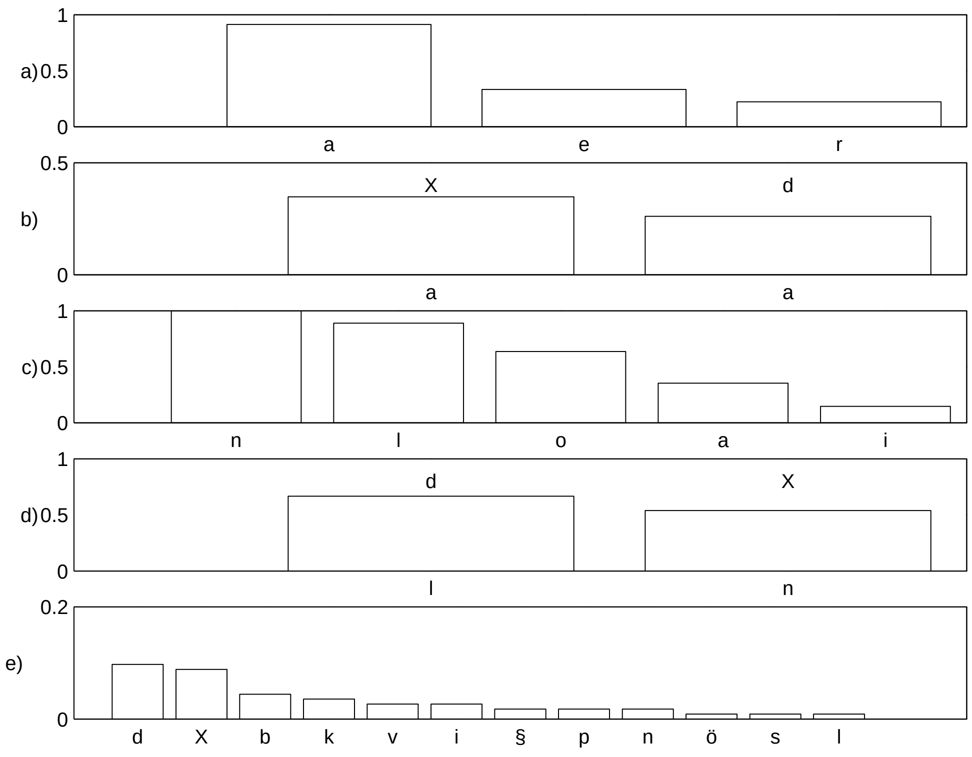

Figure C.6: Context analysis for the phoneme /g/.

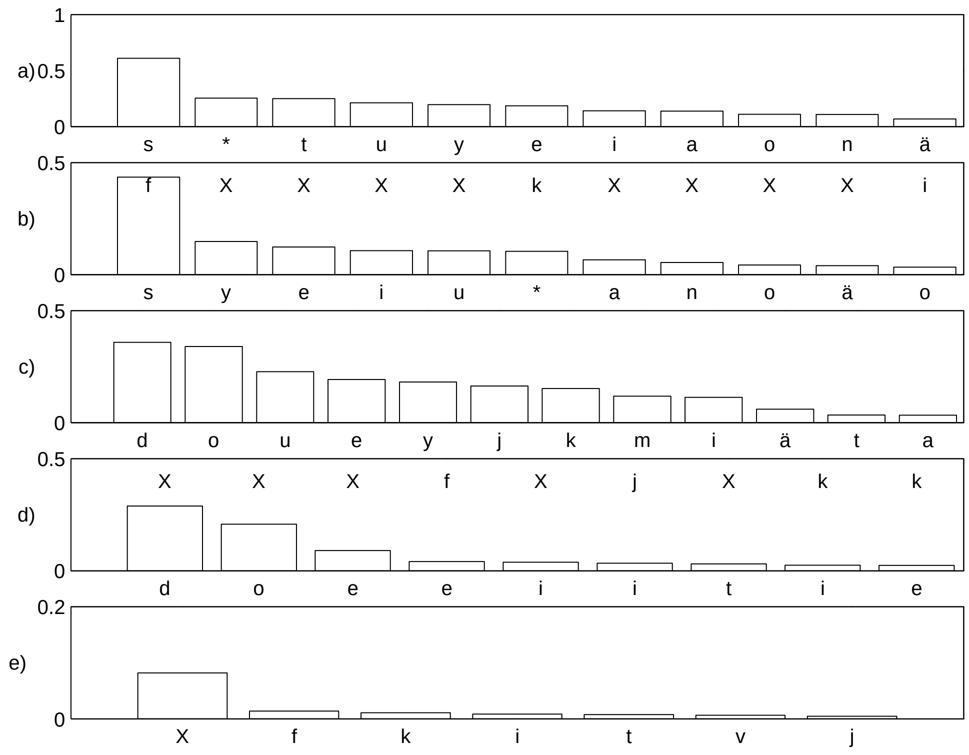

Figure C.7: Context analysis for the phoneme /h/.

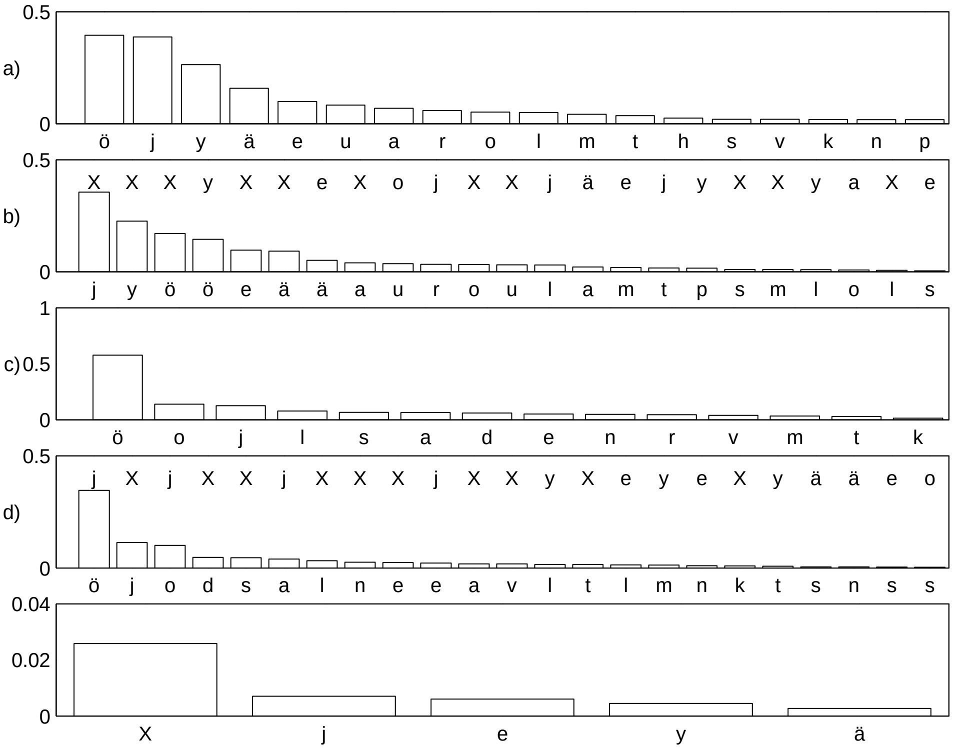

Figure C.8: Context analysis for the phoneme /i/.

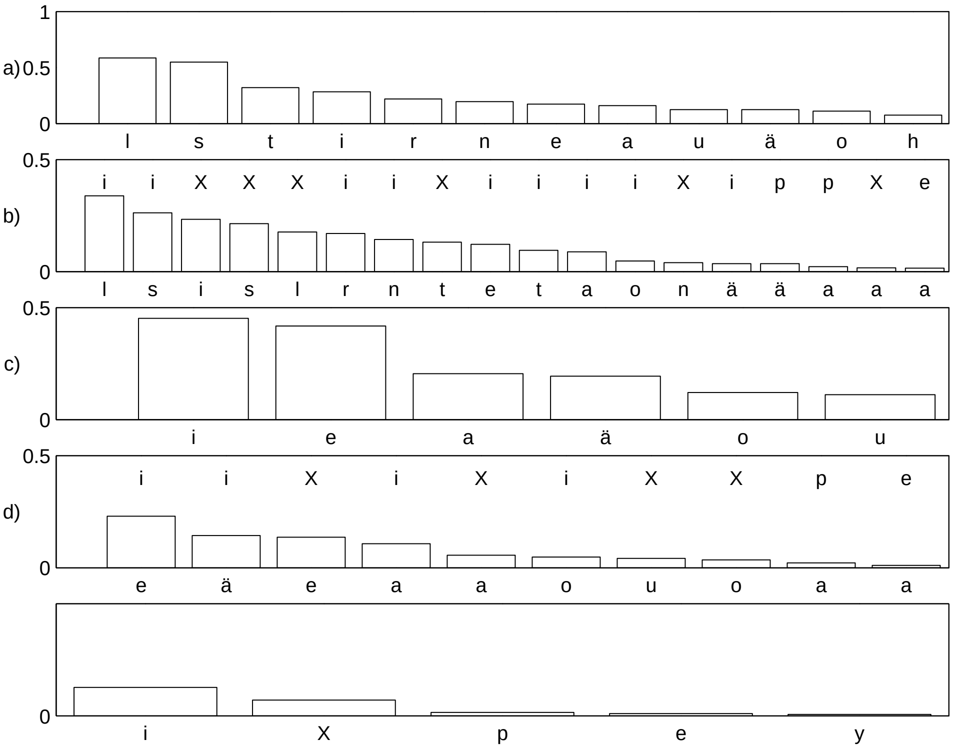

Figure C.9: Context analysis for the phoneme /j/.

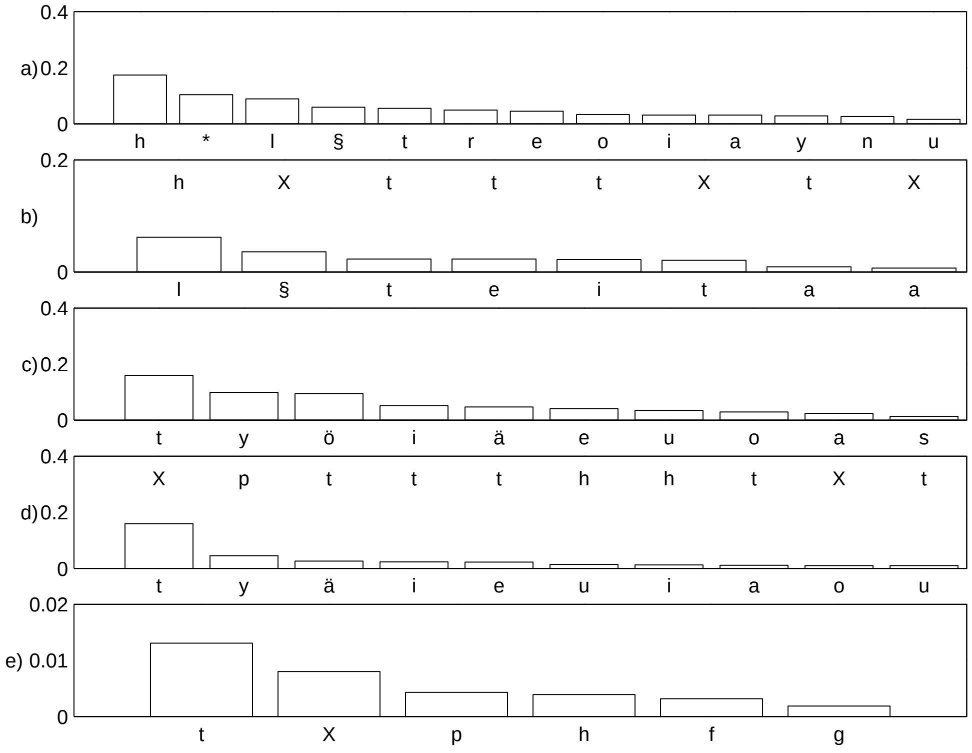

Figure C.10: Context analysis for the phoneme /k/.

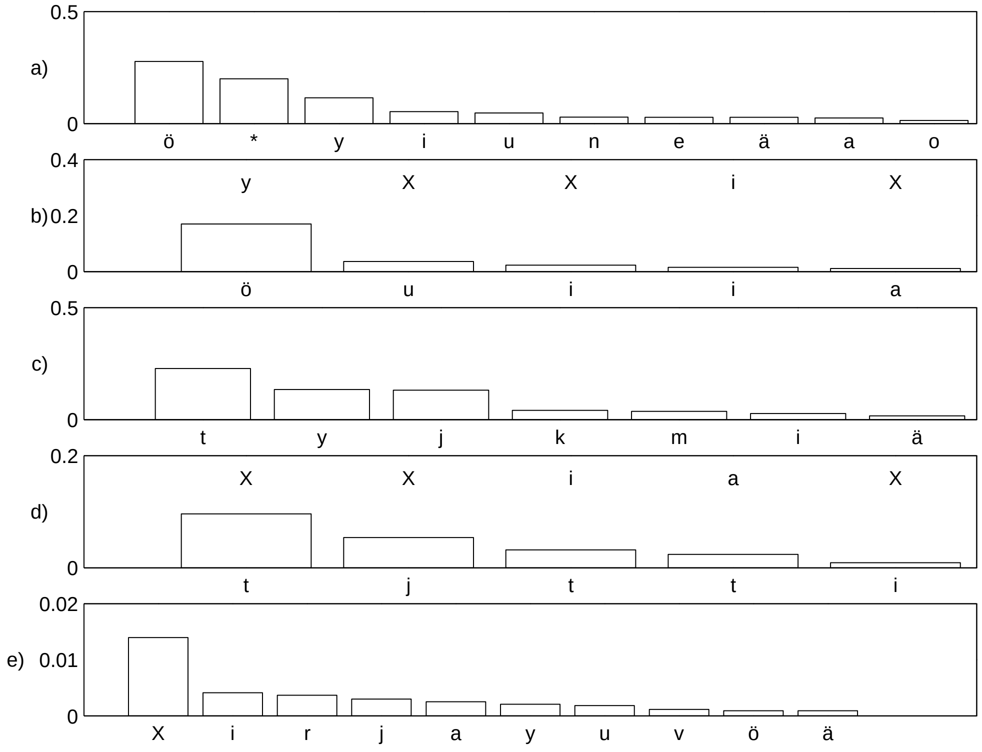

Figure C.11: Context analysis for the phoneme /l/.

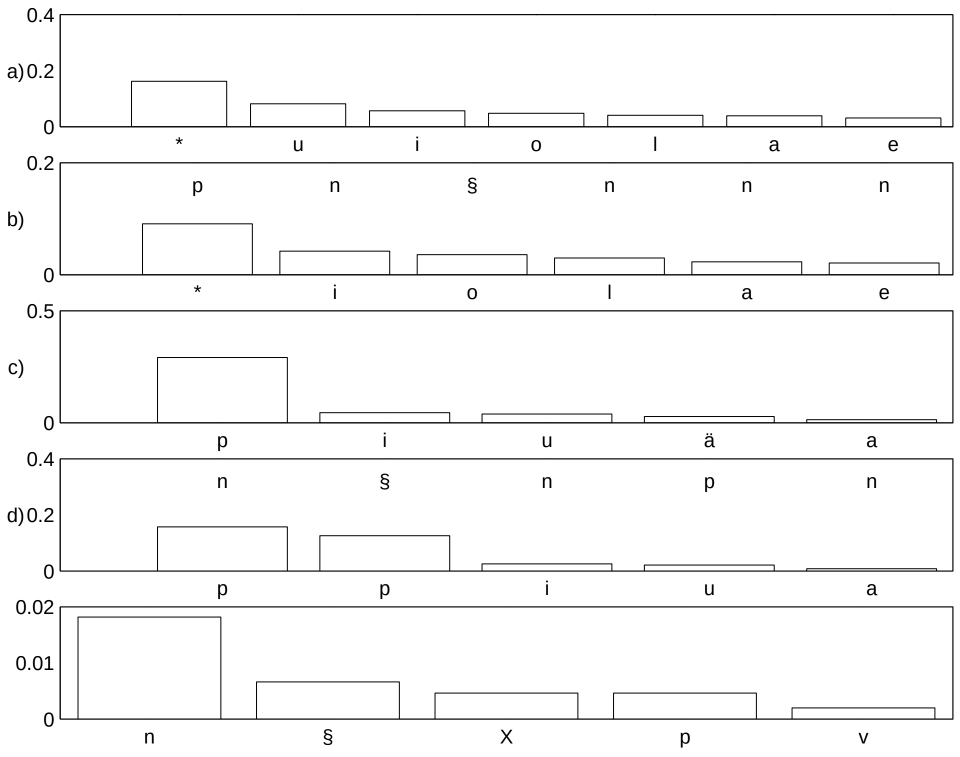

Figure C.12: Context analysis for the phoneme /m/.

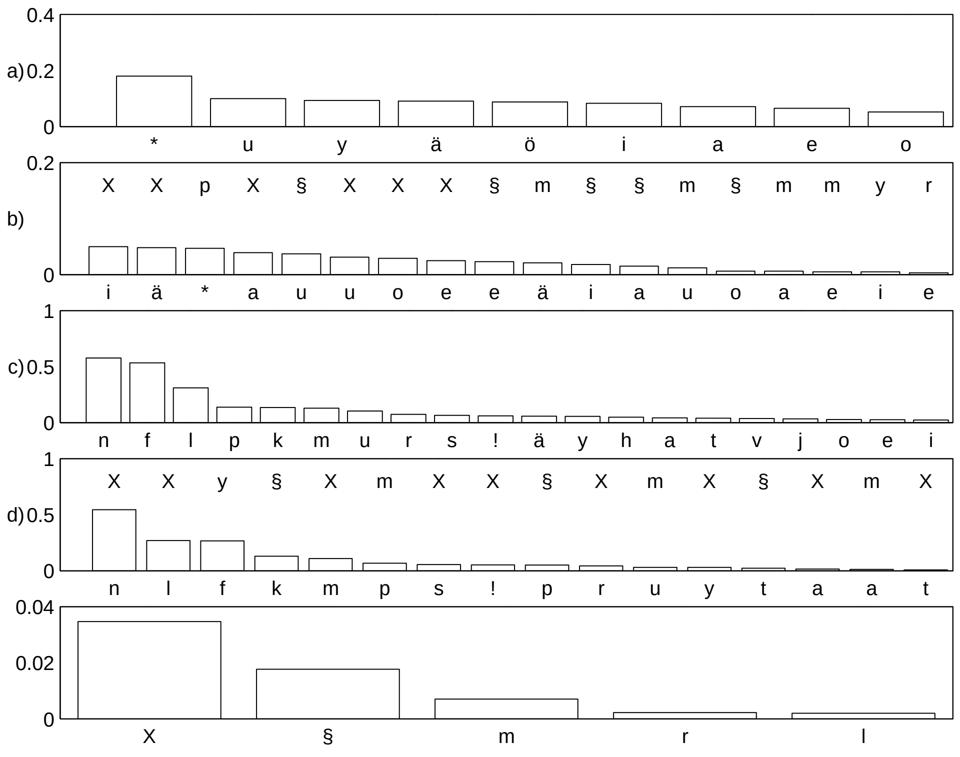

Figure C.13: Context analysis for the phoneme /n/.

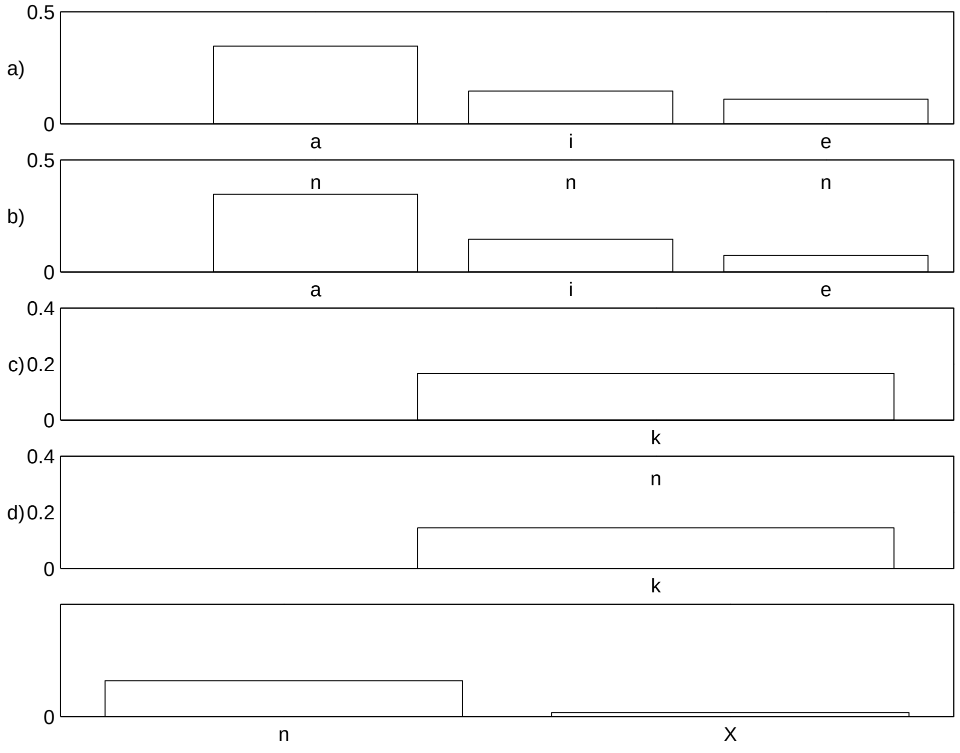

Figure C.14: Context analysis for the phoneme /ŋ/.

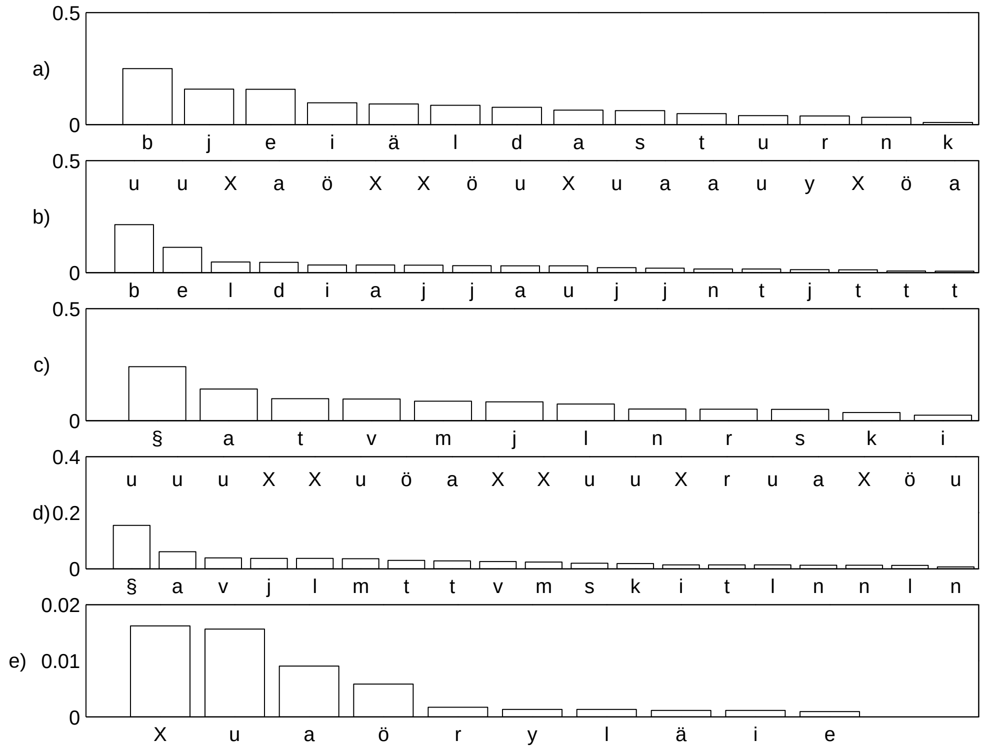

Figure C.15: Context analysis for the phoneme /o/.

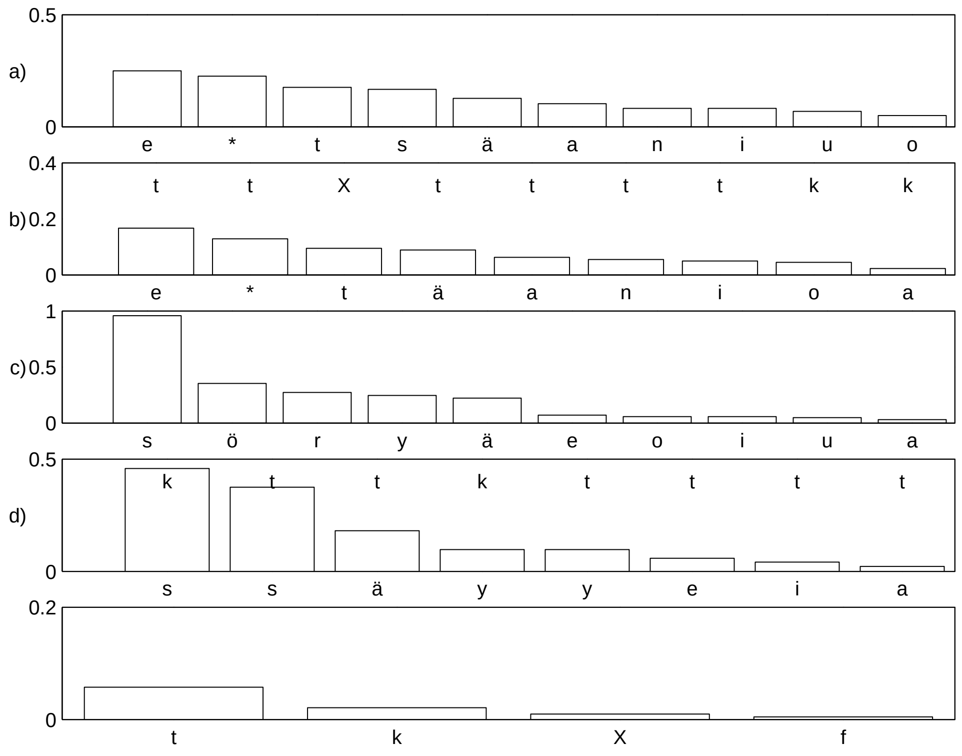

Figure C.16: Context analysis for the phoneme /p/.

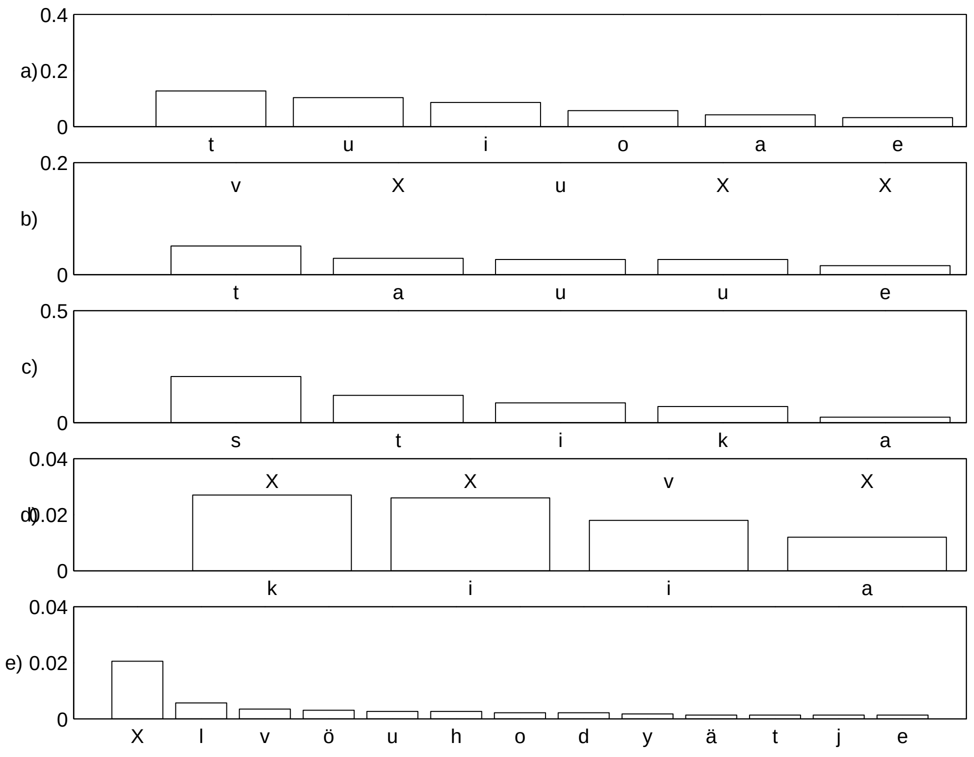

Figure C.17: Context analysis for the phoneme /r/.

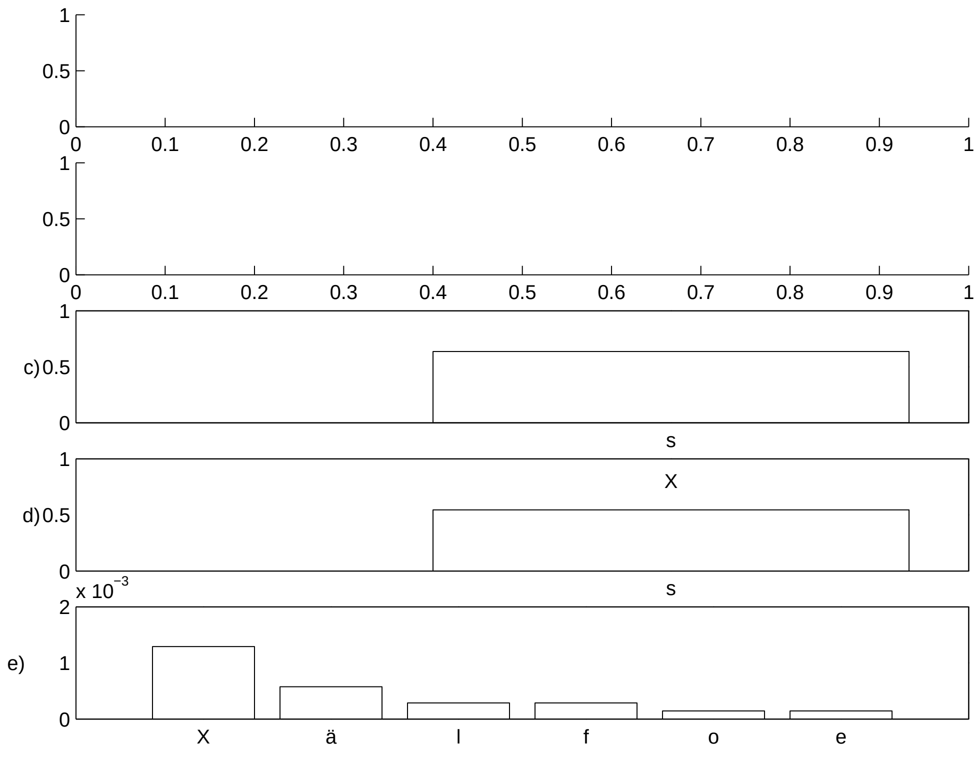

Figure C.18: Context analysis for the phoneme /s/.

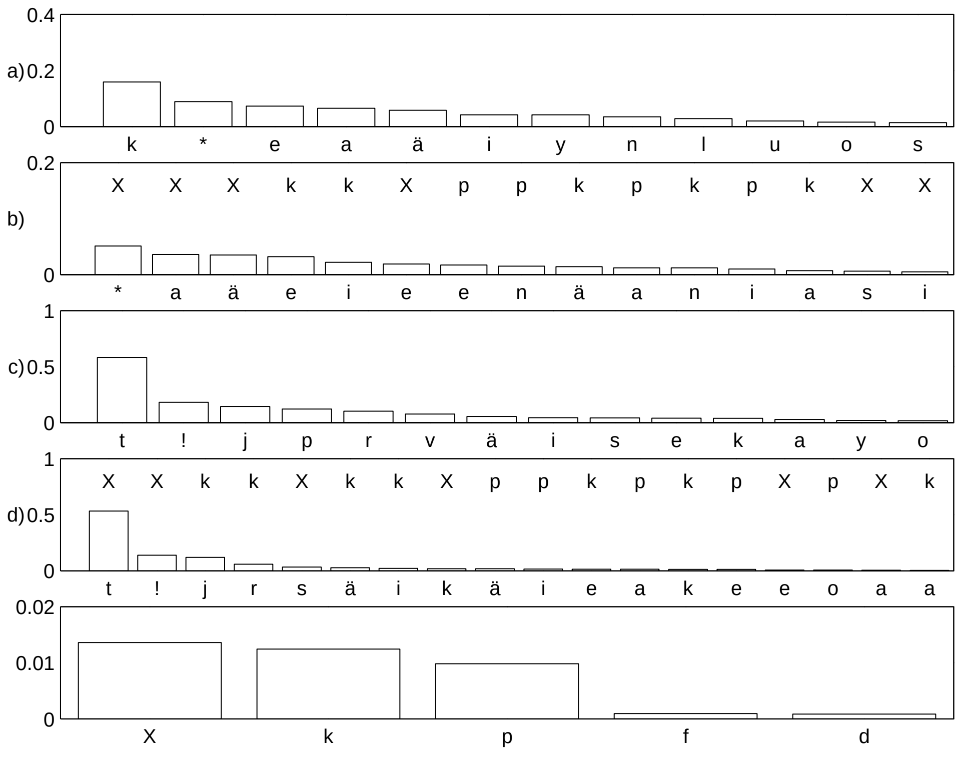

Figure C.19: Context analysis for the phoneme /t/.

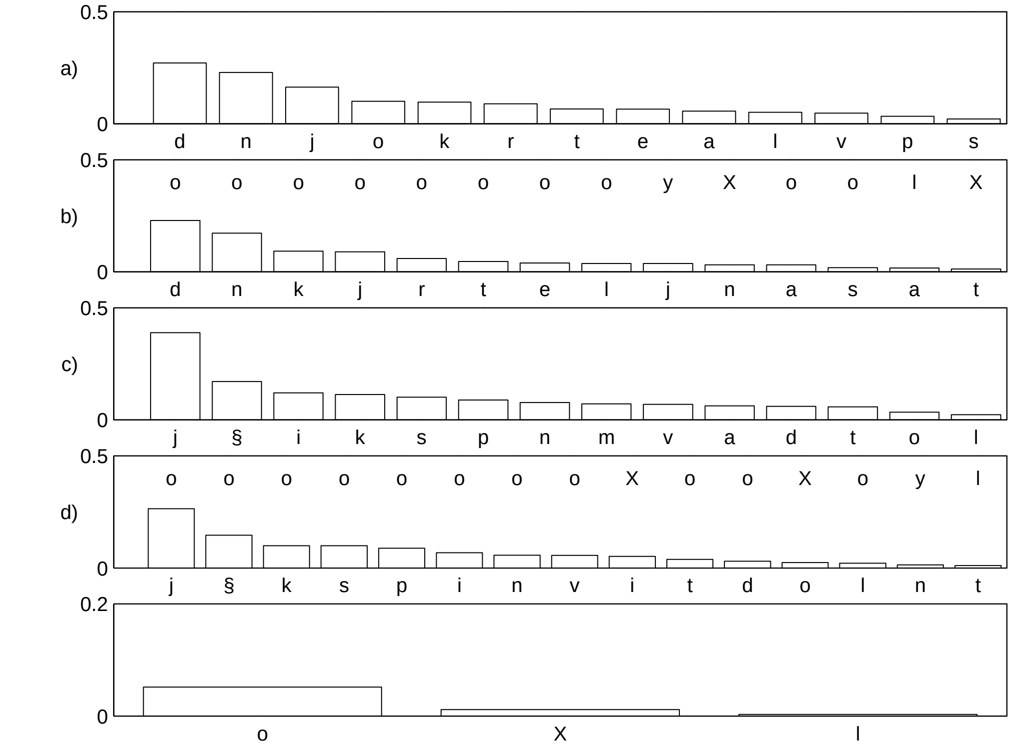

Figure C.20: Context analysis for the phoneme /u/.

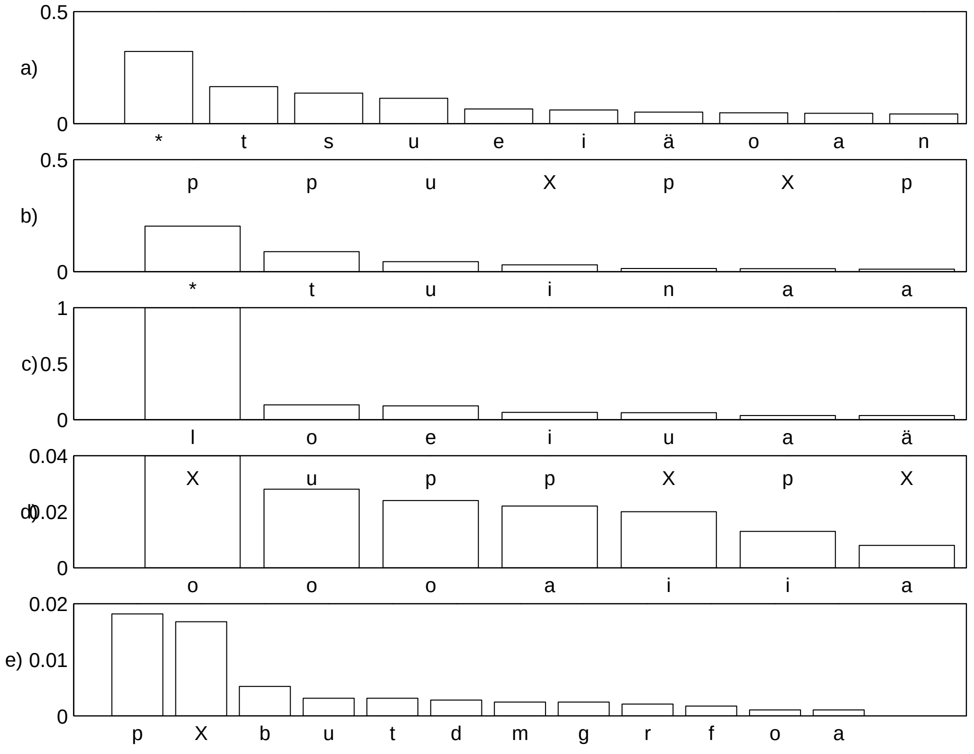

Figure C.21: Context analysis for the phoneme /v/.

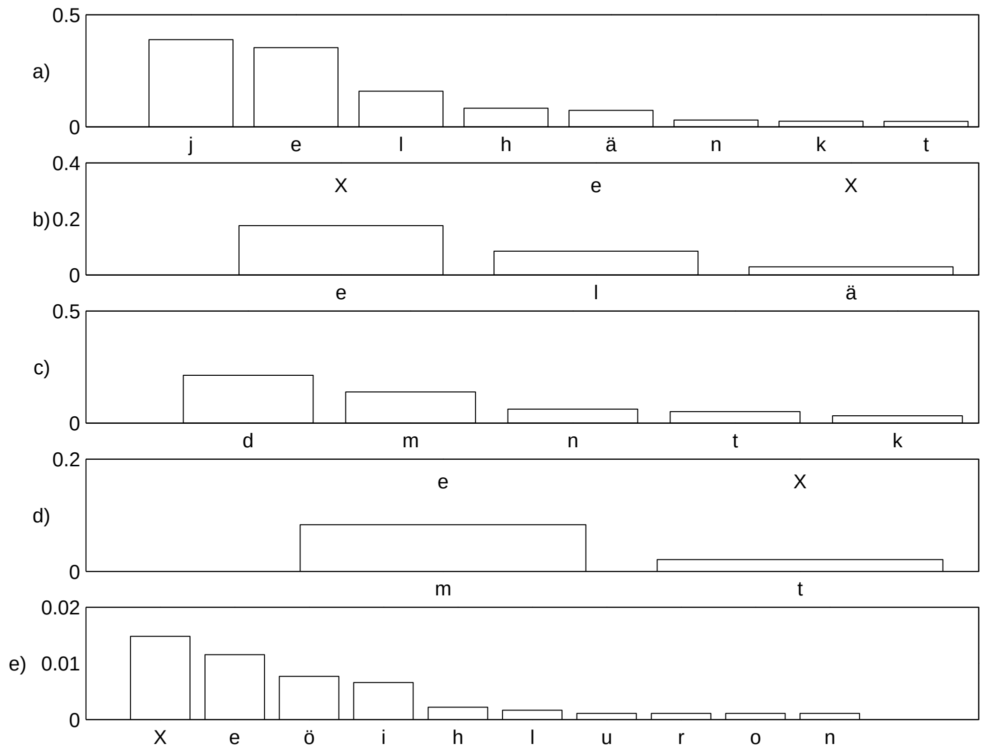

Figure C.22: Context analysis for the phoneme /y/.

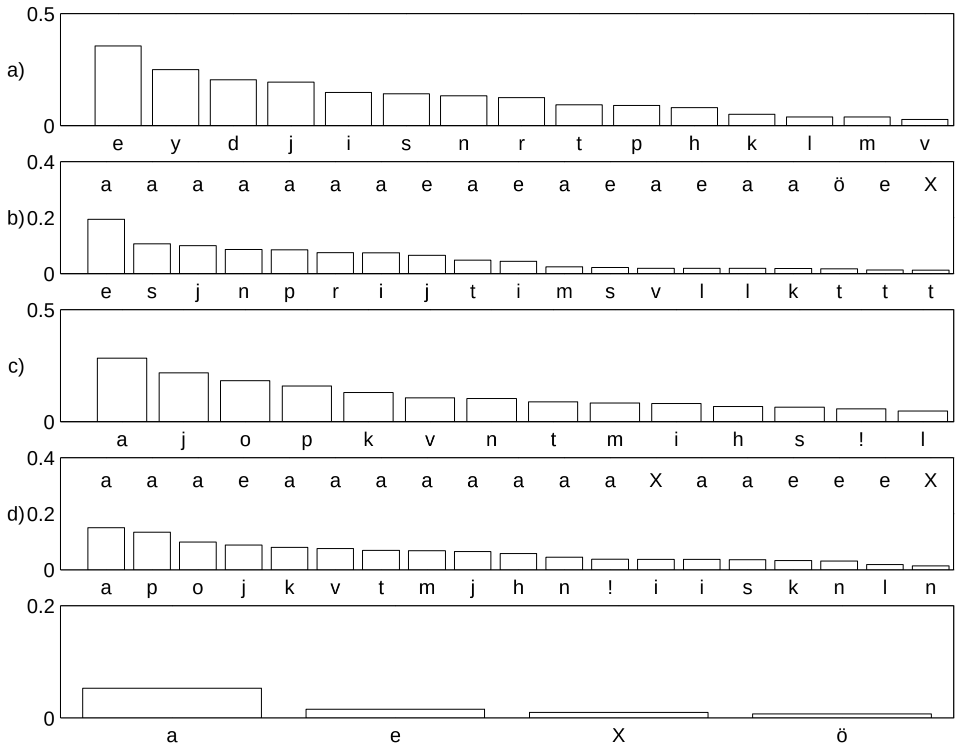

Figure C.23: Context analysis for the phoneme /ä/.

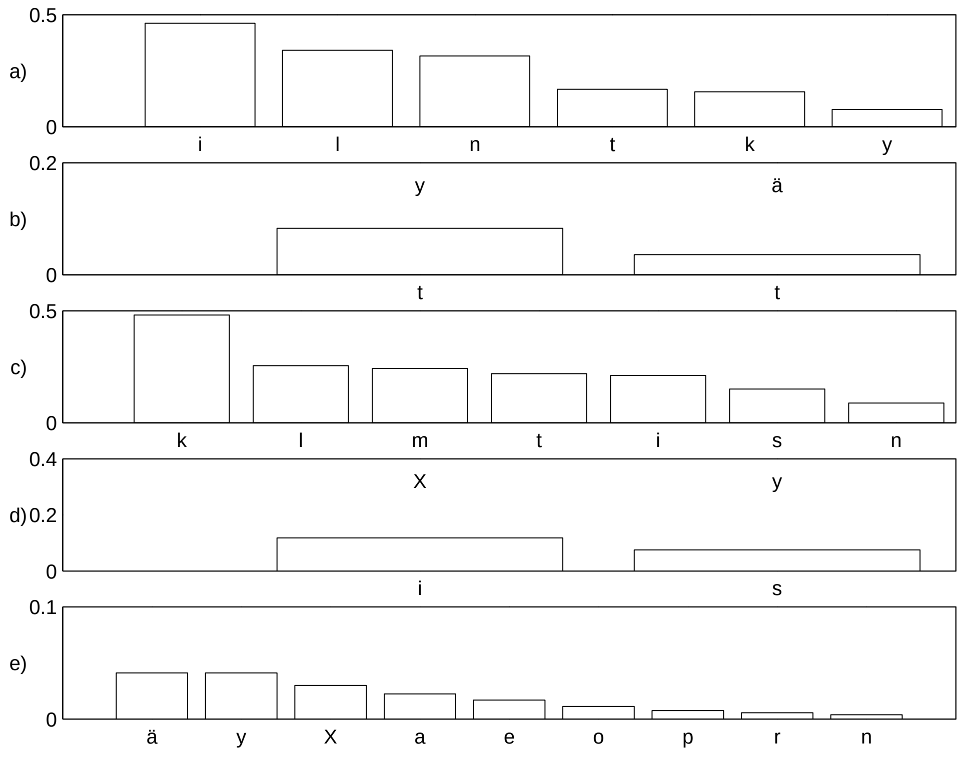

Figure C.24: Context analysis for the phoneme /ö/.