Diploma Thesis

Data-Based Automatic Phonetic Transcription

Christina Leitner

Signal Processing and Speech Communication Lab
Graz University of Technology

Assessor: Univ.-Prof. Dipl.-Ing. Dr.techn. Gernot Kubin
Supervisor: Dipl.-Ing. Stefan Petrik

Graz, September 2008

Kurzfassung

Phonetische Transkriptionen sind eine wichtige Ressource in verschiedenen Forschungsgebieten wie zum Beispiel der Spracherkennung oder der Linguistik. Die manuelle Erstellung von phonetischen Transkriptionen ist eine mühevolle Aufgabe, daher erscheint die Entwicklung einer Anwendung, die für gegebene Audiodaten automatisch phonetische Transkriptionen erzeugt, sinnvoll.

Für die Entwicklung dieses neuen phonetischen Transcribers wurde eine Methode ähnlich der datenbasierten Sprachsynthese angewendet. In der datenbasierten Sprachsynthese wird ein Wort durch die Verknüpfung von Audiosignalen aus einer Datenbank, die anhand ihrer Transkriptionen ausgewählt werden, erzeugt. Für den phonetischen Transcriber wird dieser Prozess umgekehrt: Um eine phonetische Transkription eines Wortes zu erstellen, wird die Aufnahme des Wortes mit den Audiosignalen in der Datenbank verglichen, dann werden die Transkriptionen der ähnlichsten Signale zur neuen Transkription verknüpft.

Als Datenbank wurde die ADABA (Österreichische Aussprachdatenbank) verwendet, die Aufnahmen von sechs verschiedenen Sprechern aus Österreich, Deutschland und der Schweiz enthält. Die Transkriptionen der ADABA bestehen aus mehr phonetischen Symbolen als übliche breite Transkriptionen für das Deutsche (89 Symbole statt 45 im SAMPA-Deutsch). Die Anwendung ist momentan auf einen Sprecher beschränkt, um solch detailreiche Transkriptionen zu ermöglichen. Die Audiodaten wurden in Triphone segmentiert, ein Mechanismus zur Auswahl von Kandidaten-Triphonen für den Mustervergleich wurde entwickelt und für den Vergleich wurde ein Dynamic Time Warping Algorithmus implementiert.

Die Anwendung wurde mit verschiedenen Einstellungen optimiert. Die Ergebnisse aus den Tests mit der endgültigen Implementierung wurden mit den Referenztranskriptionen der ADABA ausgewertet. Die erreichte Phone-Erkennungsrate von 91.94% ist vergleichbar mit den Erkennungsraten, die von anderen existierenden Transkriptions-Anwendungen unter ähnlichen Bedingungen erreicht worden sind.

Schlagwörter: Automatische phonetische Transkription (APT), Dynamic Time Warping Algorithm (DTW), Österreichische Aussprachedatenbank (ADABA), Automatische Spracherkennung

Abstract

Phonetic transcriptions are an important resource in different research areas such as speech recognition or linguistics. Establishing phonetic transcriptions by hand is an exhausting process therefore it seems reasonable to develop an application that automatically creates phonetic transcriptions for given audio data.

To build this automatic phonetic transcriber a new method similar to data-based speech synthesis is applied. In data-based speech synthesis a word is synthesized by recombining audio samples from a database according to their phonetic transcription. In the case of the phonetic transcriber this process is reversed: To obtain the transcription of a word its recording is compared to audio samples in the database and then the transcriptions of the most similar samples are concatenated to a new transcription.

The data is taken from ADABA (Austrian Phonetic Database) that contains recordings of six speakers from the major varieties of German. The transcriptions in ADABA contain more phonetic symbols than common broad transcriptions for German (89 symbols instead of 45 for standard SAMPA-German). The application is currently restricted to a single speaker to allow for such detailed phonetic transcriptions. The audio data was segmented into triphones, a triphone candidate selection was developed and for the pattern comparison a dynamic time warping algorithm was implemented.

The transcription framework was optimized with several settings. The results from the tests with the final implementation are evaluated with respect to the reference transcriptions in the ADABA database. The achieved phone recognition rate of 91.94% is comparable to rates reported from other existing phonetic transcription tools tested under similar conditions.

Keywords: Automatic Phonetic Transcription (APT), Dynamic Time Warping Algorithm (DTW), Austrian Phonetic Database (ADABA), Automatic Speech Recognition (ASR)

Acknowledgement

This diploma thesis was written in 2008 at the Signal Processing and Speech Communications Laboratory at Graz University of Technology.

I would like to thank Prof. Gernot Kubin for introducing me to the field of signal processing and for initiating this diploma thesis together with Prof. Rudi Muhr. I wish to thank Prof. Rudi Muhr for reawaking my interest for speech, for inspiring me with his passion, for the times of discussion and the support for this work. Special thanks go to my supervisor DI Stefan Petrik, who had always an answer to my questions, a solution for my problems and enough time to talk.

I would like to thank my parents Cäcilia and Reinhold Leitner for supporting me during my studies, financially but all the more mentally. Thank you for being there and encouraging me from the entrance examination until the diploma thesis. I like to say so many thanks to my sisters Ursi and Agnes for sharing my way, you amaze me.

Special thanks go further to my friends Karin, Doris and especially Gudrun, for being my longest friend and having always an open ear. My friends Johanna and Gudrun for welcoming me in Graz and making the start so much easier, Gerda and Birgit for being colleagues and becoming friends. Thanks also to the best flatmates Christoph, Christoph, Christof, Benjamin and Andi, for mental support, cooking coffee, and pulling the rope.

Finally thanks to all others who have shared my life in Graz and make it so much worth living here.

Graz, September 2008 Christina Leitner

Contents

Chapter 1

Introduction

The purpose of phonetic transcriptions is the documentation of the phonetic content of speech. This information is important for several different research areas. In linguistics phonetic transcriptions are used to investigate different pronunciations, e.g., to study the pronunciation styles of different speakers depending on their provenience. In speech communication phonetic transcriptions are used to build lexica with reference pronunciations. For speech synthesis they are necessary to know what a word that has to be synthesized should sound like. In speech recognition they are used to map the recognized phones onto words.

The manual creation of phonetic transcriptions is an exhausting task. It demands a large amount of time and thus is expensive. For this reason the use of an automatic transcription tool seems desirable. An automatic transcriber speeds up the transcription process. Even if the established transcriptions are not 100% correct, the amount of work for a human transcriber can be significantly reduced, because instead of creating the whole transcriptions only a validation and a correction of erroneous transcriptions is necessary.

Transcriptions from human transcribers are error prone because transcribing is not an objective task. The transcriptions of different transcribers vary. Even those of a single transcriber may deviate, e.g., due to fatigue. Automatic transcriptions contain errors too, however these errors are systematic. They can be reproduced, whereas the errors of human transcribers are random errors. The automatic transcriptions could thus form a guideline that is more objective even if the possibility of errors has to be considered. Furthermore an automatic transcriber could be used to check existing manual transcriptions, for example in speech databases, for consistency. This allows a validation of these transcriptions.

In existing systems for automatic phonetic transcription (APT) different methods are applied, but in most cases simply systems originally designed for automatic speech recognition (ASR) are used. The majority of these APT systems is model-based, i.e. they perform the recognition by using statistical models for the phones. The creation of the models can be seen as generalization of the used data, thus possibly important information may be lost. A different approach, also employed in speech recognition, is explored for the development of this application: In template-based speech recognition no modeling is applied. The utterances that have to be recognized are not compared to a model but directly to stored reference data samples. The appealing property of this approach is that in contrast to other systems no information is discarded due to modeling.

The objective of this diploma thesis was to develop a new tool explicitly designed for automatic phonetic transcription that is based on the techniques used for template-based speech recognition. This transcriber should be evaluated by comparing the automatic transcriptions to the reference transcriptions of a predefined test set. The reference transcriptions from the database that was intended to be used provide a larger phoneme set than normally considered for German in similar tasks (89 phonetic symbols instead of 45 in standard SAMPA-German). Initially, the main question was if an automatic transcription tool could achieve such detailed transcriptions at all. The used recordings are of high quality (studio environment, 44.1 kHz sampling frequency), so it was assumed that this would favor correct automatic transcriptions. Finally, the results should be put in context to other transcription systems to decide if the template-based approach is suitable for the task of automatic transcription.

The thesis is organized as follows: Chapter 2 gives an overview of existing systems for APT and deals with the matter of validation. Chapter 3 outlines the basic ideas for the implementation of the new transcription tool. In chapter 4 the database that was used for the implementation and for testing is described. Chapter 5 presents the details of the implementation and the underlying theory. In chapter 6 the results from the simulations in the development phase and the final test phase are presented and discussed. Chapter 7 gives an outlook on possible improvements and extensions and chapter 8 finally concludes the work.

Chapter 2

State of the art in automatic phonetic transcription

Most of the current automatic transcription systems are derived from systems used for automatic speech recognition (ASR). The tasks of automatic transcription and speech recognition are similar, however it has been shown that the systems optimized for automatic speech recognition do not necessarily yield the best results for automatic transcription [1].

The systems derived from ASR either create transcriptions by using phone recognition or by applying forced recognition, which is also known as forced or Viterbi alignment [2]. Other existing approaches establish transcriptions by using neural networks [3] or by application of a morph dictionary [4].

2.1 Existing Approaches

2.1.1 Phone recognition

Phone recognition can be done by using a speech recognizer based on Hidden Markov Models (HMMs). It is possible to employ phone recognition without any restrictions or to implement a phone language model. This model describes the phonotactic rules of the language and allows restricting the number of possible phone sequences.

With phone recognition usually an accuracy between approximately 50% and 70% can be achieved [2]. The phone accuracy is measured as follows: The number of inserted phones is subtracted from the number of identical phones in automatic and reference transcription. Then the resulting number is divided by the number of total phones in the reference transcription (see also section 6.1.1). Wester et al. measured a phone accuracy level of 63% in one of their phone recognition experiments [5]. For further use in other applications such phone recognition levels generally are too low. However, the results may be improved by filtering the output, this can for example be realized by the application of e.g., decision trees or forced recognition. Furthermore one advantage of phone recognition is that it does not depend on orthographic input [2].

2.1.2 Forced Alignment

Many of the currently used systems for automatic phonetic transcription employ the technique of forced alignment or forced recognition. Systems that implement this technique are for example described by Cucchiarini [2], Schiel [6], Binnenporte [7], Rapp [8], Sjölander [9], and Jande [10]. For the forced alignment approach the orthographic representation of the spoken words is needed. Additionally the lexicon of the speech recognizer has to contain the pronunciation variants of the words that allow multiple pronunciations. Then with the forced alignment procedure the best matching pronunciation within the given variants is found. This is done by employing the Viterbi algorithm [11, 12].

There are several methods for establishing a lexicon that contains pronunciation variants for the forced alignment. One possibility is to rewrite pronunciation rules that are either taken from literature or derived from speech data. These rules can then be applied to a lexicon with canonical transcriptions. Another possibility for obtaining pronunciation variants is the direct extraction from a speech corpus. The first approach, also called rule-based approach, has the advantage that it can be applied to all lexicon entries, so many pronunciation variants can be generated. With the second method only variants that actually occur in the corpus can be considered, but still there might be the advantage that word-specific phenomena can be captured that cannot be modeled by the rule-based approach [7].

From the lexicon with pronunciation variants a state transition network that describes the different pronunciations can be created for each word. The Viterbi algorithm searches the most likely path through this network and thus finds the most probable pronunciation for the recording that finally is taken as transcription [10]. Often the start and end points of each phone are determined as well, i.e. the utterance is segmented. This is of no importance for the phonetic transcription but is interesting for other applications [6, 8, 9][1].

The achievable performance of an alignment system seems to be comparable to the performance of a human transcriber as reported by Cucchiarini et al. [2]. Transcribing is a task that is based on subjective decisions so transcriptions from human listeners often tend to be different. Even transcriptions from a single listener may vary at different occasions. For the interlistener agreement - the agreement between listeners - rates from 75% to 85% were observed. For the comparison of manual and automatic transcriptions agreement levels between 72% and 80% were found, thus the results from forced alignment seem satisfying because the agreement rates are in the same range as those achieved by human transcribers [2].

2.1.3 Neural networks

A completely different approach for automatic transcription is the use of neural networks proposed by Chang et al. [3]. The system uses two neural network stages and a Viterbi-like decoder. The first neural network performs a classification of

[1]Some of the above mentioned sources focus rather on the temporal alignment than on the recognition of phones.

each frame along five articulatory-based dimensions (place of articulation, manner of articulation, voicing, lip-rounding, front-back articulation). These phonetic features are passed to the second neural network that maps them onto phonetic-segment labels. More precisely, the output of the second neural network is a vector of phone-probability estimates for each frame. From this matrix then a linear sequence of phone-labels over the whole utterance is generated by using a Viterbi-like decoder.

A comparison of the transcriptions generated by this system with the transcriptions produced by a human transcriber shows that 80% of the words are transcribed identically, so the quality of the automatic transcriptions seems comparable to those of human transcribers. Furthermore the system provides the advantage of being relatively robust in case of acoustic interference and variation of speaking-style. Another advantage is the independence from word-level transcriptions, the system does not need an orthographic representation of the text. Currently the system is tested on a corpus of American English but it may be easily extended to different corpora or even languages as most of the articulatory features are also found in other languages of the world [3].

2.1.4 Morphologically based automatic phonetic transcription

The system presented by Wothke [4] creates phonetic transcriptions from the orthographic representation of words without the use of audio data. The aim is the generation of an extended dictionary for a speech recognizer, where multiple pronunciation variants for a word are desired to make the speech recognizer more robust.

The approach provides an improvement of a simple rule-based transcription generation. Consider German as the target language: In German the pronunciation of a letter depends on the morphological context so the simple application of rules tends to be erroneous. The proposed solution for this problem is a segmentation into morphs i.e. prefix, stem, and suffix. After this first step a letter-to-phone mapping is applied and thus a phonetic transcription created. In the segmentation step multiple segmentations for words with identical orthographic representation but different morphology are created. For each segmentation the possible pronunciation variants are considered and the corresponding phonetic transcriptions are generated.

Tests showed that for 92.2% of the words in a test set the system found a segmentation and for more than 99% this segmentation was correct. Furthermore for 98% of the words a correct transcription was found, i.e. at least one of the proposed transcriptions was considered as correct.

This system performs well for the purpose of establishing a dictionary with pronunciation variants, however this task is completely different from generating a phonetic transcription of an actually spoken utterance. Nevertheless the system might be used to provide possible transcriptions to another system such as an aligner (see section 2.1.2). The aligner then chooses the most likely transcription with regard to the given audio information [4].

2.1.5 Comparison of the existing approaches

From the four proposed approaches, the phonetic recognition and the method using neural networks are most flexible. Both methods transcribe an actual utterance without restriction to given pronunciation variants and there is no need for orthographic input. The main drawback of phone recognition is that the performance is not good enough to build a powerful stand-alone transcription tool. The approach using neural networks shows results comparable to those of a human transcriber and thus seems promising.

The forced alignment method is often employed and thus proposed in many papers [6, 8, 9]. However the possible transcriptions are limited to those variants given in the pronunciation dictionary. So the accuracy of the transcriptions hardly depends on the degree of detail of the transcriptions in the dictionary. Despite this restriction the reported results show that they are comparable to results achieved by human transcribers.

The approach with the morphological segmentation finally is designed for different purposes and can only be used to cover a part of the process of automatic phonetic transcription of an actual utterance.

2.1.6 Idea for a new approach

An ideal application for automatic transcription would be able to transcribe what has actually been said in an utterance or more precisely how it has been said. It is desirable that this transcription does not depend on any given canonical transcription or its derivation. Nevertheless some prior knowledge is necessary to produce a transcription of a new utterance as it is not possible to generate it from scratch. Thus it is very important that the knowledge serving as a basis for new transcriptions is as objective as possible. The next section deals with the problem of objectivity for phonetic transcriptions, before in the following part a new approach for automatic phonetic transcriptions with a high degree of detail is presented.

2.2 Validation of phonetic transcriptions

2.2.1 Reliability and validity

Before using automatic phonetic transcriptions in an application it is necessary to determine their quality. The evaluation of automatic transcriptions is a crucial task: Often they are compared to manual transcriptions but these are either prone to variation caused by the subjectivity of any human transcriber. Phonetic transcriptions, whether produced automatically or by a human transcriber, can be seen as a kind of measurement of a speech signal. They should achieve the quality standards demanded from any other measurement. The quality of a measuring instrument can be described by means of reliability and validity. The reliability of a measuring device shows if repeated measurements of the same object yield the same results. So it is an indication for the consistency of the measurements and the accuracy of the instrument. The validity determines if the measuring device indeed measures what it is intended to measure.

One problem that occurs when transcriptions are produced by human transcribers is that of inter-subject and intra-subject variation. Obviously transcriptions by different transcribers may yield different results (inter-subject variation), but even transcriptions by the same transcriber may vary at different trials (intra-subject variation). So the reliability for human made transcriptions is not guaranteed. On the other hand automatic transcriptions by a machine are reliable as a computer can be programmed in such a way that it always produces the same results [13, 2, 14].

Another problem concerns the validity of a transcription. Finding the transcription of a word cannot be compared with other measurements in terms of validity as there is no "true" criterion score (as suggested by the test theory). Experiments showed that it is not straightforward to determine whether a phone in an utterance is present or not. For example, in an experiment described by Kessens and Strik [1] nine experts were asked to decide whether a phone in an utterance was spoken or not. In only 246 of 467 cases all the listeners took the same decision, this means that the degree of agreement is less than 53%. For experiments the quality of an automatic transcription is usually derived by its similarity to a manual transcription. However the results of the experiment by Kessens and Strik demonstrate that the validity of an automatic transcription cannot simply be determined by comparing it with any manual transcription. It is preferable to define a reference transcription for the evaluation such as a consensus transcription (see next section) [13, 2, 5].

2.2.2 Obtaining a reference transcription

Several possibilities exist for obtaining a reference transcription. One is to create a consensus transcription, this means that at least two experts work together and need to agree on each phonetic symbol of the transcription during the transcription process. The process of reaching a consensus can be seen as approach to find "true" criterion scores and to minimize transcription errors.

Another possibility is to involve more than one listener and to take only the transcriptions on which all the listeners or at least a majority of them agree, and to reject the rest of the transcriptions (majority vote procedure) [13, 2, 5].

2.2.3 Performance of a phonetic transcriber

As already mentioned, in most approaches speech recognizers are used for phonetic transcription. It seems probable that a speech recognizer with a low word error rate also provides a good transcription quality if it is used for automatic transcription. However experiments showed that this is not necessarily the case.

Kessens and Strik [1] conducted a series of experiments with a continuous speech recognizer (CSR) employed as transcription tool. The CSR was a standard HMM recognizer and the transcriptions were produced by using it in forced recognition mode. Some of the CSR properties were varied for the test series (e.g. the topology of the HMMs or the acoustic resolution of the HMMs, for more details see [1]) and then each version of the CSR realized the same task which was to decide whether a phone in a transcription variant was present or not. For each test set the word error rate (WER) was computed from the number of substitutions S, deletions D, insertions I and the total number of words N.

$$\text{WER} = 100\% \times \frac{S + D + I}{N} \tag{2.1}$$

Furthermore the agreement between the automatic and the reference transcription was calculated. The reference transcription was produced with the majority vote procedure, in this case 5 of 9 listeners had to agree on the transcription. The database that was used for the experiments is the VIOS database [15], that contains spontaneous speech.[2]

For the measurement of the agreement between the reference and the automatic transcription two measures were applied: The percentage agreement and Cohen's kappa [1]. The percentage agreement is defined as follows:

$$P_0 = 100\% \times \frac{\#\text{agreements}}{\#\text{agreements} + \#\text{disagreements}} \tag{2.2}$$

The Cohen's kappa corrects for chance agreement:

$$\kappa = \frac{P_0 - P_c}{100 - P_c} \tag{2.3}$$

with $-1 < \kappa < 1$
$P_c =$ percentage agreement on the basis of chance

With Cohen's kappa measurements that were made under different conditions can be compared.

For different test settings the values for the agreement varied between 73.9% and 79.9% and the kappa values between 0.47 and 0.58, both for spontaneous speech (VIOS database). To inspect the relation between the word error rate and kappa another test was performed with an independent test set. Normally the CSR with the lowest word error rate is considered the best. Thus it is interesting to investigate if this CSR is also the best for automatic transcription. One would expect that the CSR with the lowest WER also produces the transcriptions with the highest degree of agreement with a reference transcription. However, this is not the case, as the experiment by Kessens and Strik showed [1]: the test set with the lowest WER had the lowest kappa value, and another test set with a high kappa value had a high WER. So the assumption that a low WER results in a high degree of agreement between the automatic transcription and the reference transcription does not hold. Obviously, recognizing words is a different task than creating automatic phonetic transcriptions.

[2]Further tests that were made with read speech from another database are not taken into consideration in this summary.

This result can be explained by the fact that for speech recognition a kind of generalization has to be performed to map actual pronunciations onto models. On the other hand, in the case of automatic transcription details should be kept to allow for detailed transcriptions. Thus the objectives of automatic transcription and automatic speech recognition are contradictory.

Chapter 3

A new approach for automatic phonetic transcription

Most of the described systems for automatic phonetic transcription are modified speech recognizers. However, it has been shown that phonetic transcription is not the same task as speech recognition [1]. Therefore it is desirable to design a tool especially for phonetic transcription.

Furthermore for some purposes, like linguistic studies, it would be preferable to have more detailed transcriptions than commonly used in current approaches, e.g., to investigate different regional pronunciation styles.

Therefore I present a new approach for automatic transcriptions that is disposed for the production of transcriptions with a high degree of detail. In the following sections the underlying concepts for the design of the transcription tool will be presented. In the subsequent chapters the used database and the different parts of the transcription tool will be described.

3.1 The idea

The idea for the design of the transcription tool is inspired by the technique of concatenative speech synthesis [16]. In concatenative synthesis speech samples are stored in a database. For the synthesis of a new utterance the proper audio samples are chosen from the database according to their transcription. They are concatenated in order to form the new utterance. For the phonetic transcription this procedure is inversed: To transcribe the audio sample of a new word, the database is searched for similar audio segments. Then the best matching segments are chosen and their transcription are concatenated to a new transcription. A similar technique is used in so called translation memory systems [17, 18]. These systems assist translators by proposing translation chunks for a text that has to be translated. For this purpose segments from the text are searched in a database that contains finished translations. If a matching segment exists, it is presented to the translator who can use it, change it or create a new translation. New translations are stored in the database and thus it will be enhanced. This feature is also useful for a transcription tool: Automatic transcriptions can be created and then validated or corrected by a human transcriber. These approved transcriptions can be added to the database what makes an improvement and an adaptation, e.g., to different speakers, speaker styles or regional variants possible.

3.2 No modeling, just data

To find a matching audio segment in the database a pattern comparison between the new audio sample and the samples in the database is necessary. Most current speech recognition systems realize this task by comparing the new audio sample to a model that has been established in a training step. Normally this is done by using HMMs (for more detailed information see for example [19,11]). Although they are known to have certain weaknesses they are the dominant technique for speech recognition. The main disadvantage of HMMs is that the data is generalized to form a compact model, thus useful information like long-span time dependencies and speaker information gets lost [20, 21]. A completely different approach was proposed by de Wachter et al. [20, 22]. Instead of creating a model, pattern comparison is done straight from the data according to the motto: “no modeling, just data” [20]. This approach is called example-based or template-based speech recognition, where the underlying technique is the dynamic time warping algorithm (DTW) [12]. When using a large vocabulary, a constraint of this approach is the explosion of the search space. The extent of the search space can be reduced by applying a candidate selection that chooses potentially interesting speech samples [20].

3.3 The basics for the new transcription tool

The new transcription tool was designed to produce high detailed transcriptions for Austrian German as the used corpus is taken from the ADABA database [23, 24]. The tool should not only recognize the basic phonemes used in Austrian German but also the diacritics contained in the transcriptions of ADABA. This is a main difference to many speech recognizers or transcription tools as in most cases the transcription is rather broad. At the beginning it was not clear if the DTW-algorithm would be able to distinguish speech segments at such a high degree of detail. For simplification the first version of the tool is speaker dependent, i.e. it has only been implemented and tested for one speaker. The different units for the realization are on the one hand derived from concatenative speech synthesis (the synthesis of segments) and on the other hand from template-based speech recognition (pattern comparison with DTW). The new transcription system and the system for template-based speech recognition have the dynamic time warping algorithm in common, while the implementation of the other parts is quite different. In case of the automatic transcriber the segments in the database are triphones. In the implementations presented by de Wachter et al. monophones are used. However they point out that it is also possible to use longer segments or even combine templates of different length in one speech recognizer [22]. The candidate selection for the APT tool uses orthographic input in combination with a look up table (LUT). In template-based ASR the candidate selection is based on more elaborate algorithms (k-nearest neighbor selection combined with time filtering [20, 22]).

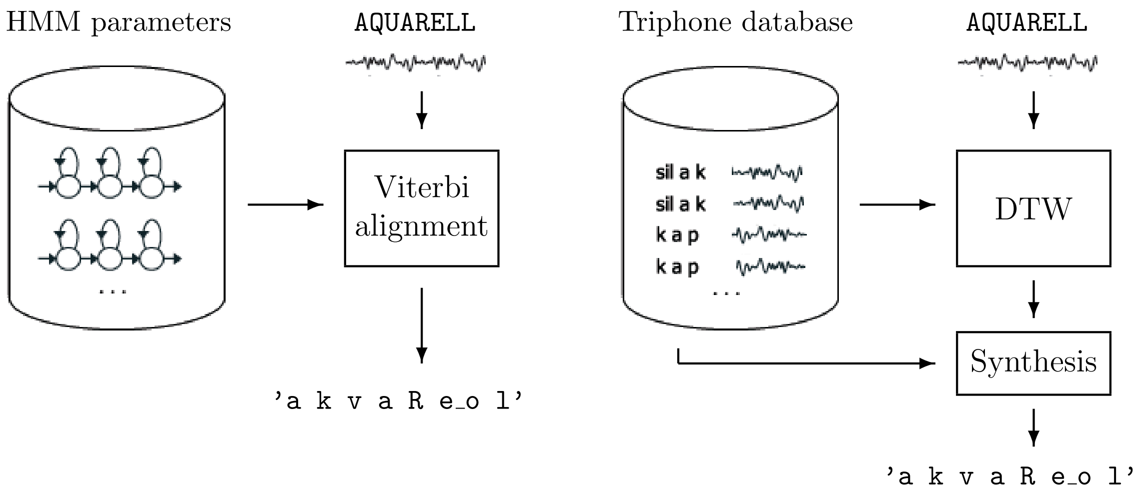


Figure 3.1: Schematic comparison of the model-based forced alignment approach (left) and the new template-based approach (right) for the automatic phonetic transcription of the word 'Aquarell'.

In chapter 4 the properties of the recordings and transcriptions of the ADABA will be described. Then in chapter 5 the realization of the new tool for automatic phonetic transcription is presented.

Chapter 4

The ADABA speech corpus

ADABA is a pronunciation database for Austrian German. It resulted from the project "Varieties of Austrian German - Standard pronunciation and varieties of standard pronunciation" that was conducted by Rudolf Muhr from 2000 to 2007. ADABA provides the first representative corpus of spoken Austrian German, one of the three major varieties of German [23, 24].

4.1 Contents of the ADABA

ADABA contains recordings of six speakers, one male and one female each from Austria, Germany and Switzerland. The aim was to investigate and demonstrate the differences in the pronunciations of the Austrian, German and Swiss speakers. Each speaker read a list of 12 964 single or multi word utterances which resulted in a total number of 75 964 audio files[1]. The word list was composed from the following sources [25, 23]:

1. Basic vocabulary: 4854 words from the word list of the Austrian Language Diploma, which constitutes a basic vocabulary for learners of Austrian German as a foreign language.

2. Frequent word forms: 5540 of the 10 000 most frequent word forms from the project "Deutscher Wortschatz" of the University of Leipzig.

3. Frequent word forms: A selection of the 30 000 most frequent word forms form the "Institut für Deutsche Sprache" in Mannheim.

4. Phonetically rich words: 753 of 1540 phonetically rich words proposed by Werner König. The list contains rare and phonetically complex words.

5. Frequent loan and foreign words: A selection of 2500 words from the Duden Dictionary of foreign words.

[1]Some of the words are contained in the database twice due to different pronunciation variants.

After elimination of entries that appeared more than once the result was first a list of 13645 entries, later this list was reduced to 12 964 words due to difficulties concerning the quality of the recordings.

The speakers are all trained speakers from broadcasting companies. For the recordings the speakers had no special instructions regarding the pronunciation except to realize the utterances that same way as they would do during their work. This should guarantee a uniform context for all speakers and enforce the naturalness and the representativeness of the pronunciation.

In addition to the word list that was read by six speakers ADABA contains further speech material. As this was not used for the automatic phonetic transcription this will not be discussed further here, more information can be found in [25] and [23].

4.2 The transcription process

The entire audio data of the six speakers was transcribed three times. This could only be done with much time and effort. In total 17 transcribers worked on the transcriptions at different times. Because of the great amount of data and the number of involved persons it is clear that during the transcription process variation occurs between the transcribers as well as between the different transcriptions of one person. Three iterations were necessary in oder to reduce inter-transcriber and intra-transcriber variation.

The transcriptions in ADABA represent the average realizations of the individual speakers as well as the realizations in the different national varieties. In some cases the actual pronunciation may vary slightly from the transcription due to following reasons: Each transcription should not only map the absolute phonetic content but should also be posed in the right relation to the realizations of the other speakers. Definitions concerning the whole corpus were necessary to make a distinction between the varieties possible. Another problem is that the differences between the pronunciations of the six speakers sometimes are diminishing. The question what differences should be considered and where to set the boundaries between different phonetic realizations had to be answered. Furthermore the symbols of the IPA proved insufficient in some special cases like e.g., the transcription of the different /r/-realizations and the different schwa-variants. These reasons led to the definition of an average transcription [25].

One further important consideration during the transcription process was the need for comprehensibility and reproducibility. A too narrow transcription was considered as not meaningful, as the potential user of ADABA was assumed to have an average knowledge of phonetics and a narrow transcription necessitates a great number of diacritics. The use of many diacritics would have resulted in a higher amount of time and further decreased the understandability of the transcriptions. As a tradeoff the number of diacritics was limited. However they were used to mark slight but relevant differences of pronunciation to ensure a correct description of the pronunciations.

4.3 The used symbol set

The symbol set used in the ADABA is based on the IPA symbol inventory [26]. Some further symbols and symbol combinations are added to the standard symbols to model certain specific pronunciation characteristics. These symbols and their explanations can be found in [25].

For machine-aided processing the transcriptions were exported from ADABA in SAMPA format as this form is easier to handle. The phoneme set is an extended version of the standard SAMPA symbol set, SAMPA AUSTRIA proposed by Muhr (see [24] for details).

4.4 Technical Details

As described earlier ADABA contains 75 964 audio files, most are recordings of one word, some of multiple words. The recordings in the published version of ADABA are available in the ogg file format (8 kHz sampling frequency, compressed). For research purposes, Muhr provided the recordings in wav file format with 44.1 kHz sampling frequency. The recordings were conducted in a recording studio of the Institute of Electronic Music and Acoustics (IEM) at the University of Music and Dramatic Arts Graz. They offer a high quality so they were considered to serve the purpose of highly detailed transcription.

For the research on automatic transcription the recordings of the Austrian male speaker were used, only some audio files were excluded for technical reasons. All in all 12 728 audio files were involved in the training, development and testing process, the total duration of all files is about five hours.

4.5 Tests and modifications of the ADABA transcriptions

Muhr proposes a basic phoneme set of 22 consonants and 14 vowels for Austrian German [25]. However with the diacritics taken into consideration the transcriptions of the Austrian male speaker showed a much higher variety. As some phones occurred only rarely the following strategy was applied: First a test for all transcriptions was performed to validate if all symbols indeed are phonetic symbols and are not included erroneously. Then the resulting phoneme set was reduced either by mapping infrequent symbols to frequent ones or by exclusion of some words that contain rare phonemes.

4.6 Testing of phoneme symbols

A list of the occurring phoneme symbols in all transcriptions belonging to the Austrian pronunciation dataset[2] was generated and then manually checked. In addition to the symbols belonging to the extended phoneme set with diacritics other symbols or combinations of symbols were found that clearly dated from errors in the transcription or from formal errors (like annotations included in the transcriptions or a wrong transcription format). All found errors were corrected; the modifications are documented in [27]. This initial analysis led to substantial feedback to the authors of ADABA which will be considered for future releases.

4.7 Modifications

After the correction of the incorrect symbols, again a list containing all phonemes of the Austrian corpus was created. This list contained 128 phonemes, 32 of them occurred only once and 7 only twice. As a phoneme that occurs only once or twice in a corpus is not representative for a language, modifications were carried out to eliminate these phonemes. Another reason was that it was planned to segment the audio data with the aid of HMMs and a statistical model that is estimated from only one or two examples is not robust against variation. The modifications were realized the following way: Either the transcription was modified in a way that it became similar to a transcription with the same or a similar phonetic content, e.g.:

Word	*Phoneme*	*Transcription*	*Modification*	*Comment*
dünn	`[y_O]`	`["dy_On]`	`["dyn]`	*as* hauchdünn

Table 4.1: Example for a phone modification

In some other cases no modifications were applied because the transcription occurred only in the dictionary and there was no file in the audio corpus. Furthermore, for some transcriptions no modifications were realized because no similar transcription was found. This concerned mainly loan or foreign words that contain phonemes that are not typical for the German language, two examples are shown in table 4.2.

Word	*Phoneme*	*Transcription*
commonwealth	`[T]`	`[ko_om@n"ve_olT]`
tagliatelle	`[l_j]`	`[tal_jia"te_ol@]`

Table 4.2: Examples for excluded words

The `[T]` in *commonwealth* clearly does not belong to the German phoneme inventory, neither does the `[l_j]` in *tagliatelle*[3]. In total 14 words were excluded for this reason.

[2]In the pronunciation dictionary of ADABA there are further entries without recordings.

[3]The corresponding IPA symbols are [θ] and [lʲ], respectively.

Finally the list of occurring phonemes could be reduced to 89. This is still a high number if compared to the number of phonemes considered normally for German (about 45, see [28]) in other speech communication tasks like recognition. Therefore a question from the beginning of the project was if the transcription task was at all possible with this high number of phonemes or if a reduction was necessary to achieve a reasonable performance. The results that answer this question are discussed in section 6.2.4.

Chapter 5

The Transcriptor

The transcriptor is a new tool for the automatic creation of highly detailed phonetic transcriptions. In this chapter the different parts of the transcription framework are described in detail. Several steps were necessary to implement the tool:

- The audio data from the database was segmented and integrated in a database.
- A candidate selection for the segments that are passed to the pattern comparison was realized.
- The algorithm for the pattern comparison was chosen and implemented.
- A synthesis procedure to create the new transcription was developed.

Some of these steps were realized using well-known algorithms or already existing tools, others were done in a rather experimental manner.

5.1 Establishing the internal database

In concatenative speech synthesis, speech segments from a database are combined to synthesize new speech utterances. For the transcription tool, the database should contain transcription segments that are combined to the transcription of a new audio sample. To find the most appropriate transcription units in the database a pattern comparison between the audio sample of the new word and the triphones in the database is performed[1]. Then the transcriptions of the best matching triphones are synthesized to the transcription of the new word.

The units used in speech synthesis are e.g., words, syllables, triphones, biphones or monophones [16]. A different approach is to use segments of different lengths. First, an approach with different length segments was discussed for this work, but to keep it simpler segments with a fixed number of phones were chosen. It was decided to take triphones as tradeoff between length and number of examples. Long segments have the advantage that they are normally more natural than the concatenations

[1]More precisely, the comparison is not done for each triphone but a choice retrieved from the candidate selection.

of short segments. On the other hand with short segments the number of examples per segment is high and thus many comparisons are necessary. The selection of triphones was inspired by a technique applied in concatenative synthesis: There segments often are cut in the middle of the phones because this is the steadiest part. By doing so the transitions between the phones are kept, whereas a concatenation at the transitions would increase the risk of distortions [16]. In our approach the segments are cut at the phone boundaries, but taking triphones ensures that the middle phone is kept with all transitions. The concatenation of the transcriptions is performed in an overlapping manner, thus the first and the third phone in a transcription serve as concatenation point.

To establish the database, both the audio samples from the data and the transcriptions had to be segmented into triphones and then joined in a data structure that can be accessed by the pattern comparison algorithm. This data structure will be called the internal database.

The steps that had to be performed for the creation of the database are explained in the following sections.

5.1.1 Segmentation of the audio data

For the segmentation of the audio samples the Hidden Markov Toolkit (HTK) by the University of Cambridge [19] was used. The realized steps were:

- Creating the dictionary
- Extracting the features
- Training the Hidden Markov Models (HMMs)
- Aligning the audio data
- Segmenting the audio data into triphones

Creating the dictionary

For the training of the HMMs HTK needs a dictionary or lexicon that contains the phonetic transcriptions of all words that occur in the training corpus. This dictionary has to be in the following form:

```
AACHEN      a x @_bs n
AAL         a l
AAS         a s
AB          a p
ABBAU       a p a_pr o
ABBAUEN     a p b a_pr o @_bs n
ABBIEGEN    a p b i g @ n
   ⋮           ⋮
```

The ADABA database provides a feature to export all transcriptions in the database in a text file[2], however the entries in the file are in a quite different form than those needed for HTK, as can be seen here:

```
Aachen [0];[[a:x@\n]]/[[a:xN=]]
Aal [0];[[a:l]]
Aas [0];[[a:s]]
ab [0];[[ap]]
Abbau [0];[["ap_(pa_(o]]/[[ap"ba_(o]]
abbauen [0];[[ap"ba_(o.@\n]]/[[...ba_(on=]]
abbiegen [0];[[ap"bi:g@n]]/[[..bi:gN=]]
     ⋮
absolut gültig [0];[["apsolu:t gyltik]]-[[..tiC]]/
                                      [[apso"lu:t ...]]-[[..iC]]
     ⋮
```

These transcriptions are from the two Austrian speakers. Transcriptions before the slash are transcriptions of the utterances of the male speaker and transcriptions after the slash are those for the female speaker. The hyphen denotes that there are two pronunciation variants that have to be split into two separate entries.

To convert the exported transcriptions to a HTK-compatible form, they were formatted with Perl scripts. The realized tasks were: Extraction of the words and transcriptions of the Austrian male speaker, splitting of entries with multiple pronunciation variants, formatting of the words, simplification of the transcriptions, insertion of spaces between the phonetic symbols, substitution of symbols that are not compatible with HTK, and checking of all occurring phones. The following two examples illustrate the steps to create the new lexicon.

Example 1:

```
absolut gültig [0];[["apsolu:t gyltik]]-[[..tiC]]/
                                      [[apso"lu:t ...]]-[[..iC]]
                     ↓   Simplification
absolut gültig ["apsolu:t gyltik]-[..tiC]/[apso"lu:t ...]-[..iC]
                     ↓   Extraction
absolut gültig ["apsolu:t gyltik]-[..tiC]
                     ↓   Splitting of pronunciation variants
absolut gültig ["apsolu:t gyltik]
absolut gültig ["apsolu:t gyltiC]
```

[2]This file was used to check the number of occurring phones as described in the previous chapter. For all tasks here the file with the already limited phoneme set was taken.

$\downarrow$ Formatting

```
ABSOLUT_GUELTIG a p s o l u t g y l t i C
ABSOLUT_GUELTIG a p s o l u t g y l t i k
```

Example 2:

```
abbauen [0];[[ap"ba_(o.@\n]]/[[...ba_(on=]]
```

$\downarrow$ Simplification

```
abbauen [ap"ba_(o.@\n]/[...ba_(on=]
```

$\downarrow$ Extraction

```
abbauen [ap"ba_(o.@\n]
```

$\downarrow$ Formatting

```
ABBAUEN a p b a_( o @\n
```

$\downarrow$ Formatting for HTK

```
ABBAUEN a p b a_pr o @_bs n
```

For the simplification of the transcriptions all stress markers, the length markers, and the symbol for a syllable break were deleted, see also table 5.1.

	IPA symbol	SAMPA Austria symbol
Primary stress	ˈ	"
Secondary stress	ˌ	%
Long	ː	:
Half-long	ˑ	:\
Syllable break	.	.

Table 5.1: Discarded phonetic symbols

The resulting symbol set contains only symbols for phonemes and diacritics. The diacritics are always treated in combination with one phonetic symbol. This leads to a final phoneme set of 89 symbols.

Some of the SAMPA symbols are not compatible with HTK, so they were substituted by combinations of letters. Table 5.2 shows all the modifications[3].

Finally all occurring phone symbols were listed to validate that all of them represent a phoneme. Non-valid phone symbols were corrected as described in section 4.6 before the dictionary was created again.

[3]Note: The backslash occurs in combination with various symbols in the SAMPA notation.

IPA symbol	SAMPA Austria symbol	Substitution
ø	2	#2
ɜ	3	#3
ɐ	6	#6
œ	9	#9
[*var*]	[*var*]\	[*var*]_bs
⁀	_(	_pr
˖	_+	_pl
ˈ	_=	_sy
ʼ	_>	_ej

Table 5.2: Substituted phonetic symbols

Extracting the features

The feature extraction was done with the HTK-tool HCopy, the chosen features are Mel Frequency Cepstral Coefficients (MFCCs) which are widely used in speech communication [12]. For every audio sample that is proposed in a list, HCopy creates a file with MFCCs, the encoding was done for each recording of the Austrian male speaker.

The settings for the MFCCs were the basic settings proposed in the HTK book [19]. The number of extracted coefficients is 12 plus the overall energy, which leads to 13 parameters per frame. The filterbank has 26 channels and the frames have a distance of 10 ms. No delta and acceleration coefficients were calculated. Delta and acceleration coefficients could improve the results, however for the basic implementation they were not used to restrict the memory requirements.

Training the HMMs

In the next step the HMMs were trained, this was done with the HTK-tool HERest. The HMMs are monophone HMMs with five states and single Gaussian probability distributions. The flat start approach was applied for training, as the training data is not segmented with timestamps. This means that initially, HMMs with identical parameters for mean and variance are used for each phone. Then the HMMs are retrained (see also section 3.2 [19]). The reestimation was performed five times.

Aligning the audio data

The main purpose of the alignment is to determine the phone boundaries to subsequently cut the audio samples. Additionally, the forced alignment procedure finds the actual pronunciation of an utterance if multiple pronunciation variants are available in the dictionary. This is normally used for a bootstrap process: First models are trained with one fixed pronunciation per word, and then the alignment is employed to find the best matching pronunciation in the dictionary. For further use the models can be retrained by using the new phone level transcriptions.

The dictionary retrieved from ADABA contains several words with multiple pronunciation variants. Accordingly, some words or utterances are recorded twice with different pronunciations. As it was unknown initially which audio file belonged to which transcription in the lexicon, the forced alignment had to be applied.

For the Austrian male speaker 256 words with each two different recordings were found. Before the forced alignment two audio files with different pronunciations are assigned with the same transcription. After the forced alignment 178 words have different transcriptions for the different variants, 78 words remain with the same transcription for the two audio samples.

This means that the alignment does not yield optimal results as words that are intended to be transcribed differently are not recognized to be different. The pronunciation variants mostly concern the pronunciation of the suffix '-ig'. The Austrian speakers sometimes pronounce it as [iç] and sometimes as [ik]. The pronunciation [iç] is preferred by the recognizer. One possible reason is that the models are not properly trained to provide a good discrimination. This may be explained by the fact that the transcriptions with [iç] occur before [ik] in the dictionary. During the bootstrap process the transcriptions with [iç] are chosen for both types of audio files and therefore the model for [ç] is also trained with data from [k]. In other words, the model for [ç] may be contaminated. However, further tests would be necessary to prove this assumption.

The aligning of the audio data was realized by using the HTK-tool HVite. After the forced alignment step the phone level transcription can be retrieved with the phoneme boundaries as shown in the example below.

```
"/AACHEN__AT_M_1.lab"
0 700000 sil
700000 2300000 a
2300000 3800000 x
3800000 4600000 @_bs
4600000 6500000 n
6500000 9400000 sil
.
```

All new phone level transcriptions with the time stamps are stored in the file `aligned.mlf`, which will be used in the next step.

Segmenting the audio data into triphones

With the help of `aligned.mlf` as a result from the forced alignment step the audio samples can be cut into triphones. To get all possible phone sequences the words were split into overlapping triphones. This is done by again using HCopy that can use the time stamps in the file `aligned.mlf` to cut segments out of the audio samples. For this purpose the file `aligned.mlf` has to be passed to HCopy, with the option `-n i [j]` the number of the phones that should be cut can be defined (e.g. with `-n 1 3` the segment from phone 1 to 3 will be cut).

To automate this procedure some processing had to be done beforehand. The transcriptions of all words were analyzed to find the number of phones in each transcription. Then several lists were generated, each containing all words with a certain minimum number of phones. From these lists a second set of lists was produced. These specify the words that can be cut at a certain phone position. Each list is used for one segment position. For example, the list of words that should be cut at the phone positions 3 to 5 contains all words with at least 5 phones. After that, a script was generated that calls HCopy for each possible segment (e.g. the segment with phone 1 to 3, phone 2 to 4 etc.). With this final script all possible segments for all words were produced in one step.

5.1.2 Segmentation of the transcriptions

HCopy provides an additional useful feature: it is possible to output the transcriptions of the new segments in a so-called master label file. This was done in every call of HCopy. As it is not preferable to have several files with transcriptions, the master label files were combined in one file for further use.

5.1.3 Creation of the database

After the establishing of both the audio and the transcription segments, a database that can be accessed by the pattern comparison algorithm was created. As working environment Matlab was chosen. To establish the database, the feature files from HTK and the corresponding transcriptions in the master label file had to be imported into Matlab. The procedure was realized as follows: First the name of the segment, its reference transcription and an intermediate transcription, which I will refer to later, is saved in a Matlab cell array. Then for each segment the corresponding feature file is searched in a specified directory, is read and the coefficients are stored in the Matlab cell array. The reading process is done by using the file `readmfc.m` that extracts the MFCCs from the feature file and returns them in a matrix.

The resulting data structure is the following:

Filename	*Reference transcription*	*Intermediate transcription*	*MFCCs*
AACHEN__01-03.mfc	'sil a x'	'sil a x'	<13x38 double>
AACHEN__02-04.mfc	'a x @_bs'	'a x n_sy'	<13x39 double>
AACHEN__03-05.mfc	'x @_bs n'	'x n_sy sil'	<13x42 double>
⋮	⋮	⋮	⋮

Table 5.3: Internal triphone database

The mentioned intermediate transcription is necessary for the candidate selection which will be explained in the next section.

5.2 Candidate selection

The triphone database contains about 79 500 entries, this leads to a search space of impracticable size. Informal tests showed that on a standard PC about 300 comparisons per minute are executed, this means that the transcriptions process for one word would need approximately 4.5 hours on this machine. For this reason a candidate selection was implemented, that proposes probably matching segments to the pattern comparison algorithm. This mechanism reduces the number of comparisons and makes the transcription process faster.

The candidate selection is based on the creation of a rough automatic transcription (in the remainder called intermediate transcription) that is based on the orthographic representation of the word. This means that for all triphones in the database and for each new word that has to be transcribed the orthographic representation has to be known. For the segments in the database this is known (the word is a part of the filename integrated in the database to maintain a maximum of possible information from the data extracted from ADABA). For a new word this information has to be provided by the user. This is a small limitation of the method, however for other approaches like forced alignment this information has to be known too.

5.2.1 Implementation

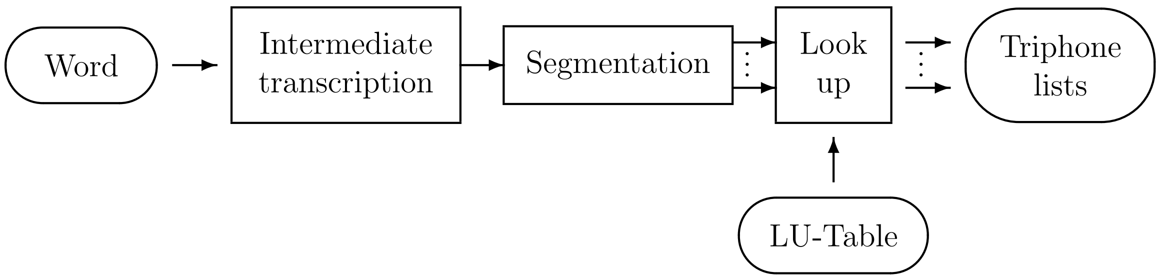


Figure 5.1: The candidate selection during the transcription process is realized by the creation of an intermediate transcription, its segmentation into triphones and a look-up for promising candidate triphones that are then passed to the pattern comparison unit.

The implementation of the candidate selection consists of two parts: In course of the creation of the internal database all triphones in the database have to be associated with a triphone derived from the intermediate transcription of the corresponding word. Then during the transcription process the following procedure is applied (see figure 5.1): For each new word the intermediate transcription has to be created and segmented into triphones. Then identical triphones are searched in a look up table that contains indices to the triphones in the database. For each segment of the intermediate transcription a list of identical triphones is passed to the pattern comparison unit.

5.2.2 Creation of the intermediate transcription

The intermediate transcription is derived with a rule-based grapheme to phoneme conversion. For the triphones in the database that intermediate transcription has to be created from the whole word and not from the triphone itself as the context of the letters in the word has to be taken into consideration. So the intermediate transcription is created from the word, then segmented and associated to the triphones of the reference transcriptions in the database (see next section).

The rule set is derived from the results of the project “Varieties of Austrian German - Standard pronunciation and varieties of standard pronunciation” (see also [23]). The applied rules work with a smaller phoneme set than that of the reference transcription to reduce the complexity. The rules were optimized to fit the data of the Austrian male speaker, as this first implementation is restricted to a single speaker. For the optimization the intermediate transcriptions were compared with simplified reference transcriptions that use only symbols from the reduced phoneme set. The number of correctly converted phones was computed for each rule. For those rules where two conversions were possible, the rule that yielded more correct phones was taken. For optimization a simple tool that computes the number of correctly transcribed phones after the application of a certain rule was used. A preliminary analysis showed that about 50% of the resulting intermediate transcriptions are identical with the simplified reference transcriptions. Future modifications could include a more sophisticated optimization process or a more detailed rule set. Another possibility is the integration of an existing grapheme to phoneme converter.

5.2.3 Alignment of the transcription segments

After the creation of the intermediate transcriptions for the internal database another problem had to be solved: The intermediate and reference transcriptions are not always of the same length, thus if both are segmented there are some segments that have no equivalent in the other transcription. For this reason an alignment procedure between the segments of the intermediate transcription and the segments of the reference transcription was realized. This was done by implementing a simple dynamic programming algorithm. For each segment the Levenshtein distance (see section 6.1.1) to all segments of the other transcription is calculated. Then the best path through the distance matrix and the alignment between the segments of the intermediate transcription and the reference transcription are computed. As result a list with the name of the segment, the segment of the reference transcription and the segment of the intermediate transcription as shown below is produced.

```
AACHEN__01-03.mfc  sil a x     sil a x
AACHEN__02-04.mfc  a x @_bs    a x n_sy
AACHEN__03-05.mfc  x @_bs n    x n_sy sil
AACHEN__04-06.mfc  @_bs n sil  x n_sy sil
AAL__01-03.mfc     sil a l     sil a l
AAL__02-04.mfc     a l sil     a l sil
      ⋮                ⋮           ⋮
```

This list is imported in Matlab to build the internal database, as described in section 5.1.3.

To ensure that the candidate selection passes a reasonable choice of triphones to the pattern comparison unit it was tested, the results follow in section 6.1.2.

5.3 Pattern comparison

In order to find the most similar triphones in the database for a new word a pattern comparison algorithm was implemented. In the next two sections some theoretical considerations for pattern comparison will be discussed. These sections are a short summary of the excellent treatise on pattern comparison by Rabiner and Juang [12]. After the theoretical parts the implemented algorithms and the details of the implementation will be presented.

5.3.1 Basic considerations

Pattern comparison can be done in many ways, the goal is to find a mathematical representation for the similarity between two speech patterns that is subjectively meaningful. This means that there has to be a correlation between the numerical value of the distance and the subjectively perceived distance of two compared speech patterns [12].

The approach chosen here is pattern-based (or equivalently, template-based [22]) meaning that the speech is represented by a time sequence of feature vectors. Thus if we have a new word we get a test pattern $\mathcal{T}$

$$\mathcal{T} = \{\mathbf{t}_1, \mathbf{t}_2, \mathbf{t}_3, \ldots, \mathbf{t}_I\}$$

with $\mathbf{t}_i$ representing a feature vector at the time i and the total number of frames of speech I. Further we define a set of reference patterns $\{\mathcal{R}^1, \mathcal{R}^2, \ldots, \mathcal{R}^V\}$, each reference pattern is also represented by a sequence of feature vectors

$$\mathcal{R}^j = \{\mathbf{r}_1^j, \mathbf{r}_2^j, \ldots, \mathbf{r}_J^j\}.$$

The aim is to compute a value for the dissimilarity, also referred to as distance, between the test pattern $\mathcal{T}$ and each reference pattern $\mathcal{R}^j$ to find the reference pattern with the minimum dissimilarity. In case of the phonetic transcription, the transcription of this reference pattern is then taken for the synthesis of the new transcription[4].

Problems that occur when searching the global dissimilarity of two patterns are as follows:

- Normally the test and reference patterns are not of the same length.

[4]More precisely, for each word the pattern comparison process is performed several times as the new transcription is established from several segments

- The patterns might not line up in a simple manner, as different sounds may vary in duration to a different degree.
- To determine the global dissimilarity of two audio samples it is necessary to evaluate the local dissimilarity between pairs of their feature vectors.

Thus for comparison a local dissimilarity measure and a global technique for time alignment is needed, with that a global dissimilarity measure can be determined by the accumulated local distances between the feature vectors in the time-aligned patterns [12].

One solution that can handle the described problems is the application of the dynamic time warping algorithm: Starting from the local distances between the feature vectors an optimal alignment is computed by minimizing the accumulated local distance. The accumulated distance from the best alignment is then taken as measure for the global dissimilarity. In the next section the problem of time alignment and normalization will be discussed. Then the general dynamic time warping algorithm and a modified version will be presented. The details of the underlying dynamic programming technique can be found in [12].

5.3.2 Time alignment and normalization

Different samples of the same speech utterance are rarely realized at the same speaking rate. However, when two utterances or tokens of utterances are compared, the dissimilarity neither should be influenced by speaking rate variation nor by duration variation. Hence a time normalization has to be performed to ensure that the dissimilarity measure provides a meaningful result.

As an example, take two speech patterns $\mathcal{X}$ and $\mathcal{Y}$. These patterns are defined by their sequences of feature vectors $(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_{T_x})$ and $(\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_{T_y})$, where $\mathbf{x}_i$ and $\mathbf{y}_i$ are the feature vectors at the time i_x and i_y, respectively. The dissimilarity between $\mathcal{X}$ and $\mathcal{Y}$ is calculated with the aid of the short-time spectral distances[5] $d(\mathbf{x}_{i_x}, \mathbf{y}_{i_y})$, in a simpler notation $d(i_x, i_y)$, with $i_x = 1, 2, \ldots, T_x$ and $i_y = 1, 2, \ldots, T_y$. The spectral distance can be measured in several ways, for more details see section 4.5 in [12].

The sequential order of a speech pattern provides important information and thus may only be changed to a certain degree. When comparing the spectral pairs $d(\mathbf{x}_{i_x}, \mathbf{y}_{i_y})$ the indices must obey specific order constraints. For example the temporal order of the feature vectors should be kept and during the alignment the modification of the time axis (the "warping") should not exceed a certain limit.

One simple solution for the problem of time alignment is a linear time normalization method. However this is not very meaningful, as a linear alignment assumes that the time variation is independent of the produced sound. This is not the case in reality, where different sounds vary in duration to a different degree.

[5] or distortions

A more general approach is to use two warping functions ϕ_x and ϕ_y, which normalize the indices i_x and i_y of two speech patterns to a common time axis k:

$$i_x = \phi_x(k) \tag{5.1}$$

and

$$i_y = \phi_y(k). \tag{5.2}$$

Based on these warping functions a measure for the global pattern dissimilarity $d_\phi(\mathcal{X}, \mathcal{Y})$ can be defined as the accumulated distance over the whole utterance

$$d_\phi(\mathcal{X}, \mathcal{Y}) = \sum_{k=1}^{T} d(\phi_x(k), \phi_y(k)) m(k)/M_\phi. \tag{5.3}$$

Here $d(\phi_x(k), \phi_y(k))$ is the short-time spectral distance between $\mathbf{x}_\phi$ and $\mathbf{y}_\phi$, $m(k)$ denotes a weighting coefficient and M_ϕ is a normalization factor.

Finally what is missing for the definition of a dissimilarity between $\mathcal{X}$ and $\mathcal{Y}$ is the specification of the path $\phi = (\phi_x, \phi_y)$. As the number of possible paths is large, the task of finding an appropriate warping function to determine the global dissimilarity is of crucial importance and has to be done properly.

One popular approach is to minimize $d_\phi(\mathcal{X}, \mathcal{Y})$ over all possible paths and define the dissimilarity between $\mathcal{X}$ and $\mathcal{Y}$, denoted as $d(\mathcal{X}, \mathcal{Y})$, such that

$$d(\mathcal{X}, \mathcal{Y}) = \min_\phi d_\phi(\mathcal{X}, \mathcal{Y}), \tag{5.4}$$

where ϕ is restricted by certain conditions, that will be discussed in the following section.

This definition of $d(\mathcal{X}, \mathcal{Y})$ seems intuitively meaningful when $\mathcal{X}$ and $\mathcal{Y}$ provide utterances of the same word. Choosing the path with the minimum accumulated distance means that the dissimilarity is evaluated along the path that results in the best alignment [12].

Time-normalization constraints

For a reasonable alignment of two speech utterances the application of some constraints to the warping algorithm is recommended. These are [12]:

- endpoint constraints
- monotonicity constraints
- local continuity constraints
- global path constraints
- slope weighting.

For the implementation of the algorithm for time alignment not all of these constraints are indeed necessary, see section 5.3.3 for the details of the implementation.

Endpoint constraints When isolated tokens of speech utterances are compared, the endpoints of the token usually define the beginning and the endpoint of the pattern. For time normalization the variations only within these endpoints are considered. This leads to the following set of constraints for the warping function:

$$\text{beginning point} \quad \phi_x(1) = 1, \quad \phi_y(1) = 1 \tag{5.5}$$
$$\text{ending point} \quad \phi_x(T) = T_x, \quad \phi_y(T) = T_y. \tag{5.6}$$

In some situations, e.g., noisy environments, the endpoints may not be reliably determined. In this case the above constraints have to be relaxed.

Monotonicity constraints For speech utterances the temporal order is of extreme importance to the linguistic meaning. For example, the two words "we" and "you" provide spectral sequences that are almost time-reversed. If no constraints for the warping function are given and time reversal is allowed, the computed dissimilarity of these two utterances would be small, what obviously is not meaningful for the speech comparison. Therefore it is useful to set a monotonicity constraint to keep the correct temporal order of a feature sequence, such that

$$\phi_x(k+1) \geq \phi_x(k) \tag{5.7}$$
$$\phi_y(k+1) \geq \phi_y(k). \tag{5.8}$$

This means that any allowed path in the distance matrix has no negative slope thus time-reversed warping is not possible.

Local path constraints For the correct recognition or transcription of a word the presence or absence of a particular sound (or phone) is often crucial. For this reason such an important sound should not be omitted during the time alignment. This can be ensured by imposing a set of local continuity constraints. The constraints can take many forms, one example is:

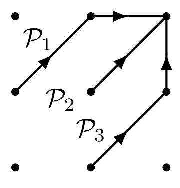
$$\phi_x(k+1) - \phi_x(k) \leq 1 \tag{5.9}$$
$$\phi_y(k+1) - \phi_y(k) \leq 1 \tag{5.10}$$

A more convenient form of representation is to define a path as a sequence of steps each given by a pair of coordinate increments, such as

$$\mathcal{P} \rightarrow (p_1, q_1), (p_2, q_2) \ldots (p_T, q_T) \tag{5.11}$$

Then the possible paths can be easily depicted by a graph as for example the three paths $\mathcal{P}_1 \rightarrow (1,1)(1,0)$, $\mathcal{P}_2 \rightarrow (1,1)$ and $\mathcal{P}_3 \rightarrow (1,1)(0,1)$in figure 5.2. Figure 5.3 shows three examples of the local continuity constraints proposed by Rabiner and Juang [12].

The best local continuity constraint for a certain application cannot be determined analytically, as it is complicated to model the speaking rate and duration changes in speech. Therefore the best continuity constraint for a special purpose can only be found experimentally.

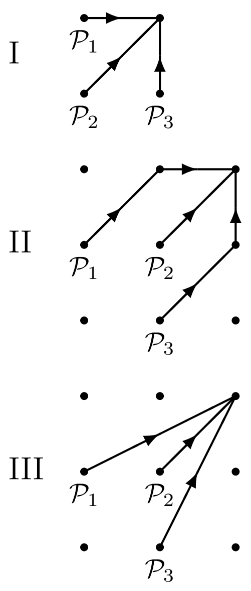


$\mathcal{P}_1 \rightarrow (1,1)(1,0)$

$\mathcal{P}_2 \rightarrow (1,1)$

$\mathcal{P}_3 \rightarrow (1,1)(0,1)$

Figure 5.2: Example for local continuity constraints [12]



$\mathcal{P}_1 \rightarrow (1,0)$
$\mathcal{P}_2 \rightarrow (1,1)$
$\mathcal{P}_3 \rightarrow (0,1)$

$\mathcal{P}_1 \rightarrow (1,1)(1,0)$
$\mathcal{P}_2 \rightarrow (1,1)$
$\mathcal{P}_3 \rightarrow (1,1)(0,1)$

$\mathcal{P}_1 \rightarrow (2,1)$
$\mathcal{P}_2 \rightarrow (1,1)$
$\mathcal{P}_3 \rightarrow (1,2)$

Figure 5.3: Local constraints and the resulting path specifications

Slope weighting Slope weighting is another possibility of controlling the search for an optimal time alignment. As can be seen in equation 5.3 the weighting function $m(k)$ determines the contribution of each local distance to the overall distance. The weighting function can either be used on a "global" scale or on a "local" scale. The first will not be discussed longer here (see [12]), as only the latter is of importance in the currently realized system. On the "local" scale the weighting function can be combined with the local path constraints, that way a preference for the paths that are allowed locally is implemented. Normally these weighting functions are referred to as slope weighting functions because of their relation to the slope of the local path constraints.

Similar to the path constraints many heuristic slope weighting functions can be applied. The following set of slope weighting functions was presented by Sakoe and Chiba [12]:

$$\begin{aligned}
\text{Type (a)} \quad m(k) &= \min[\phi_x(k)-\phi_x(k-1), \phi_y(k)-\phi_y(k-1)] && (5.12)\\
\text{Type (b)} \quad m(k) &= \max[\phi_x(k)-\phi_x(k-1), \phi_y(k)-\phi_y(k-1)] && (5.13)\\
\text{Type (c)} \quad m(k) &= \phi_x(k)-\phi_x(k-1) && (5.14)\\
\text{Type (d)} \quad m(k) &= \phi_x(k)-\phi_x(k-1)+\phi_y(k)-\phi_y(k-1) && (5.15)
\end{aligned}$$

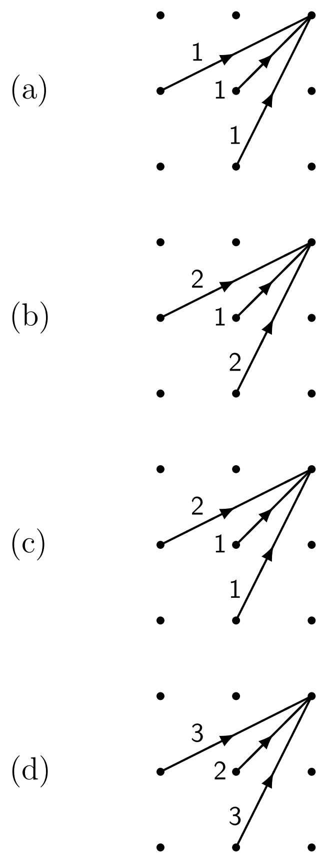


Figure 5.4: Type III local continuity constraints with four types of slope weighting [12]

The effect of applying slope weighting is demonstrated in figure 5.4. It shows the Type III continuity constraints with the above four weighting functions. The number along each path denotes the weighting value. Higher distances mark less favorable or less likely paths, therefore a higher weighting value can be used to define less preferable paths.

Normalization factor As normally speech patterns of different lengths are compared, an overall normalization is necessary to get an average accumulated distance that is independent of the lengths of the compared patterns. For this reason an overall normalization factor M_ϕ is used, whos value depends on the used slope weighting type. Usually it is defined as follows

$$M_\phi = \sum_{k=1}^{T} m(k), \tag{5.16}$$

so it equals the sum of the components of the weighting function. For the slope weighting types (c) and (d) the normalization factors can be computed in the following way:

$$M_\phi^{(c)} = \sum_{k=1}^{T} [\phi_x(k) - \phi_x(k-1)] = \phi_x(T) - \phi_x(0) = T_x \tag{5.17}$$

$$\begin{aligned} M_\phi^{(d)} &= \sum_{k=1}^{T} [\phi_x(k) - \phi_x(k-1) + \phi_y(k) - \phi_y(k-1)] \\ &= \phi_x(T) - \phi_x(0) + \phi_y(T) - \phi_y(0) = T_x + T_y. \end{aligned} \tag{5.18}$$

They are independent of the best path and the chosen constraints. However the normalization factor for the slope weighting functions of type (a) and (b) depends on the actual path. This is not practicable when the search for the best path is realized with a dynamic programming algorithm. For this reason an arbitrary but meaningful normalization factor is taken that is independent of the actual warping function. Thus for the slope weighting types (a) and (b) normally

$$M_\phi^{(a)} = M_\phi^{(b)} = T_x \tag{5.19}$$

is chosen.

5.3.3 Dynamic time warping algorithm (DTW)

The dynamic time warping algorithm (DTW) provides a solution for the time alignment problem that implies the discussed constraints. The algorithm finds the best path through a matrix containing the pairwise distances from two feature sequences in order to derive the global distance between the two underlying speech patterns. Assume a distance matrix of size T_x by T_y and let the path start at (1,1) and end at (T_x, T_y), then the dynamic programming algorithm can be summarized as follows [12]:

1. Initialization

$$D_A(1,1) = d(1,1)m(1).$$

   D_A is the matrix containing the accumulated local distances, so $D_A(1,1)$ is the value for the accumulated local distance at point $(1,1)$. $m(1)$ is the slope weighting factor.

2. Recursion
   For $1 \leq i_x \leq T_x, 1 \leq i_y \leq T_y$ compute

$$D_A(i_x, i_y) = \min_{(i'_x, i'_y)} [D_A(i'_x, i'_y) + \zeta((i'_x, i'_y), (i_x, i_y))],$$

   where $D_A(i_x, i_y)$ and $D_A(i'_x, i'_y)$ denote the local accumulated local distance in point (i_x, i_y) and point (i'_x, i'_y), respectively. The term $\zeta((i'_x, i'_y), (i_x, i_y))$ represents the weighted accumulated distance between the points (i'_x, i'_y) and (i_x, i_y), thus in each recursion step the point (i'_x, i'_y) from which the point (i_x, i_y) can be reached with the minimum accumulated distance is searched.

3. Termination
$$d(\mathcal{X}, \mathcal{Y}) = \frac{D_A(T_x, T_y)}{M},$$

   where $d(\mathcal{X}, \mathcal{Y})$ is the global pattern dissimilarity defined by the accumulated local distance in point (T_x, T_y) and normalized by the normalization factor M.

Implementation Settings

For the implementation of the transcriptor several settings for the DTW algorithm were tested. For the local path constraints the settings shown in figure 5.5 were chosen. In our special case the comparison is done between two utterances with a great difference in length, as a triphone is compared to a word (e.g., a triphone of 36 ms compared to a word of 103 ms as illustrated in figure 5.6). For this reason the possible paths should not be restricted, with these settings the path can reach every point in the distance matrix. For the same reason no global path constraints were implemented. The endpoint constraints were neglected as the audio files were assumed to be cut properly. As spectral distance measure the Euclidean distance was used.

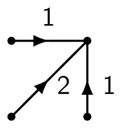


Figure 5.5: Implemented slope weighting factors for the DTW

To find the optimal implementation settings different slope weighting factors and different values for the normalization factor were tried. The best settings were achieved with the initial slope weighting factors as shown in figure 5.5 and with a normalization to the length T_x of the feature vector of the word (the results are discussed in detail in section 6.2.3).

Figure 5.6 shows an example for the resulting distance matrix D, where the word "Aquarell" and the triphone "sil a k" are compared. In figure 5.7 the matrix D_A with the accumulated distances and the best path found by the DTW are shown.

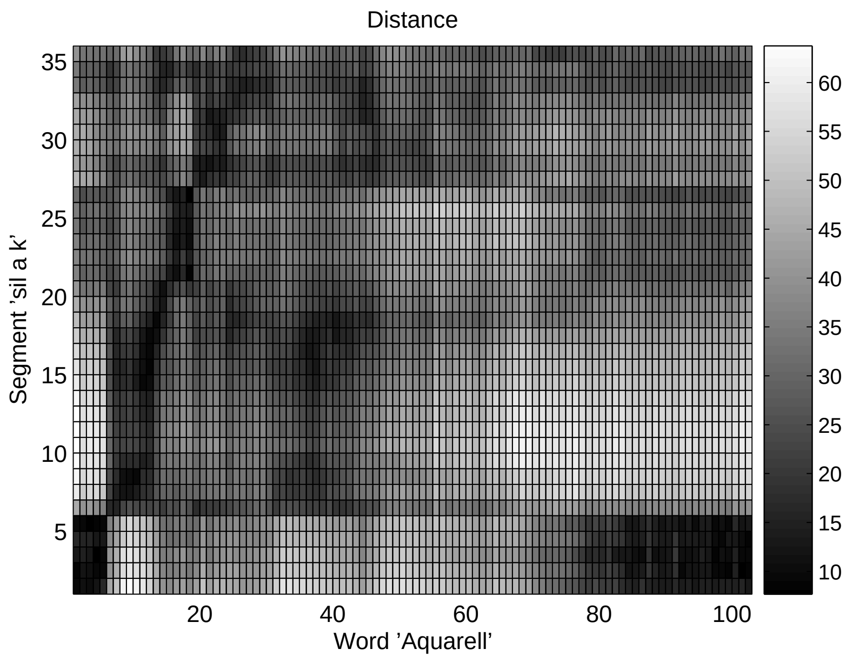


Figure 5.6: Distance matrix D between the word "Aquarell" and an example for the triphone "sil a k"; dark colors indicate low distances and light colors high distances.

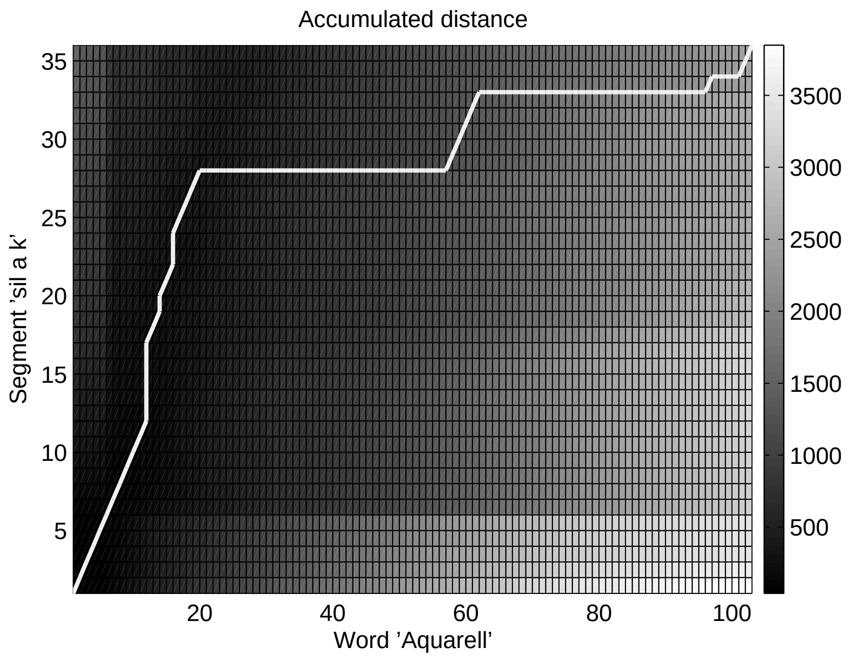


Figure 5.7: Matrix D_A with accumulated distances, computed between the word "Aquarell" and the same example for the triphone "sil a k" as in figure 5.6; the line marks the best path.

5.3.4 Segmental DTW

A variation of the dynamic time warping algorithm was proposed by Park and Glass [29]. The algorithm called segmental DTW finds matching subsequences in pairs of speech utterances. It was used for unsupervised word discovery in information retrieval and speech segment clustering. However, this solution seems to be well applicable on the problem of finding matching segments in the database for a new word that has to be transcribed.

In standard DTW two utterances with possibly the same content (i.e. phones) are aligned to determine a measure for the distance between them and to decide whether the content is the same or not. However, this is not useful if matching words within utterances composed of more words should be found. The segmental DTW algorithm provides a solution for this problem.

As in standard DTW, a distance matrix with the pairwise distances between the feature vector sequences is computed. Instead of searching the best path from point (1,1) to point (T_x, T_y) multiple paths with different beginning and ending points are investigated. This is done by dividing the distance matrix into several overlapping diagonal bands and then searching the best path within each of these bands. Figure 5.8 shows one band of a distance matrix as example. The overlapping bands serve not only the purpose of allowing multiple alignments but also avoid an extreme degree of warping that would overly distort the sub-utterances.

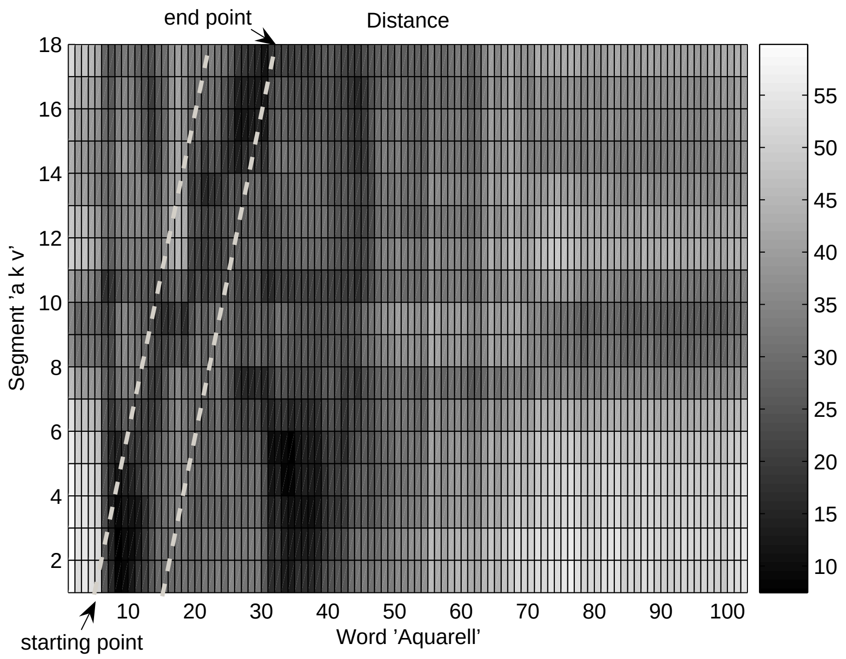


Figure 5.8: Distance matrix between the word “Aquarell” and the triphone “a k v” with the minimum distance of all examples; the dashed lines indicates the band with minimum distance, it can be seen that within this band the distances are small.

Each resulting path is then trimmed by searching the least average subsequence with the minimum length L. The minimum length constraint is implemented to avoid incorrect matches between short sub-sequences in two utterances. The sub-sequence with the minimum average represents the part of the aligned path that yields a good alignment.

The process results in one path fragment for each diagonal band in the distance matrix. The path fragment with the smallest distance is taken as the best alignment and the distance is used as the distance between the two initial speech utterances.

In the pattern comparison unit of the transcriptor a similar problem as described by Park and Glass has to be solved: In one approach a triphone in a whole word shall be identified, in the other a word in a longer speech utterance shall be found. Obviously both approaches deal with the problem of finding a smaller sub-unit in a speech pattern. Thus the segmental DTW was judged a meaningful solution that might yield better results than the standard DTW. For this reason it was implemented in addition to the normal DTW.

Implementation Settings

For the overlapping bands a width of 10 frames was chosen, this is equivalent to a duration of 100 ms. The slope of the bands was defined to be 45 degree as the two speech patterns were assumed to align roughly in a linear way. As local path constraints the same constraints as for the standard DTW were implemented. For simplicity the starting point and the end point in each sub-band were chosen to be fixed, no search for an optimal path with a certain minimum length as proposed be Park and Glass was realized. As starting point the leftmost possible point in the first row of the sub-band was taken. The endpoint was defined as the rightmost point in the last row (see figure 5.8). This might yield a result that is not optimal but the error was assumed to be negligible, as all paths in the sub-bands had the same starting and end point conditions and the possible influence of some suboptimal local distances at the beginning and the end is small. As for the standard DTW the Euclidean distance was used as distance measure.

For each comparison the band that provides the smallest accumulated distance is found and its distance is taken as the global distance between the current triphone and the new word.

The use of the segmental DTW is intuitively more meaningful than the use of the DTW algorithm as not a whole word and a triphone are compared, but the triphone is compared to a segment that is cut out of the whole word with the aid of the sub-band.

5.3.5 Implementation of the pattern comparison

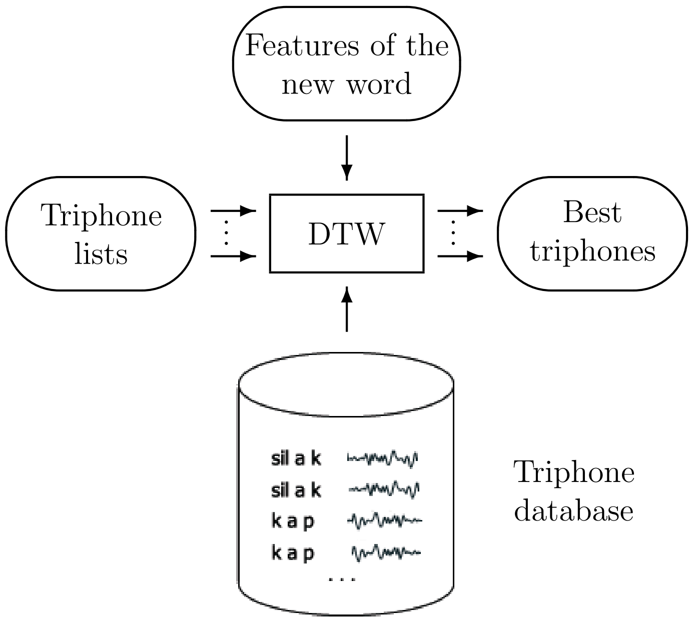


Figure 5.9: The features of the triphones chosen by the candidate selection are compared to the features of the new word, the three best matching triphones for each segment of the new word are passed to the synthesis unit.

For each new word that has to be transcribed, an intermediate transcription is created from the orthographic representation and then segmented into triphones. With these triphones the candidate triphones from the database are chosen by the candidate selection unit as described in section 5.2.1. The audio data of all candidate segments is compared to the audio sample of the new word via the DTW or segmental DTW algorithm. The resulting distances of the candidates for each triphone are stored, ordered and the three segments with the least distances are found. These segments and the corresponding distances are passed to the synthesis unit to assemble the transcription for the new word.

Fallback strategy

Sometimes it may occur that a triphone is not present in the look up table, then for this segment no candidates and obviously no best matching segments are found. For this case a fallback strategy is implemented.

For the candidate selection a parameter that defines the minimum number of candidates is given, the default value is set to three. If less than three candidates or even no candidates are found, one of the three phonemes in the intermediate transcription is replaced by its phone class then in another look up table the corresponding candidates are searched. This is done for all three phonemes separately. If still the necessary minimum number of candidates is not achieved, two phonemes at a time or even all three are replaced and the corresponding candidates are searched with these specifications. The applied substitutions are summarized in table 5.4 (note that for trills there is only the symbol 'R', thus no substitution is necessary). The phone classes are derived from the IPA-chart [26].

Phone class	Substituted symbols of intermediate transcription
Plosive	b, d, g, k, t
Nasal	N, N=, m, m=, n, n=
Fricative	C, S, f, h, s, v, x
Approximant	j
Lateral approximant	l, l=
Vowel	2, 3, 6, @\, @, a, a_(, e, i, o, o_(, p, u, y

Table 5.4: Phone classes for the fallback strategy

If for some reason, e.g., an error in the orthographic representation, no candidate at all is found the parameter that passes the segments to the transcription unit is left empty.

5.4 Transcription synthesis

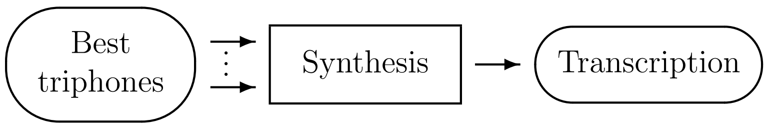


Figure 5.10: The synthesis unit creates the transcription from the best triphones.

From the pattern comparison unit the three best matching segments and their global distances for each triphone in the new word are passed to the synthesis unit. Two different approaches to identify the best phones from the triphones were tested, the minimum distance procedure (MDP) and the majority vote procedure (MVP). The majority vote procedure proved better and is implemented in the final version. The test results are discussed in section 6.2.3.

sil	f	R	f	R	a	R	a	k_h	a	g	@	g	@	sil
sil	f	R	f	R\	a	R	a	k	a	g	@	g	@	sil
sil	f	R	f	R	a	R	a	k	a	g	@	g	@	sil

Table 5.5: Three best triphones for each segment of the word "Frage"

Table 5.5 shows the most similar triphones for each segment of the unknown audio sample for the word "Frage". For both procedures the triphones are split into phones, then the phones are aligned according to their position, as illustrated in table 5.6.

sil	f	R												
sil	f	R												
sil	f	R												
			f	R	a									
			f	R\	a									
			f	R	a									
						R	a	k_h						
						R	a	k						
						R	a	k						
									a	g	@			
									a	g	@			
									a	g	@			
												g	@	sil
												g	@	sil
												g	@	sil

$\longrightarrow$

sil	f	R				
sil	f	R				
sil	f	R				
	f	R	a			
	f	R\	a			
	f	R	a			
		R	a	k_h		
		R	a	k		
		R	a	k		
			a	g	@	
			a	g	@	
			a	g	@	
				g	@	sil
				g	@	sil
				g	@	sil

Table 5.6: The phones are aligned according to their intended position in the final transcription.

5.4.1 Minimum distance procedure

The minimum distance procedure uses the global distances of the triphones to decide which phones are selected for the transcription. Based on the triphones from the pattern comparison the phone combination with the smallest overall distance is found. This is done the following way: When the triphones are split into phones, for each phone the corresponding distance is saved. Then for each phone position the phone with the smallest distance is taken. The combination of these phones determines the transcription of the new word.

5.4.2 Majority vote procedure

sil	f	R	a	g	@	sil
		R\		k		
				k_h		

Table 5.7: Ranked list of occurring phones and the resulting transcription.

In the majority vote approach the synthesis unit extracts the phones that occur most often for each position after the alignment. A list of the occurring phones is created for each position and the number of occurrence of each phone is counted. The phones are ranked according to their counter (as shown in table 5.7). If there is one phone that is the clear winner, it is selected for this certain position in the transcription. Note that in this case the global distance of the phones or triphones is not used. However if there are two or more phones with the same counter, the global distance is applied as selection criterion: for each phone the sum of the distances is built and the phone with the minimum sum is chosen for the transcription.

5.5 Transcription process summary

After the description of the different units of the transcriptor, the whole transcription process will be summed up shortly in this section.

For the transcription of a new word, the audio file or the file containing the encoded MFCCs, and the orthographic representation of the word are necessary. The transcription process can be performed either with the transcription function in Matlab directly or via the GUI, that calls the transcription function. If the input is an audio file, the data is encoded with the tool HCopy from HTK, else if the data is already available as encode feature file it is read.

For the candidate selection the string containing the orthographic representation of the word is taken and converted to the intermediate transcription. The intermediate transcription is then segmented into triphones. These two steps, the conversion and the segmentation are conducted by Perl scripts. Each of the resulting triphones is searched in a look up table to find triphones in the database that have the same intermediate transcription. If the triphone is found in the look up table the list with the corresponding triphones in the database is passed to the pattern comparison unit.

All triphones are compared with the data from the new word and the best three matching triphones are found. This is done for every segment from the intermediate transcription of the new word.

Finally the three best matching triphones per segment of the intermediate transcription are passed to the transcription synthesis unit that composes the new transcription from the proposed triphones.

Chapter 6

Tests and results

6.1 Validation

The evaluation addresses two main issues: the quality of the transcriptions for several test settings and the validation of the candidate selection.

6.1.1 Transcriptions

For the evaluation of the automatic transcriptions the transcriptions from ADABA that were already involved in the segmentation step were used as reference transcriptions. As already mentioned these contain only phonetic symbols and no length or stress markers.

The comparison between automatic and reference transcription was performed by calculating the Levenshtein distance that provides an intuitively meaningful measure for the difference of two transcriptions. Furthermore it indicates the number of modifications a human transcriber had to perform to correct possible erroneous transcriptions.

To enable a comparison with other existing applications the rate of correct phones and the phone accuracy were calculated. For some test settings additionally the number of substitutions, deletions and insertions was computed to further investigate the occurring errors.

Levenshtein distance

The Levenshtein distance, also called edit distance, is a simple measure to determine the distance between two strings. It is named after V. Levenshtein who published an article where this distances measure was probably introduced first [30].

The method to compute the distance is based on the transformation of one string into the other by applying a series of operations on the characters of the first string. These operations are insertion, deletion and substitution. For example, take the two words "vintner" and "writers": The first string can be transformed into the second by substituting the v in vintner with the w from writer, inserting an r after the w, deleting the first and the second n, and finally inserting an s at the end. The two

```
v intner
wri t ers
SIMDMDMMI
```

Figure 6.1: Transformation of the string “vintner” into the string “writer” with the series of applied edit operations: S stands for substitution, I for insertion, D for deletion and M for match (no operation) [30].

strings are shown in figure 6.1 along with the corresponding operations performed on each character, where the symbol I denotes the insertion operation, S is taken for the substitution, D marks a deletion and M means match (no operation has to be applied) [30]. The Levenshtein distance is then defined as the minimum number of operations that are necessary to transform the first string into the second. Hence, for the given example the distance equals 5.

The value for the distance as well indicates the number of operations that have to be performed by a human transcriber during the correction of an erroneous automatic transcription, i.e. when he transforms the automatic transcription to the correct reference transcription. Thus it is a meaningful measure for the degree of correctness of the automatic transcriptions and it equally describes the effort that is necessary if the automatic transcriptions should be corrected.

The computation of the Levenshtein distance is realized via a simple dynamic programming algorithm, as described in [30].

Measures for the correctness of phones

The performance of automatic transcription or speech recognition systems is often evaluated using the rate of correct phones and the phone accuracy.

The rate of correct phones is derived by the number of substitutions S and deletions D of phones [19]

$$\text{Percent correct} = \frac{N - D - S}{N} \times 100\% \tag{6.1}$$

where N is the total number of phones in the reference transcription. As the number of insertions I is not taken into consideration, good results for this measure could be achieved by permitting many insertions. This is not desirable though.

The phone accuracy considers the insertion errors and hence is more representative [19]. It is defined as

$$\text{Phone accuracy} = \frac{N - S - D - I}{N} \times 100\% \tag{6.2}$$

and thus is equal to the number of correct or identical phones $(N-S-D)$ subtracted by the number of insertions and divided by the total number of phones [31].

Another measure is the percentage disagreement[1] between the validated transcription and the reference transcription [7].

$$\text{Percentage disagreement} = \frac{S + D + I}{N} \times 100\% \tag{6.3}$$

The percentage disagreement is equivalent to the phone error rate (PER) [7]. A further measure is the percentage agreement presented in section 2.2.3.

The performance of the systems that were presented in chapter 2 is mostly measured either by the percentage agreement or the percentage disagreement between automatic and reference transcriptions. However some authors present the number of identical or correct phones, thus the results cannot always be compared directly.

6.1.2 Candidate selection

The candidate selection strongly influences the performance of the whole automatic transcription process. Its primary function is the reduction of the search space i.e. the reduction of the number of triphones that are taken from the database and passed to the pattern comparison unit. This means that the candidate selection has some side-effects. On the one hand it predefines possible transcriptions and already accomplishes part of the transcription task by simply applying rules, on the other hand in some cases it may reject possibly well matching segments and thus prevent a good transcription.

Tests have shown that this rule-based candidate selection alone achieves a rate of correct transcriptions (i.e. identical with the reference transcriptions) of around 30% for the test corpus of the Austrian male speaker (see section 6.2.4). This has to be considered when evaluating the results of the whole transcription process. The performance of the DTW algorithm alone can only be estimated objectively when the improvement after the candidate selection is evaluated. To ensure the correct function of the candidate selection the measures precision and recall were calculated for one test run. The scores for precision and recall indicate if the selection chooses useful candidates, the results are discussed in section 6.2.4 .

Precision and Recall

The measures precision and recall are used in information retrieval. They indicate if the documents that are found by an information request are relevant. For example let R be the set of relevant documents for an information request and $|R|$ the number of these documents. The processing of the information request yields an answer set A with a number of $|A|$ documents. Furthermore, consider the intersection of the datasets R and A that contains $|R_a|$ documents.

[1]Note that this definition is not consistent with the definition for the percentage agreement given by Kessens and Strik in [1] that was mentioned in section 2.2.3, i.e. that percentage agreement and the percentage disagreement here do not sum up to 100%.

Then the precision is defined as the number of relevant documents that were found divided by all documents that were found [32].

$$\text{Precision} = \frac{|R_a|}{|A|} \tag{6.4}$$

The recall is defined as the number of relevant documents that were found divided by the number of all relevant documents (that should have been retrieved) [32].

$$\text{Recall} = \frac{|R_a|}{|R|} \tag{6.5}$$

The values for both measures range between 0 and 1. For the precision a score of 1 means that all documents that were found are relevant. For the recall the score 1 denotes that all existing relevant documents could be retrieved.

In case of the candidate selection, the whole process of creating an intermediate transcription, segmenting it and reading the list of triphones with the same intermediate transcription belongs to the information request. The candidates that provide the same intermediate transcription as the new word are assumed to have a reference transcription that fits as transcription for the new word.

To verify if the candidate selection indeed chooses candidate triphones that are relevant in the sense that they provide possibly matching transcriptions, the reference transcriptions from the new words are compared to the reference triphones that correspond to the intermediate triphones retrieved by the candidate selection. For this purpose the reference transcriptions from the new words are segmented into triphones. This is the set of relevant documents R. The reference triphones that belong to the intermediate triphones returned by the candidate selection form the dataset A.

6.2 Tests

The transcriptor was developed and tested using the results of several simulations. First some informal tests were performed to estimate the performance of the overall transcription system and to ensure that the candidate selection works well. Then the standard DTW and the segmental DTW algorithm were tested with several settings in an initial test run. With the parameters that yielded the best results for the initial tests some further optimizations were performed to find the best settings for the final implementation.

6.2.1 Data preparation

For the simulations the data of the Austrian male speaker in ADABA was randomized and divided into a training, a development and a test set. The training set contains 10819 files (85% of the data), the development set 636 (5%) and the test set 1274 (10%). The audio files were used in three versions: the original wav files from ADABA, a version that was downsampled to 16kHz and another downsampled to 32kHz[2]. The initial tests were performed with each of these sets to investigate the influence of downsampling.

6.2.2 Training

The training set was used to train the HMMs in HTK, align the audio data, segment it into triphones, and establish the internal database as described in section 5.1. For the first informal tests the alignment was performed with all data, in the final evaluation this step was performed with the training data only to guarantee the independence of the development and the test set from the internal database (i.e. no information from the development or test set was used for the segmentation of the audio data for the internal database).

6.2.3 Development

The optimization of the settings for the DTW algorithm was performed with the development set. As initial test the standard and the segmental DTW algorithm were tested with different settings for the normalization factor and with the audio files of different sampling rate. Then with the settings that yielded the best results some further test runs were performed to investigate the transcription synthesis, the slope weighting factors and the minimum number of triphone candidates.

The segmental DTW yielded better results and thus was implemented in the final version of the transcriptor.

Initial Tests: DTW

The DTW algorithm was tested with three different normalization factors, with the data sets of 16 kHz, 32 kHz and 44.1 kHz sampling frequency and with the two synthesis variants majority vote procedure (MVP) and minimum distance procedure (MDP). In table 6.1 the average Levenshtein distances for the different settings

f_s	MVP			MDP		
	T_x	T_y	$T_x + T_y$	T_x	T_y	$T_x + T_y$
16kHz	**0.825**	1.256	1.028	1.181	2.445	1.675
32kHz	0.868	1.263	1.028	1.239	2.362	1.701
44kHz	0.887	1.274	1.020	1.201	2.421	1.723

Table 6.1: Standard DTW: Average Levenshtein distances for the development set with different sampling frequencies, different normalization factors and the synthesis variants majority vote procedure (MVP) and minimum distance procedure (MDP).

are shown. The distances demonstrate that the choice of T_x (the length of the feature vector of the new word) as normalization factor yields the best results both

[2]The downsampling was realized with praat [33].

for the majority vote procedure and for the minimum distance procedure over all sampling frequencies. The choice of T_x as normalization factor is reasonable as this normalization factor compensates best the influence of the length of the new word on the path and on the final global distance.

The majority vote procedure performs better than the minimum distance procedure. The Levenshtein distances for the majority vote procedure are lower for all settings. Apparently the number of occurrences of the phones allows a more robust decision than the evaluation of the distance alone. Probably the distance itself is not sufficiently objective or representative for a decision between the different triphones.

The best results are achieved with the files of 16 kHz sampling frequency. This may be explained by the fact that in the frequency range up to 8 kHz the most important information for speech is present. The higher frequency ranges probably do not contribute so much useful information to the pattern comparison unit. In HTK the filterbank is placed from zero up to the Nyquist frequency (half the sampling frequency). Therefore the bandwidth of the filters is narrower for the downsampled files. This allows for a better discrimination of the phonetic content. In other words the MFCCs of the downsampled files contain information of the more relevant frequency ranges in a higher resolution and thus yield better results.

Utterances		Phones	
Correct	313	Correct	4387
Total number	636	Total number	4807
Percentage correct	49.21%	Percentage correct	91.26%
Average LD	0.825	Phone accuracy	89.08%

Table 6.2: Best results for the development set with the standard DTW: MVP with normalization to T_x and files of 16 kHz sampling frequency.

The simulation with the files with 16 kHz sampling frequency together with the normalization factor T_x yielded the best overall result for the Levenshtein distance. The details of the evaluation are shown in table 6.2. Additionally to the Levenshtein distance the number of correct utterances and phones, the total number of utterances and phones, and the percentage of correct utterances and phones is given. The rate of correct utterances is 49.21%. This seems not high however one has to take into consideration that already one false phone leads to a utterances that is not correctly transcribed. The rate of correct phones and the phone accuracy are much more significant values as in most other applications not the percentage of correct utterances but the percentage of correct phones or the phone accuracy are evaluated. The rate of correct (or identical) phones is 91.26% and the phone accuracy equals 89.08%. These values are good compared to other applications (for a further discussion see section 6.2.4).

The details of the evaluation of the simulations with the other settings can be found in section A.1.1 in the appendix.

Initial Tests: Segmental DTW

The segmental DTW was tested like the standard DTW with three different normalization factors, the three data sets of different sampling frequency and with both synthesis variants. In table 6.3 the resulting average Levenshtein distances are shown.

f_s	MVP			MDP		
	T_x	T_y	$T_x + T_y$	T_x	T_y	$T_x + T_y$
16kHz	0.932	**0.748**	0.873	1.436	0.923	1.325
32kHz	0.906	0.778	0.854	1.442	0.983	1.379
44kHz	0.909	0.778	0.858	1.492	1.003	1.412

Table 6.3: Segmental DTW: Average Levenshtein distances for the development set with different sampling frequencies, different normalization factors and the synthesis variants majority vote procedure (MVP) and minimum distance procedure (MDP).

The results with the normalization factor T_y (the length of the triphone from the database) are the best for all data sets and both synthesis variants. This is meaningful as a normalization to the length T_x does not make sense because the distance matrix is divided into bands where the length T_x does not influence the length of the band but the length is derived from T_y. Thus the normalization to T_y is reasonable as it is able to compensate the influence of the path length best. Again the majority vote procedure yields better results than the minimum distance procedure for all settings, thus it seems to be the better choice. The results for the data sets of the different sampling frequency also show that the files with the sampling frequency of 16 kHz mostly yield the best results. They confirm the results from the tests with the standard DTW.

The best overall result is achieved with the normalization to T_y, the majority vote procedure and the files of 16 kHz sampling frequency. The details of the evaluation for these settings can be seen in table 6.4.

Utterances		Phones	
Correct	314	Correct	4435
Total number	636	Total number	4807
Percentage correct	49.37%	Percentage correct	92.26%
Average LD	0.748	Phone accuracy	90.10%

Table 6.4: Best results for the development set with the segmental DTW: MVP with normalization to T_y and files of 16 kHz sampling frequency.

Compared to the standard DTW the results for the segmental DTW are slightly better. The average Levenshtein distance is 0.748 (DTW: 0.825), the rate of correct phones 92.26% (DTW: 91.26%) and the phone accuracy 90.10% (DTW: 89.08%).

The details of the evaluation for all settings are presented in section A.1.2.

Optimization of the synthesis variant

The majority vote procedure yielded better results than the minimum distance procedure for all settings of the initial tests. One further test was performed to optimize the majority vote procedure. When the number of occurrences of two phones is equal, the distance is considered for the decision between the two phones. In one variant of the algorithm the sum of the distances is taken as criterion, in a second variant the minimum of the distances leads to the decision. During the initial tests the sum-of-distances variant was applied. For comparison, another test was performed with the minimum-of-distances variant. The tests were realized with the standard and the segmental DTW algorithm for the files of 16 kHz sampling frequency. The normalization factors were T_x for the standard DTW and T_y for the segmental DTW. The resulting Levenshtein distances are shown in table 6.5.

Algorithm	MVP with	
	sum	min.
DTW	0.826	0.829
Segmental DTW	0.748	0.744

Table 6.5: Average Levenshtein distances for the MVP with the sum and the minimum as selection criterion (f_s = 16kHz, DTW: normalization to T_x, segmental DTW: T_y).

For the standard DTW the sum-of-distances variant with the sum is slightly better, for the segmental DTW the minimum-of-distances variant results in a slightly lower Levenshtein distance. However the differences are really small and no real preference for one of the two algorithm variants can be derived. This can be explained by the fact that the cases where two phones have the same number of occurrence are rare, so the implementations do not lead to no significantly different results. As the influence on the final result is apparently not substantial the initial sum-of-distances variant was kept.

Optimization of the candidate selection

The candidate selection is implemented with a fallback strategy for the case that not enough triphone candidates are found in the look-up table. The desired minimum number of candidates can be passed to the algorithm. The default value for the initial tests was 3. During the optimization a test was performed for a minimum number of 10 candidates. Again, both algorithms were tested with the files of 16 kHz sampling frequency and their optimal values for the normalization factors. The Levenshtein distances for the tests with a minimum number of 3 and 10 candidates are listed in table 6.6.

The results show that a higher number of required triphone candidates does not lead to better transcriptions. The quality of the transcriptions even decreases. The corresponding phone accuracy rates are 89.08% with 3 candidates and 87.77% with 10 candidates for the standard DTW, and 90.10% and 89.76% for the segmental

Algorithm	Min. num.	
	3	10
DTW	0.825	0.9245
Segmental DTW	0.748	0.774

Table 6.6: Average Levenshtein distances with different minimum numbers of required triphone candidates (MVP, $f_s = 16$kHz, DTW: normalization to T_x, segmental DTW: T_y).

DTW. A possible explanation is that the candidates that are chosen by the fallback strategy are not well matching. However to fully explain this result, the effects of the fallback strategy would have to be investigated in detail. Because of these results the default for the minimum number of required candidates is set to 3.

Optimization of the slope weighting factors

Both DTW algorithms were further tested with different settings for the slope weighting factors. The tested settings are the three slope weighting types (b), (c) and (d) presented in section 5.3.2. The slope weighting factors for the local path constraints of type I are illustrated in figure 6.2.

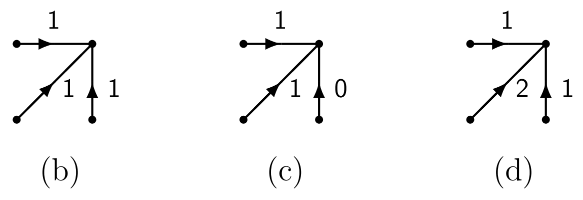


Figure 6.2: Slope weighting factors resulting from the slope weighting types (b), (c) and (d) for the local path constraints of type I.

Type (a) slope weighting is not used as it weights only the diagonal path with a nonzero factor. As default setting for the initial tests the slope weighting of type (d) was used.

Algorithm	Slope weighting type		
	(b)	(c)	(d)
DTW	0.973	0.976	0.825
Segmental DTW	0.769	0.756	0.748

Table 6.7: Average Levenshtein distances with the different slope weighting types (b), (c) and (d) (MVP, $f_s = 16$kHz, DTW: normalization to T_x, segmental DTW: T_y).

Type (d) slope weighting yields the best results for both algorithms, however for the segmental DTW the differences are small. For the final tests the slope weighting of type (d) was kept.

Optimization of the bandwidth for the segmental DTW

For the segmental DTW the width W of the bands from the distance matrix is a further parameter. The default value for the initial tests was 10 frames, what corresponds to a length of 100 ms as the features were extracted every 10 ms. The reference value proposed by Park and Glass is 75 ms [29]. Additionally to this basic setting a bandwidth of 20 samples was tested, the results are shown in table 6.8.

Algorithm	W	
	10	20
Segmental DTW	0.748	0.789

Table 6.8: Average Levenshtein distances with different values for the bandwidth W used in the segmental DTW (MVP, $f_s = 16$kHz, normalization to T_y).

Broadening the bandwidth to 20 frames does not improve the results, therefore the initial setting of 10 frames was kept.

6.2.4 Test results

After the optimization, the best settings were taken to perform a simulation with the test set. Again, the standard and the segmental DTW algorithm were tested. The results from these final tests were evaluated in detail. The analyzed measures comprise the Levenshtein distance, the rate of correct phones, the phone accuracy, and an additional statistic of the number of substitutions, deletions and insertions. To verify the performance of the candidate selection the values for precision and recall were calculated. For the intermediate transcriptions the same measures as for the automatic transcriptions were computed to illustrate the improvement from intermediate to automatic transcription.

Utterances	DTW	Seg. DTW
Correct	608	654
Total number	1273	1273
Percentage correct	47.76%	51.37%
Average LD	0.857	0.752

Phones	DTW	Seg. DTW
Correct	8559	8691
Total number	9453	9453
Percentage correct	90.54%	91.94%
Phone accuracy	88.46%	89.88%

Table 6.9: Results for the test set with the best settings from the optimization (MVP, $f_s = 16$kHz, DTW: norm. to T_x, seg. DTW: T_y).

Table 6.9 summarizes the results from the simulations with the test set. It can be seen that the segmental DTW yields better values than the standard DTW for all measures, however the differences are not great.

	DTW				Segmental DTW			
	S	D	I	Total	S	D	I	Total
Number	791	103	197	1091	654	108	195	957
In %	8.37	1.09	2.08	11.54	6.92	1.14	2.06	10.12

Table 6.10: Substitutions (S), deletions (D) and insertions (I) computed from the simulation results with the test set

In table 6.10 the substitutions, deletions and insertions are listed. Substitution errors occur more often than deletions or insertions, this is the case for the standard and for the segmental DTW. The rates of deletions and insertions are almost equal for both algorithms, while the rate of substitutions is lower for the segmental DTW. Thus the better performance of the segmental DTW is obviously based on a lower rate of substitutions.

A possible explanation why the number of deletions and insertions is rather low compared to the number of substitutions, is that the number of phones for the transcription is determined by the intermediate transcription. The intermediate transcription leads to a certain number of positions that are filled with the best phones retrieved by the pattern comparison algorithm. The length of the transcription is only varied if at the beginning or at the end two consecutive silence symbols ('sil') occur. In this case, one of the symbols is deleted and the length of the transcription is changed. In all other cases the length of the transcription is fixed. Thus it is very important that the candidate selection creates intermediate transcriptions that have the correct length. As the number of deletions and insertions is low the candidate selection obviously fulfills this condition in most cases.

Compared to the results from other applications as those presented in chapter 2, the results achieved with this transcription method are good. Commonly an agreement rate of about 80% between automatic and reference transcriptions is considered as good, as this is also the agreement rate achieved between human transcribers (interlistener agreement). Naturally, the experiments described in literature have different purposes and use different data, for this reason it is not meaningful to compare the mere figures, but they provide an orientation.

Some reported values are: 72% to 80% agreement between manual and machine transcriptions [34, 35, 2], a percentage disagreement of 12%-27% [7, 14, 13] or 80% to 88% identical labels [3, 36]. The results for the DTW lie within this range with a phone accuracy of 88.46% and a percentage disagreement of 11.54%. The values for the segmental DTW are even better with a phone accuracy of 89.88% and a percentage disagreement of 10.12%.

For the comparison to other experimental results one has to take into consideration that the transcriptor is currently restricted to a single speaker only and that the used recordings are of high quality (studio recordings), what surly is a reason for the excellent results. Still, the degree of detail of the transcriptions (the number of phones) is higher than the number used in most speech recognition or automatic transcription tasks. It was not clear from the beginning if it was even possible to create such detailed transcriptions automatically, thus the results are very satisfying. The transcriptor creates detailed transcription that are to a high degree concordant to the transcriptions from the human transcribers.

Evaluation of the candidate selection

For the analysis of the candidate selection the same measures as for the automatic transcriptions were calculated for the intermediate transcriptions. Both results were compared to investigate the improvement from intermediate to automatic transcription, so whether the achieved improvement justifies the use of DTW. If the improvement from intermediate to automatic transcription were low this would mean that the pattern comparison does not contribute much to establish a correct transcription and hence does not perform as required.

Utterances	Transcriptions	
	intermediate	final
Correct	364	654
Total number	1273	1273
Percentage correct	28.59%	51.37%
Average LD	1.394	0.752

Phones	Transcriptions	
	intermediate	final
Correct	7880	8691
Total number	9453	9453
Percentage correct	83.36%	91.94%
Phone accuracy	81.22%	89.88%

Table 6.11: Comparison of the results of the intermediate transcriptions (after the candidate selection) and the results of the automatic transcriptions (after application of the DTW) for the test set (MVP f_s = 16kHz, DTW: normalization to T_x, segmental DTW: T_y).

The results achieved with the intermediate transcription only are shown in table 6.11, opposed to the results after the application of the segmental DTW. The rate of correctly transcribed utterances after the intermediate transcription is 28.59%, thus about one third of the transcriptions can be predicted by the application of rules only. The rate of correct phones is 83.36%, meaning that the rules apparently work well for the given data set. It has to be noted that the rules are optimized for one (the Austrian male) speaker. In the case of a generalization of the transciptor for more speakers it may be necessary to modify the rules. It is possible that some rules model specific properties of the pronunciation of the Austrian male speaker.

These rules would have to be discarded or modified to enable correct results for other speakers. Of course, if the rules are generalized the results may become worse as the possible degree of detail decreases.

	DTW				Segmental DTW			
	S	D	I	Total	S	D	I	Total
Number	1459	114	202	1775	654	108	195	957
In %	15.43	1.21	2.14	18.78	6.92	1.14	2.06	10.12

Table 6.12: Substitutions (S), deletions (D) and insertions (I) computed from the simulation results with the test set (MVP f_s = 16kHz, DTW: normalization to T_x, segmental DTW: T_y).

The difference of the rate of correctly transcribed utterances before and after the DTW is about 20% absolute. The average Levenshtein distance from the automatic transcriptions is about half of the average distance from the intermediate transcriptions. The statistics on the number of substitutions, deletions and insertions in table 6.12 demonstrate that DTW causes a reduction from 1459 to 957 substitution errors. The number of deletion and insertion errors remains almost the same. This can be explained by the fact that the number of phones is fixed quite strictly as already discussed in the preceeding section. The results make clear that DTW is able to refine and improve the basic intermediate transcriptions and thus it is meaningful to include the DTW algorithm in the transcription process.

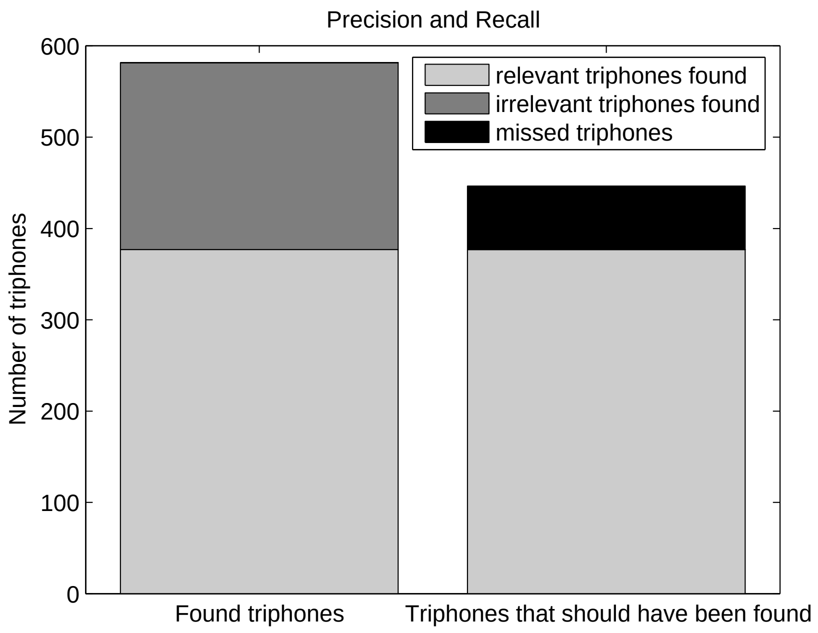


Figure 6.3: Average number of found triphones (left) and relevant triphones (right) per utterance for the test set.

Furthermore the values for precision and recall were calculated. Figure 6.3 illustrates the average number of triphones that were found and that should have been found per utterance. The left bar represents the number of triphones that were found, the lower segment illustrates the triphones that are relevant and the other segment shows those which are not relevant. The right bar represents the number of triphones that should have been found. The lower segment shows the triphones that indeed were found, the other one those that were not found. The corresponding values for precision and recall are listed in table 6.13.

Average number of triphones	
relevant & found	376.88
total found	581.68
total relevant	446.37

Average precision and recall	
Precision	0.641
Recall	0.843

Table 6.13: Statistic of found, missed and relevant triphones for the test set, and the resulting values for precision and recall.

The average score of the recall is 0.84, this means that on average a high number of relevant triphones was found. Thus the implementation of the candidate selection is obviously satisfying. The value for the precision with 0.64 is lower than the value for the recall, it shows that the search is not exact, it yields also many triphones that are not relevant. This is however not so much of a problem, it leads only to a higher number of candidates that have to be processed.

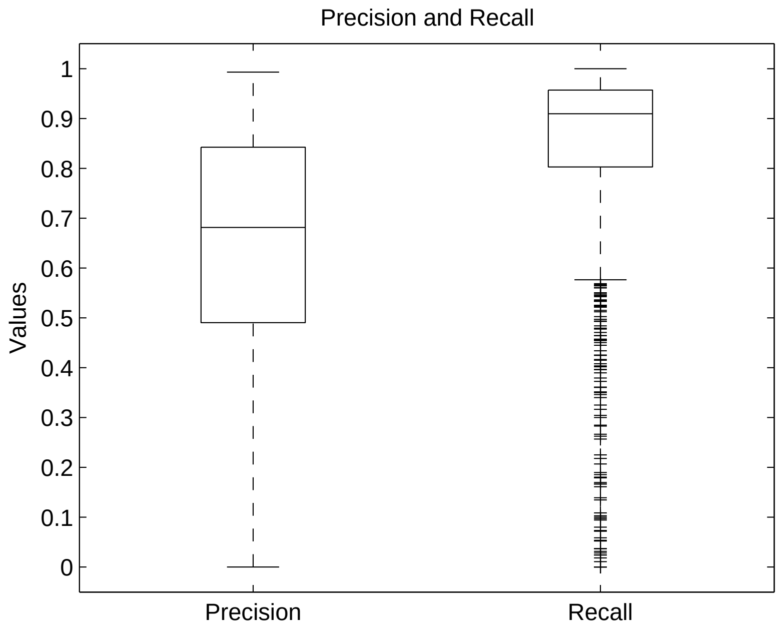


Figure 6.4: Boxplots for precision (left) and recall (right).

A boxplot for the different values of precision and recall for each utterance is shown in figure 6.4. For the precision 50% of the values lie in the range from approximately 0.5 to 0.85, hence the number of found and relevant triphones varies between 50% and 85% for half of the data. For the recall 50% of the values lie in the range from about 0.8 to 0.95 thus for most utterances the candidate selection is quite successful (0.80% to 0.95% of the candidate triphones are found for these utterances). However there are many outliers (the values from about 0.6 to 0), these are utterances where the candidate selection fails, probably because rules for the creation of the intermediate transcription do not yield a meaningful result. This can happen in the case of foreign words, as the rules for the creation of the intermediate transcription obviously are not able to predict a proper transcription for a word that does not satisfy the pronunciation rules of German.

Chapter 7

Outlook

7.1 Outlook

A new application for automatic phonetic transcription, the transcriptor, was developed. For future versions several improvements or extensions are thinkable, they will be discussed in this section.

7.1.1 Modification of the candidate selection

The current candidate selection applies a set of rules for the generation of an intermediate transcription. This works well as the test results for the intermediate transcriptions show (approximately 80% phone accuracy). However in case of an extension for more speakers the rules probably have to be modified as they are currently optimized for a single speaker.

The main reason for implementing a candidate selection is the reduction of the search space, i.e. the number of triphones among which matching segments are searched. Currently it is possible to transcribe about 100 words per hour[1], however it would be nice to speed up the process. The number of triphones that are passed to the pattern comparison unit ranges from 0 to 465 per segment of the intermediate transcription. A reduction seems reasonable if the number of triphones is high, as probably many triphones contain very similar information and thus it is not necessary to perform a comparison with each triphone. Probably a kind of clustering method could be applied to the internal database to eliminate entries that are very similar. In doing so the number of triphones is reduced and the search space is limited. This may especially be meaningful if the database and thus the search space is increased to allow speaker-independent transcription.

7.1.2 Modifications of the pattern comparison

In the current implementation of the pattern comparison the whole audio file is compared to a triphone. For the standard DTW this means that the word is not

[1]Implementation with segmental DTW, run on a standard PC.

segmented. This is not optimal, but it works. The segmental DTW performs a kind of segmentation by cutting the distance matrix into bands, however all the bands are compared with the triphones from the database.

One possibility to realize a segmentation is to segment the audio file exactly along the phone boundaries. But as they are not known this is complex.

The choice of candidate triphones is based on the segments from the intermediate transcription, so the part of the distance matrix to which the triphone should be compared is known approximately. This information could be used to segment the distance matrix roughly and to compare only the relevant parts with the triphone. As different phones have different durations, the length of the segments cannot be estimated exactly. For this reason the segments could for example be overlapping. Figure 7.1 shows the word "Aquarell" compared to one example of the triphone "a k v", a possible estimation for the relevant part of the distance matrix is marked. This estimation is derived from the average duration of the triphones plus an overlapping of 20% at the beginning and the end.

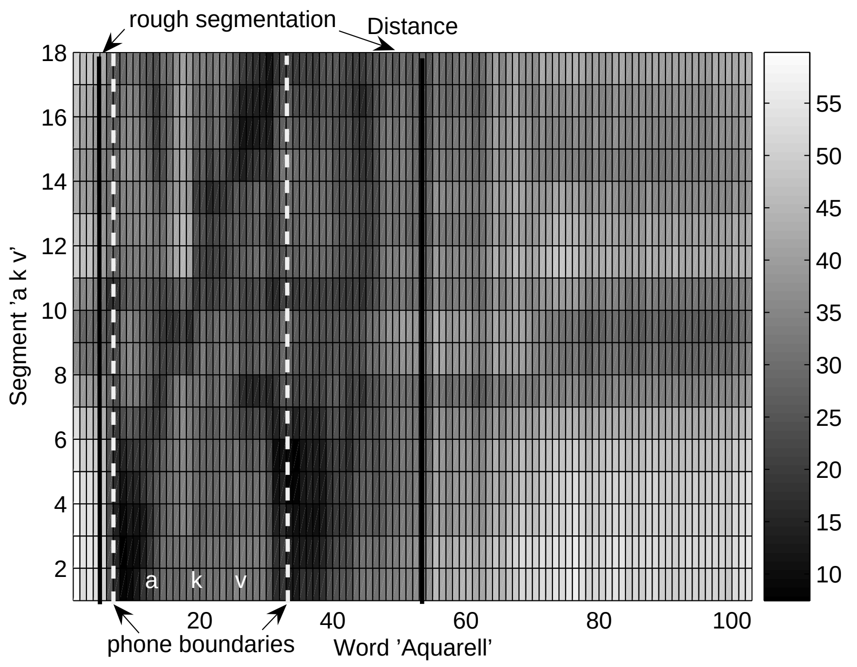


Figure 7.1: Distance matrix between the word "Aquarell" and the triphone "a k v"; the real phone boundaries of the segment "a k v" in the word are marked, an example for a rough segmentation of the relevant part of the distance matrix is given.

The expected benefits of this improvement are a higher correctness (the a priori knowledge which segment of the new word should be compared to which triphone from the database is used) and a reduced duration of the transcription process (less computations with smaller distance matrices).

Different similarity functions and settings for the DTW

In all simulations with the transcriptor the Euclidean distance was used for the DTW algorithm. Other distances could be used for example the Mahalanobis distance (see [22]). Another possible modification of the DTW algorithm is to implement other local path constraints as presented in section 5.3.2. Further refinements of the DTW algorithm include the application of flexible endpoint constraints and the application of global path constraints.

7.1.3 Extension of the database for more speakers

The implementation of the transcriptor presented here is optimized for a single speaker, the Austrian male speaker of ADABA. For further use an implementation that is applicable on the speech data of more speakers is desirable. Basically the extension for more speakers can be realized easily, however some points have to be kept in mind.

The extension can be performed by simply adding other speech files to the internal database. The only condition is that the new files must not have the same name as the existing files. Furthermore it might be necessary to modify the candidate selection, as in the current state of realization it strongly depends on the phonetics of the Austrian male speaker. For different speakers the results for the candidate selection might not be as good as with the current speaker. A possible solution is to generalize the rules to cover a wider field of pronunciation possibilities.

An informal test was performed with the data of the German male speaker of ADABA. The achieved Levenshtein distance is approximatly 2. The test was conducted before the optimization, with the candidate selection for the Austrian male speaker and without data from the German speaker in the database. The results are not as good as the results for the Austrian speaker, however they demonstrate that the procedure basically works even if no data of the new speaker is added to the database.

Chapter 8

Conclusion

The goal of this diploma thesis was to develop a tool for detailed automatic phonetic transcription.

Many existing systems for automatic phonetic transcription are based on techniques used in the field of automatic speech recognition. One common approach is to perform a forced alignment. For this method Hidden Markov Models (HMMs) are trained to model the phones. The systems that use this technique are thus model-based. In the forced alignment step Viterbi decoding is applied that chooses the best fitting transcriptions from the lexicon with transcription variants. A different approach for speech recognition is template-based speech recognition. Here no models are trained but speech utterances are compared directly, i.e., the audio data is not matched against models but it is compared to reference utterances from a database. The recognized utterances are determined by the most similar references.

The new approach for automatic transcription that was implemented within this work uses the technique applied in template-based speech recognition. For the transcription of a new word the unknown audio sample is compared to segments in a database. The transcriptions of the most similar segments are concatenated to the new transcription.

The transcription tool, named transcriptor, was implemented in Matlab. The audio files and the transcriptions for the database were taken from ADABA (Austrian Pronunciation Database). The first implementation is speaker-dependent.

To build the database for pattern comparison, the reference utterances from ADABA were cut into segments. It was decided to use segments with a fixed number of phones for simplicity. Triphones were selected as segmental units for their good compromise between shortness and the contextual information that they provide. To split the utterances, the boundaries between the phones had to be determined. This was done with the Hidden Markov Toolkit (HTK). With HTK HMMs were trained, and a forced alignment was performed. A feature of the forced alignment in HTK is the output of the phone boundaries. These phone boundaries are taken to cut the utterances into triphones. For pattern comparison usually features are extracted from the audio files. The chosen features were Mel Frequency Cepstral Coefficients (MFCCs) that are widely used in speech recognition. The feature extraction and the splitting into triphones were performed in one step using HTK.

To speed up the transcription process a candidate selection was implemented. The candidate selection passes only a choice of possibly well matching triphones to the pattern comparison unit, reduces the number of comparisons and thus the duration of the transcription process. Without a candidate selection a new word would have to be compared to each segment in the database which would lead to a long duration of the process. The triphone candidates are chosen with the aid of an intermediate transcription that is generated by the application of rules on the orthographic presentation of the new word. This intermediate transcription is also segmented, then for each segment the candidate triphones for the pattern comparison are searched in a look-up table.

The pattern comparison between the unknown audio sample and the segments in the database is realized with a dynamic time warping algorithm (DTW). This algorithm computes the similarity, also called distance, between two utterances. The pattern comparison unit finds the three best matching triphones for each segment of the new word.

After the pattern comparison the best triphones are synthesized to the new transcription. This is done by evaluating the number of occurrences of each phone. If this yields no clear result (e.g. the number of occurrence of two phones is equal), additionally the distances computed by the DTW algorithm are used as criterion.

The system was optimized with two versions of the DTW algorithm - standard DTW and segmental DTW - several settings for the candidate selection, and with files of 16 kHz, 32 kHz and 44.1 kHz sampling frequency. With the best settings from the optimization a final simulation with an independent test set was performed.

The resulting transcriptions were evaluated with the aid of the reference transcriptions from ADABA. To enable a comparison with other systems the phone accuracy and the percentage disagreement were computed. The rate of correct phones for the standard DTW is 90.54%, the phone accuracy 88.46% and the percentage disagreement 11.54%. For the segmental DTW the rates are 91.94%, 88.88% and 10.12%. Segmental DTW thus performs slightly better than standard DTW.

For other existing systems reported values are 72%-80% agreement between automatic and reference transcriptions, 12%-28% percentage disagreement and 80%-89% of identical phones. The results achieved by the transcriptor are comparable to those from other systems, the results from the segmental DTW even outperform most current systems.

For the interpretation of the results, it has to be taken into consideration that the used audio files are of high quality (studio recordings), this allows achieving good results more easily. On the other hand, the transcriptions are more detailed than the transcriptions usually treated in comparable systems. Normally for German a phoneme set of about 45 phonemes is considered, the phoneme set of the transcriptor contains 89 phonemes. From this point of view the performance of the transcriptor is remarkable, as it was not clear if such a detailed distinction is even possible. The results demonstrate that the DTW algorithm can be applied for transcription tasks. All in all it has been shown that it is possible to create detailed transcriptions by an automatic procedure that are to a high degree concordant with the transcriptions established by humans.

Appendix A

Results

A.1 Initial tests

Settings:

Used data set:	Development set
Slope weighting factors:	[1 2 1]
Minimum number of candidates:	3
Default synthesis variant:	Majority vote procedure with sum criterion

A.1.1 DTW

f_s	MVP			MDP		
	T_x	T_y	$T_x + T_y$	T_x	T_y	$T_x + T_y$
16kHz	0.825	1.256	1.028	1.181	2.445	1.675
32kHz	0.868	1.263	1.028	1.239	2.362	1.701
44kHz	0.887	1.274	1.020	1.201	2.421	1.723

Table A.1: DTW: Average Levenshtein distances

f_s	MVP			MDP		
	T_x	T_y	$T_x + T_y$	T_x	T_y	$T_x + T_y$
16kHz	49.21	33.18	40.72	35.53	15.88	25.94
32kHz	47.33	31.92	40.72	33.02	17.30	26.10
44kHz	45.28	32.08	41.04	34.43	16.35	25.16

Table A.2: DTW: Rate of correct utterances in %

f_s	MVP			MDP		
	T_x	T_y	$T_x + T_y$	T_x	T_y	$T_x + T_y$
16kHz	91.26	85.46	88.52	86.60	69.36	79.84
32kHz	90.72	85.44	88.54	85.83	70.48	79.49
44kHz	90.43	85.27	88.64	86.33	69.67	79.16

Table A.3: DTW: Rate of correct phones in %

f_s	MVP			MDP		
	T_x	T_y	$T_x + T_y$	T_x	T_y	$T_x + T_y$
16kHz	89.08	83.38	86.39	84.38	67.65	77.84
32kHz	88.52	83.30	86.39	83.61	68.75	77.49
44kHz	88.27	83.15	86.50	84.11	67.96	77.20

Table A.4: DTW: Phone accuracy in %

A.1.2 Segmental DTW

f_s	MVP			MDP		
	T_x	T_y	$T_x + T_y$	T_x	T_y	$T_x + T_y$
16kHz	0.932	0.748	0.873	1.436	0.923	1.325
32kHz	0.906	0.778	0.854	1.442	0.983	1.379
44kHz	0.909	0.778	0.858	1.492	1.003	1.412

Table A.5: Segmental DTW: Average Levenshtein distances

f_s	MVP			MDP		
	T_x	T_y	$T_x + T_y$	T_x	T_y	$T_x + T_y$
16kHz	41.98	49.37	45.28	28.30	40.57	30.19
32kHz	44.81	49.21	46.86	28.14	38.99	30.66
44kHz	43.08	48.90	45.44	25.31	38.52	28.46

Table A.6: Segmental DTW: Rate of correct utterances in %

f_s	MVP			MDP		
	T_x	T_y	$T_x + T_y$	T_x	T_y	$T_x + T_y$
16kHz	90.29	92.26	90.89	83.77	89.91	85.04
32kHz	90.51	91.87	90.99	83.54	89.10	84.15
44kHz	90.39	91.89	90.97	82.78	88.87	83.71

Table A.7: Segmental DTW: Rate of correct phones in %

f_s	MVP			MDP		
	T_x	T_y	$T_x + T_y$	T_x	T_y	$T_x + T_y$
16kHz	87.66	90.10	88.45	81.01	87.79	82.46
32kHz	88.02	89.70	88.70	80.92	87.00	81.76
44kHz	87.98	89.70	88.64	80.26	86.73	81.32

Table A.8: Segmental DTW: Phone accuracy in %

A.2 Optimization

Settings:

Used data set:	Development set Files with 16 kHz sampling frequency
Normalization factor for DTW:	T_x
Normalization factor for seg. DTW:	T_y
Synthesis variant:	Majority vote procedure with sum criterion
Default min. number of candidates:	3
Default slope weighting factors:	[1 2 1]
Default bandwith for seg. DTW:	10

A.2.1 Synthesis variant

Algorithm	DTW		Seg. DTW	
MVP with	sum	min.	sum	min.
Average levenshtein distance	0.825	0.829	0.748	0.744
Rate of correct transcriptions (in %)	49.21	48.74	49.37	49.84
Rate of correct phones (in %)	91.26	91.18	92.26	92.32
Phone accuracy (in %)	89.08	89.04	90.10	90.16

Table A.9: Results from the MVP with sum-of-distances and minimum-of-distances criterion

A.2.2 Minimum number of candidates

Algorithm	DTW		Seg. DTW	
Minimum number of candidates	3	10	3	10
Average levenshtein distance	0.825	0.925	0.748	0.774
Rate of correct transcriptions (in %)	49.21	46.23	49.37	48.11
Rate of correct phones (in %)	91.26	89.95	92.26	91.95
Phone accuracy (in %)	89.08	87.77	90.10	89.76

Table A.10: Results with different minimum numbers of required candidates

A.2.3 Slope weighting

Algorithm	DTW			Segmental DTW		
Slope weighting type	(d)	(b)	(c)	(d)	(b)	(c)
Average levenshtein distance	0.825	0.973	0.976	0.748	0.769	0.756
Rate of correct transcriptions (in %)	49.21	42.14	39.94	49.37	47.48	49.69
Rate of correct phones (in %)	91.26	89.27	89.79	92.26	91.99	92.20
Phone accuracy (in %)	89.08	87.12	87.08	90.10	89.83	89.99

Table A.11: Results with different slope weighting factors

A.2.4 Bandwidth for segmental DTW

Algorithm	Seg. DTW	
Band width W	10	20
Average levenshtein distance	0.748	0.789
Rate of correct transcriptions (in %)	49.37	48.90
Rate of correct phones (in %)	92.26	91.78
Phone accuracy (in %)	90.10	89.56

Table A.12: Results with different bandwidths for segmental DTW

A.3 Final tests

Settings:

Used data set:	Test set
	Files with 16 kHz sampling frequency
Normalization factor for DTW:	T_x
Normalization factor for seg. DTW:	T_y
Synthesis variant:	Majority vote procedure with sum criterion
Minimum number of candidates:	3
Slope weighting factors:	[1 2 1]
Bandwith for segmental DTW:	10

A.4 DTW

DTW			
Utterances		Phones	
Correct	608	Correct	8559
Total number	1273	Total number	9453
Percentage correct	47.76%	Percentage correct	90.54%
Average LD	0.857	Phone accuracy	88.46%

Table A.13: DTW: Results with the test set

A.5 Segmental DTW

Segmental DTW			
Utterances		Phones	
Correct	654	Correct	8691
Total number	1273	Total number	9453
Percentage correct	51.37%	Percentage correct	91.94%
Average LD	0.752	Phone accuracy	89.88%

Table A.14: Segmental DTW: Results with the test set

Appendix B

Phoneme sets

2	b_0	g	l=	o_(	t_>
2_o	C	h	m	O	t_h
3	d	I	m=	o	t_v
6	dZ	i	N	p	U
6_R\	d_0	i_j	n	p_h	u
9	E	i_(	n_(	R	u_o
?	e	i_R	N=	r	u_(
@	e_O	j	n=	R\	u_R
@\	e_o	k	O	r\	v
a	e_o_R\	k_>	o	S	w
a_(	e_(	k_h	O_o	s	x
a_R\	E	k_v	o_o	s_j	y
A	e	l	o_o_(	s_+	Z
a	F	l_e	o_o_R\	s_v	z
b	F=	l_e=	O_(	t	

Table B.1: Phoneme set of the transcriptor (89 phonetic symbols) in SAMPA Austria .

2	a_(	g	l=	n=	s
3	b	h	m	o	t
4	C	i	m=	o_(	u
@	d	j	N	p	v
@\	e	k	n	R	x
a	f	l	N=	S	y

Table B.2: Reduced phoneme set for the intermediate transcription (36 phonetic symbols).

Bibliography

[1] J. M. Kessens and H. Strik, "Lower WERs do not guarantee better transcriptions," *Proceedings of Eurospeech*, pp. 1721–1724, 2001.

[2] C. Cucchiarini and H. Strik, "Automatic phonetic transcription: An overview," *Proceedings of ICPhS*, pp. 347–350, 2003.

[3] S. Chang, L. Shastri, and S. Greenberg, "Automatic phonetic transcription of spontaneous speech (American English)," *Proceedings of ICSLP*, pp. 330–333, 2000.

[4] K. Wothke, "Morphologically based automatic phonetic transcription," *IBM Systems Journal*, vol. 32, no. 3, pp. 486–511, 1993.

[5] M. Wester, J. M. Kessens, C. Cucchiarini, and H. Strik, "Obtaining phonetic transcriptions: A comparison between expert listeners and a continuous speech recognizer," *Language and Speech*, vol. 44, pp. 377–403, 2001.

[6] F. Schiel, "Automatic phonetic transcription of non-prompted speech," *Proceedings of ICPhS 1999*, pp. 607–610, 1999.

[7] D. Binnenporte, C. Cucchiarini, H. Strik, and L. Boves, "Improving automatic phonetic transcription of spontaneous speech through variant-based pronunciation variation modelling," *Proceedings of LREC*, pp. 681–684, 2004.

[8] S. Rapp, "Automatic phonemic transcription and linguistic annotation from known text with hidden markov models: An aligner for german," 1995. [Online]. Available: http://www.ims.uni-stuttgart.de/~rapp/

[9] K. Sjölander, "An HMM-based system for automatic segmentation and alignment of speech," *Proceedings of Fonetik*, pp. 93–96, 2003.

[10] P.-A. Jande, "Automatic detailed transcription of speech using forced alignment and naive pronunciation rules," KTH Stockholm, Tech. Rep., 2003.

[11] C. M. Bishop, *Pattern recognition and machine learning.* Springer, 2006.

[12] L. Rabiner and B.-H. Juang, *Fundamentals of Speech Recognition.* Prentice Hall PTR, 1993.

[13] C. Cucchiarini and D. Binnenporte, "Validation and improvement of automatic phonetic transcriptions," *Proceedings of ICSLP*, pp. 313–316, 2002.

[14] C. Van Bael, W. Strik, and H. van den Heuvel, "Application-oriented validation of phonetic transcriptions: Preliminary results," *Proceedings of ICPhS*, pp. 1161–1164, 2003.

[15] H. Strik, A. Russel, H. van den Heuvel, C. Cucchiarini, and L. Boves, "A spoken dialogue system for public transport information," *International Journal of Speech Technology*, vol. 2, no. 2, pp. 119–129, 1997.

[16] S. Lemmetty, "Review of speech synthesis technology," Master's thesis, Helsinki University of Technology, 1999.

[17] A. Zerfaß, "Translation Memory - Eine Einführung," transline tecNews, 2005. [Online]. Available: http://www.transline.de/transline-tecNews/Translation-Memory-eine-Einfuehrung

[18] A. Heuberger, "Machine Translation vs. Translation Memory." [Online]. Available: http://www.multilingualwebmaster.com/library/mt-vs-tm.html

[19] S. Young, G. Evermann, M. Gales, T. Hain, D. Kershaw, X. A. Liu, G. Moore, J. Odell, D. Ollason, D. Povey, V. Valtchev, and P. Woodland, *The HTK Book*. Cambridge University Engineering Department, 2006.

[20] M. De Wachter, K. Demuynck, D. Van Compernolle, and P. Wambacq, "Data driven example based continuous speech recognition," *Proceedings of Eurospeech*, pp. 1133–1136, 2003.

[21] L. Deng and H. Strik, "Structure-based and template-based automatic speech recognition – comparing parametric and non pararmetric approaches," *Proceedings of Interspeech 2007*, pp. 898–901, 2007.

[22] M. De Wachter, M. Matton, K. Demuynck, P. Wambacq, R. Cools, and D. Van Compernolle, "Template-based continuous speech recognition," *IEEE Transactions on Audio, Speech, and Language Processing*, pp. 1377–1390, 2007.

[23] R. Muhr, "The Pronouncing Dictionary of Austrian German (AGPD) and the Austrian Phonetic Database (ADABA) – Report on a large phonetic resources database of the three major varieties of German," *Proceedings of LREC*, 2008.

[24] ADABA Homepage. [Online]. Available: http://www.adaba.at

[25] R. Muhr, *Österreichisches Aussprachewörterbuch*. Peter Lang – Internationaler Verlag der Wissenschaften, 2007.

[26] Homepage of the International Phonetic Association. [Online]. Available: http://www2.arts.gla.ac.uk/IPA/ipa.html

[27] C. Leitner, “adaba: Korrekturen und Modifikationen,” Graz University of Technology, Tech. Rep., 2007.

[28] SAMPA German reference. [Online]. Available: http://www.phon.ucl.ac.uk/home/sampa/german.htm

[29] A. Park and J. R. Glass, “Towards unsupervised pattern discovery in speech,” *IEEE Workshop on Automatic Speech Recognition and Understanding, 2005*, pp. 53–58, 2005.

[30] D. Gusfield, *Algorithms on Strings, Trees, and Sequences: Computer Science and Computational Biology.* Cambridge University Press, 1997.

[31] M. Wester, J. M. Kessens, and H. Strik, “Two automatic approaches for analyzing connected speech processes in Dutch,” *Proceedings of ICSLP*, pp. 3351–3356, 1998.

[32] R. Baeza-Yates and B. Ribeiro-Neto, *Modern Information Retrieval.* Addison-Wesley, 1999.

[33] P. Boersma and D. Weenink, “Praat: Doing phonetics by computer (version 5.0.32),” 2008. [Online]. Available: http://www.praat.org/

[34] A. Kipp, M.-B. Wesenick, and F. Schiel, “Pronunciation modeling applied to automatic segmentation of spontaneous speech,” *Proceedings of Eurospeech*, pp. 106–109, 1997.

[35] F. Schiel, A. Kipp, and H. G. Tillmann, “Statistical modeling of pronunciation: It’s not the model, it’s the data,” *Proceedings of the ESCA workshop ’Modeling Pronunciation Variation for Automatic Speech Recognition’*, pp. 131–136, 1998.

[36] M.-B. Wesenick and A. Kipp, “Estimating the quality of phonetic transcriptions and segmentations of speech signals,” *Proceedings of ICSLP*, pp. 129–132, 1996.