TAMPEREEN TEKNILLINEN YLIOPISTO
TAMPERE UNIVERSITY OF TECHNOLOGY
Julkaisu 729 • Publication 729

Juha Iso-Sipilä

Design and Implementation of a Speaker-Independent Voice Dialing System: A Multi-Lingual Approach

Tampere 2008

Tampereen teknillinen yliopisto. Julkaisu 729
Tampere University of Technology. Publication 729

Juha Iso-Sipilä

Design and Implementation of a Speaker-Independent Voice Dialing System: A Multi-Lingual Approach

Thesis for the degree of Doctor of Technology to be presented with due permission for public examination and criticism in Tietotalo Building, Auditorium TB109, at Tampere University of Technology, on the 18th of April 2008, at 12 noon.

Tampereen teknillinen yliopisto - Tampere University of Technology
Tampere 2008

ISBN 978-952-15-1946-8 (printed)
ISBN 978-952-15-2120-1 (PDF)
ISSN 1459-2045

Abstract

Speech as a means of communication is most natural to human beings. Therefore, it should be straightforward to apply the same modality for commonly used communication devices, such as mobile phones. The current state-of-the-art speech processing technology enables a wide range of applications using both speech input and output modalities. Research in the areas of speech recognition and speech synthesis has been carried out for several decades but real breakthroughs are yet to be seen. In addition to mastering the above-mentioned technologies, a computer should be able to understand the speech as humans do. Only then, natural spoken communication between a human and a machine can be accomplished in a similar manner as in human to human communications. Currently, in many cases, the users' expectations do not meet the capabilities of the technology. The evolution of voice interfaces for mobile phones is progressing despite the problems mentioned above. During the 1990s the first devices were equipped with so called speaker-dependent voice dialing or voice tagging. Most mobile device manufacturers included this feature to their devices before the year 2000. This can be considered to be the first speech application with hundreds of millions of installations and a multi-lingual user base.

In an application utilizing speaker-independent speech recognition, the user does not need to perform any training and the device is immediately available for use. This is an advantage to the user but, at the same time, presents a number of challenges to the design of the system. The biggest challenge is that the speaker-independent system is language dependent and the system requires additional development effort for every language. Speaker-independent systems are inherently more complex than speaker-dependent since the speech recognizer has to be trained with a large group of speakers for each language. In addition, a speech synthesizer is needed for spoken feedback.

This thesis considers a multi-lingual approach for a voice dialing application running on a mobile device. The proposed techniques and their applications form the core of a multi-lingual speaker-independent voice dialing system. The voice dialing system utilizes multi-lingual speech recognition, speech synthesis and language processing technologies to create a compact and complete system that can be easily extended and configured to existing and new languages. The system is split into subsystems which are designed and developed independently. The subsystems include research topics, such as multi-lingual phonetics, multi-lingual acoustic modeling, pronunciation modeling, speech recognition and speech synthesis. The subsystems communicate within a framework and share common resources. The basis for the overall system is the definition of a multi-lingual phoneme set that

represents the phonology of any supported language. The harmonized phonetic presentation is further utilized in pronunciation modeling, speech recognition and speech synthesis. Additionally, the speech recognition engine is developed in such a manner that makes it language-independent and utilizes a very compact representation of the acoustic models. The speech synthesis engine, based on a well known formant synthesizer, is both compact in size and easily runtime configurable with a description language. Finally, the pronunciation modeling engine is developed so that it supports several types of pronunciation modeling methods without dependencies to individual languages. The functionality is implemented so that a collection of runtime parameters define the operations and languages supported by any instance of the system. The contribution of this thesis includes the design and development of the subsystems and their integration to build a real application for mobile devices.

Preface

The research included in this thesis was conducted during 1996-2006 while working at Nokia Research Center and Nokia Technology Platforms. While most of the technology development was carried out at Nokia Research Center, the recent years at Nokia Technology Platforms concentrated on the final implementation onto Nokia device hardware and software. The latter part of the work has proven on several occasions more challenging than the actual scientific research towards multi-lingual speech recognition and synthesis.

I would like to thank my thesis supervisor, Professor Moncef Gabbouj, for his support during the work to complete this thesis. I would also like to express my gratitude to the reviewers of the thesis, Prof. Khaled Assaleh of American University of Sharjah and Dr. Jim Glass of Massachusetts Institute of Technology for their valuable comments and suggestions to finalize the thesis.

While working at Nokia Research Center, the biggest thanks should go to Dr. Kari Laurila, Ms. Wang Xia, Dr. Tian Jilei, Dr. Kiss Imre and Juha Häkkinen for their original ideas on multi-lingual speech recognition. At Nokia Technology Platforms I would like to say special thanks to Marcel Vasilache whose research on parameter compression was most helpful during the implementation phase. I also thank Dr. Marko Moberg and Dr. Kimmo Pärssinen for the additional motivation I got after their Ph.D. theses were finished last year. Finally, I owe the biggest thank so far in my career to Dr. Olli Viikki for his insight on developing the right speech technology and making the right decisions to get the results implemented in large scale. I also thank Olli for the valuable review he carried out on my thesis and for the coaching I have received from him.

It does not matter what you achieve unless there is someone to share the achievement. I would like to share this one with my wife Susanna and our daughter Aurora. While not exactly understanding what I was working on, they nevertheless fully supported my work. My love and sincere thanks to you both!

Tampere, March 2008

Juha Iso-Sipilä

Contents

List of Publications

This thesis is written on the basis of the following publications. In the text, these publications are referred as Publication 1 through 9 together with the reference in the Bibliography section.

Publication 1 **J. Iso-Sipilä**, K. Laurila, P. Haavisto, "Optimal Adaptive Garbage Modeling in Speech Recognition", in *Proceedings of the Nordic Signal processing Symposium (NORSIG '96)*, Helsinki, Finland, 1996

Publication 2 **J. Iso-Sipilä**, K. Laurila, R. Hariharan, O. Viikki, "Hands-Free Voice Activation in Noisy Car Environment", in *Proceedings of the Eurospeech*, Budapest, Hungary, 1999

Publication 3 **J. Iso-Sipilä**, "Speech Recognition Complexity Reduction Using Decimation of Cepstral Time Trajectories", in *Proceedings of the European Signal Processing Conference (EUSIPCO 2000)*, Tampere, Finland, 2000

Publication 4 Dong Yuan, Wang Xia, **J. Iso-Sipilä**, O. Viikki, I. Kiss, J. Tian, "Speaker- and Language-Independent Speech Recognition in Mobile Communication Systems", in *Proceedings of the National Conference on Man-machine Communication*, Shenzhen, China, 2001

Publication 5 Cao Wenjie, Xu Bo, **J. Iso-Sipilä**, "Linguistic and Acoustic Analysis of Chinese Person Names", in *Proceedings of International Symposium on Chinese Spoken Language Processing (ISCSLP)*, Taipei, Taiwan, August, 2002

Publication 6 Xia Wang, **J. Iso-Sipilä**, "Low Complexity Mandarin Speaker-Independent Isolated Word Recognition", in *Proceedings of the International Conference on Spoken Language Processing (ICSLP 2002)*, Denver, USA, 2002

Publication 7 M. Vasilache, **J. Iso-Sipilä**, O. Viikki, "On a Practical Design of a Low Complexity Speech Recognition Engine", in *Proceedings of the International Conference on Acoustics, Speech and Signal Processing (ICASSP 2004)*, Montreal, Canada, 2004

Publication 8 M. Moberg, K. Pärssinen, **J. Iso-Sipilä**, "Cross-Lingual Phoneme Mapping for Multilingual Synthesis System", in *Proceedings of the Interspeech*, Jeju Island, South Korea, 2004

Publication 9 **J. Iso-Sipilä**, M. Moberg, O. Viikki, "Multi-Lingual Speaker-Independent Voice User Interface for Mobile Devices", in *Proceedings of the International Conference on Acoustics, Speech and Signal Processing (ICASSP 2006)*, Toulouse, France, 2006

List of Acronyms

ACW	Adaptive Component Weighting
AGM	Adaptive Garbage Model
API	Application Programmer's Interface
ASR	Automatic Speech Recognition
CMN	Cepstral Mean Normalization
CPU	Central Processing Unit
DC	Direct Current
DCT	Discrete Cosine Transform
DP	Dynamic Programming
DSP	Digital Signal Processing, Digital Signal Processor
ELRA	European Language Resources Association
ERA	Expected Recognition Accuracy
FFT	Fast Fourier Transform
FIR	Finite Impulse Response
G2P	Grapheme-to-phoneme mapping, converting letters (graphemes) into pronunciation, i.e. phoneme sequence
HMM	Hidden Markov Model
HOC	Higher Order Cepstral features
HTK	Hidden Markov Model Toolkit
ID	Identifier, identity
IF	Intermediate Format
ISO	International Organization for Standardization
kB	Kilo Byte (equals to 1024 bytes)
LDC	Linguistic Data Consortium
LID	Language Identification from text
LPC	Linear Predictive Coding

LVCSR	Large Vocabulary Continuous Speech Recognition
MAP	Maximum A- Posteriori
MFCC	Mel-Frequency Cepstral Coefficients
ML	Maximum Likelihood
MLLR	Maximum-Likelihood Linear Regression
MLPA	Multi-Lingual Phonetic Alphabet
MLP	Multi Layer Perceptron
N-best	N best matches
NLP	Natural Language Processing
OOV	Out-of-Vocabulary (word)
PLP	Perceptual Linear Prediction
PMC	Parallel Model Compensation
PMR	Professional Mobile Radio
qHMM	Quantized parameter HMM
RAM	Random Access Memory
RSK	Right Selection Key
SA	Speaker Adaptation
SALT	Speech Application Language Tags
SDND	Speaker-Dependent Name Dialing
SI	Speaker-Independent
SIM	Subscriber Identity Module
SIND	Speaker-Independent Name Dialing
SNR	Signal to Noise Ratio
SW	Software
TTS	Text-to-speech
UI	User Interface
VAD	Voice Activity Detector
VUI	Voice User Interface

List of Tables

List of Figures

Chapter 1

Introduction

SPEECH Recognition has always been considered an ultimate success in usability: Who would not like to speak to their computer, car, and home appliances if the devices really understood what the user told them? Furthermore, in this ideal case, machines could enter in a conversation with the user and give similar answers and comments on the discussion as we expect from our human counterparts. A conversation with a machine is successful if the user does not mind that he/she is actually talking with a machine. Such a dream machine requires mastering several imperfect technologies such as speech recognition, speech synthesis and natural language understanding. We all know that the machine described above is only available in the distant future (if ever). The unique capabilities of the human brain to capture long history of previous conversations with other people, good memory and multi-modal interaction (not just mouth and ears) makes it very difficult to imitate the behavior of the human conversational model. Alternatively, a successful speech recognition application is such that it enables the user to accomplish the task easier, safer or with less effort than with a normal user interface.

As hard as it seems, there are thousands of researchers around the world who work on the difficult disciplines of speech recognition, speech synthesis and natural language processing. Small steps have been introduced over the years to improve the state-of-the-art methods. One particularly important enabler has been the huge progress in computer hardware, also those that are used in mobile devices. The evolution of computers has enabled the implementation of more and more complex algorithms and models for human speech perception, production and understanding. Wide availability of speech resources is also very helpful. The active contribution to various speech and text database collection activities is one of the cornerstones of multi-lingual speech processing. Without active research focused on a multi-lingual approach, the current work would not have been possible. The speech research carried out at Nokia has focused on low-complexity low-footprint speech recognition and speech synthesis systems. The efforts put on these technologies have enabled Nokia to use in-house technology for voice dialing application over the years, starting from speaker-

dependent voice dialing and currently continuing with speaker-independent voice dialing over the wide portfolio of mobile devices.

A mobile phone or device is a special kind of computer that has several limitations compared to, e.g. a personal computer used in everyday life both at the office and at home. The main limitations are the availability of computational power and memory. Today, when high speed Internet is available to most computers, mobile phones have significantly lower connection speed. The difference is in the order of a magnitude. All this together make a mobile device a much more difficult platform to develop any applications, let alone speech applications. In the thesis the term mobile phone or mobile device is used for the target platform of the voice dialing system.

1.1. MULTI-LINGUAL SPEAKER-INDEPENDENT VOICE DIALING

Voice user interface development on mobile devices has been evolving over the last decade. Originally, first voice user interface application in mobile devices was the speaker-dependent voice dialing. Until recently, speaker-dependent voice dialing has been the de-facto application on the devices of all major mobile phone manufacturers. Common requirement for such application is that the user needs to train the system in order to start using it. At the time of introduction, the benefits and drawbacks of speaker-dependent voice dialing were very apparent. At that time, there was no discussion about the possibility to utilize speaker-independent speech recognition technology in mobile devices. Some enhancement to the speaker-dependent technology was needed to overcome the main drawbacks such as user training and limited vocabulary size. The solution was to be found in multi-lingual speaker-independent speech recognition and synthesis technology. This thesis focuses on the design and implementation of a multi-lingual speaker-independent voice dialing system. Multi-linguality means the ability to cope with multi-lingual speech input (e.g. names) and support multiple languages at the same time. A speaker-independent system can be used by any user without additional training. Voice dialing refers to the application where a user may dial a contact in the device memory by saying the name of the contact. A speaker-dependent system does not require any speech synthesis technology due to the availability of the training data for feedback purposes. In a speaker-independent system, the training data is not available and due to dynamic vocabularies, such as names in voice dialing, a speech synthesizer or text-to-speech system is required to provide proper feedback to the user.

Design and implementation of a multi-lingual voice dialing system requires mastering several fields. The heart of the system is the multi-lingual speaker-independent speech recognizer and multi-lingual speech synthesis engine. These form the basis of the system. In addition to this basic technology, multi-lingual phonetics research, text processing and multi-lingual pronunciation modeling are needed to cope with all supported languages. When all these are put together, methods to combine the multi-lingual knowledge are still needed to ensure the building blocks work together seamlessly.

The development of the multi-lingual speaker-independent speech recognition system started around the year 2000 at Nokia Research Center. Early on, it was realized that to fulfill

the strict localization requirements of global business, it was necessary to develop the technology for all required languages. The mobile devices are sold in almost all countries of the world and they support 40-80 different user interface languages, depending on the market segment and platform. Nokia wants to serve the customers equally, independently of the language or region. Therefore, all applications should be localized to all supported user interface languages. There are very few exceptions to this and the voice user interface was seen to be important in this respect. The consequence of this requirement was that a lot of knowledge of languages, such as writing script and phonology, has been gathered. The development was divided into four main research areas that all needed improvement over the existing state-of-the-art technology. The areas were speech recognition, speech synthesis, pronunciation modeling and phonetics. Mastering these was not enough and several other sub-disciplines were raised, e.g. language identification from text, text pre-processing, voice UI data packaging, multi-lingual voice UI data configurations, etc. This thesis work concentrates on the main areas and the sub-areas which are dealing with text processing and voice UI data.

The voice dialing user interface is as important as the individual technical choices. In a mobile device, the starting point for voice dialing user interface design was the hands-free use. The target was to enable the user to dial any person in the device phonebook without touching the device. With the help of a wired or Bluetooth accessory, the user can accomplish this goal. Also, a tight integration with the device user interface was desired. This can be seen in the applications where speech recognition and speech synthesis are included.

The work is a joint development effort of a research team at Nokia. The author's contribution to various parts of the technology is clearly indicated at the beginning of each chapter or section.

1.2. OUTLINE OF THE THESIS

The thesis is organized as follows. Chapter 2 starts with a generic review of voice user interface components and the basics of phonetic systems used in speech research. The chapter then continues with an introduction to the fundamentals of speech recognition and speech synthesis. This summary gives the reader an understanding of the challenges and methods that are used in state-of-the-art speech recognition and speech synthesis systems. The emphasis is on low footprint speech recognition and synthesis, which are targeted at command and control type of applications, such as voice dialing. In speech synthesis the focus is on parametric speech synthesis that has the benefit of being more easily portable to several languages. Chapter 3 discusses issues related to processing the text to be suitable for speech recognition and speech synthesis technology. This includes text pre-processing, symbol conversions, text-based language identification and pronunciation modeling. The importance of Unicode standard and Romanization methods for robust handling of non-native vocabulary entries is emphasized in Chapter 3. Chapter 4 continues with the essential technical details of speech recognition and speech synthesis that define the complete multi-lingual voice dialing system. Final parts of Chapter 4 specify the voice UI data that is essential for an efficient implementation and configuration on a wide range of mobile devices. Chapter 5 concentrates

on the user interface of the voice dialing system. Finally, Chapter 6 summarizes the work with the conclusions.

1.3. AUTHOR'S CONTRIBUTION TO THE PUBLICATIONS

Publication 1 [50] improves the online garbage model introduced by Herve Bourlard et al. [11] to utilize an adaptation scheme to the current environment. The adjustable parameter of the online garbage was optimally selected to every use environment. The author's contribution was in the design of the optimization method and in development of Matlab scripts to optimize the online garbage model for the given data set.

Publication 2 [52] introduced a novel algorithm that was based on the Nokia's speaker-dependent voice dialing technology. The algorithm provided a user interface, where the system was expecting the user's "magic word" to launch voice dialing. The system provided a truly hands-free control of the mobile device in a car environment. The author's contribution in Publication 2 was the overall algorithm design with multi-level user interface and development based on an initial idea of Dr. Olli Viikki. The author also carried out the real-time experiments in car environment to verify the performance of the algorithms.

Publication 3 [53] showed that the complexity of a speech recognizer can be reduced by decimating the time sequence of feature vectors. The author's contribution in the work was the design and development of the decimation filters and the application of the decimation in the domain of speaker-dependent name dialing.

Publication 4 [28] introduced the overall framework for multi-lingual voice dialing from the perspective of the Asian languages and tone recognition. The author's contribution was in the development of the tone recognition algorithms and the overall structure of the publication.

Publication 5 [16] studied the characteristics of Chinese names and the requirements for name dialing. The special feature of lexical tone in Chinese languages was studied in detail. The work done for the publication helped in deciding on the need for tone recognition in tonal languages. The author's contribution was in the specification of the research topic and guidelines in preparing the databases used in the evaluations.

Publication 6 [128] gives a detailed solution for tone recognition approach in Mandarin speech recognition. The novelty of the tone recognition provides both good tone recognition accuracy and low memory footprint for the acoustic model. The author's contribution included the design of the modeling scheme and supervision of the implementation and testing of the feasibility of the approach.

Publication 7 [121] introduces several approaches to reduce the complexity and memory footprint of a multi-lingual voice dialing engine. The author's contribution in the publication was related to quantization of model parameters to improve speed of computation. The author also suggested quantizing the feature vector data to further reduce the memory required to perform speaker adaptation.

Publication 8 [85] outlines a method to reduce the language development effort of a multi-lingual speech synthesis engine. The publication is based on author's idea of using

another similar language to synthesize the speech of a previously unsupported language. The implementation of the idea into the multi-lingual speech synthesizer helped to increase the number of supported languages very quickly and efficiently.

Publication 9 [54] outlines the multi-lingual voice dialing technology used in the Nokia S60 mobile devices. The author's contribution included the design of the contents of the paper. The author also collected the necessary information from the previous publications to make a compact presentation suitable for the conference.

Chapter 2

Fundamentals of Speech Recognition and Speech Synthesis

AUTOMATIC Speech Recognition (ASR) and Text-to-Speech (TTS) research are the fundamental parts of any voice user interface technology. They are the most important building blocks when designing an application with voice user interface. This chapter outlines the basics of speech recognition and speech synthesis from voice dialing application point of view. This introduction covers the necessary information for the reader to understand the underlying ASR and TTS technologies. A very good and recent review of multi-lingual speech processing methods has been outlined by Schultz and Kirchhoff [100]. The chapter is first started with an introduction to general voice user interface building blocks and after that the concept of phonetics and its application to speech processing is covered in Section 2.2. A detailed description of speech recognition and speech synthesis is presented in Sections 2.3 and 2.4, respectively. Section 2.3 also highlights the importance of acoustic model training and how it can be further improved. The final section of the chapter is devoted to language data resources that are necessary in the development of a multi-lingual voice dialing system.

2.1. VOICE USER INTERFACE

Before heading to the review of speech recognition and speech synthesis technologies, let us consider a full voice user interface as a block diagram. Such a block diagram is shown in Figure 1. The figure is split into three parts: speech recognition, speech synthesis and voice UI. The first two parts are covered in this chapter. An example of voice dialing user interface is given in Chapter 5

The block diagram of speech recognition can be split into functional parts as shown in Figure 1. The main processing parts are the feature extraction front-end, word hypothesis generation and sentence hypothesis generation. The result of this chain is given to the voice UI. The result may contain several candidates in decreasing order of probability. Feature extraction processes the input speech into a representation that is both compact and contains the necessary information to differentiate spoken sounds. Word hypothesis generation takes as

input the feature vectors from the front-end and the word models defining the recognizer vocabulary. The recognizer vocabulary may be small or large, depending on the task at hand. For a number dialing task, it may be the 10 digits while for a dictation task it may be tens of thousands of words. The lexicon is shown here as a database but it may also be a processing module where the pronunciation of each word is obtained with a set of rules (cf. Section 3.6.2) or a statistical model (cf. Section 3.6.3).

The sentence hypothesis generation takes in the multiple word sequence hypothesis with their probabilities and a language model. The multiple word sequence hypothesis are contained within a so called lattice [59] and can be processed by the sentence hypothesis engine using the language model. The language model tells the sentence generator how different words can follow each other in the given language. Each word transition is associated with a probability. In the scope of this thesis, the sentence hypothesis generator is not used and therefore the technologies associated with it, including the language model, are left out of the thesis. The interested reader is advised to look at the following books [43][59][94].

The speech synthesizer is an essential part of a voice user interface. The input to the speech synthesis module is the text to be synthesized. The language of the speech should be known or a dedicated language identification module can be used to determine the language of the text in a multi-lingual context. The first part of the speech synthesis processing chain is text processing, which consists of optional language identification, text tokenization and text normalization. Language identification is handled in greater detail in Section 3.3. Text tokenization refers to the process of separating the text into words and utterances. Depending on the language this may be either trivial (most Western languages) or very complex (Chinese). Text normalization includes processing of acronyms and abbreviations known to the system and expanding them if necessary. Additionally, numbers can be handled at this stage, determining how they are pronounced in the current context, e.g. decision on using cardinal or ordinal numbers. The linguistic analysis determines the prosody for the sentence. Prosody means speech phenomena such as phrasing, intonation, duration, gain, quality and other voice characteristics aspects.

Prosody describes how speech is generated. The prosody makes the language live and also enables the prosody related meaning of the sentences to be understood by the listener. It is also an important factor in the quality of the synthesized speech and helps the listener to listen for a longer time without fatigue. Commonly, speech synthesizer prosody generation is done with statistical models using annotated speech and text data. At the final stage of a speech synthesizer, waveform generation processes the information provided by the previous modules and generates the audio signal that represents the spoken message. There are several methods to accomplish the waveform generation. Of these, the most common and relevant to the current work are presented later in this chapter.

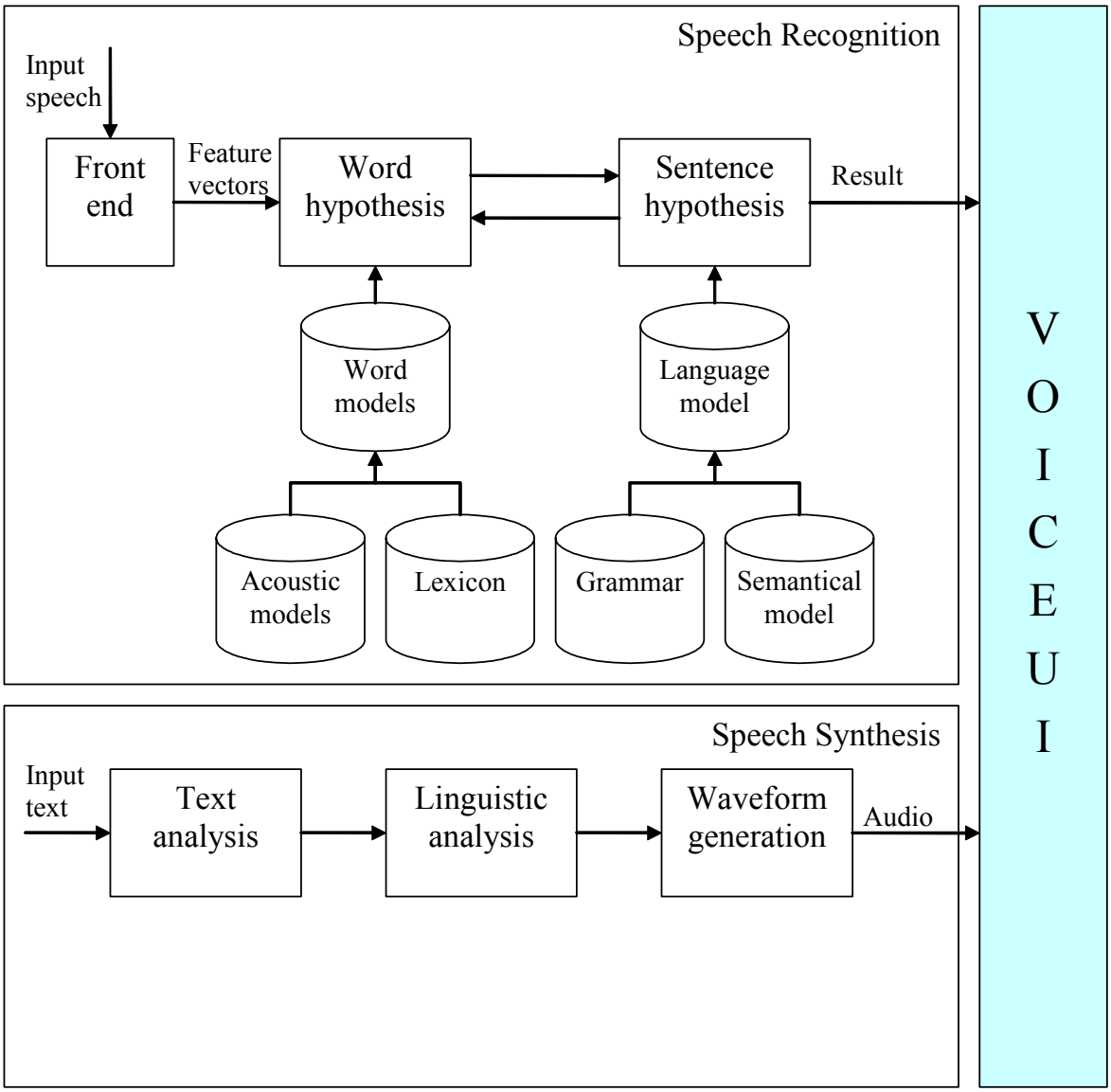

Figure 1: Block diagram of a complete voice user interface

As described above, the block diagram of Figure 1 contains several modules that are not needed in a multi-lingual voice dialing system, e.g. the semantic model and the linguistic analysis. These are not presented in the following sections dealing with technical aspects contained in a voice dialing system.

2.2. INTRODUCTION TO PHONETICS

Development of a multi-lingual voice dialing system handles multiple languages and requires knowledge of phonetics and phonology. These two terms while resembling each other consider two different areas of science. Phonetics is the science of speech in general. It does not study various languages. Phonology, on the other hand, studies topics like number of phonemes in a language and how the sounds are combined to make speech units. This section

concentrates on phonetics and phonology for multiple languages. The interested reader should read the book by Ball & Rahilly for more on detailed information on phonetics [6]. The most common public phonetic representations are described. In this section, special attention is paid on a phonetic definition suitable for a multi-lingual voice dialing system. The primary target of the multi-lingual phonetic inventory in a voice dialing system is to overcome the problem of combining languages together in a flexible manner. This requirement is essential in implementing a multi-lingual voice dialing system. The target is further emphasized in Chapter 4.

2.2.1. International Phonetic Alphabet

The International Phonetic Alphabet (IPA) is a phonetic notation that was developed to be used by linguists, foreign language teachers, translators, etc. The IPA contains approximately 100 different sounds and the categorization is done mainly according to tongue position and manner of articulation. The IPA itself contains phonetic definitions for 29 languages [49]. The number of supported languages is clearly too few when developing a multi-lingual voice dialing system and therefore other options were investigated. The basic phonetic definitions given by IPA are, nevertheless, useful and can be used as a basis for phonetic inventory in new languages.

2.2.2. SAMPA

Due to the complex outlook and encoding of IPA symbols, a machine-readable format of IPA, called SAMPA, was developed [130]. The idea of SAMPA is to use the same IPA symbols for those sounds that have a Latin based marker in IPA. For those sounds that have a complex marker outside ASCII range, an ASCII based identifier is developed, e.g. ə in IPA is @ in SAMPA notation. The markers are designed so that no phoneme separators are needed. The SAMPA encoding is therefore a sort of Huffman code [44].

SAMPA is commonly used in speech research because of its simplicity and ease-of-use on Latin based computer systems without the need for Unicode editors showing Unicode IPA symbols. Currently, SAMPA exists for just less than 30 languages [97]. Therefore, it is not sufficient for a multi-lingual voice dialing system supporting more than 50 languages. A dedicated phonetic alphabet was developed that was based on IPA and SAMPA and takes into account the special needs of a multi-lingual voice dialing system.

2.2.3. Multi-lingual Phonetic Alphabet

The development of a multi-lingual voice dialing system using speech recognition and speech synthesis requires a solid understanding of phonology of the supported languages. The importance of SAMPA is nowadays getting smaller due to introduction of Unicode capable editors and computer systems. The Phonetic Alphabet for Multi-lingual Voice Dialing (later Multi-Lingual Phonetic Alphabet, MLPA) uses its own representation of the phonetic symbols, which is based on both IPA and SAMPA phonetic definitions. This approach was

chosen since both IPA and SAMPA provide accurate phonetic descriptions of the supported languages. The design target of a multi-lingual voice dialing system differs from the phonetic research targets and therefore IPA and SAMPA are not directly used.

The IPA definition is based on phonetics research rather than on the needs of speech recognition or speech synthesis. One consequence of this is that IPA separates sounds based on the tongue position and articulation manner, not based on how the individual phones sound. SAMPA representation was used as the initial starting point for the naming convention for sound units. The Multi-Lingual Phonetic Alphabet is extended on demand when new languages are introduced or when changes to existing languages are needed.

Originally, the Multi-lingual Phonetic Alphabet was developed for the purpose of multi-lingual speech recognition. The reason for this was that the development of multi-lingual speech synthesis started somewhat later when the need for it was identified. The main drivers for the development of the MLPA framework were:

1) To limit the number of distinct sounds defined for a language
2) To limit the number of distinct sounds for a combination of multiple languages
3) To maximize the performance of the speech recognizer for a certain test database
4) To enable a speech synthesizer to use as accurate phonetic representation as needed

First, the number of phonemes within a language needs to be limited to reduce the complexity of the speech recognizer. In general, the complexity (both memory and computation) is linearly dependent on the number of acoustic units in the system. The minimum amount of sound units within a language is decided based on word confusion. A speech database can be used to estimate the number and classification of sound units needed to provide adequate recognition accuracy for a language. In a name dialing application, not all phones need to have a separate acoustic model. As an example, the phone /p/ in words *spot* and *pot* may share same acoustic model.

Second, the sound units should be designed and named in such a manner that as many as possible of the sound units have a counterpart in other languages. This has the effect that the overall number of sound units for a set of languages can be kept minimal. The viability of the chosen classification and categorization can be assessed using speech databases and recognition tasks. Sharing the sound units between languages affects the acoustic model training procedure and therefore affects the performance of the overall system. The more compact the acoustic model set is for a given set of languages, the more data will be used for training each individual acoustic model. This holds true when the amount of training data remains fixed. More training data per model is usually beneficial and results in better performance and more stable acoustic models. The importance of utilizing multi-lingual training data was observed also by Kunzmann et al. [72] especially for the case of recognizing non-native words such as names and cities.

Third, the quality of the speech recognizer can only be judged by testing. The quality of a speech recognition system is dependent on the accuracy of the phonetic representation and on the modeling capability of the acoustic models. The target of the phonetic definition is such that the speech recognizer's performance is maximized when using a given speech a database as benchmark. This can be repeated without effort when the system parameters change. Therefore, this allows the possibility to optimize and iterate and finally to achieve a desired recognition accuracy. The goodness can also be verified using real-time experiments with real test subjects. This is commonly only done during the final verification steps of the system because repeatability of real-time tests is poor. Finding the optimal parameters often requires iterative methods to tune parameters of the overall system. One such parameter is the phonetic definition used in the training and testing data of the system. The methods for finding the optimal phonetic definition for each supported language are discussed in detail in Section 4.2.

Finally, a speech synthesizer requires sufficiently accurate phonetic definition for each language. The evaluation of a speech recognizer can be performed objectively using test speech databases while the evaluation of a speech synthesizer usually requires subjective evaluation by native persons. Therefore, having a sub-optimal phonetic definition does not necessarily harm the speech recognition accuracy but the same sub-optimality can easily be detected in a speech synthesis system. A speech synthesis system requires extensive tuning of the phonetic definitions for each language. This should lead to a larger number of sound units for each language as opposed to the one specified for speech recognition. Although this is usually the case, there are some methods that can help in maintaining the number of sound units at a lower level. First, the acoustic model can utilize tying to reduce the number of components in the model (cf. Section 4.4.3). Alternatively, a more accurate phonetic definition in the speech synthesis can be mapped to a lower resolution one in speech recognizer. Additionally, processing within the speech synthesis engine can use context analysis to modify the phonetic definition within the engine. This is clearly more difficult to develop and maintain. During the development phase, the context processing rules should be defined. These rules modify the phonetic content into an internal phonetic inventory that will be used by the speech synthesizer. The maintenance becomes more difficult since changes in the system wide phonetic definition may require changes in the speech synthesizer's phonetic processing rules. Another possibility is to use a more accurate representation in the sound units obtained from grapheme to phoneme conversion and then reduce the accuracy for speech recognition while using the maximal accuracy for speech synthesis. The tone information included in the pronunciation is not necessarily used in the speech recognizer but only in the speech synthesizer, where this information is essential. This is applied also in multi-lingual voice dialing system. If the speech synthesis engine should perform additional phonetic conversions it should be made absolutely clear that the phonetic presentations obtained from grapheme-to-phoneme conversion are known to the speech synthesizer.

The difference between the requirements on the accuracy of phonetic definitions for speech recognition and speech synthesis is usually handled by using a separate dictionary or

grapheme to phoneme conversion system for the two technologies. The design constraint for the multi-lingual voice dialing system was to use the same methods for grapheme to phoneme conversion for both technologies. Both engines use exactly the same sound units and thus reduce the development time of the grapheme to phoneme conversion. This also helps the user to understand what the speech recognizer expects him/her to say. The user can listen to the voice tags in the device user interface and get hints on the expected pronunciation.

The MLPA is evolving all the time due to addition of new languages and refinements to the existing languages when the need arises. The most common reason for changing the phonetic definition of an existing language is a problem found with the quality of the current phonetic definition.

2.3. SPEECH RECOGNITION

This section outlines the basic technology that is needed to develop a successful voice user interface, such as voice dialing. The section is split into three parts: feature extraction, speech recognition and acoustic modeling. The section is started with an introduction to various types of speech recognizers before going into the technical details of speech recognition and acoustic model training.

2.3.1. Classification of Speech Recognizers

Traditional types of speech recognition are grouped according to the following criteria. The criteria that match a voice dialing system are marked in bold type face.

- **Speaker-independent** vs. Speaker-dependent
- Language-dependent vs. **Language-independent**
- Continuous vs. **Isolated word**
- **Embedded** vs. server based
- Full word modeling vs. **sub-word modeling**

When designing a speech recognition system, the above criteria need to be kept in mind. Most of them have an impact on the design complexity, development time, cost and the overall complexity of the speech recognition system. The following paragraphs explain this claim and each criterion more thoroughly.

Speaker-independent speech recognition systems are capable of coping with multiple speakers even without any additional training from each user. As can be seen in later sections, this usually requires large amounts of training data (speech) to develop a set of acoustic models that can represent the speech of a larger population. This, of course, has a direct impact on the development time and cost. This is one reason why speaker-dependent voice dialing was introduced before speaker-independent voice dialing in mobile devices [54]. Speaker-dependent systems require user interaction with the device before the system can be

used. Usually, this means a special training session, where the user speaks the keywords one or more times to create the voice tags [74]. The speaker-dependent models do not generally perform very well with other speakers, however, they are well suited to a single user device, such as a mobile phone. Speaker-dependent systems, especially for isolated word recognition have relatively short development times and do not require large training databases and most importantly, they are language-independent. The users of the speaker-dependent systems generally don't like the training session and prefer speaker-independent systems.

Language-independent most often means speaker-dependent. In such systems where the user is training the device, the user's spoken language is of less importance. Only for certain tonal languages, such as Mandarin and Thai, additional language specific processing may be required [129]. Because speaker-independent systems often are language-dependent it means that a large set of speech databases needs to be available for each supported language. The design target of the multi-lingual voice dialing system is, however, to achieve language-independence for the user. By supporting every user interface language of the device, this can be achieved. Later in Chapter 3, it is shown that text processing and pronunciation modeling have language-dependent requirements, too.

The difference between an **isolated word** and continuous speech recognition task may sound small. In reality, the biggest difference comes from the fact that for continuous speech recognition, word boundaries and the amount of words to be recognized are unknown. In the special case of digit recognition, continuous means a sequence of words with unknown length [94]. For an isolated recognition system, only two kinds of errors can be made, the utterance is either misrecognized to another word or the utterance can be rejected as being out-of-vocabulary word. For continuous speech recognition, a word can be inserted or a word can be missed. These are called insertion and deletion errors, respectively. The task is to recognize N words without errors to be able to decode the semantics of the utterance. If a continuous speech recognition system has word accuracy of 99%, then the sentence accuracy for a 6-word sentence will be $(99\%)^6$ or approximately 94.1%. In many applications, such as dictation, a single word recognition error can be handled but several errors make the correction task very challenging to the user. This is especially true, if the error correction should be carried out by voice without any keypad interaction with the application.

Embedded speech recognition is implemented in the device and it utilizes the computing and memory resources of the device. The other approach for implementing speech recognition applications is to use server-based speech recognition technology where the audio capture is performed on the device, but all the speech recognition and speech synthesis tasks are executed on a remote server. For a server-based system, the speech data should be transmitted over the mobile network to a server that performs the speech recognition task and provides the recognition result back to the mobile terminal. The normal telephone channel is not very robust against channel errors, which may degrade the recognition accuracy. Therefore, dedicated algorithms have been developed to support the distributed speech recognition approach, such as the ETSI advanced speech recognition front-end feature extraction algorithm [31]. The voice UI control can be implemented using standardized

protocols, such as VoiceXML [126] or Speech Application Language Tags (SALT) [103]. All the commercial voice user interfaces provided by Nokia are embedded, running solely on the device. A server-based speech recognition system suits best an application that is too complex to be implemented locally in the device. Also, services utilizing large data inventories that change frequently benefit from the server based ASR. Services and applications accessing private user data in the mobile device are best implemented locally in the device.

Full-word modeling refers to the situation where the vocabulary of the speech recognizer is constructed from models representing all the possible words that the system can recognize. Introduction of new words is time consuming and requires additional resources since acoustic models are trained from samples of each supported word. **Sub-word modeling**, on the contrary, provides mechanisms to enable dynamic vocabularies. This can be achieved since the words are constructed from sub-word units that can be re-used to form new words when the need arises [76]. For a speaker-independent voice dialing system, a sub-word model approach is the only solution that can fulfill the system requirement of dynamic vocabularies.

2.3.2. Feature Extraction

A speech recognition system is split into two main parts: a feature extraction module and the decoder that finds the correct result. A good feature extraction algorithm is such that it preserves important speech information and removes all unnecessary information from the speech signal. Furthermore, it should perform equally well for all genders, ages, languages and use environments. The most commonly used feature extraction algorithm (front-end) is the Mel-Frequency Cepstral Coefficient (MFCC) front-end [93].

Another commonly used approach for feature extraction is the LPC analysis of speech [3][33][57]. The current speech sample is estimated with previous samples using a linear predictor. The LPC Cepstrum can be calculated from the LPC predictor coefficients. LPC parameters are more sensitive to noise than MFCCs and therefore other methods are preferred over them [58]. The auditory model front-end was inspired by the human hearing system and tries to mimic the processing in the ear. The original work by Seneff [99] has been augmented with more recent work such as those listed by Kim et al. [67]. Hermansky proposed the method of Perceptual Linear Prediction (PLP) to overcome the problems in the LPC analysis of speech [40]. In PLP, the speech is first processed with an auditory-like pre-processor. The result is converted into prediction coefficients using low order autoregressive estimation. The method of RASTA-PLP [41] further improved the robustness on channel convolution noise effects by using a band-pass filter on spectrum coefficients.

The MFCC front-end and its most common refinements are outlined next.

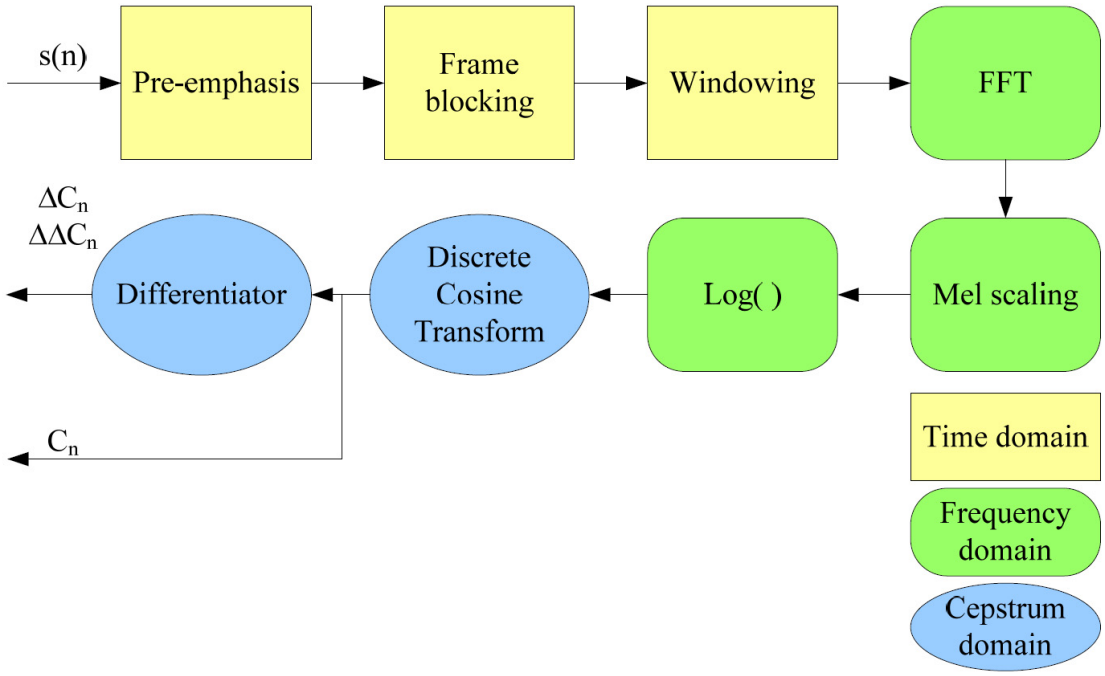

Figure 2: Standard MFCC front-end

Figure 2 shows a basic block diagram of an MFCC based front-end. The figure shows the different data domains. The three domains, time, spectrum and Cepstrum, are shown in square, rounded and oval shape, respectively.

In every speech recognition system, the parameters found in the front-end are tuned to the system and therefore slight variation from one system to another can be observed. The values given here are common to any speech recognition front-end and can be replaced with other values, depending on the application and system requirements. The values of the parameters can be utilized to adjust the complexity and time sensitivity of the front-end. The sampling frequency of the speech signal is 8 kHz and the front-end produces 100 feature vectors per second. In Figure 2, speech processing starts from the top left corner where the front-end takes the user's speech as input, denoted with $s(n)$. The first block, the pre-emphasis function, performs a simple high-pass filtering to remove any DC component and to emphasize the high frequency content of the speech signal. The filter's response curve resembles the response curve of the human ear. The filter is commonly implemented as a first order FIR filter of Equation (1).

$$y = x(i) - k * x(i-1), \quad \text{where } 0.9 \le k < 1.0 \tag{1}$$

The sampled speech stream is then divided into frames of N samples with a shift of M samples between two consecutive frames. The framing process is shown in Figure 3 for N/M equals to 3.

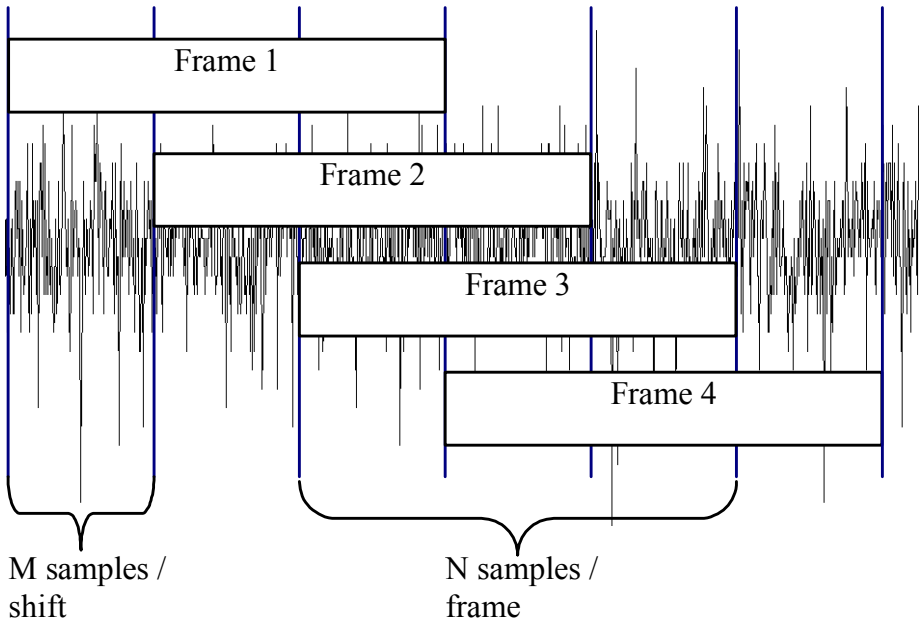

Figure 3: Diagram showing the speech segmentation into frames

The *N* samples included in a frame are then windowed with a Hamming window to help in the succeeding step, namely the spectrum estimation. The spectrum estimation is usually implemented with a Fast Fourier Transform (FFT) due to its speed and efficient implementation. Next, the power spectrum is computed from the FFT. The signal now has power values corresponding to frequencies from 0 Hz to 4 kHz, the Nyquist frequency. The power frequency data is then filtered with a filter that resembles the human ear's sensitivity curve, called Mel scale filter [23]. While the human ear is sensitive to frequency variations in lower frequency values, this sensitivity is reduced for high frequency signal components. Therefore, the filter's frequency sensitivity reduces when the frequency gets higher. The filter's response is shown in Figure 4.

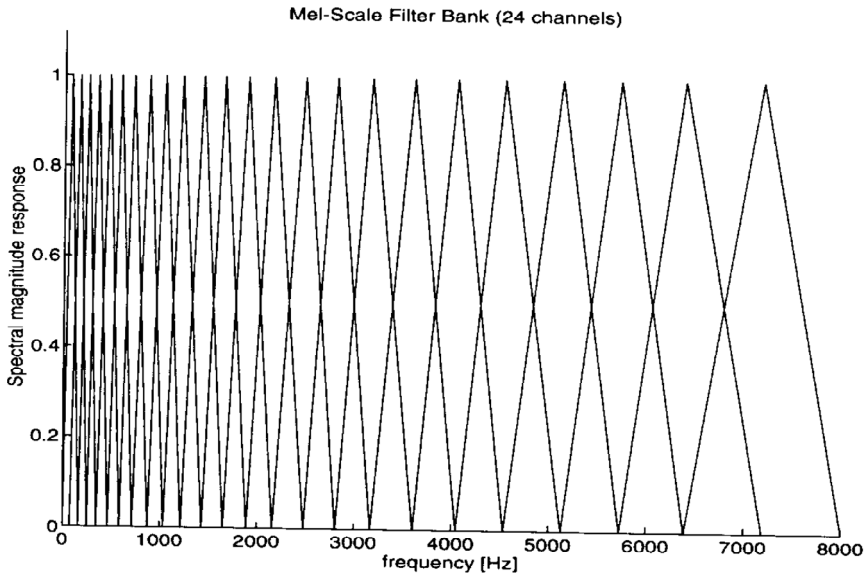

Figure 4: Filter bank as defined by the Mel scale

Following the Mel scaling the remaining signal values are compressed with a non-linear function, such as a logarithm operation. The purpose of this conversion is to limit the dynamic range of the parameters. The signal is further converted from frequency domain to the Cepstrum domain by Discrete Cosine Transform (DCT) [95]. The purpose of the DCT is to reduce the dependency between the feature vector components within a single frame. The DCT coefficients are truncated so that 10-15 Cepstrum coefficients, called the static coefficients, are extracted. The remaining coefficients, corresponding to higher cosine frequencies, are neglected since they contain mostly noise. With this process, the pitch of the voiced speech is effectively removed. The static coefficients correspond to the spectrum of a single frame. The change of each Cepstrum coefficient over time is useful in speech recognition and hence a simple regression type derivative calculation is usually applied to filter the static coefficients to produce 1st and 2nd order derivatives over time. The power of the signal is very important and is commonly included in the feature vector, along with its derivatives. The result of the front-end process is commonly a 39-component (13 Cepstrum coefficients, 13 deltas and 13 delta-deltas) feature vector, which represents the static and dynamic (derivatives) information of the signal frame being processed. This is the feature vector as it is used by the majority of the speech recognition systems around the world.

Enhancements to the Feature Extraction

The MFCC representation is affected by auxiliary acoustic characteristics of the signal, such as signal amplitude, microphone response curve, acoustic environment, etc. All these factors may affect the robustness of the overall system and therefore enhancements have been developed. A few methods are presented next and a more thorough introduction is given to the methods utilized in the multi-lingual voice dialing system.

Noise suppression methods target reducing the mismatch between noise free use environments and real use environments that inherently contain varying noise sources. The basic methodology of noise suppression for speech recognition is the same as used for speech enhancement but the objective is different. Therefore, the algorithms used are not exactly the same but refinements to optimize speech recognition accuracy have been made. The problem of noise suppression methods is that they will produce some distortion to the clean signal because of non-stationary speech signal and sub-optimal noise estimation algorithms [80].

Cepstral Mean Normalization (CMN) is a commonly applied method to remove the effect of convolution noise in speech [2]. Each feature vector component is processed separately and a mean value is estimated over the whole duration of the signal. The mean is removed from each component and the result is forwarded to the back-end of the speech recognizer (See Section 2.3.3). The major problem of plain CMN is that it needs to be computed over the whole utterance and is therefore not very well suited for real-time applications that should have a timely response to user's spoken input.

A different type of Cepstral normalization scheme was developed by Viikki et al. [123] to better cope with the real-time requirements and to achieve good statistics compensation for every use environment. The normalization scheme forms the basis for efficient parameter compression and quantization in acoustic models and possibility to combine speech databases from various sources and languages (See Section 4.4). The basic idea is that the statistics of the normalized signal should be preserved in all environments and with all speakers. The estimation is also done for every parameter as in CMN but the estimation window is time limited and moving. For a given frame, N frames backward and M frames forward are used for the statistics estimation. In the simplest case, both mean and variance are estimated and the signal is compensated with these to make it $N(0,1)$ over the estimation window. As the window is moving with the processed frame, the longer-term (e.g. over the file) statistics of the signal are slightly out of the $N(0,1)$ target. The first mean and variance estimators were implemented using rectangular window of a size of 0.8 seconds, placed 0.4 seconds before and after the current frame [123]. To reduce the computation and memory, the estimation process was changed to be recursive with a fixed look-ahead of 40 frames forward of the current frame. The method described by Viikki et al. [124] was originally for speaker-dependent voice dialing and the mean and variance normalization were applied to all feature vector components. For speaker-independent voice dialing, this was found to be less suitable since the inter-user and inter-language variation is larger and does not strictly follow the $N(0,1)$ statistics so well. Therefore, the Cepstrum coefficients (including their derivatives) are mean normalized and the power term (including its derivatives) is mean and variance normalized.

A few observations can be made concerning the feature vector normalization. First, it adds a 400ms delay in the recognition process since the forward look-ahead should be sampled before the values can be used for statistics estimation. Second, given a stationary input signal the output of the normalization is close to white noise [51][53]. Finally, the low pass characteristics of the speech modulation spectrum are preserved with mean and variance

normalization. Feature vector normalization has been found to be very noise robust in speaker-dependent voice dialing [52][124] and has proven to give excellent results for multi-lingual voice dialing, too. In addition to its applications in speech recognition, Cepstral feature normalization methods are widely utilized in speaker recognition. There, the most commonly applied methods include Cepstral liftering and Adaptive Component Weighting (ACW) [82]. Cepstral liftering accommodates for the sensitivity to spectral slope in the low-order Cepstral coefficients. Additionally, the noise sensitivity of the high-order Cepstral coefficients is improved. ACW tries to remove the channel effect from the spectrum while emphasizing the formants. Later on it will be shown that this type of normalization plays an important role in acoustic model parameter compression and quantization.

2.3.3. Speech Recognizer (the Back-end)

The speech recognition engine, i.e. the back-end, uses the feature vectors produced in the front-end and finds the model or model sequence that best matches the user's speech. Depending on the recognition task, the result can be a single word, a sequence of digits or a sentence. This introduction concentrates on isolated word recognition from the multi-lingual voice dialing perspective.

Multi-lingual voice dialing utilizes Hidden Markov Models (HMM) which are commonly used in speech recognition [93]. The HMM framework sets following assumptions concerning the speech signal characteristics:

- Speech signal (feature vectors) is stationary over a (short) period of time. Changes in the state occur instantaneously and without delay
- The probability of occupying a given state does not depend on the previous observations.

HMM can be regarded as a state machine where each individual state corresponds to an individually defined speech event. Transitions between states are governed by transition and observation probabilities. A simple left-to-right[1] HMM structure is shown in Figure 5. The HMM is defined by its elements as given in the list below [94].

- Number of states is denoted with N (3 in Figure 5) and it is commonly linked with the temporal duration of the event (phoneme, word, utterance) modeled by that HMM. Sub-word models, such as mono-phones, are commonly between 3-4 states. A word model for digits may have up to 15 states, depending on the desired temporal accuracy of the model.
- M is he number of observable symbols in a state.
- The transition probability between any two states (including staying in the same state) is given by the matrix $\boldsymbol{A}$, whose elements a_{ij} define the transition probability

[1] Speech is a directed signal in the sense that order of events is always the same for a given word.

from state *i* to state *j*. In the basic HMM framework any state can be reached from any other state, hence all a_{ij} are greater than 0. For speech recognition, a left-to-right HMM means that many of the transitions are 0, e.g., in Figure 5, only the transitions that are shown are not equal to 0.

- The observation probability **B** in a given state is denoted by $\mathbf{b}_j(\mathbf{x}_t)$ which is the probability of observing value **x** in state *j* at time *t*. Commonly, speech recognition systems use the continuous density HMM (CDHMM) framework and the probability is modeled with a mixture of Gaussian distributions. Terminology used in speech recognition often uses b-probability when referring to the observation probability.
- The initial state distribution, $\boldsymbol{\pi} = \{\pi_i\}$, defines the probability of being in state *i* at time 0.

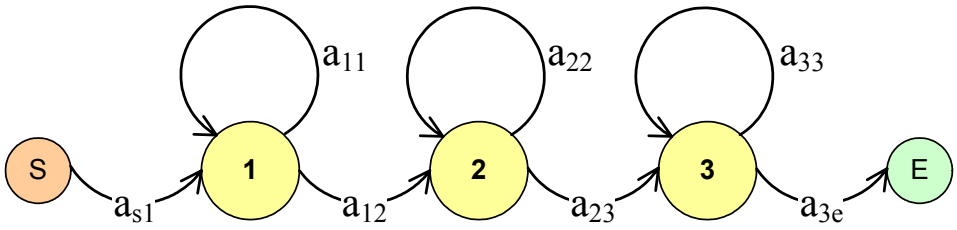

Figure 5: Diagram of a Hidden Markov Model (HMM)

The following notation is commonly used to denote an HMM:

$$\lambda = (\mathbf{A}, \mathbf{B}, \boldsymbol{\pi}) \tag{2}$$

The application of the HMM defined above requires a solution to the following three problems. First, given an observation sequence **O** = ($\mathbf{o}_1,\mathbf{o}_2,\ldots,\mathbf{o}_T$) and the model λ, how can the probability of the observation sequence, P(**O**|λ), be computed efficiently. This is called the evaluation problem and it evaluates the match between the model and observation sequence. The solution to this problem is the Forward-Backward algorithm [94]. Second, when the observation sequence **O** = ($\mathbf{o}_1,\mathbf{o}_2,\ldots,\mathbf{o}_T$) and model λ are known, how to find the optimal state sequence. This is a hidden state sequence that was generated when observing the observation sequence **O**. The solution to this problem is obtained by the Viterbi algorithm [125]. The third problem is that how to estimate the model parameters λ=(**A**, **B**, **π**) that maximize P(**O**|λ). There should be a method to estimate the model parameters to maximize the probability for observation **O**. This is the problem of training the speech recognizer. The solution to this problem is given by the Baum-Welch algorithm [8][9]. The references given for the three problems give the theoretical background needed to understand the theory, while this and the

next section concentrate on the practical issues concerning a real application. The solution to the second problem suggests the use of the Viterbi algorithm to find the optimal state sequence for the given observation **O** and the HMM λ. In speech recognition, it is common to derive several competing solutions λ_i's and the one with the highest probability is selected.

The Viterbi algorithm does not impose any restrictions on the topology of the HMM. Different types of speech recognition applications may use various modeling topologies and therefore all these need to be accommodated. For an isolated word recognizer that is using sub-word models, there has to be a mechanism to glue the sub-word models together. The probability for a sequence of sub-word models need to be evaluated. One such mechanism is the implementation of the Viterbi algorithm with the so-called token passing framework [134]. In this method, each state is associated with a token that has the following information embedded:

1. The cumulative probability up to time *t*
2. Complete history of the route of the token within the HMM network
3. Additional information that should be reported by the recognizer

The token propagates within the recognition network according to the probabilities generated within the Viterbi algorithm. The non-emitting states (S and E in Figure 5) glue two consecutive HMMs with each other. The non-emitting states occupy no time and they do not contribute to the cumulative probability. At the end of the recognition task, the winning token contains the highest probability and the full path traveled by the token. In addition to simple history information, the token may contain individual word start and end times, word confidence scores, word level probabilities, etc. This additional information may be used in the verification of the result, such as out-of-vocabulary word rejection (cf. Section 4.3.6).

2.3.4. Acoustic Model Training

The recognition accuracy of a successful speech recognition application is only as good as is the acoustic model. The front-end and back-end are important but once they have been designed, the acoustic model has the biggest contribution to the performance.

In order to generate good enough models for the application, the following needs to be in place:

- Definition of the acoustic model structure and format
- Annotated training data
- Software to implement the acoustic model training.

Recall the third problem, the training problem in the previous section, where the target is to specify the HMM parameters $\lambda = (\mathbf{A}, \mathbf{B}, \pi)$ so that they maximize the probability of the given training data. The maximization problem $\max\left[P\left(\mathbf{X}|\lambda\right)\right]$ cannot be solved directly and therefore

iterative methods have to be used. This is accomplished with maximum-likelihood training using the Baum-Welch algorithm [8][9]. The basic assumption when applying maximum-likelihood training for speech recognition is that maximizing the probability of the training data for the model also maximizes the recognition accuracy. This is actually not imposed in any way and hence a separate test or verification data is needed to estimate the recognition accuracy.

Practical Considerations in Acoustic Model Training

Well established acoustic model training algorithms, such as the Baum-Welch algorithm, create the basic framework for training. In practice, several decisions need to be made about the final configuration of the training procedure. At least the following topics should be considered.

- How does the task affect the acoustic model topology
- How much data is required vs. available training data
- What is the format of the training data
- What software is available for acoustic model training
- How much time is available for each training iteration

Definition of the acoustic models is a combination of phonetics, systems design and complexity estimation. Depending on the application, various topologies for the HMM can be chosen. Commonly, the number of states is defined according to the length of the event it is modeling. For a sub-word model, such as a mono-phone, the number of states can be between three to five states, and for a whole-word model based system, 5-15 states. In speech recognition, the most commonly applied model topology is- the left-to-right topology, also shown in Figure 5. The voice dialing system uses mono-phone models, which means that each acoustic model represents a single phone without context information of the preceding or succeeding phone. Imperl et al. [48] study training of multi-lingual context-dependent models where the number of tri-phones may become too high unless special clustering methods were used. Another study with multi-lingual context dependent models by Niesler [90] concludes that sharing the base-phones[2] between supported languages can reduce the overall amount of model parameters while preserving the recognition accuracy.

The training problem states that the model parameters λ are maximized for a given set of training data **X**. How is the training data constructed and how can it be processed to use it most efficiently? A speech database commonly includes annotation that describes the content of each audio file. The annotation can be either content-only or include time stamps for speech events in each file. In either case, databases are commonly manually annotated to avoid mistakes and include the actual content. The database annotation is either text, or text and phonetics. In the latter case, the phonetic notation needs to be aligned with the one used in

[2] Base-phone is the phone for which the left and right contexts are defined

the system under training. Prior to annotation, features are extracted from the speech database, as described earlier in Section 2.3.2.

The training itself uses words and full sentences and combines the acoustic models of smaller units to generate a master model for each full audio file. The training procedure can be speeded up if the timing information is available for the database. In the conventional Maximum-Likelihood estimation, every possible state sequence is evaluated to find the one giving the maximum probability. This search space can be reduced by setting a pruning threshold for the search so that less probable state sequences can be skipped and hence make the training faster. Another approach is to use the time annotation to limit the search within the annotations. This can significantly improve the training speed. When combined with the strict time limits and a certain degree of freedom to the annotated boundaries can both preserve the training speed and achieve high recognition accuracy. The method proposed by Huggins-Daines and Rudnicky [45] suggests that adding limitation to valid phoneme sequences into Baum-Welch estimation can improve the training speed without sacrificing the recognition accuracy. A different approach was proposed by Malfrère et al. [81] where a speech synthesizer was utilized to find the phonetic alignment of the training corpus. This alleviates the need for having initial segmentation for the database.

There are two ways to generate the original annotation, one is based on the actual contents of the database and the other is based on the prompts in the database. Obviously, the former results in more accurate annotation. Regardless of how annotation was done, timing of the words is normally not included in the annotation. The acoustic model definition needs to be aligned with the annotated training data. The database developer commonly provides the training data annotation and therefore some adjustment is needed, in case the phoneme definition differs from the original version of the database. The training software, such as HTK [42], can be used to iterate over the acoustic models and the training data to obtain optimal models in maximum likelihood sense. Further improvement of maximum-likelihood training can be achieved using several approaches that take the recognition accuracy into account during the training procedure. These approaches are called discriminative training methods [63][75].

2.4. SPEECH SYNTHESIS

A text-to-speech synthesizer (speech synthesizer is user interchangeably with text-to-speech synthesizer) converts a given text to an audio form that can be played through a loudspeaker or stored in a storage device for later use. Important factors relating to the speech synthesizer are the quality of audio, intelligibility of the speech and naturalness of the intonation. These factors limit the scope of text-to-speech from techniques using recorded speech samples or concatenation of isolated words. The basic requirement for a text-to-speech synthesizer is that it should be able to output speech from the phonetic inventory of the system applying appropriate intonation modeling to make the speech sound natural. The quality of the speech signal produced by the text-to-speech synthesizer largely determines the size of the system. The discussion here is general to text-to-speech synthesizer but more emphasis is put on the

low-footprint formant based synthesizer suitable for a multi-lingual voice dialing system with strict memory requirements.

A generic view on text-to-speech synthesis is given first. Introduction to natural language processing and speech signal processing provides a common framework to text-to-speech synthesis technology. Next, the details of natural language processing for speech synthesis are outlined. In Section 2.4.2, speech generation techniques are discussed and most common methods for formant and waveform concatenative speech synthesis are introduced.

From a processing perspective, the input to a text-to-speech system is a piece of text, such as a sentence. The output of the system is spoken speech that contains the information given by the text input, pronounced in such a way that sounds natural and matches the meaning of the sentence. Therefore, the task of the speech synthesizer is not just converting the text into sounds but also to make sure the naturalness and intonation of the speech matches the intent in the sentence to be spoken by the system.

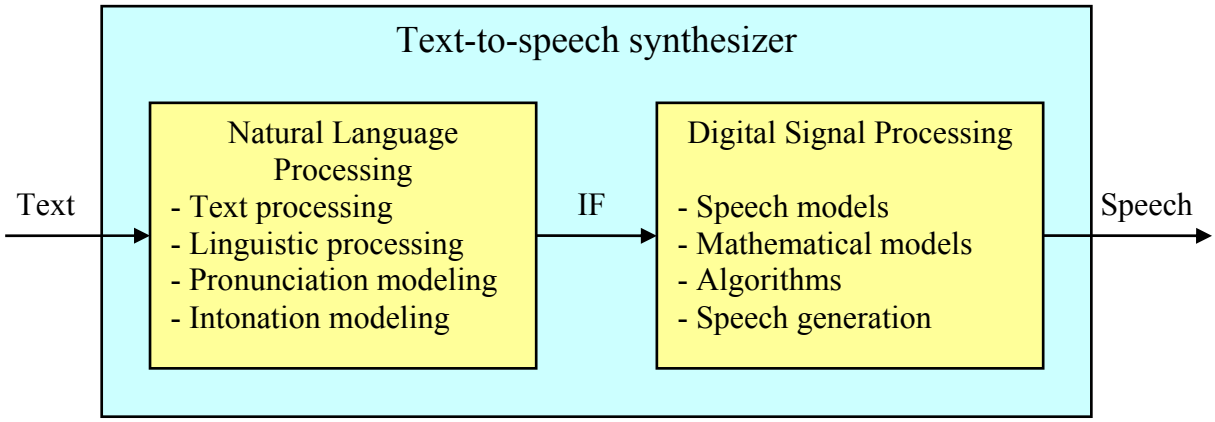

Figure 6: Simple diagram of a text-to-speech synthesizer

Figure 6 outlines the basic building blocks of a text-to-speech synthesis system [29]. The input to the system is text and the output on the system is speech. The Natural Language Processing (NLP) module converts the text into an intermediate representation and the Digital Signal Processing (DSP) module generates the speech signal from the intermediate data. The Intermediate Format, IF, between the two processing blocks represents the information that is passed between the two modules. There is no single correct list of needed information to be included in the IF data and hence it may contain anything that the designer of the system considers important. The IF data contains at minimum phonetic presentation and the necessary intonation information to generate speech in the DSP module. Both modules are important to the overall quality of the generated speech and it is difficult to split the quality characteristics between the two modules.

2.4.1. Natural Language Processing for TTS

The start of the NLP module is the text-processing that converts the input text into a format that can be handled by the succeeding parts. The first task of the text-processor is to separate

the text sentence into words, i.e. tokens. This is trivial for most Latin-based languages but several Asian languages, such as Chinese, Japanese and Thai, do not consider word breaks within a sentence. For these languages, additional word tokenization methods should be used [62][91][105]. In addition to word tokenization, the sentences commonly include numbers, abbreviations and acronyms. Numbers may be natural numbers or amounts of currency, time, distance, etc. In any case, these need to be converted into text representation that is equivalent with the meaning of the sentence. An example sentence: "I will meet Dr. Jones at 12 o'clock at the jct. of Main st. and 3rd st." should be converted into

I will meet [doctor] Jones at [twelve] o'clock at the [junction] of Main [street] and [third] [street]

The processed words are denoted with the square brackets. The example sentence contained items such as time, abbreviations and ordinal number. It should be noted that *dr.* can mean either a *drive* or a *doctor*, depending on the context. Similar examples can be found in other languages. Abbreviations are particularly problematic, since they are language-dependent and there is no standard set of abbreviations that can be used.

Morphological analysis processes the text and decides the function and structure of each word. This is needed for the prosody analysis, since the same written format for a word can have varying meaning and therefore affects the intonation of the sentence. An example in English is the word *record* that is either a noun or a verb and consequently has two pronunciations. Morphological analysis considers language peculiarities such as inflections, tense, case, gender and number. All this information is needed to perform phonetic transcription and intonation modeling [34].

The phonetic transcription in its simplest form can be done with a dictionary that contains all words appearing in the text-to-speech input data. In practice this is not feasible due to inflections and many new words that appear to every language on a daily basis. In addition to dictionaries, rules and statistical methods can be applied to pronunciation modeling. More details about the methods available for pronunciation modeling can be found in Chapter 3.

Prosody is a combination of audible properties of a speech signal that are manifested by pitch, loudness and syllable duration. In addition to these core properties, rhythm and timing can be added as prosodic features. In many cases, intonation is used for prosody although it strictly relates to the change of pitch in the speech signal. Loudness is less prevalent as a prosodic feature. Pitch and duration have the biggest effect to the prosody for a given sentence. The prosody generation should be such that the result mimics the human speech as naturally as possible. It should be noted that in tonal languages, such as Mandarin Chinese, Thai and Vietnamese, the pitch assumes a lexical characteristic for a given word. For Mandarin Chinese, there are four lexical tones that alter the meaning of the syllable. As an example, the syllable *ma* can have varying meaning depending on the pitch contour, as shown in Table 1.

Table 1: Example of the lexical tone in Mandarin Chinese

Ideograph	Pinyin	Tone	Meaning
妈	mā	High	mother
麻	má	Rising	hemp
马	mǎ	Falling-rising	horse
吗	mà	Falling	curse

As the NLP and DSP of a text-to-speech system do not heavily depend on each other, they may be designed somewhat independent of each other. The obvious advantage is that a complex task of a full TTS system can be split into smaller sub-problems that can be tackled by specialists of each sub-area. Furthermore, the general interface (IF) between the processing units allows the DSP part to be replaced.

2.4.2. Digital Signal Processing for TTS

Digital Signal Processing (DSP) is responsible of taking the information generated by the NLP module and generating speech corresponding to the information given. The quality of the sound depends on the NLP module but also on the quality and features of the DSP module itself. Two main techniques for the DSP block of a speech synthesizer are outlined next.

The first class of speech synthesizers is formed by the formant synthesizers. The perfect example of a parametric synthesizer is the Klatt synthesizer [70]. These synthesizers, while having inferior speech quality as opposed to the synthesizers of the second type, are nevertheless important. The formant synthesis is often called rule-based synthesis. Simple tuning of the voice between male and female [71] without sample data and phonetic foundation makes them unable to discriminate any language. Finally, the parametric synthesis technology has a long history and hence it is a mature technology. The reasons that the voice dialing system utilizes the Klatt synthesizer are further elaborated in Section 4.5. The rule based synthesizers are defined with a set of parameters for each sound unit, such as a phone. The parameters include information about formant frequencies, their bandwidths and other information and can be represented by a compact set of data. A thorough presentation of acoustic and phonetic features formant based synthesizer is given by Holmes [43].

Aside from formant synthesizers using signal processing and speech production modeling to produce speech, waveform concatenation based speech synthesizers contain a database of speech segments that are concatenated to form new words and sentences. The speech is pre-recorded and usually compressed to save memory. The requirement for the speech database is that the phonetic content and prevalent coarticulation information should be captured in the database to be able to generate natural sounding speech. The smallest units of speech in these systems are commonly a phone, diphone, half-syllable or a triphone [29][43]. While a phone is a single unit of speech, a diphone is a segment of speech that starts

from the mid-point of one phone and extends to the mid-point of the following phone. It therefore contains the coarticulation between two consecutive phones. It is nevertheless not robust enough to produce natural sounding speech since only a single instance of each diphone is stored in the system. The half-syllable and triphone contain at least one full phone, the half-syllable a full vowel and triphone full phone with information from the preceding and succeeding phones. Often, the coarticulation effects of natural speech span longer than a single phone or even triphone. The basic units given above can be extended to more general unit selection based text-to-speech systems [46][88]. They store a large speech database and can optimally select segments from the speech storage to form new words and sentences. A highly simplified example of unit selection is shown in Figure 7.

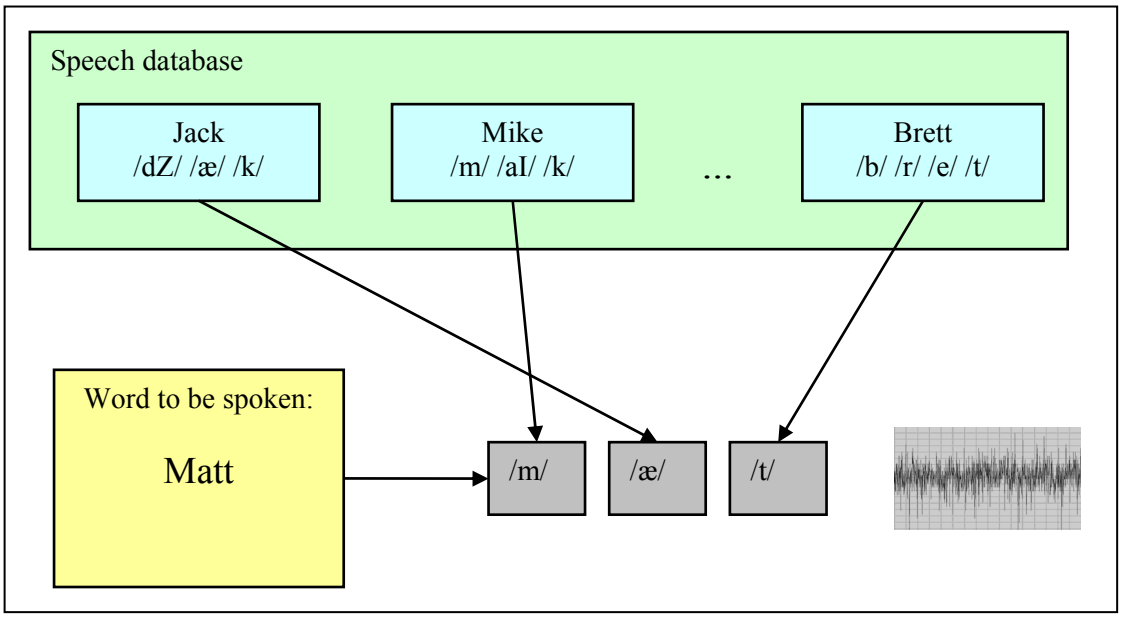

Figure 7: Example of a unit selection speech synthesis

In the example of Figure 7 the speech database consists of names and their phonetic labeling. Simplified IPA has been used for the notation. The system is requested to generate speech for word *Matt*. The synthesizer takes the three phones needed from three individual utterances and concatenates the segments to form the desired speech output. It should be noted that, in practice, the speech database contains full sentences with carefully designed content so as to be able to generate any given speech with a desired quality level. Additionally, the segment length can be considerably longer than a single phone. In this case, the synthesis quality is more natural due to units selected from longer and natural speech segments.

Recently, a new method for text-to-speech synthesis has been proposed. This framework utilizes the well known Hidden Markov Model (HMM) framework. The HMM is used for modeling the speech spectrum, fundamental frequency and the speech prosody. These can be estimated jointly with a single HMM or split into separate models. The basic theory for HMM based speech synthesis was proposed by Yoshimura et al. [133]. A clear advantage of the HMM based speech synthesis is that the voice characteristics can be altered

in an easier manner. Also, frameworks, such as speaker adaptation may be applied to this type of speech synthesis.

Due to the high memory requirements of the concatenative speech synthesizers, most current embedded speech synthesis applications use formant-based synthesis. As the memory available on mobile devices is getting closer to the desktop computers (largely thanks to the development in FLASH memory capacity), the waveform concatenation speech synthesizers are expected to be more common in the near future. If the Hidden Markov Model based synthesizers can fulfill the expectation of smaller footprint with higher speech quality they are likely to become more common in embedded solutions.

2.5. LANGUAGE DATA

Language data is essential in the development of a multi-lingual voice dialing system and has significant impact on the number of supported languages and on the final performance of the system. Two main types of data are available through database providers or through independent collection efforts. Textual language data can be used for the development of grapheme to phoneme converters, text based language identification and general text processing. These methods are presented in Chapter 3. The other type of data is spoken data or speech data that is needed e.g. to generate the acoustic models or to estimate a prosody model for a speech synthesizer. In general, it is necessary to have language resources, both text and speech, for each language that is included in the system. It will be shown in Section 4.2 that in some cases a language can be supported even without any speech data of that language. Aside from the personnel costs, the database collection and license costs are the second biggest investment when implementing a speech recognition and speech synthesis system. The lack of databases and the price of the existing ones is one of the major reasons why multi-lingual speech recognition and speech synthesis systems take such a long time to develop. Language resources needed in the development of a multi-lingual voice dialing system are explained in detail in the following sub-sections.

2.5.1. Text Resources for Multi-Lingual Voice Dialing

Multi-lingual grapheme to phoneme conversion methods, including text-based language identification and symbol conversions are outlined in Chapter 3. Text data is needed for the development of pronunciation modeling methods, application of language identification from text and text pre-processing methods. In addition, text data may be useful for the annotation of speech recognizer training and testing data. It may also help understand the problems in certain languages. Text resources commonly appear as either a lexicon or an annotated text database. A lexicon commonly contains the following information of each word/utterance it includes.

- Word in text format
- Pronunciation with the given phonetic annotation, such as IPA

- Frequency in the language
- Part of speech information

One additional form of text data is the language N-grams, which are useful for the development of language models. In this case the N-gram is constructed from words [83] and gives the relative frequency of each word sequence of length *N*.

2.5.2. Speech Databases for Multi-Lingual Speech Recognition

Multi-lingual speech data is primarily required to generate and test multi-lingual acoustic models. A multi-lingual phonetic set definition is obtained through iterative steps to obtain the best combination of performance and complexity. In order to include a certain language in the voice dialing system, speech data for that language is needed. In certain cases, however, the phonemes of a new language can be obtained from other languages and in this case no training data is needed for the new language. This approach is applied if there are no easily accessible language resources available for that language. The training data should have wide coverage of the phonetic content and the context information of the language. The amount of training data per acoustic model unit should be adequate. It is also beneficial to have many databases coming from same source or that they at least use similar equipment and have a common specification. Examples of such collections of databases are the Speecon [102] and SpeechDat (II, M, Car and E) [104]

Test data is needed for each language that is supported by the multi-lingual voice dialing system. Test data should be representative of the language and should include enough data to cover the phonetics of the language. Test data should also reflect the target real use case to be able to give credible estimate of the final performance in a product. Final requirement of the test data is that it should reflect the use environment of the target platform, such as a mobile device. The last target is the most difficult to satisfy since speech database recordings are usually carried out in a quiet recording environment. A common methodology is to artificially mix environment noise to the audio data. This way, the effect of changing noise level can be investigated. This method does not take into account the user's tendency to change the voice and effort when speaking in a noisy environment. This effect, called the Lombard effect, is explained in detail by Junqua [65].

A multi-lingual speech recognition system also targets at recognizing non-native speech. Therefore, the training and testing data should contain enough samples from users uttering non-native speech. In practice, such data is rare and takes additional effort to collect.

2.5.3. Speech Databases for Multi-Lingual Speech Synthesis

The development of a formant-based speech synthesizer requires multi-lingual speech data for the phoneme parameter estimation, prosody modeling and system verification. Audio data is needed for each new language to be added to the multi-lingual speech synthesizer. There is an exception to this called text-to-speech language morphing, which is explained in Section

4.5.2. The definition of the phoneme parameters of the Klatt synthesizer requires the developer to listen to native speech. Even though the sound units can be shared over languages in the speech recognizer, they most often cannot be shared in a speech synthesizer. This is due to minor and major differences between similar sound units in different languages.

The prosody model used in the multi-lingual voice dialing system is optimized for short utterances, such as names. The prosody model for each language is trained using accurately annotated speech data. The collection and annotation of the prosody training data is the major part of language development effort of a speech synthesizer. The test data of a speech synthesizer is needed to verify the correctness of the synthesizer output in various conditions and on various phoneme combinations. In this case, the testing is merely a comparison of the synthesized speech with the recorded samples in the speech database. The most important method to verify the goodness of a speech synthesizer is subjective listening evaluations.

2.5.4. Common Distributors for Language Data

There are several providers of language resources. Each provides all three types of data. The usefulness of each type clearly depends on the development methods used for a particular voice UI application. The Linguistic Data Consortium (LDC) [78] is hosted by the University of Pennsylvania, USA. It was established in 1992 to create, collect and distribute speech and text resources and lexical for research and development purposes. It serves both research and commercial organizations. ELRA [30] is the European counterpart of LDC. It is distributing language resources to non-commercial and commercial licensees.

Chapter 3

Multi-Lingual Grapheme-to-Phoneme Conversion

GRAPHEME to Phoneme conversion (G2P) is the process of taking a word and producing a sequence of phonemes, i.e. the pronunciation, that represents the pronunciation of the word in that context. In this thesis, due to the multi-lingual aspects, this definition is extended to the following:

Grapheme-to-phoneme conversion takes a word in an unknown language and produces multiple pronunciations of the word that are likely to be spoken by the user and will be prompted to the user as a confirmation message.

The above statement can be split into three important parts that are explained next:

1. The language of the text is not known. The users of mobile devices can have acquaintances of various nationalities and the nationality of the user is not always clear. Therefore, words of any language should be expected.
2. The process of G2P determines the language of the given word and based on it produces several pronunciations to increase the probability to include the pronunciation the user of the device is going to say.
3. The spoken feedback to the user's voice dialing is given with text-to-speech synthesis, which is very sensitive to correct determination of the language of the word. Even small errors in the pronunciation of the word can be annoying to the user.

With this in mind, the following sections explain the methods used for multi-lingual G2P in great detail, introducing the major problems and solutions associated with them. The author's contribution in this chapter is significant and relevant parts are highlighted before each particular topic is presented. The term phoneme is used generally to denote a sound unit since

the terminology in grapheme to phoneme conversion often uses this term. This exception applies only to the current chapter.

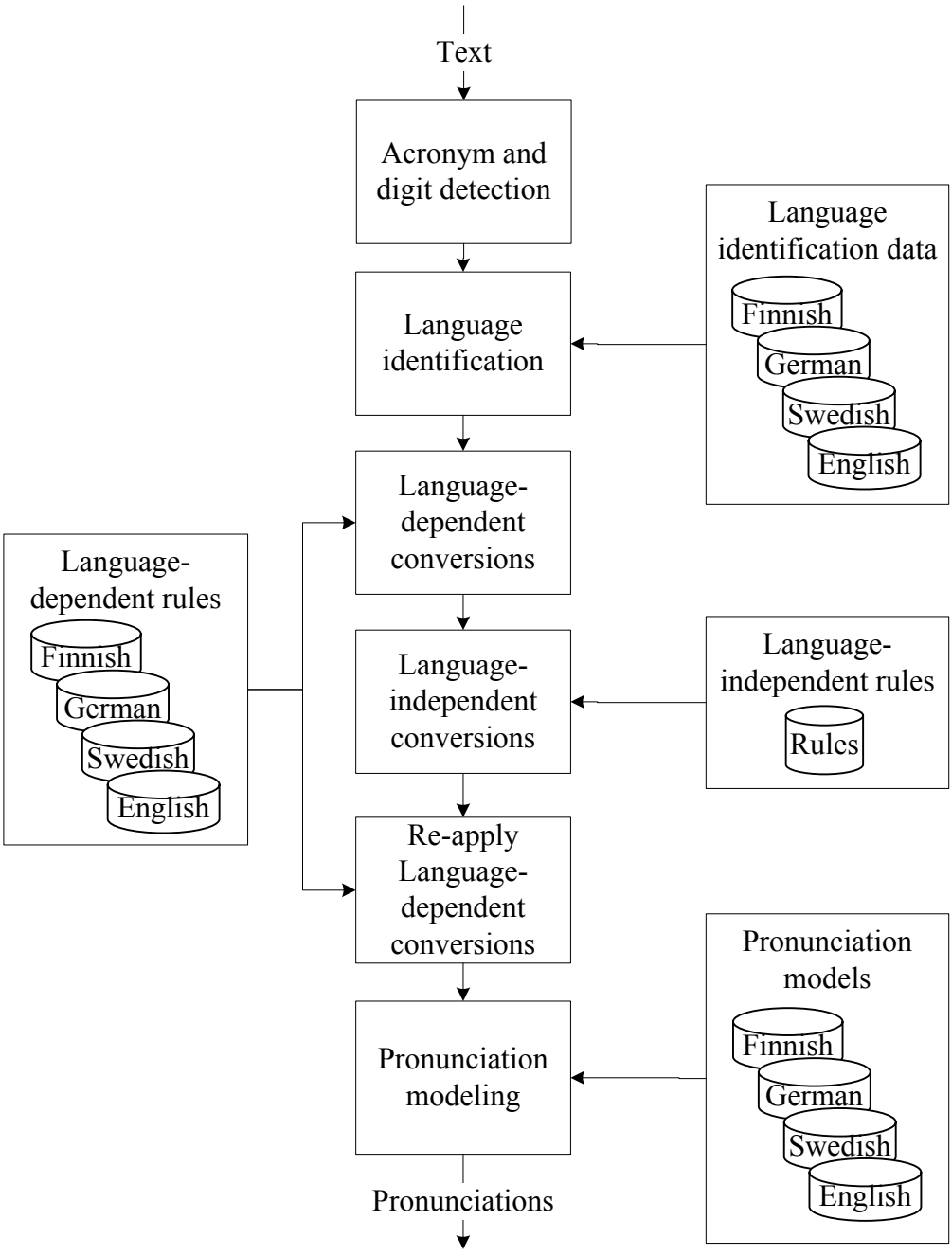

Figure 8: Processing flow of grapheme to phoneme conversion

Before heading to the writing scripts, the figure above contains all the data processing elements found in the grapheme to phoneme conversion system. The processing starts with acronym and digit detection followed by language identification. Various text processing takes place in the language-dependent and language-independent processing modules. The final step is the pronunciation modeling that generates the pronunciation for the word in one

or more languages. All the processing steps are presented in Sections 3.3 through 3.6 and follow the order given in Figure 8.

3.1. WRITING SCRIPTS AND THEIR EFFECTS ON G2P

In the world of multi-lingual speech processing, writing scripts play an important role. The reason for this is that the conversion of the written text into pronunciation, i.e. the phoneme sequence, requires the knowledge of the actual text to be converted. In order to properly decode the text into a pronunciation, the encoding of the text should be known. The basic coding that most computer users are familiar with is the ASCII code, which uses a 7-bit representation of the (English) alphabet to give a code index for each character. This is certainly adequate for one language, such as English, due to the limited number of characters in the language alphabet. For many other languages and especially a combination of several languages, this is not adequate. Especially, in mobile devices that are sold globally, country-specific encodings cannot be utilized. Therefore, a common encoding that covers a large enough set of languages is needed. Unicode [118] provides it.

Unicode specifies a unique identifier (number or code word) for each character that is included in the Unicode system. This value can be used in various complying systems and the number always refers to the same character no matter what the language or writing script is. The Unicode contains characters for most of the world's languages. Unicode itself does not deal with languages but with character encoding. Therefore, several languages can share the same encoding. Currently, Unicode supports over 50 different character encoding. The Unicode website gives a list of some 250 supported languages. Therefore, Unicode is a good choice for the text representation in a multi-lingual voice dialing system. Before Unicode, there were several different, overlapping, encoding schemes used all over the world. The interested reader should refer to the Unicode standard [118].

Another reason for adopting Unicode is that Symbian operating system natively supports 16-bit Unicode, and hence all texts in Series 60 mobile devices are stored with the Unicode encoding.

3.1.1. Use of Unicode in a Voice Dialing System

Once the Unicode has been selected as the prominent encoding for text, a selection is needed between various types of Unicode encoding. There are three basic types of Unicode encoding, Unicode-32, Unicode-16 and Unicode-8. The short names for these are UTF-32, UTF-16 and UTF-8, respectively. The latter two are considered and used in this work.

UTF-16 is the basic representation of the Unicode and supports more than 63,000 characters using 16-bit values for each (hence the name UTF-16). Each character takes 16 bits (2 bytes) of memory to store. For Latin-based languages, such as English and Finnish, there is some redundancy in the storage since all characters of the language could be represented with just one byte. The redundancy can be overcome with common compression techniques. The

pros (+) and cons (-) of UTF-16 with respect to speech processing are shown in the lists below.

+ Can represent all current languages used in a mobile device with single set of 16-bit code points, i.e. characters
+ Each character is easy to access using the index of the character within the array
+ Adding and deleting characters in the middle of text is simple
+ Requires less storage space for Asian scripts than UTF-8
- Wasting space if e.g. Latin script is used
- String operations become cumbersome and standard C programming language string operators cannot be used, e.g. string length and string concatenation.

The other option for the text presentation is UTF-8, which is a variable length encoding method. The basic idea is that existing ASCII encoded text can be directly used with UTF-8 capable software. Another benefit is the good efficiency of the encoding for most of the Latin based languages and characters. Furthermore, C programming language standard string operations are valid, since the end of UTF-8 encoded string is given by the NUL character ('\0'). The drawback of UTF-8 is that the number of characters in the string cannot be determined by the length of the string. The whole string needs to be analyzed before the number of characters is known. Also, adding and deleting characters in the middle of the string is difficult. The following list summarizes the pros (+) and cons (-) of UTF-8 encoding.

+ C-programming language standard string operations can be readily used
+ Existing software that uses only ASCII encoding does not need any changes
+ Good data efficiency with Latin based languages
- Length of the string in characters is difficult to determine
- Adding/deleting characters in the middle of string is more difficult
- Less efficient than UTF-16 for Asian languages such as Chinese and Japanese, using Han characters

Even though both of the encodings have several pros and cons, it was quite straightforward to select one when deciding the internal text representation of a voice dialing system. UTF-16 is easier to handle, especially when the processing requires modification of existing text strings, adding and deleting characters in the middle of the string. If voice dialing was to be used in a system with external interface in UTF-8, it would be easy to make a conversion from UTF-8 to UTF-16. If the external link was with some other encoding than Unicode, the support for over 50 languages and their writing script would have been very demanding to implement. It should be noted that conversion between UTF-8 and UTF-16 is unambiguous and that the length of the resulting word sequence is unknown until the whole

text is processed. When converting from UTF-8, the number of characters in the source sequence is not directly known. Conversely, conversion from UTF-16 results in a UTF-8 stream whose length cannot be determined even if the number of characters in UTF-16 is known.

3.2. ROMANIZATION IN VOICE DIALING

Romanization converts a word in a non-Latin writing script to a representation using a Latin writing script. Transliteration converts a written word to use Latin script. Transcription is the equivalent for a spoken word. Before the time of modern computer systems and the Unicode, there was a need to represent a complicated written language in Latin writing. Computers could not simply display or take in these special encodings and no fonts were available to display the text. In addition, representing proper names from these languages in other languages was not possible with the local alphabet due to character encoding conflict. Therefore, transliteration of these languages was necessary and methods were generated for several such languages. Among these are Greek, Chinese, Russian and Japanese. Another important factor to support Romanization is language learning. An Englishman probably finds learning Chinese next to impossible if the only thing he can get is plain Chinese ideographs and no Romanized version of the language learning material is ever available. Common to all Romanization schemes is that they provide a phonetic representation of the foreign text. This way, it helps both non-native writing systems and language learners to cope with the text.

In the modern computer age, the need for Romanization often arises from the fact that written text needs to be typed into a computer. For Mandarin Chinese, there are more than 20,000 commonly used characters and therefore it is impractical to have a keyboard with all the characters. In this case, a special pronunciation based Romanization scheme is used, which is then further transformed into text using Chinese[3] ideographs. The Romanization system used in P.R. China is called Pinyin [56]. The basic Chinese input with Pinyin uses single syllables and the user should select the correct ideograph for each syllable separately. This is due to the high amount of homonyms in Chinese languages. To improve the input speed, several sentence level Pinyin to character input methods have been proposed. These use the sentence level pronunciation information to guess the correct ideograph for each syllable of the sentence. The sentence level methods highly increase the input speed of writing Chinese language.

Some languages have a standardized Romanization scheme while others have several non-standard ways to Romanize a piece of text. The non-standard Romanization becomes problematic in a voice dialing system because no single method can obtain the correct pronunciation for the Romanized word as expected by the grapheme to phoneme conversion. One user may use one Romanization scheme while another uses his/her way of Romanizing the words. This cannot be properly handled by the voice dialing system and may lead to errors

[3] Chinese here refers to Mandarin Chinese spoken in the P.R. of China.

in the transcriptions. Romanization is supported in the following in the multi-lingual voice dialing system,

First, Romanization from a native script to a Latin form is performed in order to facilitate the grapheme to phoneme conversion in Latin based languages. For example, a Russian word written with Cyrillic letters can be converted into Latin representation and consequently, English and Finnish pronunciation can be generated for the Russian word. This is also a good approximation of the pronunciation of a non-native word when spoken by a Finn or an Englishman. This case may happen when a Finnish person has a Russian acquaintance and wishes to store his/her name in the original form written in Cyrillic letters.

Secondly, many people use Romanization in their mobile devices because of difficult input methods for their native script. It is important to be able to support this approach to storing names in the device. Therefore, the standard or a non-standard Romanization of certain languages is included in the native pronunciation rules to support the possibility of using Latin input in these languages. These are important features in the multi-lingual support of the voice dialing system to cope with multi-lingual names and multi-lingual users. The Romanization methods of a voice dialing system are outlined in detail in Section 3.5. Some languages that commonly utilize Romanization are shown in Table 2.

Table 2: Some languages that define Romanization

Writing language/Country	**Standard Romanization scheme**	**Non-standard Romanization scheme name**	**Unicode support for the encoding**	**Standard Romanization Supported by voice dialing**
Greek	ISO 843	N/A	Yes	Yes
Russian	ISO 9:1995	N/A	Yes	Yes
P.R. China	ISO 7098:1991	Wade-Giles	Yes	Yes
Hong Kong	N/A	Jyutpin	Yes	No
Taiwan	N/A	Tongyong Pinyin, Zhuyin	Yes	Yes
Thai	ISO 11940	N/A	Yes	No
Japanese	N/A	Hepburn	Yes	No
Hindi	N/A	N/A	Yes	No
Arabic	ISO 233-2	N/A	Yes	No
Russian	ISO 9:1995	Multiple	Yes	Yes

As can be seen in Table 2, for most of languages, there are several Romanization methods. This is a major problem since the voice dialing system cannot support several Romanization methods for a single language. Secondly, even though ISO standards exist, they

are not always used but a more prevalent local Romanization is a common practice. Due to the writing script itself, Arabic is the most difficult language to Romanize. The reason is that the normal writing style does not include short vowels and it requires experience and knowledge of the language and context to know how to Romanize a given word. It should be noted that the reason for not supporting many of the Romanization schemes is two fold. First, the Romanization may not be commonly used for actual names in the device but rather as a means for inputting native script, e.g. Pinyin for Chinese. Second, in many languages, Latin script is used for foreign names and may be considered as English, e.g. in Japan. The latter statement is explained in more detail in Chapter 5.6.2.

3.3. HANDLING OF ABBREVIATIONS, ACRONYMS AND DIGITS

Before going into the details of abbreviations, acronyms and digits, the terminology should be clarified.

- An abbreviation is a string of letters ending with a period. It refers to a full word or phrase that has been made shorter. Example: the word background may be abbreviated as backgr.
- An acronym is a sequence of capital letters (upper case) that is formed from initial letters of the words defined by the acronym. For example, the acronym for International Business Machines is IBM. The acronym can be either pronounced as a full word, e.g. NATO or spelled letter by letter, e.g. I B M.
- A digit is any contiguous sequence of Arabic numerals (0,1,…, 9)

In multi-lingual processing of abbreviations, acronyms and digits, each language has its own way of processing them to ensure that any language peculiarities are taken into account. Nevertheless, each language's behavior is determined purely on the basis of the data. No language-specific code is used in the processing. The handling of acronyms and digits follows the patent application co-authored by the thesis author [108]. The approach to acronym and abbreviation handling was optimized from a voice dialing perspective. The users' tendency to use acronyms like "GSM" was especially considered and included in the method.

3.3.1. Handling Abbreviations

From the beginning of voice dialing technology development, it was obvious that supporting abbreviations in a low memory footprint system will be very challenging, if not impossible. Furthermore, the multi-lingual approach did not help in this matter, either. In fact, abbreviations are language-dependent and can have very user
-specific forms. Users have their own way of abbreviating words in the limited space provided in the device phonebook and there are no commonly used abbreviations for all languages. Hence, for multi-lingual voice dialing system does not support abbreviated terms other than

those that have been explicitly added to the system for the sake of supporting abbreviated terms in the system voice commands. The problem of abbreviated voice commands is discussed in Section 5.6.

3.3.2. Handling Acronyms

The basis for acronym processing uses the following definitions for an acronym.

1. An acronym is a word written fully with capital letters, separated by space from adjacent words, or a single lower or upper case letter optionally followed by a period.
2. A single word written in capital letters is not an acronym unless its length is one or two letters. This takes into account the users' tendency to store names with capital letters only. Acronyms longer than two letters can be either pronounced directly or spelled letter by letter, which makes them difficult to process.

From these basic definitions, acronym processing rules are defined, as given in [108]. It should be noted that some additions have been made in order to cope with certain less familiar languages, such as Thai and Vietnamese. Acronyms are commonly specified for all languages, even for those that do not normally use a Latin character set. Also, special character sets, such as Greek and Cyrillic, have been included in the acronym detection.

As in the case of symbol conversion, the system can only detect a letter to be capital if it has been included in the list of known capital letters. The list includes supported capital letters with their lower case counterpart to enable case conversions. Case conversion is done on all capital letters that are not part of an acronym. Therefore, the pronunciation rules only have lower case symbols included. Acronym detection is the first step in generating the pronunciation for the given text.

3.3.3. Number Handling

A natural number can be converted into spoken format but the conversions are language-dependent and in some languages may be very difficult. The conversion requires, e.g. conversion of number 10 to "ten" and number 123 to "one hundred and twenty three". Most of the languages use thousands and millions to represent large numbers. In Chinese, large numbers are represented with tens of thousands and hundreds of millions. These kinds of differences make natural number conversion difficult.

As natural number detection and processing is complex, the voice dialing UI supports only single digits. This means that a digit sequence 123 should be spoken (in English) as "one two three". This way, digit support was easily added to all languages. Moberg [86] presented a novel method for multi-lingual natural number processing. The task of converting a

sequence of numbers, possibly followed by a measure, such as time or distance was proven to be feasible in multi-lingual environment.

3.4. LANGUAGE IDENTIFICATION FROM TEXT

The next important part of the grapheme to phoneme conversion process is the text based language identification that can help in generating the correct pronunciation for the input words. The author's contribution in this section relates to the development of a novel language-independent language identification method to improve the data packaging that is introduced in Section 4.6. The introduction in this chapter highlights the main techniques used for text based language identification. The target of text-based language identification is to determine the most likely language(s) of the input text.

Why is language identification needed in voice dialing then? It becomes obvious in Section 5.6. The language of the text that should be processed is not always known and within a single device, text material from several languages may exist. Text-based language identification is commonly used for document language recognition in common PC software, such as Microsoft Office. Usually in such applications, there are plenty of words to be used for the decision process. Unfortunately, this is not the case in a voice dialing system, where a common entry in the phonebook contains just a given and a family name. Sometimes, only a nickname is given. This sets an extremely high target for the language identification process. It has to be emphasized that the voice dialing system does not include any online language identification process involved that could estimate the spoken language from the input speech. From now on, language identification refers to text-based language identification only. Fortunately, for voice dialing purposes the language identification does not need to be perfect.

Traditionally, language identification uses N-grams that are matched against the input text [138]. N-gram defines a sequence of *N* symbols that can be associated with a probability or frequency. After data, such as document text, has been processed, the language of the text can be estimated. Generally, the more there is data the better the estimate will be. Tian et al. [112] compared neural networks, decision trees and letter N-grams for text-based language identification.

The language identification module is needed to choose the languages for the pronunciation-modeling module. Usually, the user interface language is used as the default language but it is beneficial to include other pronunciations, too (cf. Section 4.3.1). The languages given by the language identification module is pre-defined based on the languages available in each device. Other languages are not fully supported by the voice dialing UI. To efficiently use the language identification in every product configuration, the method should be independent of applicable languages and should be easily configurable to match the device requirements. Several methods to cope with short input utterance and mismatch in writing scripts, along with multi-lingual speech recognition and speech synthesis requirements have been developed. Such a system can be implemented using an N-gram statistical model. The models are described in the following section together with some other prominent methods

for language identification. After the introduction in the next section, it should be clear why the N-gram method was chosen.

3.4.1. Language Identification

Language identification in the context of multi-lingual voice dialing identifies the language of a written word or sentence. This is necessary due to the multi-lingual nature of the user vocabularies in a mobile device. The author contributed to the development of language identification methods by suggesting a letter N-gram based method that enables flexible language configurations. N-gram decoding is the most commonly applied method for language identification from text [138]. Performance is good when the amount of text is relatively large, on the order of several sentences. The idea is simple and a set of N-grams is trained per language and, with proper weighting of the N-grams, language identification can be obtained with relatively small memory and computational load. As shown by Cavnar et al. [17] language identification can be done quite efficiently and with low memory requirements using N-grams. Instead of using N-gram probabilities, the order of the N-grams was stored. When an unknown language sample was received, the same process was applied and the difference in the ranking result was used to determine the match between the input and the reference languages. The methods proposed were utilized on rather long text messages and therefore the results cannot be directly compared with the methods used in multi-lingual voice dialing system.

Neural networks are known to be efficient and compact in various statistical problems and they have also been tried for language identification. Tian et al. [114] proposed a method using neural networks with a scalable input layer based on the currently supported languages. The neural network was used in such a way that the input layer to the MLP neural network [39] consisted of the letters (or their indices) with appropriate left and right context. The output of the neural network had a node for each language. The output node having the highest probability value was the best match for the input text language. When combining several languages with varying scripts the number of letters in the input layer gets bigger. The proposed approach makes use of language-independent letter transformations to reduce the size of the neural network input layer. In the same approach, an alphabet based scoring method was introduced. Alphabet scoring gives a rudimentary classification of the languages when knowing the alphabet inventory of each language. Obviously, this works extremely well by itself for two languages with separate writing script and alphabet. As an example the alphabet scoring gives a perfect separation between the Cyrillic language and Latin languages. For a Finnish word, there is high probability that the word is not English, due to the Finnish (and German) letter ‘ä’. After this step the neural network score is calculated and the two scores are scaled and combined to obtain the final result for language identification. The result is an ordered list of languages with their probabilities.

Finally, methods that should be most efficient work on word dictionaries to search for full words from the available dictionaries and, based on the dictionary scoring, a decision on the language can be made. The author has been investigating this type of language

identification method and it is likely that it is included in future versions of the language identification module. The drawbacks of this approach are the size of the data package and the increase in processing time for language identification. It should be noted that N-gram based language identification can also utilize full word matching. In this case, longer and variable length N-grams are used and they comprise of full words. All the methods presented suffer from short text samples for identification. In a multi-lingual voice dialing system, the input is commonly a name consisting of a given and family name, often only given name. Therefore, the accuracy of the language identification is not very good as presented by Häkkinen et al. [112]. This is the main reason why multiple pronunciations for each item are applied in multi-lingual voice dialing. More detailed presentation on the multiple pronunciation variant approach is given in Section 4.3.1.

Due to the reasons to be presented in Section 4.6, the language identification method should be such that the data can be isolated into language specific data packages. The neural network based approach presented in earlier in this section requires that all languages supported by a device are known when the neural network is trained. Therefore, it cannot be efficiently used and does not fulfill the requirement for flexible configuration. The author suggested developing a low footprint method that enables flexible language configurations. Consequently, a low footprint language identification module was best implemented using N-gram based methods. The basis for a low memory footprint language identification technique is the proper selection of uni-grams, bi-grams and tri-grams for each language. The N-grams are calculated from a set of training text and the probabilities are stored. Adjusting the performance of the language identification system makes the optimization of the data size possible. Finally, the size of the N-gram package is very small, in the order of a few kilobytes per language. Additional benefit is that language identification can be trained with domain specific data and the resulting data packages for each language are independent of each other. The system is therefore configurable in such a way that varying and optimized language identification data may be used for different applications.

3.5. TEXT MANIPULATION AND SYMBOL CONVERSION

This part of the thesis contains several ideas proposed by the author. There are also several parts that are common development done by the voice dialing development team. The author's contribution is clearly marked, where applicable.

Before text can be processed in pronunciation modeling (see Section 3.6) a certain amount of text processing is needed. In this thesis the process is called symbol conversion and refers to any text manipulation, whether language-independent or language-dependent. The major task of symbol conversion is to ensure that any valid information in the input text is preserved and everything that cannot be transcribed into a pronunciation in the target language is removed. What is valid and what is not will become evident in this section. The types of symbol conversions can be divided into three classes based on the purpose of the conversion:

1. Removal of unknown characters from the input text to ensure the language-dependent pronunciation rules have only valid input.
2. Language-independent Romanization for a non-native word to alleviate its pronunciation modeling in the target language.
3. Make special conversions for the input text to make certain complex pronunciation modeling schemes easier.

The first class of conversions ensures that the pronunciation modeling can be done to valid utterances and that the pronunciation is something that the user expects the system to produce. Therefore, special attention should be paid to the handling of non-native symbols. The second class of conversions is responsible of Romanization and other processing methods that are needed to ensure a proper pronunciation for non-native words. The third class of conversions helps in some languages that require special conversions to improve the efficiency of the pronunciation rules or make the language specific processing possible. All the classes are presented in greater detail in the following sub-sections.

Symbol conversion plays a significant role in the robustness of the system. Robustness here means that the system is capable of processing any input and can produce a voice tag for any given text. The input cannot be limited to allow only common words or symbols, but any type of word, e.g. names in the phonebook, can be used. Secondly, the user may have non-native names in the phonebook that cannot be transcribed correctly with the rules of the default language. The solution in this case is to Romanize the word, if possible, and then generate the pronunciation using the language specific pronunciation modeling methods. The characteristics of some scripts/languages[4] require use of special conversions in order to support the pronunciation modeling methods better. The origin of most writing scripts is from the era before computers and hence the writing script and its link to pronunciation may be very complex. The following sections describe the three text conversion steps used in a multi-lingual voice dialing system.

3.5.1. Step 1: Language-Dependent Symbol Conversions

The first step of the symbol conversion is the application of language-dependent conversion rules. The rule-set is denoted with Rule set 1 in this work. These rules convert one sequence of characters into another sequence of characters. Alternatively, they may translate character sequences that are non-native to the language but still occur often, such as two consecutive 'z' characters for Finnish. These rules only contain exceptional cases and most common native character sequences are handled in the pronunciation rules. If the letter (or character) is not found among Rule set 1, the characters cannot just be omitted but need to be processed with Rule set 2, described in the next section. After Step 1, the text may still include symbols that are unknown to the system. Furthermore, after Step 1, only the sub-strings (usually single

[4] Such as Thai and Vietnamese

letters) that matched the rules may have changed. Since Rule set 1 is also used in Step 3, native letters are also included in Rule set 1. The unconverted part of the text is left intact.

The purpose of Step 1 is to label the known characters so that they are not processed by the language-independent rules in Step 2. In some languages, special conversions can be made already at this stage. The example of Thai in Section 3.5.4 is one such conversion and is done at this stage.

3.5.2. Step 2: Language-Independent Symbol Conversion

The second set of symbol conversion rules, Rule set 2, is applied to the input text in a language-independent manner. This set of rules contains conversions common to all languages. These include Romanizations of characters that are used in the minority of the supported languages, such as 'ä' to 'a' or 'é' to 'e'. In this step, similar to Step 1, only parts that match the rules are changed. All other text remains unchanged until Step 3. It should be noted that in the previous step, all known characters were matched and converted. Therefore, characters that matched in Step 1 are not processed now. Step 2 has all known removals of diacritics and general Romanizations, such as Greek and Russian. After this step, all valid symbols have been identified and all others will be removed in Step 3. Before sending the text to the pronunciation model, one final step is needed.

In the normal use case when the word in question is native, there is no need for language-independent symbol conversion, i.e. Step 2. Only unknown characters, such as commas, periods, slashes, etc are removed. The system is fully data driven and each and every character has to be introduced in the system so that it can be recognized and used. This applies also to the normal ASCII characters. This has the benefit that languages that do not use a Latin based writing script do not implicitly need to support ASCII. These languages include e.g. Arabic, Hebrew, Thai and Chinese.

3.5.3. Step 3: Reapply Language-Dependent Symbol Conversion

Language-dependent rules, Rule set 1, need to be applied once more since language-independent rules may have introduced symbols that are not known to the currently processed language. After step 3, it is guaranteed that for the current language there are no unknown symbols in the text and that pronunciation rules can be safely applied. Characters that are not known to the pronunciation model affect the performance and should therefore be removed. Note that after all conversion steps, each language variant to be generated may have a different string to be processed in the pronunciation model. This is depicted in Table 3.

Table 3: Examples of symbol conversion result in three different languages

Language \ word	Hääkakku	Pizza	Håkan
Finnish	hääkakku	pitsa	hokan
English	haakakku	pizza	hakan
Swedish	hääkakku	pizza	Håkan
Chinese	NULL	NULL	NULL

As it can be seen in Table 3, the Chinese conversion rule set does not contain the ASCII letters and therefore the text is empty (denoted with NULL) after symbol conversion.

3.5.4. Text Processing Examples

The following examples in Table 4 show what happens to the example strings in three languages when the rule steps are applied.

Table 4: Examples of symbol conversions

Operation	English pronunciation rules	Finnish pronunciation rules	French pronunciation rules
After acronym detection	Jack / Jill	Pizza %	Börje
After step 1	**Jack** / **Jill**	Piza %	**B**örje
After step 2	**Jack** / **Jill**	Piza %	**Borje**
After step 3	**Jack Jill**	**Pitsa**	**Borje**

The examples above are such that the word "Jack / Jill" is otherwise valid but the slash '/' character needs to be removed since it is not defined in the English conversion rules. For the Finnish case, the double 'zz' sequence is first converted into a single 'z' and then in step 3 the z is converted into letter sequence 'ts'. Finally, for the French example, the unknown letter 'ö' is transformed by the language-independent rules into 'o' and then the final step confirms that this is a valid letter in French and preserves it in the text string.

Romanization in the context of grapheme to phoneme conversion means the conversion of the native form of a word into another form that can be processed with pronunciation rules of a language whose writing script does not match the writing script of the original word. Two examples to follow:

1. When the Russian word Анастасия is given to the system, it needs to be pronounced by a native Swedish who knows some Russian. The original word Анастасия is Romanized to Anastasiya and the pronunciation is generated with the Swedish pronunciation rules.

2. Finnish word Päivi is given to the system. The word needs to be pronounced by native English. The original word is Romanized to Paivi and the English pronunciation rules are used to generate the final pronunciation for the word.

In some cases, the first Romanization step does not generate a proper set of characters for the target language. In these cases, the so-called native Romanization rules are used to map the unknown characters to the native character set. An example from a Russian word pronounced in Finnish: the Russian name Александра is Romanized to the word Alexandra. Character 'x' is, however, non-native to Finnish alphabet and needs some further processing. The Finnish native Romanization rules say that character 'x' needs to be converted into the 2-character sequence 'ks' to be aligned with the Finnish alphabet. Finally, the Finnish pronunciation rules can be applied to the word Aleksandra to obtain the Finnish pronunciation. The examples with non-native scripts can be slightly difficult to justify since not too many Finnish speakers can even read the Cyrillic alphabet let alone pronounce the words.

Example language-dependent conversions for Finnish are shown in Table 5. For Finnish, the normal ASCII range is preserved as well as the native special characters 'ä', 'å' and 'ö'. Furthermore, some accented characters are preserved to be able to obtain more accurate pronunciation modeling for names originating from Swedish.

Table 5: Examples of Finnish language-dependent symbol conversion rules

Convert from	Convert to	Notes
0x0061-0x007a	0x0061-0x007	ASCII characters [a-z] are preserved
á	á	Some non-native accent characters are processed in rules
ö	ö	Native Finnish letter
ä	ä	Native Finnish letter
å	å	Native Finnish-Swedish letter

Thai pronunciation has some peculiarities that must be taken into account when designing text conversions and the pronunciation modeling scheme. The pronunciation rules are normally straightforward in Thai but there are a few exceptions that need special handling. A special phenomenon for Thai is the change in order of a vowel and consonant pair in certain contexts. The Thai pronunciation model is implemented as a decision tree (cf. Section 3.6.3) and it cannot cope very well with this phenomenon. Therefore the text conversion does the re-ordering when needed. The author suggested performing the character reordering in the text processing module preceding the decision tree decoder of the pronunciation modeling module. The following examples show some of the Thai special text processing rules.

Table 6: Examples of Thai language-dependent symbol conversion rules

Convert from	Convert to	Notes
แก	กแ	Thai special collation, change of order of phonemes
โง	งโ	Thai special collation, change of order of phonemes
ฑ	ฑ	Normal Thai letter
ฒ	ฒ	Normal Thai letter
ณ	ณ	Normal Thai letter

It should be noted that there is no support for Latin letters in the Thai pronunciation method.

Another example is the use of combining characters in Vietnamese. Vietnamese uses an extended Latin type of writing script [118] and the storage of the writing commonly utilizes the combining characters specified in the Unicode standard. A combining mark is such that it is following the main character and commonly adds a diacritic to the main characters. The Finnish letter 'ä' can be written in Unicode with combining marks as follows.

ä = a + ¨

In a similar manner, any number of combining marks can be added after a character. In Vietnamese, there are two kinds of combining marks used. One set used for changing vowel type and another for marking the tone of the vowel. A character having both types of diacritics can be written in three different ways.

Representation 1. The whole character as a single Unicode character
Representation 2. The vowel plus modifier as a single Unicode character followed by the combining mark for the appropriate tone
Representation 3. The vowel as a single Unicode character followed by two combining marks, one for the vowel modifier and one for the tone mark

This set requires processing text in such a way that the text can be input to the system in any of these formats. Nokia devices store the Vietnamese text according to Representation 2. The reason for this storage type is that a user can delete the tone mark alone from the vowel without deleting the vowel. This is useful if the user has mistakenly typed in a incorrect tone mark. In a multi-lingual voice dialing system, representations 1 and 3 are converted to representation 2. This way, only a single set of pronunciation rules need to be supported for Vietnamese. The author proposed to use this approach and to add the support for all combining markers for Vietnamese. In other languages, combining characters are possible but in most cases, the mobile device does the conversion to single characters when the data is inputted to the device. Therefore, e.g. Finnish pronunciation rules do not need support for

combining characters. Since the Vietnamese alphabet contains the full ASCII range, it does not have Romanization rules for non-native words. The non-native words are particularly difficult for Vietnamese, since they resemble the native writing.

3.6. METHODS FOR GRAPHEME TO PHONEME CONVERSION

Now that the text string has been prepared for pronunciation modeling, the appropriate pronunciation methods can be applied to generate the pronunciation of the text in the given language. This is the pronunciation that will be used both in speech recognition and speech synthesis. At this point of the processing the voice dialing system may have requested several pronunciations corresponding to multiple languages. The text input to each language pronunciation modeling contains only allowed characters for that language.

The grapheme is the basic unit of a written language. Latin based languages, such as English and Finnish, use a common ASCII alphabet with extensions for written text. Several other languages use characters that do not so clearly represent a single letter or grapheme. These more complicated languages include Chinese, Thai, Arabic, etc.

The purpose of the grapheme to phoneme (G2P) system is to convert a sequence of graphemes into a sequence of sound units that correspond to the spoken representation in the given language. This section outlines the methods that are used for the conversion and gives the arguments for the choice of G2P method used for certain languages. The G2P conversion methods are divided here into four classes and they are presented in the following sections.

3.6.1. Lookup Tables and Dictionaries

The most intuitive type of grapheme to phoneme conversion method is the lookup table. A lookup table, or dictionary, contains words and/or sentences and their corresponding pronunciations. Lookup tables are fast and accurate. Furthermore, if the lookup table entries are ordered alphabetically, the search can be done in $O(\lg N)$ time using a binary search. Lookup tables can be applied efficiently to languages that have few inflections. An inflection is a term in language morphology that describes the modification of a word according to the language grammar, such as gender, number, tense or person. Lookup tables commonly match full words and must be limited by word-boundary characters.

The drawback of lookup tables is clear with inflectional languages: there needs to be a large amount of rules to match the inflections. Such languages, e.g. Finnish and Turkish, can be handled more efficiently with pronunciation rules. Due to the limited amount of space on the mobile device display, many of the display strings need to be abbreviated to fit in the screen. The abbreviated terms may be used also for voice tags. Abbreviations are very difficult to handle and if the abbreviation itself is not included in the system dictionary, the pronunciation of the voice tag will not be what is expected by the user.

Lookup tables may be compressed with conventional text compression methods. This improves the storage requirements for possibly large dictionaries but at the same time slows down the search and may even require more dynamic memory while processing the

dictionary. A novel compression method was proposed by Tian [115][116] that interleaves the word with the pronunciation to enhance the compression ratio. For languages that do not have word boundaries, lookup tables can be used once the word segmentation has been obtained.

3.6.2. Rules

Rules can be applied to languages that have a certain level of consistency between the written and spoken form. For such languages, e.g. Finnish, Italian, Romanian and Estonian, a compact set of rules can be developed.

The set of rules should cover all possible native and non-native text that is given to the rule processor. The reason for this is the multiple pronunciation support and the fact that the language of the given word cannot be unambiguously decided. Even though symbol conversions (cf. Section 3.5) guarantee that there are no unknown characters left in the text, the rules should also provide some non-native text processing, if possible. The most important rules are those that cover native words. An augmented set of rules should be prepared to handle non-native words to reflect the most probable pronunciation for them. The pronunciation rule processor of multi-lingual voice dialing utilizes a special marker to denote the start/end of the word. This can be used in languages, where the pronunciation of a certain character sequence is different depending on the location of the sequence in the word, such as Finnish. As an example, part of the Finnish pronunciation rules is presented in Table 7. First, a part of the rules to process native Finnish text is shown followed by the rules that have been developed to handle the letter 'c'. The letter 'c' is not native to Finnish and can have a pronunciation similar to letter 'k' or 's', depending on the context.

Table 7: An excerpt of the Finnish pronuciation rules

Rule	Pronunciation	Example word
aa	a:	saapas
sjö	s ö:	sjöberg
@å	o:	Åland
ci	s i	city
z	ts	Lazio
prepaid	p r i: p e i d	prepaid
zoom	ts u: m	zoom
c	k	Carl

As can be seen from the native rules of Finnish in Table 7, the rules are very regular and most often have a one-to-one correspondence between the written text and the phoneme output. The basic rule for letter 'c' converts it to a /k/ phoneme. In addition to the basic rule, there are tens of exception rules to handle the two possible pronunciations (/k/ and /s/) for letter 'c'. The

storage space required for the Finnish pronunciation rules is on the order of 1.5 kB. For most languages the pronunciation rules may occupy 1.4 kB to 30 kB. To speed up the rule processing, the rule set for each individual language is stored in sorted order to alleviate the use of binary search in the matching algorithm.

The rules that are defined should be compact and comprehensive so that all the required cases are processed correctly. In addition to this, in case of unseen data, the rules should provide the best possible approximation of the pronunciation. For certain rule-based languages the storage space requirement is much higher and advanced storage and processing methods are required. This is explained next.

Chinese Pronunciation Rules

For Chinese spoken languages, e.g. Mandarin and Cantonese, the number of graphemes is large compared to other languages. Mobile phones in China currently support more than 20,000 different characters. In Hong Kong, there is an additional set of 4,500 characters in use that are not covered by the basic set. Furthermore, these additional characters are spread over the Unicode Basic Plane and therefore cannot utilize the following advanced storage system presented for the normal Unicode range of Chinese characters. The processing and storage of Chinese rules are discussed next.

The large amount of characters sets high requirements for the storage of the rules. Furthermore, several characters have multiple pronunciations, depending on the context and meaning. If these additional variations are to be taken into account, rules soon become less powerful and dictionary based approaches should be used to obtain better accuracy in G2P. An additional challenge in the Chinese languages is that there are no word breaks in the written text so the word separators need to be estimated, normally with statistical methods such as the one presented by Sproat et al. [105]. This type of processing requires so much memory that it is not possible in the context of multi-lingual voice dialing. Nevertheless, the voice dialing system contains a certain set of multi-syllable rules that cover the most common multi-pronunciation characters.

Chinese languages have another interesting feature that relates to the pronunciations of characters. Even though the number of characters (estimates vary from 20,000 to 60,000) is large, there is a significant amount of common pronunciations. If the change of intonation (lexical tone) is not taken into account, there are only about 400 different pronunciations (syllables) for all characters. If the tone is taken into account, the number of pronunciations increases to about 1,400. This means that many characters have exactly the same pronunciation and the confusability of the spoken language is high. Native speakers sometimes have problems in understanding each other if the context of the discussion is not clear. The knowledge that several characters share the same pronunciation can be used to reduce the memory storage requirement of the rule set.

Due to memory constraints of the voice dialing system, most Chinese pronunciation rules are mono-syllabic, which means that each character can only have a single pronunciation. The most frequent pronunciation is used in the storage optimized rules. The Chinese voice commands that are pre-defined in the device have manually prepared

transcriptions and they contain several cases where mono-syllabic rules would not result in the correct pronunciation. It is a known fact that for Chinese names, even the most complex rules make errors since names do not bear the context that could help in finding the correct pronunciation for each character. Last names are more problematic than first names; Chinese last names commonly have one single syllable while given names have either one or two syllables. For the two syllable given names, a context exists and it can be used in pronunciation modeling.

Chinese pronunciation rules can be efficiently stored in three parts: The first part is the basic character set of about 21,000 characters[5] that covers a contiguous sequence of Unicode points, from code point 0x4E00 to 0x9FBB in hexadecimal notation. The second part is the exceptional cases mentioned above that are stored separately as a look-up table. The third additional part is the Hong Kong special character set that is distributed very sparsely over the Unicode range and cannot be stored efficiently. The storage of the Basic Unicode set would take quite a lot of memory if stored in a conventional manner. Each character occupies two bytes in UTF-16, each pronunciation is about 4 phonemes (including the tone for TTS) and each phoneme can be indexed with a single byte. This results in 20,924*6 bytes, which equals to more than 120 kB. This is clearly too much for a single language pronunciation rules and it does not even contain the special cases or the Hong Kong special character set. The solution is to use the index of the character for the lookup and this way the characters do not need to be stored at all. The author's contribution was the original implementation of the idea together with finalizing the applicable character ranges. The author also designed the search algorithm for grapheme to phoneme conversion with optimized memory and CPU resources. The novel and efficient storage is shown in Figure 9. As can be seen in Figure 9, the sharing of items is increased when moving from left to right in the picture. Also the length of the data vector gets shorter when moving from left to right. For illustrational purposes the picture does not reflect the real content of the data. In the scheme of Figure 9, the left-most array is not stored at all and hence 42kB is saved. Furthermore, each pronunciation is indexed separately (recall that, there are only 1400 different pronunciations). This way, the pronunciation of each character is replaced with a 2-byte index pointer. Pronunciations occupy about 1,400*4 bytes which equals to approximately 6kB. Hence, the overall storage space needed for the Basic Unicode range is 20,924*2 bytes + 6kB equals to 48kB. This is a 60% saving in storage compared to the conventional storage type. Although the savings does not sound large but in a device where multiple applications should be included in the area of some tens of megabytes, this additional saving makes a difference. The Chinese product variants commonly use more ROM memory, e.g. for storing the more complex font and therefore should utilize storage saving techniques if possible.

[5] Exact number in this range is 20,924

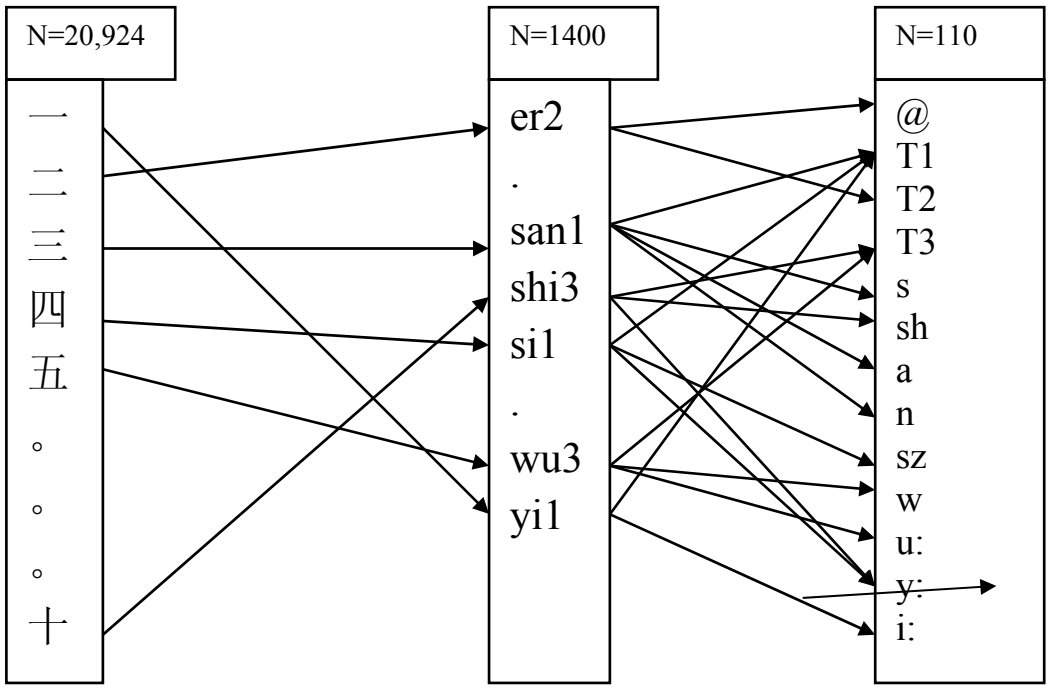

Figure 9: Example of efficient storage for Chinese pronunciation rules

When searching for the pronunciation with the Basic Unicode set, the pronunciation for each character can be found easily since the index (the Unicode point) of the character immediately tells the pronunciation and no further search is needed. Note, that to be able to use these rules, the range of Unicode points needs to be mostly contiguous. Empty slots waste space of two bytes per empty slot. It should be noted that the Hong Kong special character set requires extra storage space since the Unicode code points have been split over the basic Unicode plane. Therefore, the method described above cannot be applied and the normal rule storage methods should be utilized.

3.6.3. Decision Trees

A decision tree is commonly used in data classification and analysis tasks in data mining and statistical analysis. A decision tree can be used to store a statistical model of a cause and consequence relation between observed data and the associated output. In a decision tree, the observations are stored in the tree nodes and the output values are in the leaves of the tree. In grapheme-to-phoneme conversion, the nodes of the tree are filled with context information from the training data observations while the leaves of the tree represent each associated phoneme of the training data [106]. An advantage of a decision tree is that the size of the tree can be optimized to the given training data by so called pruning. Suontausta and Tian [107] showed that pruning of a decision tree can be done efficiently to reduce the storage space requirement for the English pronunciation model. A decision tree can perform well when the language has irregular pronunciations with respect to the written format and/or the amount of exceptions is high. It also requires a representative amount of training data to generalize the decision tree for any unseen observations. In the multi-lingual voice dialing system, decision

trees are used in languages such as English and Thai. For Thai, special pre-processing may be necessary to enable efficient decision tree processing, such as the one presented by Ding et al. [27].

3.6.4. Neural Networks

Neural networks have been successfully applied to several fields of science, including speech processing. Approaches to acoustic modeling and feature classification are just a few examples. Neural networks have been applied to pronunciation modeling, too. Neural network commonly refers to a multi-layer Perceptron based system where the input layer contains the graphemes to be processed and the output is the phonetic match to the input text. Jensen and Riis used a neural network to carry out pronunciation modeling for speaker-independent multi-lingual voice dialing and the results showed the feasibility of neural network for this task [60].

Chapter 4

Multi-Lingual Speaker-Independent Voice Dialing

THE multi-lingual speaker-independent voice dialing system utilizes a combination of methods from various disciplines that were introduced in the previous chapters. This chapter concentrates on the overall speech recognition and speech synthesis system design. It also describes the combination of various modules to complete a functional system. In addition to this, detailed implementation and system related methods are presented in this chapter in the form of Publications. The author's contribution to this chapter is significant. The parts where the contribution is less significant are clearly indicated. While this chapter concentrates on the underlying technical issues, Chapter 5 continues with the voice user interface design methodologies and the introduction to the Series 60 voice dialing user interface.

In the beginning of this chapter, Section 4.1 outlines the basic building blocks of the multi-lingual voice dialing system. This gives the reader the link between the processing modules and the data that is utilized in each module. Section 4.2 continues by showing why multi-lingual phonetics is essential in controlling the complexity of the voice dialing system. Section 4.3 focuses on the main features of the speech recognition engine that enable the recognizer to handle multiple languages at the same time in a single device. Section 4.4 concentrates on the memory reduction methods found in the multi-lingual speech recognizer. The author’s contribution to multi-lingual speech synthesis is detailed in Section 4.5. The section presents the rapid speech synthesis language development with language morphing. Concluding Chapter 4, voice UI data is presented in Section 4.6. As presented earlier, code and data are separated in the multi-lingual voice dialing system and therefore the data makes the system behave as specified in each language. This applies equally to text processing, speech recognition and speech synthesis.

4.1. MULTI-LINGUAL SPEECH RECOGNITION AND SYNTHESIS FRAMEWORK

The processing modules of a multi-lingual voice dialing system can be seen in Figure 10. Each module has a set of input and output nodes to denote the data processing involved in each module. The multi-lingual voice dialing engine can be divided into four separate functional entities that execute independent tasks which are linked to each other.

The phonetic inventory defines the acoustic model (HMM) parameters of the supported languages. In addition, the phoneme definition of each language is provided for the pronunciation modeling block. The third type of phonetic data is utilized by the speech synthesizer. The pronunciation modeling module takes in the pronunciation modeling methods and text to produce a phoneme sequence. Figure 10 omits the text processing part of pronunciation modeling for the sake of simplicity. It is nevertheless included in the pronunciation modeling module. The phoneme sequence is in turn given as input to the speech recognizer or the speech synthesis engine. The speech recognition engine takes the phoneme sequences, the acoustic model and the user's speech as input and produces a list of results to be shown to the user. The last module, speech synthesizer, takes the phoneme definitions and the phoneme sequence as input and generates an audio signal matching the phoneme sequence.

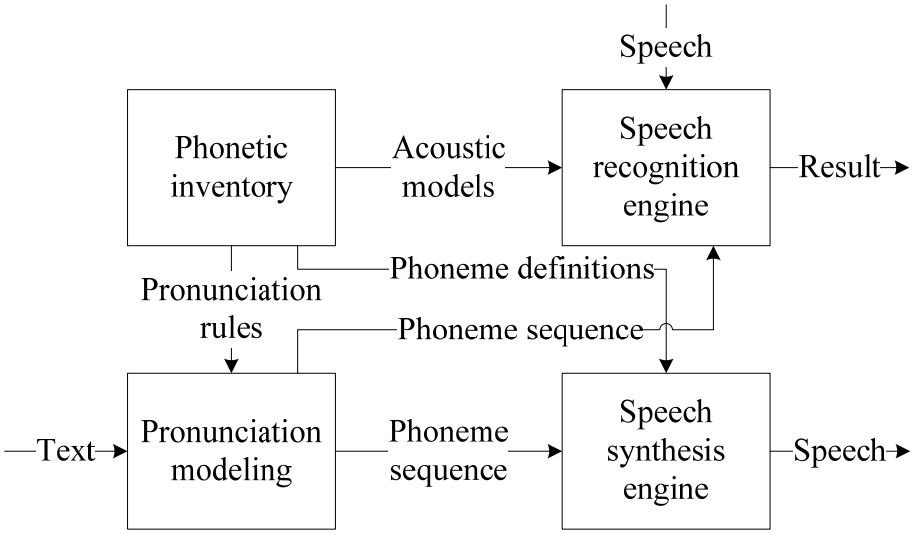

Figure 10: Interfaces between the modules of the multi-lingual voice dialing system

One of the key requirements for the multi-lingual system is the independence of the SW modules to every supported language. This means that the SW modules themselves perform the processing based on the data transmitted between the modules resulting in the desired functionality. The novelty of this system is that data alone specifies how the system functions for each supported language. This makes language-specific updates simple since no code needs to be updated when introducing support for a new language. The Nokia Series 60 devices are shipped in almost 50 different languages and clearly changes to one of the

languages cannot trigger a code change. The author's contribution is significant in making sure each module is implemented according to this principle and that all languages can be supported with a single code base. The author's expertise in Asian languages, especially Mandarin Chinese, made it easier to build support for Asian languages.

4.2. MULTI-LINGUAL ACOUSTIC MODELING IN VOICE DIALING SYSTEM

The International Phonetic Alphabet, IPA, is the basis for combining phonemes of several languages together. IPA was introduced in Section 2.2.1. In many cases, when two sounds are separated in IPA, they are nevertheless so close to each other that in a voice dialing system it does not really require the distinction. The target of the phonetic definition of multi-lingual voice dialing system is to minimize the overall number of sound units. As a rule of thumb, one should combine as many sounds together as possible to reduce the overall amount of sound units in the system. This will have the following four consequences.

1. The overall amount of sound units is reduced hence making the acoustic models to fit in a smaller memory space.
2. The amount of training data per sound unit is increased, which in turn should provide more robust and more accurate acoustic models.
3. Acoustic models can be re-used for languages that have no training data available.
4. An error is introduced to the combined phonetic inventory and the larger variance of the shared acoustic models can cause recognition errors.

Furthermore, the combination of several sound units within a single language can cause reduction in speech synthesis quality due to inaccurate phonetic inventory. This can be heard as pronunciation errors in the synthesis output. Therefore, one should be careful when combining sounds of a single language to reduce the size of the mono-lingual phonetic inventory.

Another rule of thumb is that speech recognition is more robust to combination of sound units than speech synthesis. Subjective user evaluation is more important and relevant in the case of speech synthesis as opposed to speech recognition. Even minor deviations in pronunciation of a speech synthesizer are easily noticed by the user. On the other hand, if the recognition result is correct, the user does not care if the phonetic definition is accurate or not.

In most of the languages supported by the voice dialing system, native phoneticians have checked and confirmed the chosen phonetic inventory for each language. In several cases, fine-tuning is applied to reduce the overall number of sound units to make the system compact. Fine-tuning is made on the condition that the system performance does not degrade. The author has no expertise in phonetics and phonology and therefore the contribution in this part is minimal.

How is the final phonetic inventory set for each language obtained then? First of all, the accuracy of the phonetic inventory of each language has to be designed according to the requirements of speech synthesis since the user tolerance for pronunciation error is small. This is the first condition and all the subsequent optimization and combination has to be done keeping this in mind. Next, each language is processed separately to understand which sound units are necessary for that language. This stems from the previous step to maintain the TTS pronunciation accuracy. Finally, with the help of IPA and acoustic model training combined with recognition experiments, a compact set of sound units is selected that is large enough to maintain the accuracy within a language and compact enough to limit the total amount of sound units for a set of multiple languages.

In the current version of the voice dialing system, the following separate phonetic inventories have been defined.

1. Europe
2. Middle East
3. India
4. Asia and Pacific
5. The Americas[6]

The phonetic inventories are overlapping in most of the regions but the acoustic model training data for each area mostly consists of area-specific data. In addition to the major geographical areas, a certain level of fragmentation is used within each region. This is especially true for Asia and Pacific area, which is further divided into smaller groups of languages, where a more detailed acoustic modeling is applied. On the contrary, a single acoustic model for Europe is used in all 30 supported European languages.

In the multi-lingual voice dialing system, the design goal for speech recognition engine was to make it as language-independent as possible. To reach this goal, each acoustic model is processed in a similar manner and has the same internal HMM structure. The speech recognition engine takes as input the acoustic model sequences corresponding to the vocabulary items. The engine constructs a recognition network and matches the input speech against the network. The maximum amount of different acoustic models active at any given time has to be limited to make the decoder faster and use less memory. Garcia et al. [26] report successful combination of two to three languages with a context-dependent modeling scheme. They, however, fail to maintain the recognition accuracy when introducing more languages in the system.

The Multi-lingual Phonetic Alphabet (MLPA) was designed so that it is easy to use in a speech recognition engine. The combination of sound units of several languages was already discussed in the previous chapters. The motivation was to reduce the overall amount of

[6] The Americas refers to North, Central and South America

acoustic models in the system to limit both memory and computation. The defined combinations have been verified with testing data covering all supported languages. One novel method introduced in the recognition engine is the capability to replace given multi-lingual acoustic model with a language-specific model. Although many acoustic models are shared between several languages, there are cases when a language-specific acoustic model is needed to replace the multi-lingual model. The recognition engine can make a run-time selection to replace the multi-lingual acoustic model if a language specific acoustic model is included in the acoustic model set. This processing step is done when the new acoustic model sequence is introduced to the recognition engine.

4.3. MULTI-LINGUAL SPEECH RECOGNITION FOR VOICE DIALING

This section introduces several methods to improve the speech recognition framework. Also two studies on tone recognition are presented. These were important factors defining the final features of the speech recognition system. Most of them are based on the author's Publications. The author's contribution is highlighted before discussing each method. This section also introduces other main features found in the multi-lingual voice dialing system.

4.3.1. Multiple Pronunciation Variants in the Recognizer

When a speech recognition system is developed for a single language, it has the following limitations. First, it can be only used by native users speaking that language. Second, the system cannot handle non-native names efficiently and in the worst case cannot provide any voice tag or spoken output for such words. To overcome the former problem, a multi-lingual voice dialing system should handle users speaking multiple languages and vocabularies having multi-lingual content.

Many users have contacts from foreign countries, i.e. non-native names. This leads to a double problem in the pronunciation modeling part of the multi-lingual voice dialing system. What is the language of the word and how should it be transcribed to generate a voice tag? How is the user of the device going to pronounce the word? Is he capable of pronouncing the word in a non-native manner or is he going to speak the word as a native word in his own language?

The multi-lingual voice dialing system utilizes multiple pronunciation variants to cope with the above-mentioned challenges. The first target of multiple pronunciation variants is to provide several alternative pronunciations to each voice tag in order to have the user's pronunciation available. The fact is that text based language identification is challenging and the uncertainty in a user's pronunciation make a single pronunciation system perform significantly worse.

The second target of multiple pronunciation variants is to provide a native pronunciation for a non-native word to the user. In many cases, the user tends to pronounce the non-native word incorrectly and closer to what the native pronunciation rules and

phonetics are. This way, the user gets a better recognition accuracy and improved experience when using a system with multiple pronunciation variants. The multiple pronunciation variants approach used in the multi-lingual voice dialing system was introduced by Tian et al [113]. The results shown prove the feasibility and need for the multi-lingual pronunciation modeling and multiple pronunciation variants.

In addition to variation in the pronunciation, several languages include a lexical tone. This means that the change in intonation when saying a word, changes the meaning of the word. Such languages may need a special tone recognition capability to alleviate a voice dialing system for such languages. Tonal languages, such as Mandarin Chinese, Thai and Vietnamese, pose additional challenges to the development of multi-lingual voice dialing system. Two considerations for tone recognition are presented in this thesis. First, the author suggested using a specially designed tone recognition system [128]. The technique can be applied to multi-lingual speech recognition engine with minor changes. The novelty of the approach is the combination of acoustic models and tone models to distinguish between two vocabulary items that differ only by the tone. Second, a study on the characteristics of Chinese person names and the need for tonal recognition in Mandarin Chinese is presented [16]. The work included a novel method of estimating recognition accuracy without testing data to estimate the effect of tone to the overall recognition accuracy. The conclusions made in the study helped to decide on the need for tone recognition for tonal languages.

4.3.2. Handling Tonal Variability in Speech Recognition

To develop a speaker-independent speech recognition system for Mandarin Chinese, the main differences between Mandarin Chinese and western languages should be identified. The most important differences are that Mandarin Chinese is a tonal language and that words are relatively short. The fact that different tones of a Chinese syllable will result in totally different meaning makes tone recognition very important in Chinese speech recognition. Tone recognition research is therefore also a good reference to other tonal languages, such as Thai and Cantonese.

It is known that the tone information is related to the pitch contour. Experimental results indicate, however, that it is very difficult to extract the correct pitch information, especially in noisy environments [127]. At the same time, the Higher Order Cepstral (HOC) [129] features can significantly improve the tone recognition accuracy. Experimental results by Wang et al. [129] show that HOC features improve tone recognition accuracy in a Mandarin Chinese speech recognition system, but do not provide similar improvement for a Western language speech recognition system.

Two variants of the proposed method for Chinese tone recognition are compared with a baseline system. The baseline system for Mandarin Chinese utilizes syllable Initials and Finals for modeling Mandarin syllables. Due to the large number of models and tonal variants, this system has the highest complexity both in terms of computational complexity and memory consumption. Two schemes are proposed for Mandarin isolated word tone recognition to reduce the complexity while retaining the high performance. In the first

method, the final part of syllable is split into phoneme models and tone is modeled on the main vowel of the syllable. The second scheme uses so called lexical tone model, resulting in total of four tone models, hence significantly reducing the system complexity. The second proposed scheme has also clearly less complex model structure.

Speaker-Independent Speech Recognizer with Tone Recognition

The multi-lingual voice dialing system is based on sub-word level acoustic models. For Chinese speech recognition systems, acoustic models are not the most common selection for the sub-word units. The most commonly used acoustic modeling units in Chinese speech recognition are syllables or Initial/Final pairs. The syllable scheme is expensive for an embedded system since, for Mandarin, there are over 400 base syllables and over 1,300 syllables with tone. In the baseline system, the Initial/Final approach was utilized, since it provided a good performance with a reasonable amount of models. The HOC coefficients were used for modeling the tonal information.

In the baseline system, the tone classification capability was enabled in the acoustic modeling phase. A left-to-right HMM without state transition matrix was trained for each selected model unit. Three states were allocated to Initial models and six states for Final models. This is due to the relative length of the initial and final part. The finals were modeled with tone association, which was called Toneme by Chen et al. [19]. The baseline system is called Elephant in the following series of tone recognition experiments. Elephant obtained a good recognition performance of 87.9% on a tone confusable name database. The Elephant system included totally 174 models, which resulted in high memory consumption and high computational complexity compared to a non tonal recognizer.

Generally, each syllable initial contains only one phone, but each syllable final can be composed of several phones. For example, the syllable final 'ian' is a combination of phonemes /i/, /ε/ and /n/ using IPA notation. Apparently, there is a great deal of overlap among the syllable finals. In order to reduce the overlap and, at the same time, to reduce the number of models, a more compact system was developed based on phoneme level modeling. Mono-phone was selected as the basic modeling unit since bi- or tri-phone units are not memory-efficient enough to be a practical solution in embedded systems.

In a mono-phone model based system, the biggest problem is to solve how to model the tone. As stated previously, tone is associated with the final part of each syllable. Considering phone-level tone recognition, which phone should be chosen to carry the tone information? How to decide the alignment between phones and the tone? In this research, a concept of main-vowel as the critical part to identify the tone information of a syllable is proposed.

By taking a closer look at a Chinese syllable, it can be divided into smaller segments: the pre-tonal part, the tone nucleus, and the coda part, as observed and defined by Zhang et al. [137]. The pre-tonal part spans from the syllable initial and the gliding period from the syllable initial to the nucleus of syllable final. The tone nucleus part carries the syllabic tone information, spanning the major portion of the syllable final. The coda part indicates the coda of the syllable final, and it is less important for tone perception of the syllable. In the

experiments, the main vowel, i.e. the middle vowel of a syllable final is treated as the tone nucleus. This scheme is called Tiger. In Tiger, the number of acoustic models was decreased by 27% to 127 by utilizing phone level modeling, compared to Elephant.

The number of models can be further reduced with the Monkey scheme, which includes 47 mono-phone acoustic models and 4 separate tone models for tone recognition. The four tone models were trained from all the syllable finals with the same tone mark in the database, using the Higher Order Cepstral (HOC) parameters. Therefore, unlike the Tiger scheme, there is no information lost by assuming the location of the tone information, as in Tiger. Not only the number of models, but also the computation in decoding was decreased dramatically, because the 47 acoustic models of the Monkey scheme have only 39 components. The experimental results show that Monkey is compact, cheap and viable for speech recognition of tonal languages.

Complexity Considerations

Recognition accuracy is the most common method for evaluating the performance of a speech recognition system. In the case of embedded devices, the complexity of the chosen algorithm is equally important. The baseline system and the proposed methods are analyzed with respect to their complexity, both in terms of memory and computational complexity.

The baseline system is the toneme based recognizer called Elephant, which has separate syllable initial and final models so that the syllable final models include tone information. The differences between the baseline and the proposed systems are described in more detail in Table 8. The computational complexity of the system is dependent on the size of the vocabulary and the number of densities in the models. Assuming the vocabulary size to be the same in all three systems, the complexity difference is dependent only on the number and size of densities in use, i.e. the speed of computing observation probabilities for the HMMs. The memory figures given in Table 8 are not absolute but indicate a relative difference between the methods.

A training database of 100 speakers and an evaluation database with 19 speakers were used in the speaker-independent tone recognition experiments. The training corpus contained speech from 50 male and 50 female speakers. The database was recorded in a quiet office environment with an 8 kHz sampling rate. The total number of training utterances was more than 100,000.

Pitch99 is the database designed for the verification of pitch estimators and tone classifiers. It consists of 64 pairs of easily confusable Chinese names, uttered by ten female and nine male speakers, in which 3-syllabic names (long names) and 2-syllabic names (short names) are equally present. Each name was repeated four times. The two names within a pair are identical if tones are ignored. Here are two examples, "fu4 zuo4 yi4" vs. "fu2 zuo4 yi4", and "bao1 yun2" vs. "bao4 yun2", where number indicates the tone as given by Table 1. Noisy test data was obtained by adding car, cafeteria or music noise to clean utterances. The signal-to-noise (SNR) ratio of the noisy database was uniformly distributed between 5 dB and 20 dB.

Table 8: Complexity comparison of Elephant, Tiger and Monkey

System	Elephant (baseline)	Tiger	Monkey
Number of models	174	127	51
Tone models	152	80	4
Tone model FV	54	54	54
Phoneme model FV	54	54	39
States per model	initial 3, final 6	all 3	phoneme 3, tone 6
Overall states	978	381	165
Overall memory	52.8k	20.6k	6.8k
Complexity % to baseline	100%	39%	13%

Database and System Description

Two front-ends were used in Elephant, Tiger, and Monkey, namely standard Mel-Frequency Cepstral Coefficients (MFCCs) and Higher Order Cepstral Coefficients (HOC coefficients). The former resulted in 39 Cepstral coefficients, i.e. 12 static coefficients plus energy and their 1st and 2nd order derivatives. The latter resulted in 54 Cepstral coefficients consisting of the 17 Cepstrum coefficients plus energy and the 1st and 2nd order derivatives. Feature vector normalization [124] was applied to both feature sets. HOC coefficients contain the fine harmonic structure characteristic of the glottal driving source [4] and therefore, they can be applied to the tone recognition task. Many researchers prefer using pitch or voicing features for tone modeling [19][111][117], and they do get performance gains in clean environments. These features are, however, vulnerable to ambient background noise. Furthermore, pitch and voicing features are only valid for voiced speech. For unvoiced speech, some extra processing, such as continuation, is needed to obtain a continuous feature stream [19]. On the other hand, HOC coefficients are easy to calculate, easy to utilize, and most importantly, they work well for both speaker-dependent and speaker-independent systems.

A left-to-right HMM with a diagonal covariance matrix was trained for each model unit and tone models. The number of states was set to three for the syllable initial and phoneme models, six for the syllable final and for the lexical tone models. The Viterbi algorithm was used for decoding [125].

Tone Recognition Performance

The baseline speaker-independent system is based on the syllable initial and final models, as described in Section 0 and is named Elephant. To reduce the phonetic overlap between the syllable final models, Tiger is established based on mono-phone models and tone modeling in

the main vowel of the syllable. The HOC front-end dramatically outperformed the standard MFCC front-end in both schemes. However, if tone is ignored, i.e. only the baseforms are considered, there is no clear difference between the two front-ends. The results in Table 9 clearly show that HOC parameters are effective in tone recognition. Baseform in Table 9 refers to the case when tone is ignored and FV is the feature vector size.

Table 9: Standard MFCCs vs. HOC Coefficients

Experiment	FV	Clean (%)		Noisy (%)	
		tonal	baseform	tonal	baseform
Elephant	39	76.4	94.4	70.8	87.5
	54	87.9	96.9	83.5	93.2
Tiger	39	78.5	96.6	73.0	90.8
	54	87.8	97.3	82.3	91.9

In the Monkey scheme, phoneme models and the four lexical tone models are trained separately. Phoneme models are trained using the standard MFCC front-end, while tone models are trained using the HOC front-end. To evaluate the robustness of the tone models, an experiment was carried out on the Pitch99 database. In this experiment, the recognition vocabulary was defined so that the recognition accuracy for each tone could be calculated. The results in Table 10 show an acceptable performance in both clean and noisy environments.

Given the good-performing tone models, Table 11 shows that Monkey scheme obtained an accuracy of 82.7% in clean. Although there is still some room to catch up with the HOC version of Elephant and Tiger, it is clearly better than Elephant and Tiger with just 39 feature vector components. Furthermore, 87% reduction in model memory and computation was obtained, compared to Elephant.

Table 10: Tone classification performance

Tone	Clean (%)	Noisy (%)
Tone1	85.6	78.3
Tone2	88.0	80.3
Tone3	79.4	67.4
Tone4	92.2	84.1
Average	87.1	77.5

Table 11: Comparison of Elephant, Tiger, and Monkey with 39-component feature vector

Experiment	Clean (%)	Noisy (%)
Elephant (39)	76.4	70.8
Tiger (39)	78.5	73.0
Monkey (39)	82.7	74.4

Speaker adaptation is useful for improving the recognition accuracy of a given speech recognition system for a single speaker. Bayesian adaptation was applied to the Tiger and Monkey systems. The adaptation method was supervised online adaptation, which means that after a correct recognition result the models were updated. For Tiger, all the models were adapted while for Monkey, two settings were used. First, only acoustic models were adapted and then both acoustic and the lexical tone models were adapted. The results can be seen in Table 12. The results in parentheses for Monkey are for adapting only acoustic models. Clear improvement, especially in noise, can be seen for Tiger and Monkey. The benefit of adapting only acoustic models is small. When performing speaker adaptation on acoustic and tone models, the maximum performance is obtained. The conclusion is that the adaptation of the tone models is feasible. One reason for the improved performance is that pitch/tone is a highly speaker-dependent feature and therefore speaker-adaptation should improve the recognition accuracy.

Table 12: Tone adaptation results

System	Environment	Speaker-Independent	Bayesian Adaptation
Tiger	clean	87.9 %	89.7 %
	noisy	82.3 %	87.3 %
Monkey	clean	82.7 %	83.9 % (83.0 %)
	noisy	74.4 %	80.2 % (75.4 %)

Now that tone recognition has been found feasible, it is also of interest to know whether the changes should be incorporated into the multi-lingual voice dialing system. Applying the proposed tone recognition scheme is significantly different from the normal speech recognition system and requires a degree of customization to the engine. The changes are not difficult to implement but it is difficult to maintain a single code base that can be easily configured to use either the non-tonal recognizer or the one having tone recognition capability. The need for tone recognition in Mandarin Chinese voice dialing is assessed next.

4.3.3. Name Analysis for Multi-lingual Speech Recognition

Voice dialing of user contacts is the main application of multi-lingual voice dialing technology. To improve the performance of the system it is beneficial to perform analysis on the names of various languages to make adjustments to the system requirements and features. One such study was performed on Chinese names together with Cao Wenjie and Bo Xu from the Chinese Academy of Sciences [16]. The study proved that tone recognition is not necessary for Mandarin when building a voice dialing system for Chinese languages.

In many applications, it is important to identify proper names and especially person names. Chinese text contains only 1-2% of person names, but the errors made in parsing the text are mainly caused by person names. In one study by Ji and Luo, the parsing errors of person names caused up to 90% of all parsing errors [61]. Therefore, a comprehensive analysis of Chinese person names was carried out. Another study by Cao et al. [15] address the automatic person name tagging of a Chinese text database. According to the study, the quality and amount of training data is important in automatic tagging of person names.

A Chinese person name is formed by a family name and a given name. Most surnames are monosyllabic and only a few are disyllabic. The disyllabic names represent one family name, or a combination of two monosyllabic family names. The geographical distribution of family names is different in various parts of China. Almost all given names are monosyllabic or disyllabic. People's selection of names, number of syllables and characters used in given names, is influenced by many factors. These include gender, dialect, time, psychology, geology, belief, culture, etc. [20]. Compared to names of most western languages, Chinese names are more flexible, and it is impossible to include all the different Chinese names in a relatively small vocabulary. Some researchers have found rules in surnames and character selections in given names. Yet monographs on the analysis of Chinese names, especially those based on a large and representative name corpus, are rare.

A large name corpus of 1,000,400 entries was utilized to calculate the statistics. The name entries were mainly extracted from newspapers, and some were written manually. Approximately 992,500 names were analyzed. Of the remaining 8,000 names, 6,000 entries are transliterated names and the rest could not be confirmed to be names. In the rest of the discussion, Corpus refers to the 992,500 names that were analyzed.

Overview of the Chinese Name Corpus

Altogether 4,253 different Chinese characters were present in the name database. 4,064 different characters cover the names in the Corpus. Of the 4,064 characters, 956 were used in surnames, and 3,994 characters were found in given names. Characters used in given names can be positive or neutral. They can also be derogatory, which, especially in villages, is believed to help children with such names to grow up more easily.

How is the length of the name distributed in the corpus? There are three kinds of names according to the length of the name. Most of the full names are tri-syllabic, whose number is about 891,750 (89.9%). The second biggest group is disyllabic names, which consist of 100,400 (10.1%) names. The least frequent names are tetra-syllabic (the surnames

are two characters), which contain 340 names with the proportion of 0.03%. The ratio of disyllabic names to tri-syllabic names is 1:8.882. Traditionally, people tend to select disyllabic given names, and most full names are therefore tri-syllabic. For a time (1970's-1990's), there was a fashion to select monosyllabic given names for babies in many places. This trend originated from some cities, and was later adopted in most parts of the countryside. But people found that monosyllabic given names are more prone to be the same as other people's and expresses less information than disyllabic given names do. From the late 1980's, the trend gradually faded and 2-syllable given names became more common.

Analysis of Chinese Names

The first task in the analysis of the surnames is to find the surnames from the database. A 570-surname set [79] was used to do a first-step extraction. Next, by using the statistical information of the corpus, the possible surnames were depicted. For each surname candidate that was uncertain, the abundant information of the Internet and modern Chinese dictionary was used to confirm the name type and to find out its correct pronunciation. 974 surnames were extracted from the Corpus, out of which 951 were monosyllabic and 23 were disyllabic.

Table 13: A comparison of the two statistics of Chinese surnames

$Rank_1$	$Cumulative_1$ (%)	$Rank_2$	$Cumulative_2$ (%)
10	49.8%	14	49.5%
19	59.9%	24	60.1%
34	70.1%	39	69.8%
58	80.0%	64	80.0%
113	90.0%	114	90.0%
427	99.0%	365	99.0%

Table 13 presents a comparison between the current study and a spot check about a census of 7 regions in China in 1982 [20]. The spot check was a statistics of 174,900 names, involving 729 monosyllabic surnames, and 8 disyllabic surnames. $Rank_1$ is the number of names giving the cumulative frequency shown on same row in the $Cumulative_1$ column. The information in row 4 shows that the 58 most common family names cover 80% of the population. The values with subscript 2 refer to the statistics of the spot check data. A comparison of the four most common family names can be found Table 14. While the most frequent family name is the same (王, wáng), there is a slight difference in the order of the succeeding family names in the table.

Table 14: A comparison of the four most frequent surnames

Current study		Spot check	
Surname	**Frequency (%)**	**Surname***	**Frequency* (%)**
王	10.5%	王	7.3%
张	9.3%	陈	7.1%
李	8.8%	李	6.9%
刘	6.7%	张	6.2%

The main reason for the difference in the two statistics may be caused by the regional distribution of surnames. Also, the selection method of the corpus according to regions may influence the result. Yet there are similarities in both order and amounts. Both cases can reflect the surnames used by unusually large number of people in China, although the order may be different.

The analysis of given names revealed that 3,994 different characters are used. Both monosyllabic and disyllabic given names were analyzed. In the Corpus, about 100,450 given names are monosyllabic. The character set used in monosyllabic given names contains 2,341 characters. About 892,000 given names are disyllabic. 3,075 characters are used in the 1st syllable of the disyllabic given names and 3,406 characters are used in the 2nd syllable of the disyllabic given names. Table 15 shows the cumulative frequency of the monosyllabic given names. Only 66 different characters cover 50% of all monosyllabic names. As opposed to the findings in family names, the characters used for given names are distributed more equally.

Table 15: Cumulative frequencies of monosyllabic Chinese given names

Number of Characters	Cumulative Frequency
66	50.2 %
98	59.9 %
147	70.0 %
229	80.0 %
425	90.0 %
687	95.0 %
1,492	99.0 %

Table 16: Cumulative frequency of characters in disyllabic given names

1st syllables of given names		2nd syllable of given names	
Number of Characters	Cumulative Frequency	Number of Characters	Cumulative Frequency
39	50.1 %	51	50.2 %
60	60.1 %	73	59.9 %
91	70.0 %	108	70.1 %
140	80.0 %	170	80.0 %
251	90.0 %	305	90.0 %
421	95.0 %	490	95.0 %
1,053	99.0 %	1,211	99.0 %

Table 16 shows the cumulative frequencies of the 1st and 2nd syllable of Chinese given names. The difference in statistics is marginal and shows that there are less than one hundred characters for both 1st and 2nd syllable that are used in the majority of the names. Full names were also analyzed. The following is a comparative table of the statistic data compared to that of the spot check. The spot check data is marked with an asterisk.

Table 17: Comparison of Chinese full names

Length of given names	Total number	Proportion of multiple occurrences	Unique names	Proportion of unique names that have multiple occurrences
1-syllable	100,450	80.4%	32,110	38.6%
2-syllable	891,920	52.6%	549,110	22.1%
1-syllable*	21,400	54.1%	12,835	23.6%
2-syllable*	153,500	22.9%	130,670	9.4%

From Table 17, it can be seen that the proportion of repeated names in our corpus is much higher than that of the spot check. This is mainly because of the different size of the two corpora. We can also see that shorter names have more often multiple occurrences in the Corpus.

The four most frequent disyllabic full names are: 王军 (282 occurrences), 李军 (263), 张军 (243), and 刘军 (237). This is in line with the spot check. The most frequent three tri-syllabic names are 王建国 (312), 张建国 (292) and 李建国 (283), while in the spot check it is 李秀英, 王秀英 and 张秀英. This is because the proportion of male and female is quite different in the two corpora and name selection is dependent on the person's gender.

Handling of Multi-pronunciation Characters

In Mandarin Chinese, there are over 1,000 multi-pronunciation characters. There are 396 multi-pronunciation characters in the Corpus, out of which 387 characters appeared in given names, and 138 characters in surnames. As opposed to normal text, handling of multi-pronunciation characters in names has particular problems because of little context information. Yet several characteristics can be found, such as:

- When used in surnames, most multi-pronunciation characters have only one pronunciation (Only 6 exceptions have been found).
- Many multi-pronunciation characters have only one pronunciation in names, for example: 华 (hua2), 红 (hong2).
- Some Chinese names have repeated characters, such as 丽丽. The 2nd character is pronounced as li5 (neutral tone).
- Some characters have a specific pronunciation in specific position in a name, for example: 为 in the last syllable of names is often pronounced as wei2.

Considering these characteristics, multi-pronunciation characters can be handled based on knowledge and rules. The multi-pronunciation characters were analyzed and 176 characters were selected that usually have only one pronunciation in names. A lexicon was prepared of these properly annotated characters. Some rules were made for repeated characters in given names and for pronouncing specific characters in specific positions.

About 315,600 names in the Corpus have multi-pronunciation characters. First, the correctly annotated surnames were filtered from the Corpus. Second, the lexicon and rules were used to process the remaining names. In the first step, 149,300 unique names were extracted. In the second step, additional 156,600 unique names were extracted. There were still 9,700 names whose pronunciation was not possible to solve unambiguously. These 9,700 names were left out of the following analysis.

Tonal Pattern of the Names

For a tri-syllabic Chinese person name with pronunciation, such as "cao2 wen2 jie2", its tonal pattern is denoted with "2 2 2". The analysis was done to find out people's preference towards the tonal pattern of a given name. Also of interest was the influence of family name on the selection of given name. In this study both phonetic and tonal difference was regarded as multi-pronunciation character.

The tonal patterns of both full names and given names were analyzed. Because the amount of 4-syllable names was too small, they were excluded from the analysis.

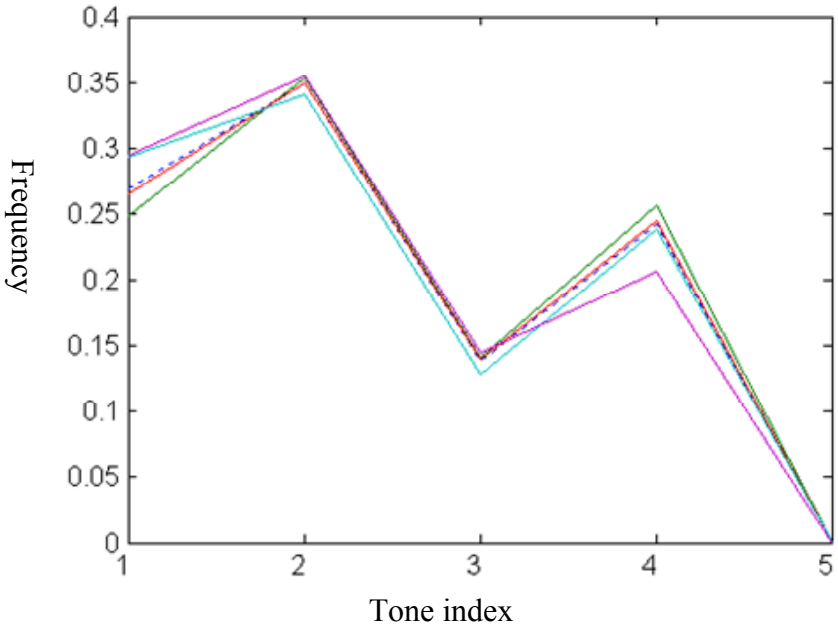

Figure 11: Tonal pattern distribution of monosyllabic given names

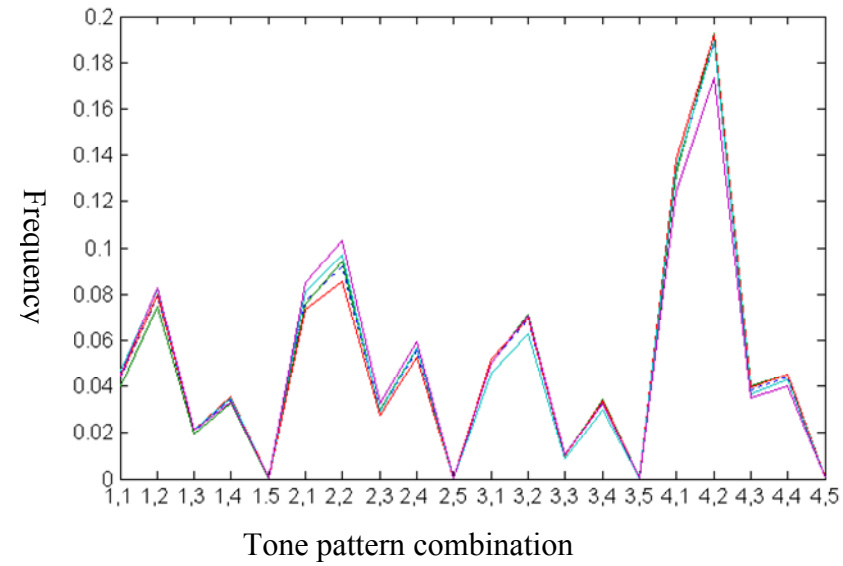

Figure 12: Tonal pattern distribution of disyllabic given names

Figure 11 and Figure 12 show the distributions of tonal patterns of monosyllabic and disyllabic given names, respectively. The horizontal axis of the figures represents the tonal pattern of given names, and vertical axis represents frequency of each tonal pattern. Each figure has 5 lines. Each solid line in the figures represents the tonal pattern distribution of given names that have surnames with the same tone. The two dotted lines are average values.

From the figures, we can see that some combinations are clearly more frequent than others. People do have a preference towards the tonal pattern of given names. For disyllabic given names, tone patterns "4,2" and "4,1" are the two most common. Monosyllabic given names have most frequently tone "2", i.e. the rising tone. The tone of the surname has only a small influence on the tonal pattern of given names.

Estimation of Recognition Accuracy

In this part, the confusability of Chinese names was evaluated using statistics obtained from a speech recognition engine. Random sets of *N* names were extracted from the corpus, where *N* equaled to 20, 50, 100, 200, and 500. A statistical model was used to estimate the confusion probability between each pair of names of the randomly generated name list. Then, based on these estimated confusion probabilities, an upper-limit to the recognition accuracy of each name list was estimated. A similar concept for estimating recognizer accuracy without testing data was proposed by Deng et al. [24]. These methods become more useful when the task gets more complicated. In many cases, the data collection efforts take biggest share of the development costs.

A large database on LVCSR (Large Vocabulary Continuous Speech Recognition) speech has been collected by National Laboratory of Pattern Recognition at Chinese Academy of Sciences. It contains a confusion matrix between the LVCSR's input and output Chinese syllables. The input of the LVCSR was generated by broadcasters whose pronunciation could be considered as canonical. The corpus can accurately reflect the real confusion faced in any Mandarin speech recognition task.

Although Mandarin Chinese has only 409 syllables (without considering the tone), the syllable confusion matrix was not used directly. This is because the frequencies of the syllables in the corpus are quite different, and the lowest syllable frequency may be too low to estimate the confusion probability between this and other syllables accurately (sparse data problem). To resolve this problem, two phone confusion probability matrices were estimated from the corpus: a confusion matrix of 24 consonants (including 'y', 'w' and nil-consonant), and a confusion matrix of 37 simple or compound vowels. Postulating that a consonant cannot be confused with simple or compound vowels, and vice versa, a syllable confusion matrix was re-estimated with these two phone confusion matrices. Supposing two syllables s_1 and s_2, where $s_1=c_1v_1$, $s_2=c_2v_2$ (c_i are the beginning consonants of s_i and v_i are the final vowels of s_i, $i=1, 2$), the probability of confusing s_1 with s_2 is given by Equation (3):

$$\text{Prob}(s_1, s_2) = \text{Prob}(c_1, c_2) * \text{Prob}(v_1, v_2) \tag{3}$$

When computing the confusion probability between two words, a model like a dynamically-built finite state network was assumed. Each syllable of the shorter transcription (a sequence of syllables of a word) acts as a state in the network, and syllables of the longer transcription acts as the arcs of the network. The probability of each arc equals to the confusion probability between the syllable of the arc and the syllable of the state at which the

arc points. Dynamic programming (DP) matching was used and the boundary condition was such that the first and the last syllables of the two words should be matched.

As Figure 13 illustrates, transcriptions of two names are $t_1=s_1s_2$ and $t_2=s_1's_2's_3'$, where s_1="zhang", s_2="li", s_1'="chang", s_2'="lin", s_3'="qiu". The probability of confusing t_2 with t_1 should be estimated. Let syllables of t_1, i.e. *"zhang"* and *"li"*, be the states of the network, and the sequence t_2, i.e. *"chang lin qiu"*, be the output sequence.

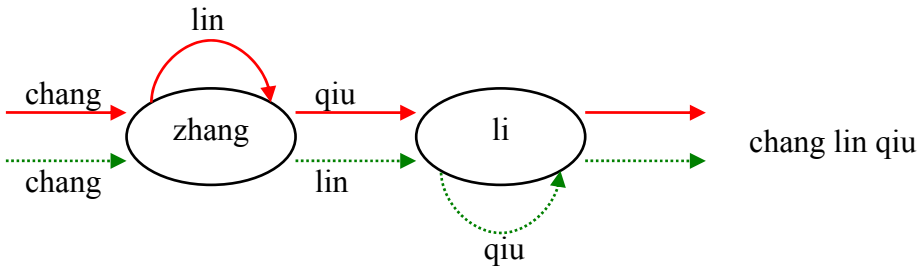

Figure 13: Finite state network showing confusability between "chang lin qiu" and "zhang li"

Then, the state shift path could be the sequence labeled by the solid lines, or the sequence labeled by the dashed lines. The probability of each path is the product of the probabilities of the arcs in the path. The highest probability is the confusion probability between t_2 and t_1.

For a randomly generated name list of size N, the recognition accuracy for each name can be estimated. The only thing that should be computed is the confusability of all name pairs for the given name list and to normalize the summation. The expected recognition accuracy (ERA) could be expressed as in Equation (4). ERA is the recognition accuracy estimate based on the underlying speech recognizer and the confusability measure defined by Equation (3).

$$\mathrm{ERA}(i) = P_{ii} / \sum_{j=1}^{N} P_{ij} \tag{4}$$

The extraction and computation was repeated for 2,000 random sets for each N. For each extraction the number of name pairs, triplets, and quadruplets, etc., in which the names have the same pronunciation (tone omitted), was checked. The expected recognition accuracy can be seen in Table 18.

Table 18: Average ERA and average number of repeated transcriptions

Extraction Number	Average ERA (%)	Average number of repeated transcriptions
20	99.99%	0.003
50	99.85%	0.043
100	99.90%	0.144
200	99.69%	0.607
500	99.14%	3.819

It can be observed from the values above that the recognition accuracy of this speech recognizer is very high. Also, the number of similar phonetic names is not very high when the number of words is less than 500. The study showed that probability of having two names with the exact same pronunciation is very small. This was the second part of deciding on the need for tone recognition. Based on the studies it was decided that no tonal support is included in the multi-lingual speech recognizer.

4.3.4. Speaker Adaptation

When the user starts using the voice dialing system, the acoustic modeling units are common and do not include any information of the current user's voice. The models are speaker-independent and do not necessarily perform optimally with the current user's voice. The reason for this can be the use of non-native names in the system, a heavy dialect or accent. Several methods can be applied to improve the recognition accuracy of the voice dialing system during the use of the device. Many of the approaches, however, require the user's attention.

Speaker adaptation methods can be roughly divided into two classes: supervised and unsupervised. The former class consists of methods that rely on the correctness of the recognition result. In this case only correct data is used for adapting the acoustic model, which ensures that the model set does not diverge from its original distributions but actually improves the match to the user's speech. The latter class of algorithms does not impose any restrictions on the recognition result but performs speaker adaptation regardless of the correctness of the result. Both methods have pros and cons that are highlighted in Table 19 below.

Speaker-adaptation algorithms are commonly based either on Maximum A-Posteriori (MAP) adaptation [36] or the Maximum-Likelihood Linear Regression (MLLR) algorithm [77]. Of these, the former is more attractive in a mobile environment due to its lower computational and memory requirements as shown in Section 4.4.2. The biggest benefit of MLLR is that it can adapt to unseen parameters via its use of global (or several local) transformations [77].

Table 19: Characteristics of the supervised vs. unsupervised adaptation

Characteristics	Supervised adaptation	Unsupervised adaptation
Adaptation speed	Somewhat fast	Fast
Divergence	No	Yes
User interaction needed	Yes	No

As described above, supervised adaptation is desired due to its improved stability during the lifetime of the device. From the user's perspective, however, the speaker adaptation should not add any additional steps into the user interface but adaptation should be performed seamlessly to the user. The use of unsupervised adaptation has triggered several studies on its feasibility in real-world applications. Hybrid methods have also been proposed where semi-automatic verification of the result precedes the speaker adaptation. The possibility of divergence should be minimized and several approaches have been proposed such as the work presented by Nguyen et al. [89].

Therefore, a simple transparent verification is used in the voice dialing system whose user interface is shown in Chapter 5. The rule for carrying out speaker adaptation is based on the user's acceptance of the recognition result. If the user accepts the recognition result and consequently a phone call is made or the action associated with the recognized command is activated, speaker adaptation is performed. This is an adequate requirement to fulfill supervised adaptation. Sometimes, the user may not cancel a wrong result in a timely manner and the speaker adaptation is performed on the basis of wrong result. According to experiments, the speed of adaptation is set to a level that even 50% of the adapted utterances can be wrong and still the overall performance is improved from the original speaker-independent system. The effect of the parameter quantization to speaker adaptation was studied more closely since it was a possible source of problems, to be discussed thoroughly in Section 4.4.2.

4.3.5. Utterance Detection

The basic speech recognition theory does not include methods to detect the interesting speech segments. In practice, the input signal to the speech recognizer contains both speech and background noise. Noise in this context refers to any signal that is not carrying information about the user's intention and should be ignored by the speech recognition system. The terms noise or background noise are used interchangeably in this section to refer to the part of signal that is of no interest. The two main approaches to utterance detection are outlined in this section. First, a dedicated background model can be trained to model the non-speech events in the input signal. The speech recognition grammar is constructed in such a way that the non-speech events can be detected by the background model. The second alternative is to detect speech events in the signal by speech processing methods, such as a Voice Activity Detector (VAD). This approach, together with post-processing to remove any spurious speech activity,

gives a set of speech events that are then processed in the speech recognition engine. The discussion is started with the most common methods including work done by the author. Finally, the method used in the multi-lingual voice dialing system is described. The reasons for using the chosen method are also elaborated.

When the noise and speech are separated with a dedicated noise model, the question often arises about what kind of acoustic model should be constructed. Depending on the task of the speech recognizer, a reference noise signal is commonly used to train an acoustic model that captures the noise characteristics. The noise signal often contains events, such as total silence (microphone noise), car noise, traffic noise, street noise, background speech, hesitations, lip smacks, etc. Each of the noise types can have their own noise model or a single noise model can be trained to represent all of the noise types. A known problem with the conventional trained noise models is that the training material cannot, in practice, include noise from all possible usage environments. Mismatch between the training and use environments usually results in reduced recognition accuracy. In addition to the mismatch between the noise models, there is also a mismatch between the input speech signal and the speech acoustic models due to the new use environment. One solution is to use an adaptive noise model and a technique called Parallel Model Combination (PMC) as suggested by Gales and Young [35]. The method incorporates an adaptive background model that is used to transform the speech acoustic models to the current environment. The transformation requires a reliable estimate of the current noise condition and assumes that the noise environment is stationary. In the results shown later in this section, the word models are assumed to be fixed i.e. no PMC is used.

A garbage model is a background noise model that tries to detect events that do not match the speech models very well. The garbage model and its optimal adaptation was proposed by the author with Laurila and Haavisto [50]. As opposed to modeling noise events that may change during the use of the system, the garbage model measures the similarity of the input speech signal to the speech models. If the match is not good enough, the evaluated signal frame is classified as noise. Bourlard et al. [11] proposed to use the observation probabilities of the speech acoustic models to obtain a reference score for out-of-vocabulary word detection (cf. Section 4.3.6). The idea of the garbage model is extended here to an Adaptive Garbage Model (AGM), which can be used as the only noise model. In addition to using it as the only noise model, the AGM can be optimally adapted to the current noise environment and to the current set of word acoustic models. The garbage model ensures a match between the use environment and the acoustic models since the noise model is estimated online from the match between the input signal and the acoustic models.

Garbage Model Definition

The garbage model is dependent on the word models currently used in the speech recognizer. This ensures that there is no mismatch between the word models and the garbage model. For each input frame, the observation probabilities of each state in each model are sorted in descending order. After all observation probabilities have been calculated, the garbage model observation probability is calculated as follows:

$$b_{GM} = b_W{}^{r^K} \tag{5}$$

where b_{GM} is the garbage mode observation probability and $b_W{}^{rK}$ is the r^{Kth} largest observation probability of the word models. The rank r^K is calculated as follows:

$$r^K = \left[1 + (1 - K) * (S - 1)\right] \tag{6}$$

where K is a factor determining the rank order used and its value is between zero and one. S is the number of active states in the recognizer. The r^{Kth} largest observation probability of the word models is set as the garbage model observation probability. In addition to this a linear interpolation between nearest observation probabilities if r^K is not an integer, is used. In the similar approach proposed by Bourlard et al. [11] the mean of the word model observation probabilities was used instead of rank order. Rank ordering gives a more reliable estimate against outliers and adjustment of the noise model scoring is easy by changing the desired rank order. As an example, for S=100 and K=0.85 gives the rank value r^K of 15. This means that the 15th largest observation probability of the current frame is used as the background model observation probability. In practical terms the rank tells that what ratio of the observation probabilities of the word models is smaller than the garbage model probability for each signal frame.

If the factor K is small and S is large, the calculation of the garbage model observation probability becomes more complex since there are more observation probabilities to be sorted. To overcome this problem an approximation scheme can be used. When calculating the observation probabilities for the word models, the maximum and the mean are stored. Then, the garbage model observation probability is interpolated between the maximum and the mean. This reduces the complexity significantly without decreasing the performance. The rank value becomes the interpolation factor between the mean and maximum probability. With value of zero, garbage model gets the mean observation probability. For rank value of one, the maximum observation probability is used. Notice, that since K<1, the garbage model never produces the highest observation probability within a frame. This creates one fundamental difference between the conventional background models and garbage model. In some cases the conventional background models can mask parts of a word model or even a full word. This is likely to occur with 'noise-like' sounds, such as the beginning of the word "*six*". The conventional background models can give higher probabilities than the beginning states of the model "*six*", and thus the beginning of the word "*six*" can be neglected in the recognition process. This is not the case with garbage model since it masks only the low-scoring word states.

Test for the Garbage Model

The database used in the test consisted of 1,900 7-digit strings in three different environments from 56 speakers. The environments were parking place (Park), city and highway and the recordings were carried out in a car. The database was divided into training and test sets, so that half of the speakers were included in each set. The baseline test was made with 10-state, 2-mixture speaker-independent acoustic models that were trained using parking place, city and highway training sets. Three conventional background models (3-states and 3-mixtures each) were trained separately, one for each environment. The baseline test results can be seen in Table 20. In all the tables *ins* stands for insertion error, *del* for deletion error and *sub* for substitution error.

Table 20: Baseline test result using conventional background models.

Environment / Set	String accuracy	Word accuracy	Sub	Del	Ins
Park training	85.3%	97.6%	21	1	32
Highway training	60.5%	91.6%	140	39	9
City training	78.9%	96.1%	44	1	41
Park test	89.2%	97.8%	31	4	13
Highway test	65.9%	93.6%	112	19	7
City test	75.3%	95.5%	53	8	36

In Table 20, there are more deletion errors in highway and more insertion errors in parking place and city. Moreover, it is hard to guess what happens outside these environments. Variability in the database quality is the reason for better test set results in parking place and highway.

Another test with a fixed garbage model rank was performed to verify the usefulness of the garbage model. The test was performed in all environments with fixed garbage model rank of K=0.88. The value was chosen so that the average performance in all environments was maximized. The following results were obtained.

Table 21: Result with fixed garbage model.

Environment / Set	String accuracy	Word accuracy	Sub	Del	Ins
Park train	90.3%	98.4%	15	2	19
Highway train	64.6%	92.8%	94	59	7
City train	87.1%	97.8%	32	4	14
Park test	89.6%	98.1%	28	6	6
Highway test	68.8%	94.0%	88	36	6
City test	82.1%	96.9%	42	19	6

The string recognition accuracy was increased in all three environments. Average improvement compared to the conventional background models was 2.7%, 3.5% and 7.7% in parking place, highway and city, respectively. There are less substitution errors in Table 21 than in the results shown in Table 20. There are still many deletion and insertion errors and what is more important, there are more deletion errors in highway and more insertion errors in parking place. The rank adaptation of the garbage model is a solution to the unbalance between the insertion and deletion errors. The adaptation scheme and the results given in Iso-Sipilä et al. [50] verified that the garbage model with rank adaptation can cope with varying use environment in the recognition task of continuous digit dialing. The adaptation of the garbage model improved the recognition accuracy from 3% to 8%, depending on the environment.

The background modeling scheme in the Nokia speaker-dependent voice dialing system was based on the garbage model with a fixed rank. This was first evaluated for the speaker-independent voice dialing system, too. The speaker variability in the training databases and the multi-lingual acoustic modeling together caused a reduced recognition accuracy compared to the system with a trained noise model.

End-of-Utterance Detection

In addition to separating speech from noise, a decision when to stop recognition should be made by the recognizer. The basic behavior of a Viterbi based speech recognizer is that it will find the optimal path through the set of acoustic models given the data. It does not, however, tell when to stop, i.e. when the best path is good enough. The simplest form of an utterance detector uses the power contour of the signal to determine when the speech starts and when it ends. As described earlier in this chapter the voice dialing system uses a dedicated background model for detecting non-speech signal. Therefore, no special start-of-speech detection is needed. Only the decision of the end of recognition is needed. Therefore, the rest of this section concentrates on methods for end-of-utterance detection.

Several power based end-of-utterance schemes have been proposed and their performance is relatively good in normal conditions. The work by Hariharan et al [37] showed that with proper post processing of the signal power a robust end-of-utterance system can be

designed. By applying multi-band spectral analysis with robust filtering the performance was improved in noise conditions. When the amount of noise increases and has non-stationary characteristic, the performance is reduced.

The end-of-utterance method used in the multi-lingual voice dialing system relies on the speech recognition decoder's decision. It assumes that when a good result candidate is found, no other candidate will outperform it. When a pre-defined time period has been passed without a better candidate, the recognition result is considered confirmed and the recognition process is finished. In practice, several exceptions can be found and additional criteria should be defined. The exceptions require, e.g. that no other prospective results coming through the Viterbi decoder and that there are no other candidates that have the same prefix as the current candidate result. With these additional restrictions, the end-of-utterance algorithm is both robust and uses very little computational resources.

4.3.6. Out-of-Vocabulary Word Rejection

The speech recognizer is normally designed to find the best path through the models, given the data. This definition does not address whether any of the results is a valid match to the input data. The problem of classifying the result into either valid or invalid is difficult to solve. Introduction of such algorithm commonly decreases the performance of the system for valid input, too. The set of algorithms dealing with valid and invalid input is called Out-of-Vocabulary (OOV) word rejection methods, here referred to as rejection. Some of the most common methods are outlined first and then, the algorithm used in the multi-lingual voice dialing system is introduced shortly. Research carried out by the author is a basis for the rejection functionality of the multi-lingual voice dialing system.

The detection of incorrect hypotheses is a challenging task. The fact that the speech recognizer always finds a match makes it hard to design a rejection algorithm that is speaker- and vocabulary-independent. Obviously, if the user vocabulary has the word "Jack" and the user says "Mack", it is a demanding task to decide whether the word spoken is "Jack" or if it should be rejected as not belonging to the active vocabulary. Nevertheless, most speech recognition systems are equipped with rejection capability. The simplest approach to rejection is to compare the probability of the recognized word with the probability of a reference model and make the decision based on the log-likelihood ratio of the two probabilities. The reference model can be a noise model, a general speech model or an online garbage model like the one described in Section 4.3.5. Usually a garbage or filler model is utilized for the detection of OOV words [96][131]. In the work of Bourlard et al. [11] a special garbage model was introduced, where the probability of the garbage model was calculated from the word model probabilities. The work presented by Junkawitsch and Höge in [64] uses inter-frame correlation for utterance verification.

The garbage model was found to be effective for speaker-dependent voice dialing and its good rejection capabilities were utilized in the design of a fully voice controlled system to be described next. The system utilizes the concept of continuous listening, where the

recognizer is active at all times and waits for a certain key phrase to be spoken. It is the ultimate challenge for out-of-vocabulary word rejection.

Currently many recognizers require users to express themselves with some non-speech modality to start speaking to the device. In human-machine dialogues, the users are often given only a few seconds to speak, otherwise the system takes over. Systems which continuously listen to the user are rare. One approach towards continuous listening is to utilize word spotting techniques [96][131]. The approach given by Iso-Sipilä et al. [52] provides user a hands-free and eyes-free way to initiate a dialogue with a machine by speaking a predetermined command or phrase. The approach is here called voice activation. The system is a combination of out-of-vocabulary word rejection technology and clever utterance detection combined with user interface design to meet the criteria for an efficient voice activation algorithm. The contribution of the author is the overall system design which included the design of a multiple state user interface and a three-level confidence threshold to improve the rejection performance. The author also designed several additional criteria to improve the accuracy and performance. The design target was to accept an activation phrase if it is repeated within a short time period. If a false rejection does not cause adjustment in the system operational point, the user may end up repeating the same word again and again to no avail.

Sub-word HMM Approach

A high rejection rate of OOV words is a key requirement of voice activation systems. The user finds it very irritating if a system gets activated from background speech, noise, or other non-keyword speech input. In order to make voice activation applications attractive to the users, the false alarm rate should be as low as possible. On the other hand, the acceptance rate of valid activation phrases should be as high as possible. To meet both of these high performance objectives, special techniques focusing on OOV word rejection should be developed.

An effective approach to improve the rejection performance is to use sub-word modeling to model the activation phrases. Conventional left-to-right whole-word HMM structure can provide only single state order. When having sub-word models this constrained structure can be relaxed as illustrated in Figure 14. If sub-word models are recognized with a loop grammar, i.e. in any order, an additional degree of freedom is introduced. When the correct utterance has truly been spoken, it is likely that the sub-word HMMs are recognized in the proper order. In the case of out-of-vocabulary utterance, the sub-word models are recognized in an arbitrary order.

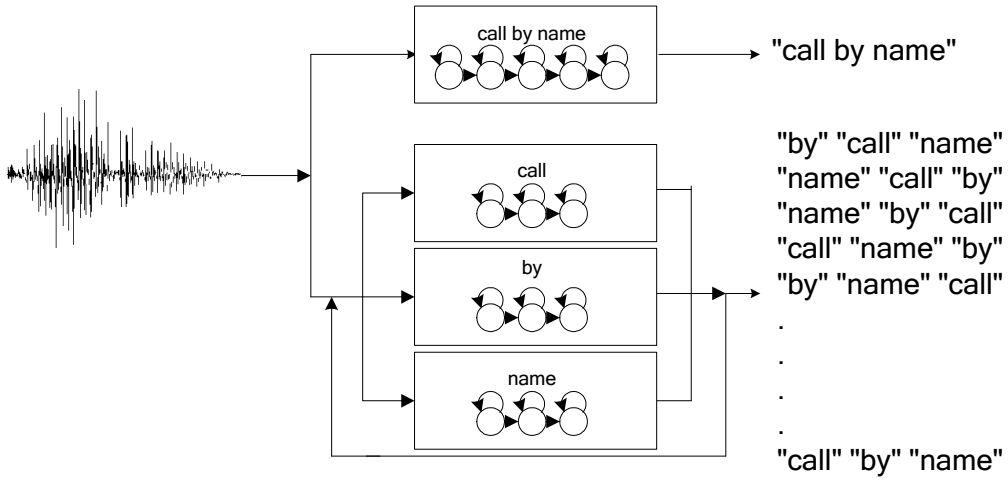

Figure 14: Whole-phrase and loop grammar sub-word modeling approaches to voice activation.

When sub-word HMMs are recognized in the proper order, the validity of the recognized utterance should be further tested using confidence measures, such as log-likelihood ratio tests. Using the sub-word approach, the rejection performance is proportional to N^N where N is the total number of sub-word models. When aiming at an extremely high OOV word rejection rate without substantially reducing the acceptance rate of valid activation phrases, the grammar validity test is essential as OOV words occasionally tend to produce high confidence value, which leads to false activations.

Since phonemes are one of the basic units of speech, it is straightforward to apply them as sub-word HMMs. However, a typical activation phrase consists of several phonemes. In the example phrase shown in Figure 14 may contain 10 phonemes corresponding to 10^{10} possible combinations. As the number of sub-word models increases, it becomes more difficult to recognize sub-word models in a proper order. To solve this problem, one can accept invalid combinations which are likely to occur when the correct activation phrase is spoken. Rather than using the phonemes as sub-word units, a simplified approach can be utilized that is suitable for systems using whole word modeling.

One can directly construct the sub-word models from the whole-word HMM that describes the activation phrase. Sub-word models are obtained by dividing the whole-word HMM into N sub-word units as illustrated in Figure 14. In practice, the number of sub-word models should be $N \geq 3$ so that a sufficient OOV word rejection rate can be achieved. When creating the sub-word models, special attention should be paid to the splitting process, to prevent the sub-word models from being too similar. If two or more sub-word HMMs are too close to each other, it is difficult to recognize these models in the proper order even in the case of a valid activation phrase. In addition to the sub-word modeling approach, user interface design should be such that it reduces the number of false alarms while makes sure that the user can get the message to the speech recognizer with reasonable effort. This is described next.

Multi-Level Approach

It is natural for humans to repeat unrecognized commands, i.e. when the user has spoken a command with no response from the system. This is common in human-to-human conversation and it is exploited in the voice activation system to further enhance the performance. To guarantee a high OOV word rejection rate, it is necessary to require high confidence thresholds for accepting the result. This may lead to a low acceptance rate of correct utterances. To solve this problem, one can continuously monitor whether utterances have confidences close to the required values. If such confidences are observed, the user might have said the proper activation phrase, and is likely to repeat it within a few seconds.

In the voice activation system, each result candidate is classified into one of the three classes: *accept*, *reject* or *potential*. *Accept* means that the confidence of the utterance is high enough and the result should be accepted. *Reject* means that the confidence is too low and the recognition is rejected. *Potential* means that the confidence is neither too low nor high enough for the two former classes. In this case, the recognizer changes state for T seconds and then returns to normal operation mode if no further command is received from the user.

The multi-level approach consists of M states. Each state is assigned with a confidence threshold and the higher states use lower thresholds. This means that each time a state change occurs to a higher state, the result can be accepted with a lower confidence value. Figure 15 shows the voice activation state machine with $M = 3$. Here, the 1st level corresponds to the normal operation mode of the recognizer. A state change occurs when a result candidate is *potential*. The state is changed back to the 1st level when a time-out or *reject* event occurs. The result candidate can be accepted in each level of the state machine.

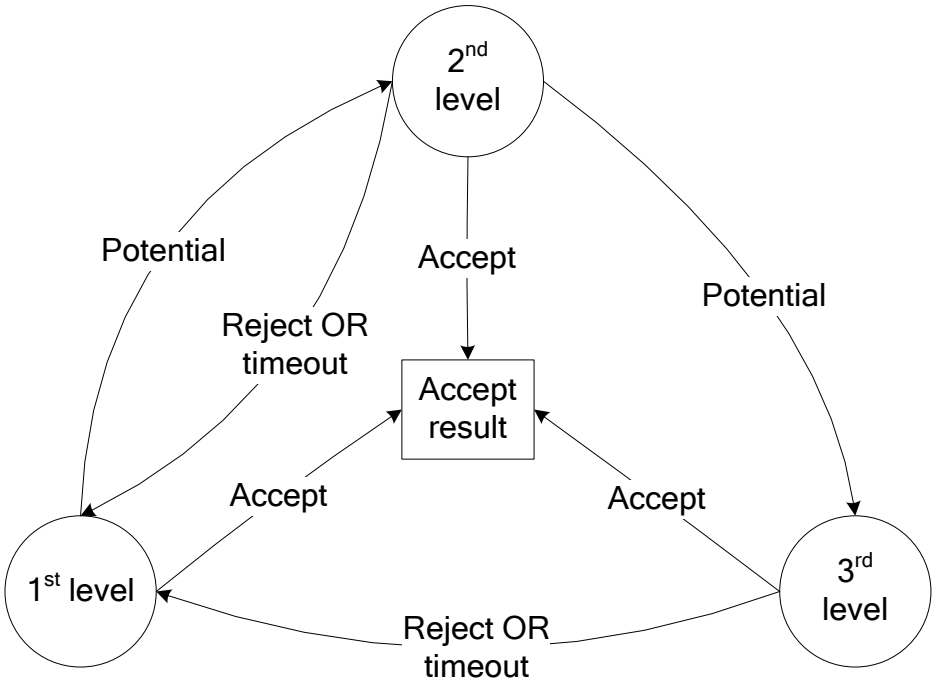

Figure 15: Multi-level voice activation approach with three levels.

Lower thresholds in higher levels increase the probability of false alarms. To avoid this, the successive repetitions are further compared in order to ensure that they are the same.

Whenever a result candidate is obtained in a higher level *m* (*m* > 1), the utterance in level *m* is compared to the utterance in level *m*-1. The comparison is done at the feature vector level and it uses a Viterbi-algorithm based time alignment scheme [125]. The distance between the two repetitions is compared with a threshold to decide on the similarity of the utterances. If the utterances are close enough and the recognition is classified as *accept*, the result candidate at level *m* is accepted. If the distance is too high, the result candidate is rejected and the level is changed to the 1st level.

Sometimes OOV input makes the recognizer change the level. It is very unlikely that two such inputs that produce high confidences occur within *T* seconds. Furthermore, since the repetitions are compared the possibility of OOV input to activate the recognizer is reduced significantly.

A confidence measure is used to determine whether a recognition result is correct or not. This confidence is based on the difference between the word model and the garbage model log-likelihood scores [122]. The confidences of each sub-word model are normalized with the length of the recognized sub-word. Hence, *N* confidences are obtained for each recognized activation phrase. These confidences are compared against the pre-defined confidence thresholds. Final requirement for the utterance to be accepted is that the power contour of the signal should follow a pre-determined pattern as described detailed by Iso-Sipilä et al. [52].

Voice Activation Experiments

The experimental results given in [52] show that the accuracy of the voice activation system can be improved significantly by introducing the sub-word modeling and multi-state user interface approaches. Finally, real-time speaker-dependent experiments with 12 novice users were carried out to verify the algorithms. The speakers were from eight different countries and in each language there was a different set of activation phrases. The task of each test person was to utter an activation phrase and if the recognizer did not respond, the activation phrase was repeated. If the recognizer did not respond with three activation phrases, the recognition was discarded and marked as a failure. Otherwise, the recognition was marked as successful. It was also marked, whether the successful utterance was the 1st, 2nd or 3rd. The experiments were carried out in a car. The environments were parked car, city traffic, road 80 km/h, road 80 km/h with music and highway 120 km/h. The results for each environment are shown in Table 22. The overall acceptance with maximum of three trials was 99.7%. The OOV word rejection capability was tested off-line with a dedicated database that contains meetings, radio programs, music and noise. One false alarm per 25 hours of rejection test data was observed using speaker-dependent models.

Table 22: Results of the real-time speaker-dependent experiments with 12 test persons

Environment	One of the three trials	1st trial	2nd trial	3rd trial	Failures
park	99.7%	95.8%	3.2%	0.8%	0.3%
city	99.8%	92.1%	6.7%	1.0%	0.2%
road	99.9%	95.1%	4.6%	0.2%	0.1%
road + music	99.1%	86.3%	10.0%	2.8%	0.9%
highway	99.7%	90.3%	8.4%	1.0%	0.3%
average	**99.7%**	**91.7%**	**6.8%**	**1.1%**	**0.3%**

It has also been observed that the multi-lingual approach and the feature vector normalization cause a certain uncertainty in the phoneme models leading to lower rejection accuracy, compared with mono-lingual models. The reason behind this is the use of data from several languages to train a single acoustic model. The assumptions made for the similarity of the sound units is adequate for finding the best match but it seems to be less optimal for the assessment of the result validity.

4.4. ASR COMPLEXITY REDUCTION METHODS

The available memory and computational resources are the major limiting factors in a mobile device. These limit the use of various signal processing algorithms to perform speech recognition and speech synthesis. The amount of RAM memory on a desktop PC is in the order of Gigabytes while a mobile device is commonly equipped with RAM memory of several Megabytes. In the computational capabilities the difference is not that large and the desktop CPUs are normally a decade faster in integer calculation than the mobile counterparts. Therefore, the memory and computation need to be optimized in a mobile platform.

The starting point for an implementation on a mobile device is commonly the code using floating point arithmetics, which is often referred to as the reference implementation. The floating point representation is not suitable for mobile device CPUs that most efficiently compute with integer arithmetics. The first step, details are not included in the thesis, is to convert the floating point arithmetics into integer arithmetics. This is a rather straightforward process where data ranges play important role and where the overall performance of the system is commonly preserved while the computational complexity and memory requirements are reduced. A good coverage of the basic concepts with integer arithmetics is given by Yates [132].

The rest of this section concentrates on the methods to reduce the memory and computational requirements on a mobile processor/platform. The author's contribution is presented with two publications made in this area and the contribution is highlighted in the

beginning of each section. First a method to reduce the computational complexity and RAM use by utilizing a time decimation approach is presented. Second, several memory and computation reduction methods used in the multi-lingual voice dialing system are outlined.

4.4.1. Complexity Reduction via Decimation in Time

In small vocabulary speech recognition systems, the observation probability computation requires most of the processing power. The repeating computation of Euclidian distance between the speech feature vector and the acoustic model density is therefore a good candidate to look for reduction in computational complexity. It is obvious that either reducing the number of components in the feature vector or computing the observation probability less frequently both result in a linear reduction of complexity. The basic concepts of the former are shortly introduced in Section 4.4.4. The latter of these approaches is presented next.

It has been shown in the work of Arai et al. [1] that the modulation spectrum of MFCCs has most of the information within the frequency range from 1 Hz to 16 Hz, assuming a feature vector sampling frequency of 100 Hz. This suggests that the maximum observable frequency of 50 Hz, i.e. the Nyquist frequency, is not necessary to maintain all information in the feature vector stream but the feature vectors could be decimated to reduce the amount of processing. The following is based on Publication 3 [53] and shows that feature vector stream can be decimated in speaker-dependent voice dialing.

The MFCC front-end used in the speaker-dependent voice dialing is similar to the one presented in Section 2.3.2 but only Cepstral coefficients and their 1st order derivatives are included in the feature vector. For the same amount of densities, the speaker-dependent system requires only 67% of the computation needed in the speaker-independent system. The feature vector stream contains 100 feature vectors per second and this study concentrates on the time behavior of each individual feature vector component.

When the statistics of each individual feature vector coefficient time trajectory is studied, the following observation can be made. The modulation spectrum is limited to the frequencies between 0 and 16 Hz [66][136]. However, with feature vector rate of 100 frames per second, the highest frequency found in the modulation spectrum is 50 Hz. The average spectrum for three Cepstral coefficients over several utterances in a quiet environment is shown in Figure 16.

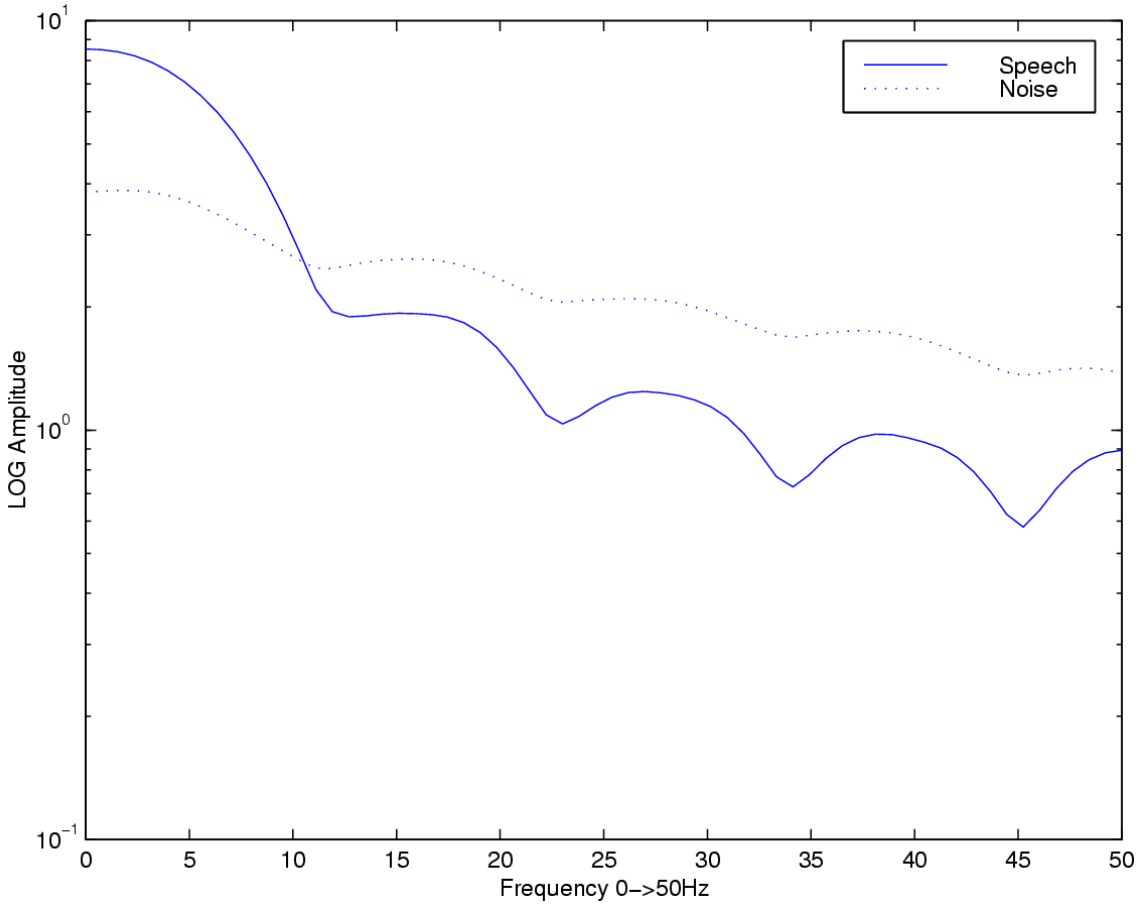

Figure 16: Spectrum of Cepstral time trajectories for speech and noise

In Figure 16, the amplitude of the speech spectrum decreases rapidly as a function of frequency. Most of the energy is concentrated in the frequency band 0-12 Hz. The frequencies above 12 Hz have less information content and the information is also difficult to model with a HMM based system. Based on these assumptions and basic signal processing theory [47], the band-limited feature vector stream can be decimated without significant loss of information.

Decimation of the Cepstral Time Trajectories

Since the Cepstral time trajectories are band-limited, the feature vector rate can be reduced through decimation. The decimation process can be seen in Figure 17 where F_s is the original sampling frequency. The low-pass filter is necessary to avoid aliasing after removing every other sample of the stream.

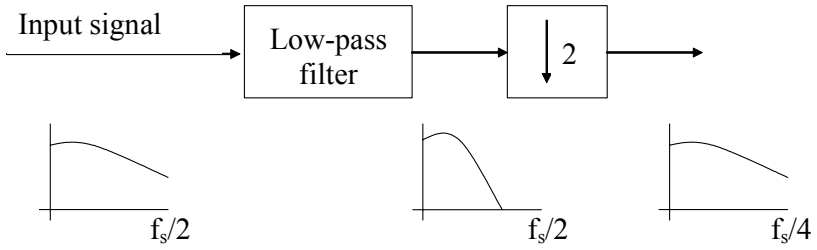

Figure 17: Process decimation and its effects to the signal spectrum

At the output of the decimator, in Figure 17, the rate of feature vectors is half of the original rate. Generally, the original rate is decreased by the decimation factor. Since the speech is band-limited to 12 Hz, a decimation factor of 4 could be used. Now, the problem of decimation is that the information stays there but the exact timing of all the events is not known. After decimation, different feature vector component streams can have different timing and this can seriously affect the accuracy of an HMM based recognition system. This is especially true if the decimation is applied without touching the acoustic models.

Minimum-phase FIR filters were first tried for the decimation due to the short time delay. Also, more efficient filtering can be obtained with a shorter non-linear phase filter. Since the minimum-phase filter has a non-linear group delay the delay of different modulation frequencies varies. This variation produced bad results in the experiments and linear-phase filters were designed to obtain linear group delay. Each linear-phase FIR filter had 21 taps and the stop-band started at the critical frequency 50 Hz/*f*, where *f* is the decimation factor. The stop-band attenuation of the filters was at least 20 dB and the pass-band deviation was limited to 1 dB.

Acoustic Modeling and Experiments

A speaker-dependent name dialing system was used in all experiments. The acoustic model for each utterance was trained with a single noise-free training sample with varying state count, according to the length of the training sample. Since only one training utterance was utilized, variance information was not used in the models and furthermore the use of feature vector normalization set the variance of each feature vector component equal to 1 over a given time interval. State duration constraints [74] have successfully been used in speaker-dependent noise robust speech recognition. State duration constraints force the frame-state alignments to be similar in training and recognition, no matter what the environment is. The minimum and maximum duration limits were set individually for each state and Viterbi decoding was performed so that these duration limits were met. An on-line garbage model was used for noise modeling [11][50].

With decimation, the length of each utterance is shortened. A decimation factor f=2 resulted in a decimated utterance containing half of the original feature vectors. This had an effect on the state duration limits, which had to be halved, meaning that the minimum and maximum duration limits were divided by the decimation factor *f*. To avoid non-integer values, the following rounding was applied. In Equation (7), the function *round* rounds its

argument towards the nearest integer. If D_{min} or D_{max} gets too small after the division, the default value of 1 is used.

$$\begin{aligned} \hat{D}_{\min} &= \lfloor D_{\min} / f \rfloor \\ \hat{D}_{\max} &= round\,(D_{\max} / f) \\ if\ &\left(\hat{D}_{\min} < 1\right) \Rightarrow \hat{D}_{\min} = 1 \\ if\ &\left(\hat{D}_{\max} < 1\right) \Rightarrow \hat{D}_{\max} = 1 \end{aligned} \tag{7}$$

The database used in the experiments consisted of 9 speakers, 6 males and 3 females. Each speaker had two sets of 30-utterance vocabularies: one with short first names (short set) and another with longer full names (long set). Every speaker in the database had his/her own vocabulary. Each utterance was spoken 12 times and the first repetition was used for acoustic model training. To evaluate the recognition performance in noisy environment, car noise was artificially added to original noise-free speech waveforms at three Signal-to-Noise Ratios (SNR): 0 dB, -5 dB and –10dB. In the results, the environment labeled *clean* is the original speech that was recorded in a noise-free environment. The speech recognition algorithm utilized the state-duration constrained HMM framework with token passing [134]. The baseline result for short set and long set can be seen in Table 23.

Table 23: Baseline result for speaker-dependent name dialing

Environment/Set	short set	long set
Clean	99.32%	99.80%
SNR = 0 dB	96.26%	99.46%
SNR = -5 dB	90.16%	98.68%
SNR = -10 dB	70.97%	86.70%
Average	**89.18%**	**96.16%**

In addition to the proposed decimation approach, the following complexity reduction methods were also evaluated. Firstly, the reduced-dimension feature vector with only the static Cepstral coefficients reduces the observation probability computation by a factor of 2. Secondly, changing the overlap of consecutive frames to be 80 samples instead of 160 samples also reduces the complexity by the same amount. The results for these experiments can be seen in Table 24. The row titled *Difference* shows the absolute recognition rate difference with respect to the baseline average result. The reduced-dimension feature vector is clearly inferior to the baseline system in all environments and both sets. The smaller overlap method is clearly worse than the baseline system for the short set. The smaller overlap method surprisingly provides better result for the long set than the baseline.

Table 24: Result with a reduced-dimension feature vector and smaller overlap

	Reduced dimension		Smaller overlap	
Environment/Set	short set	long set	short set	long set
Clean	98.98%	99.70%	99.22%	99.86%
SNR = 0 dB	94.71%	99.12%	95.11%	99.39%
SNR = -5 dB	86.29%	97.34%	87.47%	98.62%
SNR = -10 dB	63.26%	80.06%	67.30%	88.18%
Average	**85.81%**	**94.05%**	**87.28%**	**96.51%**
Difference to baseline	-3.37%	-2.11%	-1.90%	+0.35%

The decimation experiments were carried out with decimation factors ranging from 2 to 5. The results for the short and the long set can be seen in Table 25 and Table 26, respectively.

Table 25: Decimation results for the short set

Decimation factor	2	3	4	5
Clean	99.19%	98.95%	98.88%	96.46%
SNR = 0 dB	95.58%	94.78%	94.98%	87.54%
SNR = -5 dB	88.98%	87.91%	88.82%	78.61%
SNR = -10 dB	69.36%	66.76%	69.83%	60.27%
Average	**88.28%**	**87.10%**	**88.13%**	**80.72%**
Difference	-0.90%	-2.08%	-1.05%	-8.46%

Table 26: Decimation results for the long set

Decimation factor	2	3	4	5
clean	99.59%	99.26%	99.53%	99.22%
SNR = 0 dB	98.15%	98.48%	97.44%	95.69%
SNR = -5 dB	97.94%	96.49%	97.30%	95.85%
SNR = -10 dB	88.65%	78.92%	87.03%	87.98%
Average	**96.08%**	**93.29%**	**95.32%**	**94.68%**
Difference	-0.08%	-2.87%	-0.84%	-1.48%

Observation from Table 25 and Table 26 is that the performance degradation was minimal with decimation factors up to 4. The reason for the better performance with decimation factor 4 than decimation factor 3 was caused by the integer rounding of the state duration parameters. With decimation factor 5, the performance started to degrade more rapidly for the short set while the performance of the long set stayed very close to the baseline system. The behavior of the short set was a clear consequence of the fact that the signal was band-limited to 0-12 Hz and a higher degree of decimation started to remove essential information. Comparing this to the result in Table 24 for the reduced-dimension feature vector shows that the information obtained from derivative calculation was important. Therefore, decimation is a better approach to complexity reduction than the reduced-dimension feature vector. Decimation of the short set provided consistently better results than the smaller overlap method. With decimation, a higher decimation factor is superior in complexity reduction if compared to the smaller overlap method.

Comparing Table 25 and Table 26 to Table 24, it should be pointed out that the performance in noisy environments can be maintained with the decimation approach. The degradation with decimation factor of 5 for the short set is mainly due to the already short utterances. The length of the short utterances can be as little as 25 frames. After decimation by a factor of 5, only 5 frames are left.

Complexity Reduction Analysis

The complexity reduction obtained with the decimation approach is twofold. Firstly, the complexity of the Viterbi-decoder is reduced by the factor of decimation[7]. This means that when using the decimation factor of 3, the Viterbi decoder has 1/3 complexity compared to the baseline system. The MFCC front-end has the same complexity in all cases except for the additional filtering of each FV component with the 21-tap FIR filter, which is negligible, compared to observation probability computation. Secondly, the memory required for the state-duration constrained token passing implementation is lowered by the percentages given in Table 27.

Table 27: Memory saving in token passing with decimation

Decimation factor	short names	long names
2	40%	44%
3	64%	62%
4	64%	69%
5	78%	74%

[7] Assuming that observation probability computation takes most of the processing power

The reason for not being able to save memory directly proportional to the decimation factor is seen in Equation (7). Since the duration limits can only have integer values and the maximum duration is rounded towards the closest integer, the saving is not that big.

With this method, a larger vocabulary size can be enabled in the device while maintaining the target complexity. A similar approach has been suggested for speaker-independent voice dialing by Kiss and Vasilache [68]. Later on, the work combining data driven low pass filtering together with feature vector normalization was granted a US patent [51]. In this patent, a method combining feature vector normalization (mean and variance) together with data driven low pass filtering was found to improve the noise robustness of the speech recognizer in speaker-dependent and speaker-independent domains. The system included the filter estimation process and the noise removal behavior of the combination was considered to be the inventive step. Figure 16 shows that the noise modulation spectrum has larger amplitude on higher modulation spectrum frequencies than speech. Therefore, filtering the feature vector also increases the SNR of the signal. The multi-lingual voice dialing system utilizes feature vector decimation with an approximated approach. Next, the most important complexity reduction methods utilized in the multi-lingual voice dialing system are presented.

4.4.2. Quantization Methods to Reduce Memory and Computation

This section is adapted from Publication 7 [121]. The following methods further reduce the memory and complexity requirements of the multi-lingual voice dialing system. The author proposed to apply parameter quantization also on the observations to get rid of the helper tables proposed by Vasilache [119]. The author also suggested performing speaker adaptation after every recognized utterance to reduce challenges in the implementation caused by prolonged storage of parameters over several recognition tasks. Finally, the author also suggested that the speaker adaptation data should be quantized in order to store it in a more compact manner. All these methods together improved the memory efficiency of the multi-lingual voice dialing system without reducing the recognition accuracy of the system. More recently, work done by Köhler et al. [73] has also successfully utilized the parameter quantization method to reduce the memory footprint and computation. The following sub-sections illustrate the abovementioned complexity and memory reduction techniques and their influence in the design process of a practical speech recognition engine.

Reducing Memory and Computational Requirements

Various compression options to reduce memory taken by HMMs have been proposed, the vast majority of them being focused on quantization procedures for the density mean and variance vectors [32]. As previously presented by Vasilache [119], joint scalar quantization is a very effective and, possibly, one of the simplest methods. Similarly to the alternative approaches, it is capable of achieving both a reduced memory representation as well as a reduced computational cost.

The decoder complexity is heavily influenced by the cost of computing state emission likelihoods (which is later referred as B-probs). This part of computation can consume more

than 60% of the total recognition processing even with the efficient approach allowed by the quantized HMMs (qHMM) [119]. The most expensive part is the evaluation of the Mahalanobis distance for each density. For a single component the formula reduces to

$$d_{ki} = \frac{(\mu_{ki} - o_i)^2}{\sigma_{ki}^2} \tag{8}$$

where μ_{ki} and σ_{ki}^2 are the i^{th} mean and variance component of the density k and o_i is the i^{th} component of the observation vector. The sum of d_{ki} for all dimensions of the observation vector (the feature space) gives the required distance value for the density k. With a quantized representation for means and variances it is readily apparent that given the observation vector the set of possible values for d_{ki} has the size of the product of mean and variance quantizer levels. With a low rate quantization this results in a small set which makes pre-computing these values advantageous. By storing them into tables, one for each feature vector dimension, the d_{ki} values are obtained by indexing with the joint index of mean and variance quantizer. This reduces B-prob computation to table indexing and summations. Vasilache [119] gives estimates on the number of floating point operations required by the original and by the indexed approach. It is shown that, for a certain critical level of densities, it is more effective to use the helper tables. Since, in practical systems, the arithmetic operations alone are not uniquely responsible for processor cycles both methods should be evaluated in order to determine the fastest approach.

Once B-probs have been obtained the computation of the best alignment of the HMM states to the input sequence of feature vectors is obtained using the Viterbi algorithm. A token passing approach presented by Young et al. [134] is used. For a phoneme based isolated word recognition task, it is advantageous to minimize the Viterbi token passing structure with the help of prefix sharing. By this, a tree structured recognition grammar is established. It is known [87] that optimal structures can be derived in terms of the total number of phonemes required. Usually the optimal structures are less regular and much more expensive to construct dynamically. The tree structure provides a good trade-off between compactness and representation regularity. Its major advantages are that it requires minimal additional resources to keep its structure (as low as one bit per phoneme instance), offers the possibility of memory localized decoding and hence good processor cache utilization. Last, but not least, in comparison with a linear grammar representation, it has better behavior when a beam search with the Viterbi algorithm is utilized.

If in Equation (8), the feature vectors are also quantized, the computation of the helper tables at each frame can be completely avoided, provided that the number of quantization levels is manageable in terms of the required storage. Since the HMM parameters are already heavily quantized, the feature vectors can be quantized with a higher or comparable rate without significantly influencing the adaptation performance. In this respect, the existing scalar quantizers for the mean parameters of qHMMs can provide a simple and efficient

solution. This approach reduces the memory space required to store the most recent utterance for speaker adaptation.

Speaker adaptation is a mandatory procedure to maximize the recognition performance for every user of the system. Among the available alternatives, Bayesian adaptation was chosen due to stability, performance, simplicity and minimal complexity overhead. A more in-depth description of the adaptation algorithm and its performance is given by Vasilache and Viikki [120]. In the following, the focus is only on the complexity aspects. Once enough adaptation utterances have been observed the adapted model set would then replace the original speaker independent one. However, doing this will also substantially increase the memory requirement for adaptation parameter storage. In a supervised framework it was observed that adaptation after each utterance produces good results. Therefore, with careful programming of the adaptation equations, the memory requirements for estimation accumulators are reduced to the equivalent of adapting a single Gaussian density. Furthermore, no long-term storage of accumulators is required. The following experimental results verify these assumptions.

Experiments

The evaluation was carried out using a multi-lingual speaker-independent name recognition task. Feature extraction was performed using MFCCs. The 1st and 2nd order time derivatives were included which resulted in 39 dimensional feature vectors. For noise robustness a normalization scheme [124] was enabled as the final stage of the front-end. The three state phoneme models had a left-to-right structure with no skips. Maximum Likelihood training was performed on a diverse set of languages. Although the system was trained to simultaneously handle a large set of languages, the results are shown only for two out of the 25 supported languages. The selected languages are German and English. The recognition grammar consisted of names composed of one or several words. Slightly more than 100 name entries were active for each language and a total of 11,000 utterances were selected for testing. Two testing environments were used; "clean" which contains the original speech waveform recordings and "noise" which was created by mixing various noise types with SNR ranges from 5 to 20 dB.

The results shown next are limited and the full results can be found in Publication 7 [121]. In the first set of experiments, using the original set of models, the effect of switching the engine from floating point mode ("float") into fix point mode ("fxp") was evaluated. Typically, 16 bit integer representations were used in fixed point implementation. Both speaker-independent ("SI") and speaker-adapted ("SA") rates are presented below in Table 28.

Table 28: Fixed point vs. floating point performance

Environment	SI fxp (%)	SI float (%)	SA fxp (%)	SA float (%)
Clean	95.05%	95.14%	98.02%	97.97%
Noise	84.83%	85.09%	92.40%	93.07%

The negative impact is most visible in the more challenging noisy environment. Since floating point computation is prohibitively expensive in an embedded environment both because of memory as well as computational complexity (e.g. emulation may be required) this design step and performance impact were unavoidable.

The following experiments focused on HMM parameter quantization. A large range of quantization rates for means and variances was evaluated, as illustrated in Table 29 and Table 30. The quantization rate for means is placed at the start of each row and the rates for variances are on top of the columns. All the quantizers were non-linear, Lloyd-Max trained on the parameters of the original models. No re-training on acoustic models was applied after quantizer training. From Table 29 and Table 30 it is observed that with less quantization the performance saturates close to the baseline models. Due to practical considerations (aligned packing of mean-variance pairs into bytes) "5m3v" and "3m1v" are the most interesting design points. These are highlighted in all tables. If pressed by hard complexity limits, even extreme quantization rates, such as a 2-bit rate for the means with a global variance, can be considered.

Table 29: Recognition rates with HMM quantization

Clean SI (%)					
m \ v	0	1	2	3	4
3	92.75	**94.40**	95.03	95.14	n/a
5	93.08	94.56	95.08	**95.01**	95.05
7	93.03	94.48	95.11	94.79	94.87
Noisy SI (%)					
m \ v	0	1	2	3	4
3	82.03	**83.77**	84.23	84.58	n/a
5	82.52	84.30	84.50	**84.76**	84.65
7	82.48	84.50	84.44	84.79	84.76

Table 30: Recognition rates with HMM quantization and speaker adaptation

Clean SA (%)					
m \ v	0	1	2	3	4
3	97.05	**97.51**	97.77	97.66	n/a
5	97.45	97.86	98.17	**97.97**	98.01
7	97.54	97.82	98.04	97.97	97.98
Noisy SA (%)					
m \ v	0	1	2	3	4
3	90.24	**90.86**	91.34	91.30	n/a
5	91.11	92.40	92.50	**92.82**	92.56
7	90.61	92.16	92.80	92.62	92.51

Although the required storage for the quantizers is small (e.g. only 40 values for a 5-bit + 3-bit pair) the performance of optimal linear quantizers was analyzed. In comparison with the non-linear ones, they had similar performance except for the lower rates where a higher degradation was noticeable. Only the "Clean SI" results are shown in Table 31 while other cases provided similar performance.

Table 31: Recognition rates with uniform HMM quantization

Clean SI (%)					
m \ v	0	1	2	3	4
3	92.44	**94.03**	94.38	94.32	n/a
5	93.04	94.59	94.72	**95.14**	95.18
7	93.10	94.52	94.96	95.08	94.94

As presented earlier in this section, it is beneficial to avoid computing the B-prob helper tables at each frame. In order to do this the feature vectors are quantized only for the purpose of optimized B-prob computation. The results of these experiments are shown in the columns "SI" and "SA" of Table 32. The naming convention encodes the quantization rates for means, variances and features (i.e. "3m1v4f" refers to rate of 3 bits for mean, 1 bit for variances and 4 bits for the feature vector components). For reference, the original models "original" and quantized models with accurate B-prob computation are also included (i.e. the table entries which are not ending in "f"). Based on these results the proposed method has negligible impact on the recognition performance. Surprisingly, in a few cases even better performance is measured.

Table 32: Results of feature quantization for B-prob computation and adaptation

Clean	**SI (%)**	**SA (%)**	**qSA (%)**
3m1v4f	94.52	97.70	97.55
3m1v	94.40	97.51	97.58
5m3v4f	95.02	97.97	97.95
5m3v	95.01	97.97	97.99
Original	95.05	98.02	97.96
Noise	**SI (%)**	**SA (%)**	**qSA (%)**
3m1v4f	83.90	90.83	90.76
3m1v	83.77	90.86	90.69
5m3v4f	84.66	92.76	92.78
5m3v	84.76	92.82	92.69
Original	84.83	92.40	92.25

As discussed in previous section, storing quantized features can substantially reduce the memory requirements of speaker adaptation. The impact on the speaker adaptation performance is visible in the "qSA" column of Table 32 where a 4-bit quantizer is used in all cases. This quantizer is identical to the one used in the B-prob acceleration for "3m1v4f" and "5m3v4f". A minimal effect is observed for all cases.

Finally, following the ideas presented by Kiss and Vasilache [68] the influence of halving the frame frequency of the B-prob computations was tested. Two most interesting design targets are shown in Table 33. In this case, Viterbi beam search is computed at every frame by using the same B-probs for two consecutive frames.

Table 33: Results with computation of B-probs every 2nd frame

Setting	**SI (%)**	**SI/2 (%)**	**qSA (%)**	**qSA/2 (%)**	**Environment**
3m1v4f	94.52	94.38	97.55	97.51	Clean
5m3v4f	95.02	94.87	97.95	97.82	
3m1v4f	83.90	83.47	90.76	90.34	Noise
5m3v4f	84.66	84.25	92.78	92.04	

As shown in Table 33 in columns "SI/2" and "qSI/2", the impact is only marginal in comparison to the reference rates in columns "SI" and "qSA". Nevertheless, the complexity reduction is substantial, 50% saving in b-prob computation.

Finally, after all these steps, it can be concluded that in practice, for this design example, "5m3v4f" provides a good solution, with substantial complexity reductions, when

high performance is desired. For stronger complexity constraints, "3m1v4f" provides a good compromise.

4.4.3. Model Tying

Model tying can be used to reduce the size of the acoustic model or to increase the amount of training data per density [135][38]. Remember from Section 4.4.1 that observation probability computation takes major part of the overall complexity. Therefore, the number of probabilities to compute has a linear dependency to the complexity. Reducing the number of densities via model tying reduces the complexity proportionally. Also, for a fixed amount of training data, reduction in the number of densities to estimate increases the density specific training data. The multi-lingual voice dialing system utilizes model tying in some Asian acoustic models that have more phonemes than most of the other regional acoustic model variants. Tying reduces the memory overhead of the acoustic model and also reduces the computation while retaining the recognition accuracy at an acceptable level. This allows maintaining a given target size for the acoustic model and limit the computational requirements associated with the size of the acoustic model.

4.4.4. Feature Masking

The purpose of the feature vector computation module is to find a set of parameters that best describe the speech signal and can be used to train a set of statistical acoustic models to perform speech recognition. This process does not always result in an optimal set of parameters and some parameters are more important than others. This can be seen if a given parameter is removed from the feature vector and the models are re-trained and the evaluation is done.

Following this idea, feature masking has been proposed [109]. The methods to find the optimal set of parameters commonly involve a sequential, iterative approach to omit and include parameters to maximize the performance of the system. At the same time, the model size is minimized. This results in a trade-off between the performance and size. Reducing the size of the acoustic model also results in faster decoding since there are less acoustic model parameters to compute. A complete description of the latest methods can be found in Cooke et al. [21].

4.5. MULTI-LINGUAL SPEECH SYNTHESIS FOR VOICE DIALING

The fundamentals of text-to-speech synthesis algorithms were reviewed in Chapter 2. The required steps that should be taken to turn a mono-lingual speech synthesizer into a system that is capable of handling multiple languages are described here. Once a speech synthesis algorithm is developed for a single or for a few languages, it should be expanded to fit a multi-lingual framework. There are a few basic requirements that should be fulfilled to achieve this. First, phonetic parameters should be shared between supported languages, if possible. Second, to alleviate fast development for new languages, shortcuts may be

necessary. One such option, called language morphing, is introduced. Third, the speech synthesis software should be language-independent and the data must be separated from code. Out of these three requirements, language morphing and language-independent speech synthesis engine were originally proposed by the author to be used in the context of the multi-lingual voice dialing system.

4.5.1. Sharing Phonetic Parameters between Languages

It is beneficial to share phonetic parameters between languages. This was shown in Section 4.2 for speech recognition. The same approach can be applied to speech synthesis with some restrictions and additional requirements. Firstly, the sharing of the phonetic parameters cannot be usually done in a straightforward manner. Secondly, the language specific sound units can be derived from the shared sound units with less work. What do these requirements mean?

The target of sharing the sound units in the speech recognizer was to reduce the complexity and to ensure enough training data for each sound unit. The effectiveness of the sharing can be evaluated by using a test speech database. The recognition accuracy will tell how well the shared sound units perform on each language. For speech synthesis, no general purpose objective evaluation methods exist and the evaluation needs to be done in a subjective manner. Therefore, the evaluation of shared acoustic parameters for synthesis very easily results in bad subjective quality. Due to this, the phonetic parameters for each language need to be optimized to maximize the subjective quality of the speech synthesis engine for each language. In many cases, rule based speech synthesis framework can generate various sounds units from the phonetic inventory. If this is not possible, a separate sound unit should be defined in the system.

To make it easier to develop the language specific phonetic parameters for speech synthesis, the Multi-lingual phonetic alphabet (MLPA) is used as the starting point. The language-specific version of each sound unit is developed by starting with the shared sound unit and adjusting the Klatt synthesizer parameters until the desired quality is achieved. The fact that the MLPA has been defined beforehand significantly helps in the development phase of a new speech synthesis language. In some cases, the MLPA needs to be refined if the speech synthesis phonetic definition should be more accurate than what MLPA can provide. In these cases, the pronunciation models and acoustic models need to be re-designed and re-trained, respectively.

There are situations when the language portfolio needs quicker expansion than the normal language development efforts permit. In these cases, special approach is needed to speed up the language development of the speech synthesizer. This is discussed next.

4.5.2. Rapid Development of Text-to-Speech Languages

The development speed of languages for speech synthesis is generally slower than that of speech recognition. First, the sharing method for sound units described in the previous section is not that efficient for speech synthesis. The consequence of this is that each sound unit in

every language should be optimized and adjusted for maximum subjective quality. Second, more knowledge of each new language is needed in the speech synthesis language development as opposed to speech recognition. Again, this stems from the fact that subjective quality assessment is needed to evaluate a speech synthesizer. As a third restriction on speech synthesis development, each language consumes memory and when several languages are stored simultaneously in a single device, the memory figures may become restrictive.

To overcome the inherent slowness of language development, a method called language morphing is proposed. In this method, the speech synthesis engine of an existing language is used to synthesize speech in a new language that does not have a native speech synthesizer. The approach was originally proposed by Campbell for waveform concatenation based speech synthesis [14]. The author's contribution is the original idea to utilize an existing language to enable the formant speech synthesizer for a new language without considerable development effort. This approach certainly introduces reduction in subjective quality but with proper design these can be avoided in many cases. A system supporting many languages can thus be considered as a competitive advantage for companies operating globally. Capability to support multiple languages in a TTS system can be achieved either by providing differentiated versions of the product for different regions or by having a single, truly multi-lingual system supporting several languages. Work presented by Badino et al. [5] showed that language morphing is also viable when foreign words are embedded in text to be spoken out by a speech synthesizer. The benefit of language morphing in this case is that the same synthesis voice can be used for both native and non-native parts of the sentence.

Development of multi-lingual speech synthesis systems usually requires considerable effort. New speech databases must be recorded and processed for systems based on waveform concatenation. In case of formant synthesis, a new set of synthesis parameters need to be tuned for the sound units of the new language. In addition to the actual waveform generation there are many other differences in languages that must be taken into account. The ones that mostly affect the TTS development are in grammar, phonology and prosody. Orthographic information together with syntactic and morphological analysis provides input for grapheme-to-phoneme (G2P) conversion. Syntactic analysis provides information for sentence level stress assignment, which controls pitch and duration. Alternatively, prosody can be modeled using data driven statistical methods such as classification and regression trees [12][110]. In general, the more accurate synthesis model of a language is needed, the more thorough knowledge of the language is required [6][29].

In addition to development effort, the memory requirement of a multi-language text-to-speech system is significantly higher than that of a single language TTS system. Multi-lingual speech synthesis systems based on a speech database have usually been designed so that the data and software architecture are completely separate. This way, the speech corpus can be changed without affecting the software architecture of the overall system. Thus, development of additional languages is relatively straightforward procedure. The size of the synthesizer, however, increases every time a new language is added. The most memory intensive parts of the whole TTS system are commonly the pronunciation lexica and the

speech databases. The size of each language module varies from several hundred kilobytes to several megabytes, depending on the system target use environment and TTS technology. The size of the speech database can be reduced utilizing various speech coding methods on stored speech units or by reducing the number of the stored units. Some of these memory reduction methods may also be applied to formant synthesizers although the reduction may not be as significant as in the case of waveform concatenation systems. [69][101][18][13][84]

Memory consumption of several megabytes per language is rarely acceptable in mass produced mobile devices, especially in mobile phones, due to the price of the ROM memory. For that reason, support for multiple languages on a single device with the limited memory resources is a major challenge for both design and implementation.

Language Morphing for Text-to-Speech Synthesis

In this approach, an existing synthesis language is re-used to create a close approximation of a new language that should be included in the TTS system. The existing synthesis system can be a full scale TTS system with a large dictionary, syntactic parser and modules generating sentence level prosody or it can be a small footprint isolated word synthesis system. The notation used refers to the existing TTS language as target language while the new language to be added is called source language.

The source language and the target TTS language are linked together at the phonetic level. The pre-requisite is that both languages should have a way of generating proper phonetic transcriptions. The TTS system is naturally already taking care of the G2P conversion for the target language but in this approach such a conversion is also needed for the source language. A phonetic transcription of the input word can be obtained using a lexicon. For smaller memory footprint, a set of rules or statistical trees might provide an attractive alternative for implementing G2P conversion [22]. In the multi-lingual voice dialing system, the G2P module is shared between the TTS and speech recognition and is mandatory for each new source language.

In addition to G2P conversion for the new language, a special mapping module is needed to match the sound units in the source and target language. The mapping module provides the rules for representing the sounds of the source language with the given set of phonemes of the target language. The top-level framework for the phonetic mapping is presented in Figure 18.

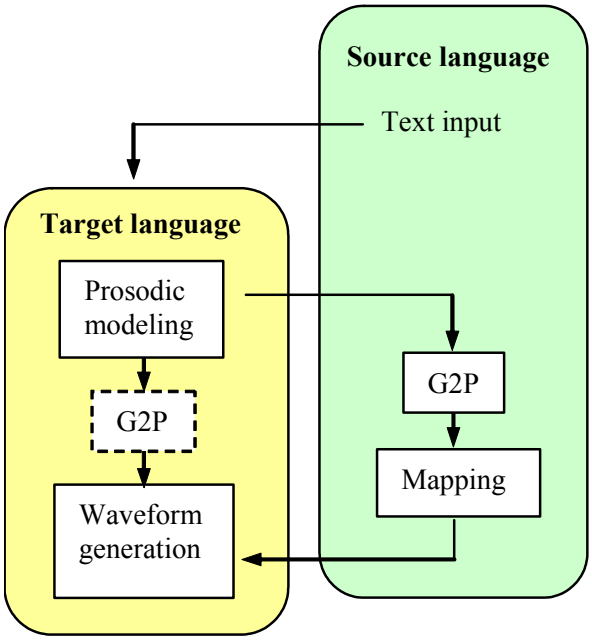

Figure 18: Top-level presentation of a TTS system using cross-lingual phonetic mapping.

The text input given in the source language is processed with the TTS rules and models of the target language. As the figure shows, the prosodic modeling of the TTS system will be based on the target language. The G2P of the target language will be by-passed by the corresponding source language G2P module followed by the phonetic mapping module.

The waveform generation i.e. synthesis module is completely unaffected by the phonetic mapping. The synthesis will operate in the same way as it would in the case of synthesizing text from the target language. The phone sequence, which enters the waveform generation module, contains only those sound units that are supported by the TTS system of the target language. Therefore, the phonetic mapping module is tailor made for each source-target language pair.

The target and the source languages should be linguistically similar. For example, it is more feasible to map Estonian or Hungarian to Finnish than to American English since Estonian, Hungarian and Finnish all belong to the same language family. The goal of the phonetic mapping is to utilize the sound units of the target language to create a sequence of sounds that are as close to the original source language as possible. If the languages are phonetically similar the mapping rules can be easily found and there is no need to derive more elaborate rules. In some cases it is necessary to define more advanced phonetic mappings to improve the subjective sound quality. The reason for this is that the target language and the source language are not necessarily sharing certain sound units and therefore the source sound unit has to be formed applying e.g. two separate sound units from the target language phonetic inventory.

Example of mapping rules from Dutch to German is shown in Table 34. Only the four phonemes shown differ between these two languages. German does not have /A/ (open back

unrounded vowel), /G/ (voiced velar fricative), /r/ (alveolar trill) or /y/ (close front rounded vowel) phonemes so they need to be replaced with approximations from German. The closest match can be obtained using German phonemes /a/, /g/, /R/ and /y:/, respectively. For all the other Dutch sound units there is a counterpart in German using the same symbol so no special mapping is required. Of course, there might be slight differences in phonetic quality between the languages although certain sounds are described with the same phonetic symbols. The phoneme notation used in the example is based on SAMPA [97] and is shown in Table 34.

Table 34: Mapping Dutch into German

Source language: Dutch	**Target Language: German**
A	a
G	g
r	R
y	y:

Below, in Table 35, is an example showing how the phonetic transcription of a Dutch name can be converted to utilize a German TTS system using cross-lingual phonetic mapping.

Table 35: Complete mapping rules from Dutch to German

Dutch	G	e:	r	t	r	y	d	@
↓	↓	↓	↓	↓	↓	↓	↓	↓
German	g	e:	R	t	R	y:	d	@

The phonetic mapping can provide a substantial change in the TTS output in the case of languages whose pronunciation rules differ significantly. This can be demonstrated with an example where a Swedish word "gilla" (to like) is synthesized with an US-English TTS system with and without the phonetic mapping. The example in Figure 19 shows that the sequence created by the cross-lingual phonetic mapping module is closer to the Swedish pronunciation than the one produced by the US-English G2P. Both sequences can be fed into the US-English TTS system to obtain the speech output. The change in pronunciation can be considered especially useful in improving the intelligibility of the synthesized words.

Another example in Table 36 shows the mapping rules for converting Hungarian into Finnish. Notice that in the given example the last two rules split the source phonemes into two target phonemes. Palatal nasal /J/ is presented as a combination of nasal /n/ and palatal approximant /j/, and an approximant /ts/ is presented with unvoiced alveolar plosive /t/ and unvoiced alveolar fricative /s/.

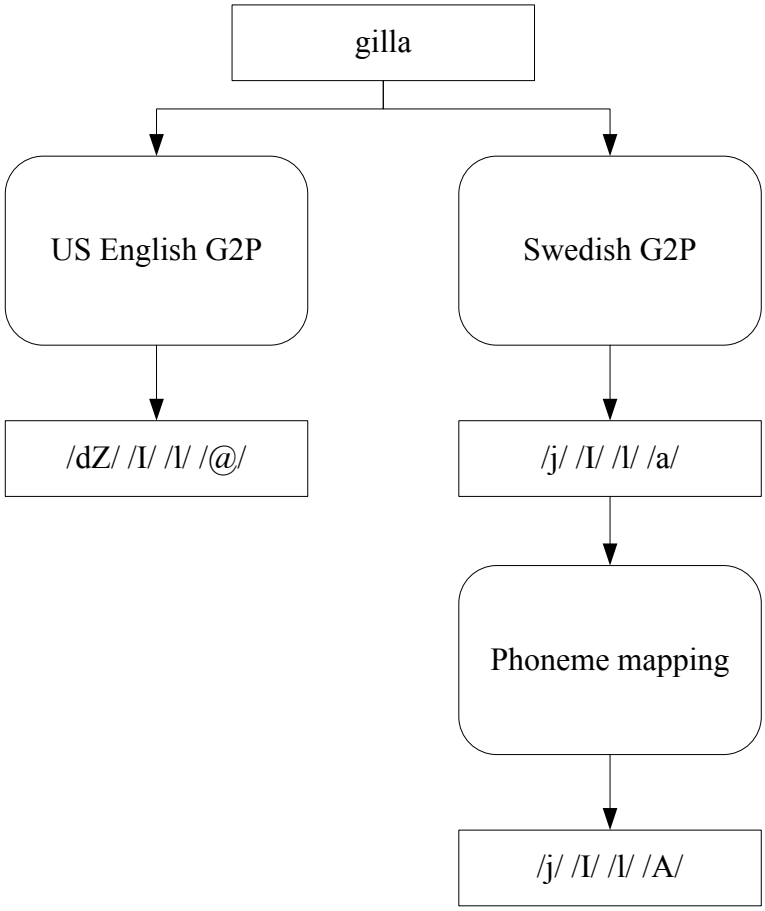

Figure 19: Language morphing between Swedish and US English

Table 36: Mapping Hungarian into Finnish

Source language: Hungarian	**Target language: Finnish**
A	a
a:	a
i:	i
E	e
e:	e
J	n + j
ts	t + s

Similar mapping tables can be easily constructed for any two languages keeping in mind that the closer the languages are to each other, the less approximation is needed in presenting the sound units. Additional use case for the language morphing is to provide a

pronunciation for non-native words with the native TTS system. In the example of Figure 19, the word "gilla" might be labeled as Swedish and the US English TTS system could be used for speaking out the word using the language morphing method. This would provide the English speaking user a close approximation how to pronounce the word as a non-native word. Also, the voice of the US English TTS can be used for speaking out non-native words. Finally, there may be several target languages whose phonetic data could be used to find the optimal language morphing from the source language. As proposed by Black and Schultz [10], this approach may give additional advantage when a reasonably large amount of languages have already been developed. This approach has not been in used in the voice dialing system but it may give additional benefit in case when single target language cannot provide good enough mapping.

4.5.3. Language-Independent Speech Synthesis Engine

The basic rule of software localization and internationalization is that the data should be separated from code, i.e. the software. In a desktop PC application this means that all the display texts and graphics need to be kept in separate data storage for each supported language. This is valid also for mobile devices and makes the development and maintenance of the code easier and less expensive.

In a similar manner, the voice dialing software should be designed to be language independent. This was also the target for the speech synthesis engine. The speech synthesizer consists of several modules that all need to be implemented keeping this restriction in mind. The challenge was overcome by introducing a data configurable text-to-speech system. The author's contribution to the language independent text-to-speech was the observation that the former approach of language-specific software based processing was not adequate. The author proposed that the code based processing should be replaced with a language-independent processing method, such as the one presented by Pärssinen and Moberg [92]. The main benefit of the separation of code and data is the improved portability and maintenance. Furthermore, the language developers do not need the competence for software development when making adjustments to existing languages or when developing a new language.

4.6. VOICE UI DATA

Software that can easily be internationalized and localized is such that its data has been separated from the code. In a multi-lingual voice dialing system this means that the voice UI data should be separated from the code that implements the voice dialing algorithms. This section concentrates on the detailed introduction of each data type and how they are combined into a data package suitable for mobile devices. It shows how the requirement is applied to all processing modules of the multi-lingual voice dialing system.

The basic building blocks of the multi-lingual voice dialing system have been introduced in the preceding chapters. These include multi-lingual phonetics, text processing, language identification, multi-lingual acoustic modeling, pronunciation modeling and multi-

lingual speech synthesis. The data associated with the technologies should be constructed and stored in a way that the data can be utilized in a similar manner for every language. The data should also be efficiently packed to minimize storage requirements. The storage type should also be such that the flexibility of configurations, scalability of memory and computational requirements are maintained. The benefit is that changes in product configuration do not cause unexpected problems during the voice UI localization and voice UI language data package definitions. A practical requirement is that the voice UI data should be such that can be easily configured from one set of languages to another. The following sub-sections describe the role of each data type in the final voice UI language package and its requirements and its effects to the configurations. The following definition should be kept in mind: A **voice UI language package** consists of a combination of language-specific data that is used for a given **UI language package**.

Each mobile device that is shipped to customers includes a pre-defined set of supported user interface languages. The design decision for the voice UI follows the distribution of the visual UI languages. English is commonly accompanied with one or many local languages, depending on the country where the device is to be sold. As an example, a device sold in Finland, commonly supports all Scandinavian languages and English. The voice UI supports the same languages. It is known that this may be a limiting factor since many people can read English while they prefer using their native language when communicating with speech. Therefore, spoken languages have bigger variation within a small region and decision on voice UI data package configuration may lead to many non-native users to the system.

Each voice UI data package contains the following data:

- Acoustic model for the languages supported in the package
- Text-processing data for each language
- Language identification data for each language
- Pronunciation models for each language
- Speech synthesis data for each language

These data packages together and solely define the behavior of the voice dialing system. No other data or language specific assumptions are made in the code. Therefore, the code is fully language-independent to enable full support of the target languages, including any new languages to be added at a later phase. The challenge in the definition of the data to be placed in each variant can be split into two parts. First, the data files that are only dependent on the individual languages should be collected. Second, the data files that are dependent on the exact configuration of the language variant should be evaluated. Both challenges are described if they are applicable to each data type.

4.6.1. Acoustic Model

The acoustic model defines the phonetic content of the languages that are supported by the voice UI data package. The number of sound units included in a single variant varies but is typically between 60 and 120 sound units. This has a direct impact on the size of the acoustic model. A rule of thumb in the multi-lingual voice dialing system is that each sound unit contributes about 1 kB in the final size of the acoustic model.

The master copy of the acoustic model is stored in the device read-only memory while a working copy is available for the current user. The speaker adaptation algorithm is applied on the user specific acoustic model. The user specific acoustic model can be reset to the default one if the user of the device is changed. The option to reset the acoustic model can be found in the Voice Commands application (cf. Section 5.3.2).

Acoustic model data is affected by the latter problem described in the previous section. The data within the acoustic model is shared between several languages and an acoustic model for a single language is not available. A single sound unit can be found in several languages and the acoustic model data is shared in most cases between languages. This makes the acoustic model selection somewhat more difficult for each voice UI language package. Remember that there are acoustic models for five geographic regions, Europe, Middle East, India, Asia Pacific and the Americas. Each set of acoustic models (there may be several within a region) supports a fixed set of languages. For a given voice UI language package, the optimal acoustic model should be selected among the possible candidates. Given the languages in the voice UI language package, it is straightforward to choose the acoustic model if only one acoustic model matches the desired languages. In this case, the single acoustic model supporting all languages in the voice UI language package is selected. There are two other possibilities to occur. First, there may be several acoustic models matching the voice UI language package. The second case is that there are no matching acoustic models.

In the former case, the selection should be made anyway and the choice is either to determine it manually or by taking the first acoustic model with least supported languages. The latter option requires some more explanation. It is common in the Asia Pacific region to develop several acoustic models with few supported languages and a wide overlap between supported languages between two acoustic models. The reason for this is that the languages spoken within a sales region may be significantly different from each other and the phonetic sharing described in Section 4.2 may reduce the recognition accuracy. Hence, the acoustic models need less multi-lingual data and more language-specific data to train them. As an example, Table 37 gives the supported languages for two acoustic models in the Asia Pacific region. The language names are written according to the ISO 639-1 standard [55].

Table 37: Example of two acoustic models and their supported langauges

Acoustic model	Supported languages
HMM1	en, ms, id, tl
HMM2	en, ms, id, tl, vi

The HMM2 supports the same languages as HMM1 except that it has also support for Vietnamese. The need for this type of acoustic models stems from the linguistic differences of languages. To achieve best possible recognition accuracy for each language, within-region variants of the acoustic models are necessary. Now, if the voice UI language package contained languages en, ms and id, the correct acoustic model to choose would be HMM1. On the other hand, if the languages were en, tl and vi, HMM2 would be the correct choice. Selecting another acoustic model may decrease the recognition accuracy.

When there are no matching acoustic models to the requested voice UI language package, the system developers need to train a new acoustic model to support the new configuration that has not been seen earlier. This is the only case when variant configuration cannot cope with the given voice UI language package automatically but require the developer to include the support to the overall system. Luckily, this does not happen too often since every new acoustic model increases the likelihood that a new voice UI language package is already supported by the existing acoustic models.

4.6.2. Text-Processing Data

The data needed to perform text-processing contains both language-independent and language-dependent data. The language-independent data specifies the case conversions that are possible in all supported languages. It also contains the language-independent symbol conversion rules that were described in Section 3.5. The data is stored in a single data file that is common to every voice UI language package. The size of the data file is 2 kB.

The language-dependent data specifies the symbol conversions needed for each language. It also defines the valid character set for each supported language. The character set data is used in both symbol conversions and in language identification. The data size varies among languages and the data is embedded in the language specific pronunciation data, to be presented in Section 4.6.4.

4.6.3. Language Identification Data

Language identification is carried out by an N-gram based method that takes into account the language-specific alphabet in the scoring as described in Section 3.4. For each language there is a language-specific data file containing the N-gram definitions. Additionally, there is a common data file for language identification that contains voice UI language package specific information. The size of the N-gram data for a single language is approximately 2 kB.

4.6.4. Pronunciation Models

The pronunciation modeling data is language-dependent and is applied once the language of the voice tag is known. The data consists of pronunciation rules, pronunciation lookup tables and decision trees and it is stored into a single data file. The rules and lookup tables are sorted according to the keys enabling a fast binary search to be applied when they are applied. The decision tree data is compressed with methods introduced in Suontausta and Tian [107]. The size of the pronunciation model data file varies between 1 kB and 100 kB, depending on the language. For very regular languages, such as Finnish and Italian, the data size is very small. Mandarin Chinese, Cantonese require relatively large data file size due to the large amount of characters whose pronunciation should be stored. The language-dependent symbol conversion data is also stored with the pronunciation modeling data.

4.6.5. Speech Synthesizer Data

The language-independent speech synthesis engine, described in Section 4.5.3, utilizes language specific and language-independent data to generate the synthesized speech. The language-independent data specifies default processing in case of non-matching rules. It also determines the way how language morphing is used (cf. Section 4.5.2). The data in the language-specific part of the synthesis data is formatted with a description language that has been packed into a binary representation. It contains data for phonetic parameters, intonation model as well as the Klatt parameter processing and tone generation. The size of the language-independent data is about 5 kB and the language-specific rules are commonly between 15 kB and 20 kB.

4.6.6. Finding the Right Data

When a new variant is generated for the given product, the problem is to collect all the data to form the voice UI language package. Looking at the previous sub-sections, pronunciation model, text processing, speech synthesis and language identification are straightforward since they mainly contain language specific data. The parts that are dependent on the actual variant can be easily formed during the variant creation process. The acoustic model processing is more difficult due to the reasons explained in Section 4.6.1. Once all the data files have been located they are copied to the device variant data package. The overall size of the voice UI language package depends on the languages in each variant and varies between 150 kB and 350 kB. This can be considered to be very compact presentation.

Chapter 5

User Interface of the Voice Dialing System

ONCE the voice UI technology is available, the voice user interface design can be started. These two phases partially overlap since certain technical issues are only uncovered once the user interface prototype is available and it undergoes extensive testing, and vice versa. This is particularly true for the language specific challenges described later in this chapter. The user interface is the link between the device and the user. A good user interface anticipates the user's intentions and next actions. An excellent user interface exceeds user's expectations on the ease-of-use and convenience of the use. These are the basic user interface design principles and can be applied to speech user interfaces in a similar manner.

The chapter starts with a short introduction to speech as a message container and how the speech signal is linked with the meaning in a language. The chapter continues with an introduction to voice user interface design in Section 5.2. Next, in Section 5.3, the user interface for voice dialing in Nokia S60 devices is described. In addition to the visual user interface, the implementation includes several software modules that are needed for smooth handling of the device contacts and the integration of the voice dialing UI with the visual UI of the device. The voice UI engines and the associated processing flow are introduced in Section 5.4. Section 5.5 continues with the introduction to the software architecture for voice dialing that links the technologies with the visual and voice dialing UI. Finally, Section 5.6 highlights some challenges of a multi-lingual voice dialing user interface. The author's contribution to this chapter is in the user interface design, application flow design and the phonebook synchronization high level algorithm design. The author has also taken part in the design of the language handling in the UI where multiple languages and writing scripts are involved. The author has not taken part in the voice dialing UI software implementation work.

5.1. SPEECH AS A MESSAGE CONTAINER

Although speech is the most common interaction method between humans, there are several implications to machines in order to be able to support the requirements of a voice user

interface design. Humans have developed the languages over the course of thousands of years. Additionally, the automation of such complex systems, as languages, requires extensive knowledge. This challenge can be represented with different layers of processing speech as a signal to convey the meaning to the listener or to the machine. The layers are shown in Figure 20.

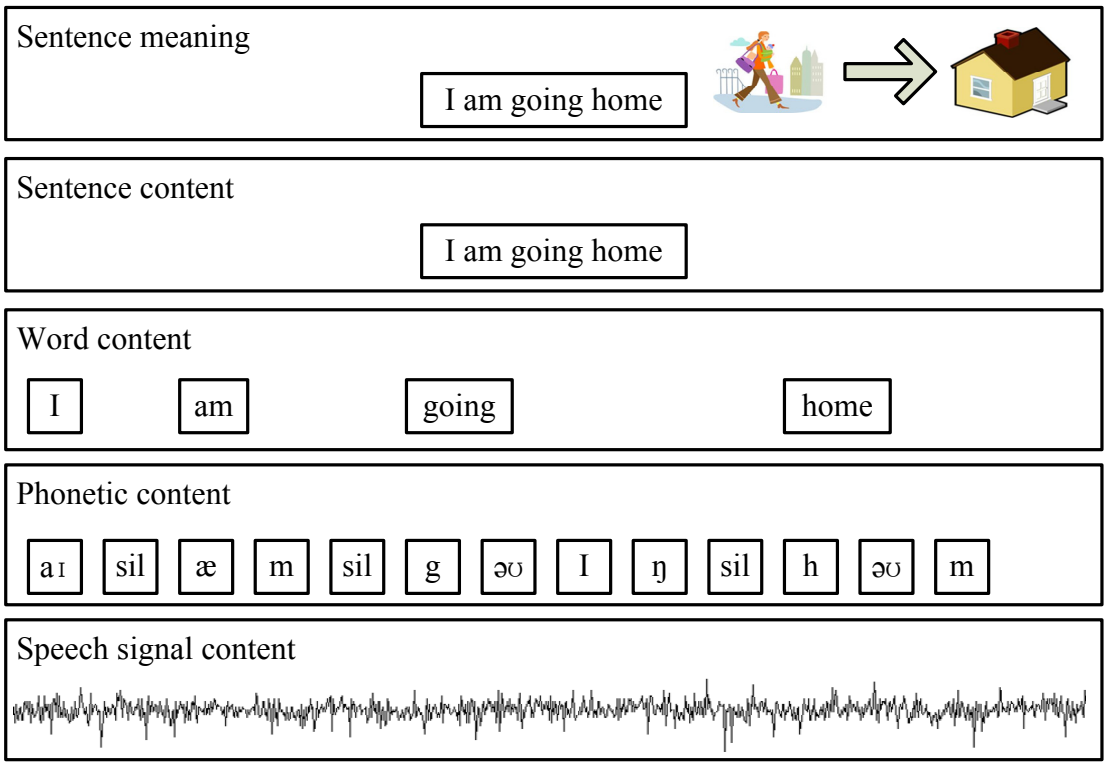

Figure 20: Levels of speech message

Speech as a signal is well known and studied signal. Its production methodology is generally known and the modeling methodologies are well established. This knowledge is utilized in e.g. speech coding where the speech signal can be compressed into a small memory space to alleviate efficient mobile networks or other speech transmission and storage systems. Without the knowledge of the source signal, i.e. speech, these applications would not be possible. Speech as a signal is in most parts language independent with rather small amounts of language specific characteristics. These differences can be taken into account when designing the speech model that is used.

On top of the speech signal is the phonetic content of the speech utterance. This is the first step towards the meaning and is a necessary step to formulate the later steps of comprehension. Although, the later steps may help in understanding even with imperfect phonetic understanding, it is the single most important information source when it comes to the meaning of the speech message. The combinations of phones and phonemes that can follow each other are language dependent and therefore contain the first level of language

specific characteristics of the speech message. The phonetic content of the example sentence of Figure 20 is given with the IPA notation.

The sounds that are uttered by the human form the words. The word level information can be decoded from the phonetic content of the message. In addition to the phonetic content, the upper level sentence information is also utilized (at least by humans). The words are commonly specific to a given language and cannot be decoded with methods developed for another language. This is very much true for humans as well: Two humans speaking different languages cannot understand each other even though the words may be understood as a sequence of certain sounds. Therefore, the phonetic content and the word level content is not enough to understand a spoken language.

Words are combined together to form sentences and phrases. The way words follow each other depends on the language and may depend on the type of information that is being delivered. In a human machine interface, the sentences are commonly formed from words with the help of a language model. The language model can help in correcting errors in the phonetic content and in the word decoding done in the lower layers of information processing.

The meaning of the spoken utterance can be decoded from the sentence. This requires additional layer of spoken language understanding to figure out fine details of the meaning. Depending on the task of voice user interface all or only some of the layers described can be present. In the multi-lingual voice dialing system the three lowest layers are included.

5.2. VOICE USER INTERFACE DESIGN

Voice user interface comprises of interaction between the human and the machine that is mainly based on the spoken language. This may happen in two directions, either from human to machine or from machine to human. The former is referred to speech recognition and the latter as text-to-speech synthesis. In addition to speech, other modalities such as display, key input, etc. may be used. Combined systems are commonly referred to as multimodal systems.

The reason to utilize speech in a user interface stems from several reasons. The majority of reasons lie within the ease of use since speech is a natural means of communication for humans. Additionally, for many user groups, such as the blind or illiterate, speech may be the only means that they can comfortably use when interacting with a device or a service. According to the World Health Organization (WHO) [25], there are 45 million blind in the world and 135 million people have low vision. This does not include temporal blindness that may be caused by environment, lack of glasses or some medical reason. Therefore, one large group of users for voice based interfaces is the blind and visually impaired. Even for people with good sight voice can provide benefit over other modalities when applied properly. It may speed up the completion of a task more reliably or generally more conveniently.

The design steps to develop a voice user interface are similar to that of a normal user interface. They both need use cases or user stories to support the design and most often concept work and usability studies. In that sense, design of a voice user interface does not differ from the design of user interfaces utilizing other modalities. The use of spoken

language brings in several possibilities and challenges that can be utilized or must be taken care of. Speech as an input modality does not provide deterministic response for different users and quite often not even for the same user. Additionally, speech user interfaces are relatively new and users do not have the same experience as opposed to using e.g. graphical user interface in a desktop PC. At this moment, there are no common methodologies of using voice user interfaces that are known by every user. Due to these, the performance of speech input technology does not always match with the expectation. This leads to severe problems in usability and task completion if it is not taken into account in the user interface design. This is to say the error conditions in a voice user interface need much more attention than those for other modalities. As an example of a voice user interface design the multi-lingual voice dialing system user interface is presented in the next section. The author's contribution is limited to minor details of the user interface, making sure that the main use case of hands-free dialing is working as expected.

5.3. SERIES 60 VOICE DIALING UI

The voice dialing UI of the Series 60 3rd edition has three main parts, the voice dialing UI, voice commands application and the phonebook voice UI. These are described next. S60 applies user centric design in the user interface design. Usability studies are commonly carried out to confirm the design applicability.

5.3.1. Phonebook Voice UI

In general, each contact stored in the device phonebook contains a voice tag that can be used to dial a telephone number associated with the contact. An exception to this is due to a mismatch in writing script and the device supported languages occurs or when the contact name does not contain any characters that can be transcribed into a pronunciation by the multi-lingual voice dialing system. Such cases are e.g. Arabic name in a Finnish device or contact name consisting of special characters. The device phonebook also shows a voice tag icon to denote the phone number that is associated with voice dialing, as seen in Figure 21. The telephone number is selected based on a predefined precedence list that considers the number type and the default voice call number if given. The device phonebook also utilizes the multi-lingual text-to-speech engine to play sample of the voice tag stored into the voice dialing system of a given phonebook item as shown in Figure 22.

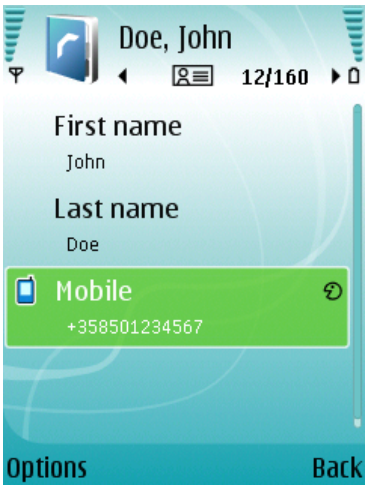

Figure 21: Voice tag icon in the phonebook

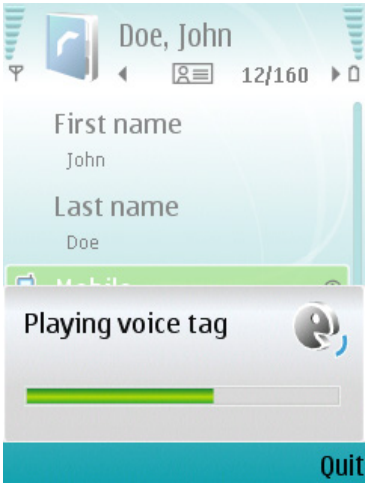

Figure 22: Voice tag playback dialog in the phonebook

5.3.2. Voice Commands Application

The Voice Commands application is responsible for automatically training the voice commands that can be used to start applications or change the device profile. The voice commands application initially contains a set of applications and profiles that can be launched or activated. All other pre-installed applications can be launched once they have been added to the list of active voice commands. Activation does not require any training actions from the user but the user simply adds the command from a list. If the pre-defined command name is not what the user prefers to use for launching the application, the user may modify the command. The user simply types in a new text string, which is then converted into a voice tag by the multi-lingual voice dialing system. An example of this is seen in Figure 23 where the user has added his/her own command "sinihammas" for toggling Bluetooth on and off.

The Voice Commands application also controls the settings associated with the voice dialing system. Currently[8], there are two settings available for the user. First, the user may enable or disable the speech synthesis feature that provides spoken feedback during voice dialing. The user may also reset the speaker adaptation (cf. Section 4.3.4) which may be useful if the main user of the device changes. In this case the working acoustic model described in Section 4.6.1 is replaced with the original acoustic model from the device ROM.

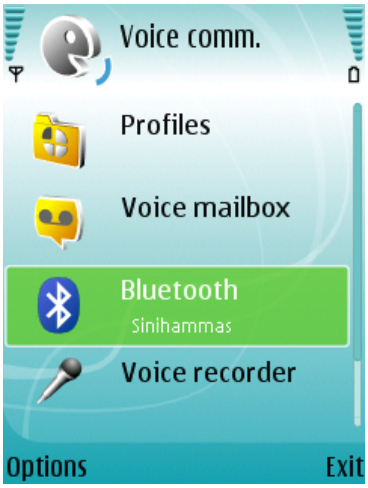

Figure 23: Voice commands application view with one user edited voice command

5.3.3. Voice Dialing UI

The target of the voice dialing application is to make hands-free call as simple as possible, using a headset. The headset can be connected to the mobile device either as a wired or a wireless accessory. In the latter case, the connection is established with the Bluetooth technology and the voice dialing is initiated with a dedicated Bluetooth voice dialing message.

Each contact can have a single voice tag that is associated with a single telephone number. The fact that the phonebook should show the location of the voice tag in the phonebook UI limited the number of voice tags to one. Also, the API available via the Symbian contact engine did not support multiple voice tags per contact. Therefore, only a single voice tag per telephone number and contact was created. As an exception to this, existing nickname field of the contact card has additional voice tag but it is calling to the same number as the default given and family name based voice tag.

Since the hands-free use was the main use case for voice dialing, there was no possibility to add extra verification steps for the result but the recognition result is accepted automatically after a short timeout. Two alternative options to automatic dial are manual verification and voice verification. The former was considered problematic since it adds a

[8] Series 60, 3rd edition

new user setting and changes the control flow of voice dialing. The latter requires special characteristics from the speech recognition engine and at the time of voice dialing UI development; this technology was not available for all supported languages.

Sometimes, the speech recognition engine can make an error and shows the wrong result to the user. This situation can lead to either calling the wrong person or the user discontinues using voice dialing since it is considered unreliable. To avoid this, the user is given option to show other results than the best selection. By selecting the correct result from the so called N-best list, the speaker adaptation can be done efficiently.

The voice commands and phonebook voice tags are recognized in parallel and the result is shown containing entries from both vocabularies. It may be confusing to the user that the result can be either a command or a name. But, taking into account that there are no limits to the voice tag content in voice commands and phonebook, this confusion was not considered to be a major issue. Also, it makes the voice dialing UI simpler since no separation is necessary.

Keyboards on mobile devices are changing significantly from one model to another. While several mobile phones have a dedicated key to launching voice dialing, there are many devices where no key can be solely assigned to voice dialing. In addition to this, there are some devices, such as Nokia N71, where the voice key is shared between voice dialing and Push over Cellular (PoC). To accommodate this, the launching of voice dialing can be configured to be done with voice key or the right selection key (RSK). In either case, a long press is needed to launch voice dialing. The difference is that long press of voice key launches voice dialing in every UI context while long press of RSK only works in the idle view. In addition to the device keys, a supported headset key can be used to launch voice dialing. The additional benefit is that there is no need to touch the device to make a phone call. This is especially convenient when using a Bluetooth headset. Also, in this case, the voice dialing can be launched while the device keypad lock is enabled.

In many cases, the user hesitates during speaking or speaks the wrong input to the voice dialing system. In these cases, the result can be wrong. To help users cope with this problem, the voice dialing system should be able to detect if an invalid input has been spoken. The technology to make this happen is not perfect and some invalid utterances cannot be detected but they are accepted as valid. On the other hand, some valid spoken sentences are rejected as invalid. To overcome this, there should be an option to adjust the rejection sensitivity.

Although the voice dialing system is multi-lingual and speaker-independent, some user's voice may be such that the satisfactory performance cannot be obtained immediately in the first use. To alleviate this problem, the voice dialing system includes automatic speaker adaptation functionality that converts the acoustic model to better match the user's voice. The speaker adaptation is efficient in improving recognition accuracy in case of heavy dialect, accent or environment mismatch. The speaker adaptation is applied to every accepted recognition result whether it is the best result or selected via the N-best view. The important requirement from the user is to always choose the correct result.

The following diagram in Figure 24 shows the voice UI control flow in detail.

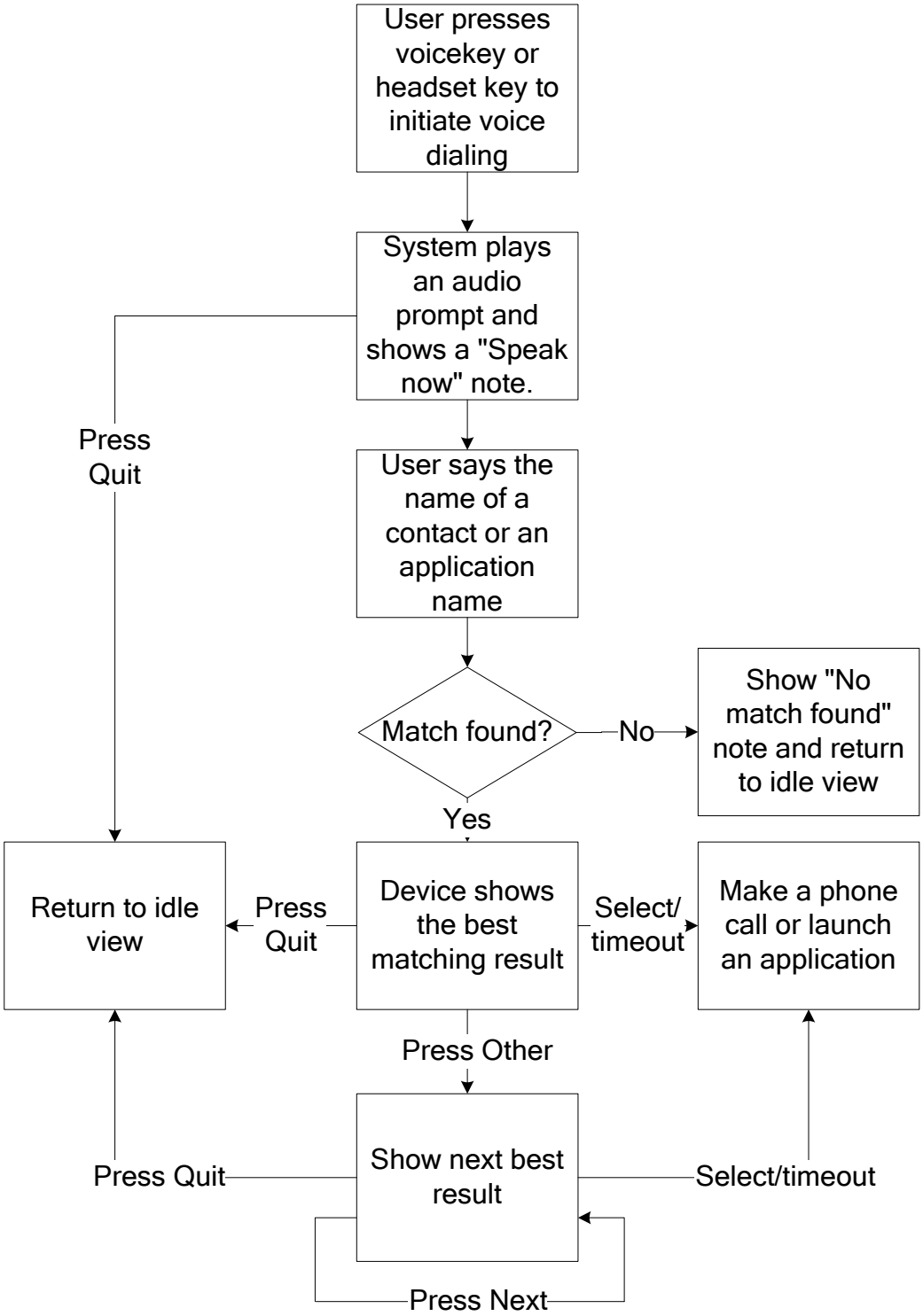

Figure 24: Voice dialing control flow

The start of voice dialing is indicated to the user via audio and visual modalities. The user hears a jingle tone that is only used by the voice dialing to indicate start of speech recognition event. The user also sees the prompt shown in Figure 25 right after the audio tone has finished

playing. The user now has five seconds to speak the command or name. The speech recognition is stopped as soon as a valid match has been found.

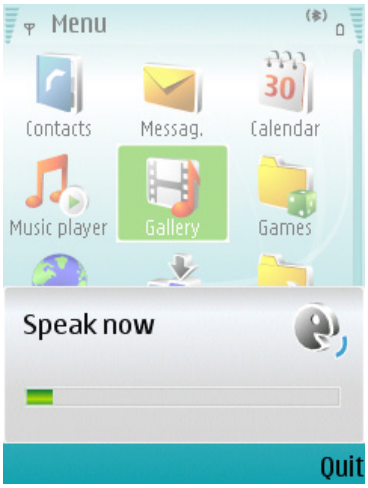

Figure 25: Voice dialing UI

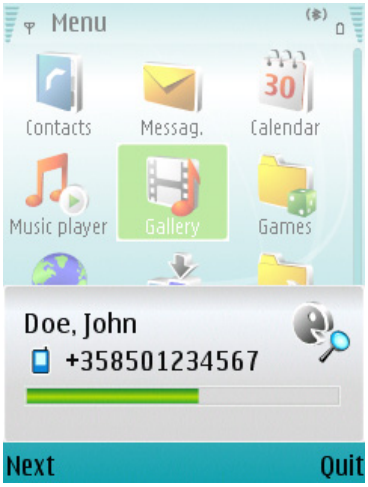

Figure 26: Result of voice dialing

The result is given to the user both by spoken audio and visual indication on the device screen. The audio is the synthesized voice tag and the result and the associated number is shown on the screen. After a short timeout, two seconds, the call is initiated or the command is executed. The result display indicates the type of the selected number with a graphical icon. For user's convenience the telephone number is also shown so that the user knows where the device is about to make a phone call.

5.3.4. Voice UI Data Handling

The voice UI data, introduced in Section 4.6, is linked with the API architecture since all file and data access operations are handled via the API layer. The speech recognition and speech synthesis engines do not have a connection to the device file system and therefore cannot

access the data by themselves. This approach was chosen to maintain portability of the voice UI engines between the Nokia Series 40 and Series 60 environments. The API layer, on the other hand, does not know about the contents of the voice UI data but only delivers the data when requested by the voice UI engines. The data can be located in either device static flash area or in the device file system.

5.3.5. Phonebook Synchronization

The voice dialing system needs to be synchronized with the device phonebook at all times. The contacts in the device may have various origins such as SIM card, vCards from memory card, PC office software via Nokia PC Suite and Internet via syncML synchronization servers. This sub-section highlights the methods used in the voice dialing system to cope with the occurring changes to the phonebook.

The synchronization is based on change events provided by the Symbian contact engine. These events are received by the voice dialing synchronization agent which in turn controls the internal contact database of the voice dialing system. The event may be one of the following: add a contact, delete a contact or modify a contact. In addition to this, the event may be unknown which commonly means an unexpected change in the device contacts. In this case, the whole voice dialing contact database is re-synchronized with the device phonebook.

A full re-synchronization is also triggered by a change of user interface language in the device. All the voice tags are re-trained with the new default language, corresponding to the user interface language. All the major changes are detected during the device boot process and individual changes are processed as they occur. The synchronization takes about 1 minute per 300 phonebook contacts, depending on the hardware of the device.

The phonebook synchronization process does not categorize the names but delivers them to the training module. All errors in the voice tag training are informed to the phonebook so that the voice tag icon can be shown or omitted in the phonebook application.

5.4. VOICE UI ENGINES

The voice UI engines can be split into four functional parts: speech recognition, speech synthesis, pronunciation modeling and language identification. These individual modules have already been introduced in the earlier chapters.

Even though the engines are separated in the block diagram, there are tight links between them. They have both functional and data dependencies. The control flow between the parts of the voice dialing engine is shown in Figure 27. Notice that the modules shown in Figure 27 do not contain any user interface modules.

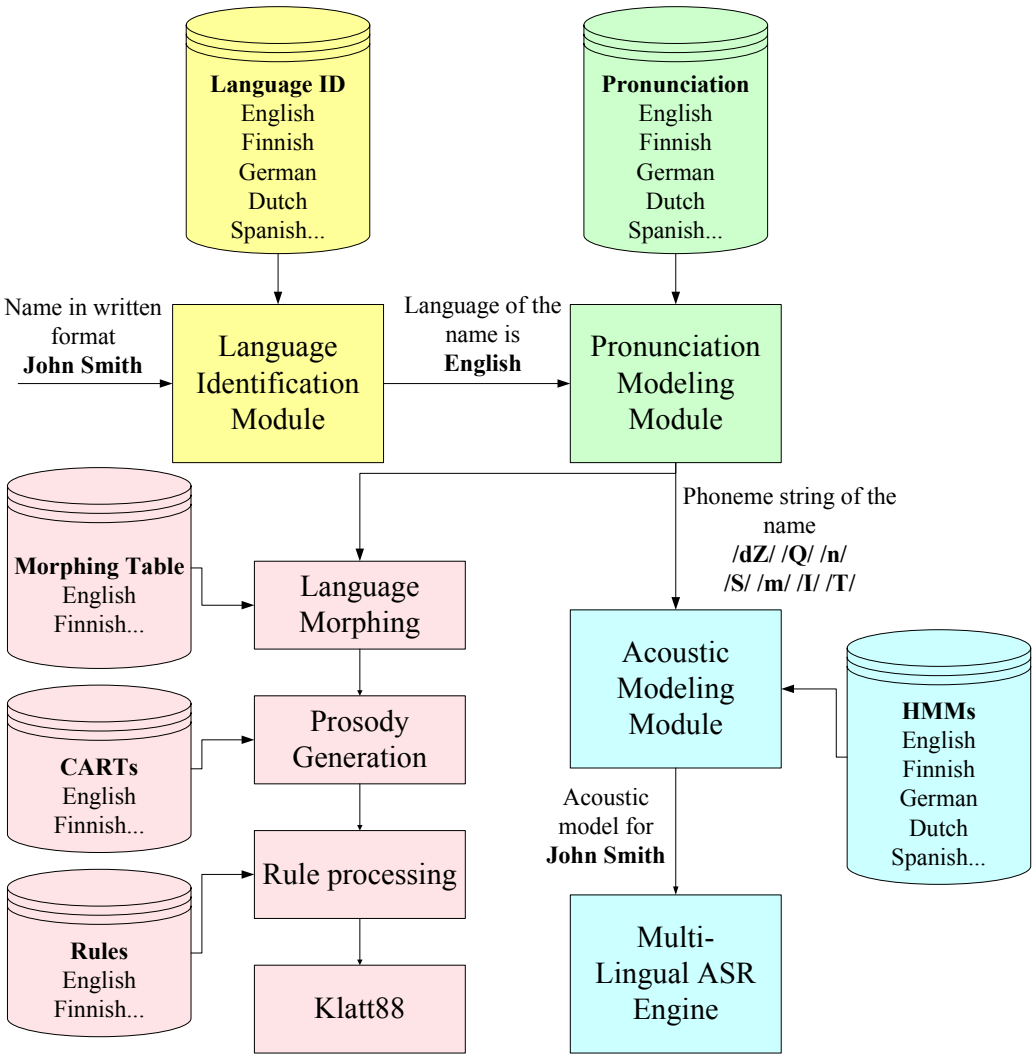

Figure 27: Detailed block diagram of the voice dialing engine components

The implementation of the engines has been done in such a way that the data can be loaded from outside of the engine layer to reduce dependency to the current operating system and hardware. One benefit of code and data separation is that the same engine code is used in the multi-lingual voice dialing system of Nokia S40 devices. The API SW layer is responsible of storing and loading the data and provides it via the API functions to the engines.

5.5. VOICE DIALING SW ARCHITECTURE

The SW architecture of the voice dialing system can be seen in Figure 28. The three layers shown are linked together with SW APIs. The application layer was already discussed in Section 5.3 and the engine layer was introduced in Section 5.4. The API layer links the user interface components to the engines that implement the necessary technologies.

There are two reasons to develop a dedicated SW API to support voice dialing rather than directly accessing the technology from the API. First, many operations that are needed in the engine layer are of no interest to the user interface. Second, the operations need to be done

in a certain order and nothing should be omitted for the system to function as expected. The SW API provides the user interface designer a set of methods that can be used without possibility to misuse the voice dialing system.

From the engine layer point of view, the most important part of the API layer is the abstraction of audio and data handling. This means that the engine layer does not physically read or write any data from/to the device file system. The engine also receives the audio data through the SW API. This makes the engine SW platform independent and it is easier to port to other platforms, if necessary. The SW API between the API layer and Engine layer provides abstraction to the voice dialing engines and therefore makes it possible to switch the implementation, if needed.

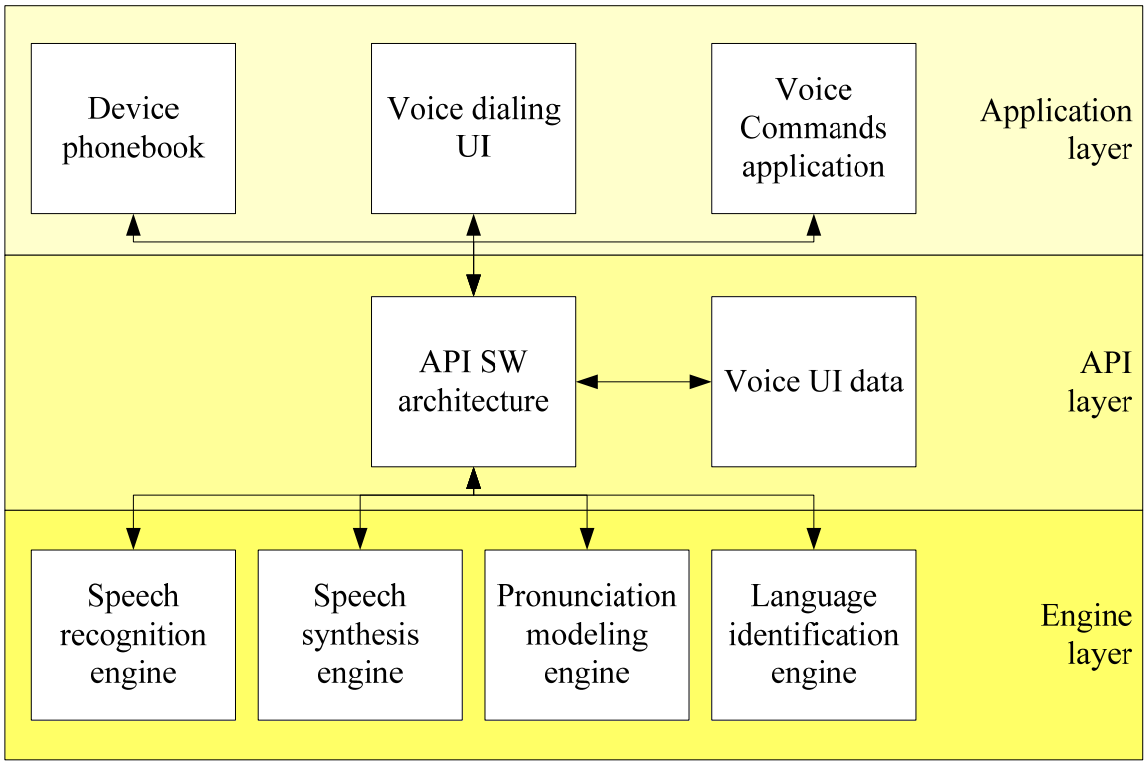

Figure 28: Voice dialing SW architecture diagram

5.6. UI CHALLENGES RELATED TO LANGUAGES AND MULTI-LINGUALITY

The mobile device can commonly handle a wide range of writing scripts. For geographical and memory reasons some fonts are included only in the areas where the font is needed. The Chinese font is only included in the mobile devices sold in Chinese speaking world and therefore a Western mobile device cannot handle Chinese text.

Each device contains a set of languages that are also directly supported by the voice dialing UI. In areas where foreign languages are commonly used, this may pose a problem to the voice dialing system. The user may have contact names that are of different origin from the user's own native language. The voice dialing system can partially cope with languages that are included in the device but other languages cannot be handled properly.

An optimal multi-lingual voice dialing system would have the capability to recognize and synthesize every language within a single device. To achieve this, there should be a perfect language identification system that can classify the language of each name into the correct language. Furthermore, there should be a method to improve the recognition accuracy of non-native names that are inherent in a mobile device. The device does not know how properly and with what kind of accent the user is going to pronounce each name. The problem of accented speech was studied in great detail by Bartkova and Jouvet [7]. They concluded that both non-native training data and pronunciation modeling adaptation may be needed to achieve adequate recognition accuracy on non-native speech. They also emphasized the need for speaker adaptation.

When a name is introduced to the system it is processed in a similar manner, independently of the original language of the name. The methods that are available to cope with the non-native and/or non-supported language names are the following. First, the multi-lingual voice dialing system includes a Romanization scheme that converts most of the foreign script letters into Latin and can generate a pronunciation in most of the languages. Second, to further improve the probability of including the correct pronunciation, the multi-lingual voice dialing system trains several voice tags per name according to the UI language and automatic language identification. With these methods, the support of non-native names can be improved. All of the challenges cannot be solved with these and the remaining challenges are discussed next.

5.6.1. Speech Recognition

Due to memory, computation and recognition accuracy reasons, the user is expected to pronounce name in a certain way. This means that, in most languages, the given name should be spoken before family name. For some languages, the preferred order is opposite and it is normal to pronounce the family name before the given name. These languages are included in the system and for them the voice dialing system trains the voice tag in the preferred order.

To overcome the question of order of names, the voice dialing system could also train the voice tags with either order. Alternatively, the user could choose the preferred order as in the device phonebook when viewing the contacts. In the latter case, the challenge is that there is no direct visual indication of the order in the voice UI and the user may not remember the order that was chosen in the system settings. The former case makes the user's task easier but increases the memory footprint of the system and also reduces the recognition accuracy. Due to these reasons, no user setting has been enabled in the multi-lingual voice dialing system.

The display texts in a mobile device are designed to match the current display in each device. At the moment, QVGA resolution display is the most common with 240 (width) by

320 (height) pixels. To fit all texts in all languages in every display location, the localized texts need to be either designed to fit or to be cut to fit. The latter case is especially problematic for voice user interfaces. When abbreviating a term for display, the abbreviation is commonly not known beforehand. The voice dialing system should, nevertheless, be able to generate a voice tag for the applications included in the voice commands application. The terms that are not abbreviated cause no problem since the pronunciation model provides a pronunciation to all native words. Abbreviations should be known beforehand so that the full form pronunciation can be provided for those terms. This requires a link between the localization process and the pronunciation model design of every language. To cope with this, a recent implementation of the multi-lingual voice dialing system includes two text strings for each voice command: One for the text to be displayed and one for the spoken form. The spoken form is used when training the voice tag and when playing the sample with text-to-speech. It is also mandated that the spoken string should not contain abbreviations to make the task of pronunciation modeling easier. Another type of problematic cases arises from acronyms that are used in e.g. Instant Messaging. It is commonly shortened as IM in display texts. The spoken term should be expanded for the user's convenience.

Another difficult situation arises in languages that use a non-Latin script. Most of these languages can be Romanized to Latin using either a standard or a non-standard method. In either case, the language identification module should detect these languages from the Romanized format. This would be the only way of providing a correct pronunciation to the recognition system (and to the speech synthesis as well). The problem is that Romanized words would be very difficult to distinguish from English words that are commonly used in non-Latin languages. Latin is also used for writing any other Latin name, too.

Consider an example from Thai where the user writes all his/her Thai contacts with Thai script. He/she also has foreign friends whose name cannot be written in Thai script. Therefore, they need to be written in Latin letters. In this case, the language identification properly detects all Thai words and considers all other words to be English (since the device probably only supports English as Latin language). For Thai words, correct language is chosen and for Latin words, English.

Now, if the user wrote the names with some Romanized format of Thai instead of native writing script, the following problem would emerge. The system should detect between the Romanized Thai names and real English names. For short names that are commonly used in Thailand, this would result in a lot of wrong decisions for the names using Latin script. Due to this, a decision was made to support only native writing script in most of the non-Latin languages.

5.6.2. Speech synthesis

A speech synthesizer is used for the spoken feedback of the recognition result to the user. It can be launched from the voice commands application or in the device phonebook to playback the voice tag to hear a sample of what the recognizer expects the user to say.

The speech synthesis language should be the correct but due to limitation in the accuracy of the language identification from text, the chosen language would be very often wrong. Therefore, a compromise has been chosen and the UI language is always used for spoken feedback. This makes the feedback consistent while sometimes providing wrong and even funny pronunciation of the word in question. There is only one exception to this and it occurs when the name is written in a writing script that cannot be transcribed with the current UI language. This can happen e.g. in the English UI when the device contains names written in Chinese characters. In this case, the system detects that no spoken output can be generated in the UI language and only in this case the language identification is used to find a pronunciation in the most likely language.

A few research options have been identified to provide a more robust language selection for the text-to-speech feedback. These include the decision made by the speech recognizer out of the several language variants. Also, the text based language identification can sometimes provide quite reliable estimate of the language. Combination of these two information sources together with the knowledge of the UI language could provide better spoken feedback accuracy for the user. Finally, one option is that the user can adjust the language of each contact and this way helps the voice dialing system to provide the feedback in correct language.

Chapter 6

Conclusions

The development of a multi-lingual voice dialing system requires competences from various fields of speech and language processing. Mastering technologies such as multi-lingual phonetics, multi-lingual pronunciation modeling, speech recognition and speech synthesis, is essential. In addition to laying out the basic technical frameworks that can handle the individual requirements for each technology, the technologies should be combined in such a way that the user interfaces for the desired applications can be implemented on the target hardware platform. The hardware platform has restrictions on computational capabilities as well as on available memory to execute the chosen realizations for the technologies. Therefore, the design and implementation of a multi-lingual voice dialing system is usually a trade-off between system performance, ease of implementation and complexity.

This thesis introduced a multi-lingual voice dialing system that is used in Nokia mobile devices. The system is capable of recognizing and synthesizing speech in more than 50 different languages. The contribution of the author supported various fields of expertise in the overall system. To date, this is the only speaker-independent speech recognition and speech synthesis system that supports more than 50 languages. This is achieved using a single software code base and data driven language configurations. At the time of writing this thesis, the implementation is included in over forty S60 and S40 devices sold world-wide.

It was shown that a multi-lingual voice dialing system can be developed to include support for tone recognition. The method described increased the computational and memory requirements only slightly while providing high tone recognition accuracy compared to other alternatives. The study on Chinese names, on the other hand, concluded that tone recognition is not necessary in a name dialing application. Taking these two into account, tone recognition support was left out of the final multi-lingual voice dialing system.

The minimization of the speech recognizer's computational complexity was achieved through data quantization and frame-rate reduction. The methods introduced can reduce the CPU load by as much as 50% in the case of a speaker-dependent recognition task. For speaker-independent voice dialing, the combination of frame-rate reduction and parameter quantization were the key success factors. It was also shown that quantization can be applied

to most parts of the speech recognition data path to reduce both memory and CPU requirements.

Furthermore, an online garbage model was introduced with adaptation capability to the use environment. While Nokia speaker-dependent voice dialing system relied on the garbage model, it was found not to be suitable for speaker-independent voice dialing technology. Instead, the multi-lingual speaker-independent voice dialing system utilizes a trained background model.

The system's capability to handle out-of-vocabulary words and incorrect user input has an effect on the overall usability. The methods proposed for keyword spotting showed that with proper user interface design, both speaker-dependent and speaker-independent keyword spotting can be implemented without sacrificing the system performance. The basic rejection methods developed are used in the multi-lingual voice dialing system.

The language morphing method enabled a quick expansion of language support in the speech synthesis technology. A simple phoneme mapping and proper selection of target language provided adequate speech quality. This method is used in the multi-lingual speaker-independent voice dialing system. For some of the originally morphed languages, a native TTS has been developed to maximize speech quality.

The various complexity reduction methods presented in the thesis paved the way to early implementation of speaker-independent voice dialing. Almost three years after the introduction, it is still without much competition. This is true both in terms of performance and simplicity. It should be noted, though, that design of the user interface should be done together with the developers of the ASR and TTS technology so that all the possibilities and limitations are covered in the UI design. Otherwise, the system will not function as expected.

In addition to the basic technologies for voice dialing, the configuration of the speech recognizer and speech synthesizer has to accommodate requirements from several products. The data driven approach for language functionality enabled a flexible configuration mechanism that combines individual language data into each new language variant, when required. This feature is of extreme importance when several new devices with large volume sales per model are shipping every year.

A common drawback of any speech system is that its use is limited to a given application or a set of applications. The speech synthesizer and speech recognizer are commonly tailored to the applications and extension to new applications domains usually requires additional efforts in the technology development. This is also true for the multi-lingual voice dialing system. One prospective mobile application is the mobile dictation of messages and e-mails. This application requires extensive development effort to turn the speech recognition and speech synthesis technologies from voice dialing application to support the application of dictating text.

The technologies needed to build a successful multi-lingual voice dialing system were presented in this thesis. All the sub-areas, such as pronunciation modeling, speech recognition and speech synthesis, are evolving all the time. At the same time, the computational resources found in a mobile platform are increasing. It is obvious that more and more complex speech

processing systems can be built into mobile devices. One recent area of development is multimodality where users may combine several input and output technologies, such as keyboard and speech, to interact with the device. Multimodal systems are likely to offer the user the possibility to choose among supported interaction methods or to utilize several methods simultaneously to achieve the target faster, more conveniently or more safely, depending on the application and the user.

Chapter 7

Bibliography

[1] T. Arai, M. Pavel, H. Hermansky and C. Avendano, "Intelligibility of Speech with Filtered Time Trajectories of Spectral Envelopes", in *Proceeding of International Conference on Spoken Language Processing*, pp. 2490-2493, Philadelphia, USA, 1996

[2] B. Atal, "Effectiveness of linear prediction characteristics of the speech wave for automatic speaker identification and verification", *Journal of Acoustic Society of America*, Vol. 55, No. 6, pp. 1304-1312, 1974

[3] B. Atal and S. Hanauer, "Speech Analysis and Synthesis by Linear Prediction of the Speech Wave", *Journal of the Acoustic Society of America*, Vol. 50, issue 2B, pp. 637-655, 1971

[4] J.-J. Aucouturier and M. Sandler, "Segmentation of Musical Signals Using Hidden Markov Models", in *Proceedings of the 110th Convention of Audio Engineering Society*, Amsterdam, The Netherlands, May 12-15, 2001

[5] L. Badino, C. Barolo and S. Quazza, "Language Independent Phoneme Mapping For Foreign TTS", in *Proceedings of 5th ISCA speech synthesis workshop*, pp. 217-218, Pittsburgh, PA, USA, 2004

[6] M. J. Ball and J. Rahilly, *Phonetics, the science of speech*, Oxford University Press, pp. 101-122, New York, USA, 1999

[7] K. Bartkova and D. Jouvet, "On Using Units Trained on Foreign Data for Improved Multiple Accent Speech Recognition", *Speech Communication*, Vol. 49, pp. 836-846, 2007

[8] L. E. Baum, T. Petrie, G. Soules and N. Weiss, "A maximization technique occurring in statistical analysis of probabilistic functions in Markov chains", *The Annals of Mathematical Statistics*, Vol. 41, No. (1), pp. 164-171, 1970

[9] L. E. Baum, "An Inequality and Associated Maximization Technique in Statistical Estimation for Probabilistic Functions of Markov Processes", *Inequalities*, Vol. 3, pp. 1-8, 1972

[10] A. Black and T. Schultz, "Speaker Clustering for Multilingual Synthesis", in *ISCA Workshop on Multilingual Speech and Language Processing*, Stellenbosch, South Africa, April 9-11, 2006

[11] H. Bourlard, B. D'hoore and J.-M. Boite, "Optimizing Recognition and Rejection Performance in Wordspotting systems", in *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing*, Vol. 1, pp. 373-376, Adelaide, Australia, 1994

[12] L. Breiman et al., *Classification and Regression Trees*, Wadsworth Inc., Belmont, CA, USA, 1984

[13] N. Campbell and A. Black, "Prosody and the Selection of Source Units for Concatenative Synthesis", *Progress in Speech Synthesis*, Springer-Verlag, New York, 1997

[14] N. Campbell, "TALKING FOREIGN - Concatenative Speech Synthesis and the Language Barrier", in *Proceedings of Eurospeech*, pp. 337-340, 2001

[15] Cao Wenjie, Zong Chengqing, Xu Bo, J. Iso-Sipilä, "Chinese Person Name Identification Based on Rules and Statistics", in *Proceedings of International Symposium on Chinese Spoken Language Processing (ISCSLP)*, Taipei, Taiwan, August, 2002

[16] Cao Wenjie, Xu Bo and J. Iso-Sipilä, "Linguistic and Acoustic Analysis of Chinese Person Names", in *Proceedings of International Symposium on Chinese Spoken Language Processing (ISCSLP)*, Taipei, Taiwan, August, 2002

[17] W. Cavnar and J. Trenkle, "N-Gram-Based Text Categorization", in *Proceedings of Third Annual Symposium on Document Analysis and Information Retrieval*, pp. 161-175, UNLV Publications/Reprographics, Las Vegas, NV, USA, 1994

[18] D. Chazan, R. Hoory, Z. Kons, D. Silberstein and A. Sorin, "Reducing the Footprint of the IBM Trainable Speech Synthesis System", in *Proceedings of the International Conference on Spoken Language Processing*, pp. 2381-2384, Denver, 2002

[19] C. J. Chen, R. Gopinath, M. Monkowski, M. Picheny and K. Shen, "New Methods in Continuous Mandarin Speech Recognition", in *Proceedings of Eurospeech*, pp. 1543-1546, Rhodos, Greece, 1997

[20] *Analysis and Statistics of characters in surnames' and given names' usage*, Chinese Academy of Social Sciences Yuwen Press (in Chinese), 1990

[21] M. Cooke, P. Green, L. Josifovski and A. Vizinho, "Robust Automatic Speech Recognition with Missing and Unreliable Acoustic Data", *Speech Communication*, Vol. 34, No. 3, pp. 267-285, Elsevier, 2001

[22] W. Daelemans and A. van den Bosch, "Language-Independent Data-Oriented Grapheme-to-Phoneme Conversion", *Progress in Speech Synthesis*, pp. 77-89, Springer-Verlag, New York, 1997

[23] S. Davis and P. Mermelstein, "Comparison of Parametric Representations for Monosyllabic Word Recognition in Continuously Spoken Sentences", *Readings in Speech Recognition*, pp. 65-74, Morgan Kaufmann Publishers Inc., 1990

[24] Y. Deng, M. Mahajan and A. Acero, "Estimating Speech Recognition Error Rate without Acoustic Test Data", in *Proceedings of the Eurospeech Conference*, pp. 929-932, Geneva, Switzerland, September, 2003

[25] A. Foster, J. Humphries, I. Kocur, R. Pararajasegaram, U. Raman, N. Rao, S. Resnikoff and H. Taylor (editors), "State of the World's Sight", *VISION 2020: the Right to Sight 1999-2005*, International Agency for the Prevention of Blindness, International Health Organization, 2005

[26] E. Garcia, E. Mengusoglu and E. Janke, "Multilingual Acoustic Models for Speech Recognition in Low-Resource Devices", in *Proceedings of the International Conference on Acoustics, Speech and Signal Processing*, Vol. 4, pp. 981-984, Honolulu, USA, 2007

[27] Ding Guohong, Wang Xia, Cao Yang, Ding Feng and Tang Yuezhong, "High Quality Thai Text-to-Phoneme Converter", US Patent application 20060259301, 2006

[28] Dong Yuan, Wang Xia, J. Iso-Sipilä, O. Viikki, I. Kiss and J. Tian, "Speaker- and Language-Independent Speech Recognition in Mobile Communication Systems", in *Proceedings of the National Conference on Man-machine Communication*, Shenzhen, China, 2001

[29] T. Dutoit, *An Introduction to Text-to-speech Synthesis*, Kluwer Academic Publishers, Dordrecht, 1997

[30] European Language Resources Association (ELRA), www.elra.info

[31] ETSI DES/STQ-00008a, Speech Processing, Transmission and Quality Aspects (STQ - Distributed speech recognition - Advanced front-end feature extraction algorithm - Compression algorithms

[32] K. Filali, X. Li and J. Bilmes, "Data-driven Vector Clustering for Low-memory Footprint ASR", in *Proceedings of the International Conference on Spoken Language* Processing, pp. 1601-1604, Denver, USA, 2002

[33] J. L. Flanagan, *Speech Analysis, Synthesis and Perception*, Springer-Verlag, 1972

[34] H. Fujisaki and H. Kawai, "Realization of Linguistic Information in the Voice Fundamental Frequency Contour of the Spoken Japanese", in *Proceedings of the International Conference on Acoustics, Speech and Signal Processing*, Vol. 1, pp. 663-666, USA, 1988

[35] M. Gales and S. Young, "Cepstral Parameter Compensation for HMM Recognition in Noise", *Speech Communication*, Vol. 12, pp. 231-239, 1993

[36] J. L. Gauvain and C.-H. Lee, "Maximum a Posteriori Estimation of Multivariate Gaussian Mixture Observations of Markov Chains", *IEEE Transactions on Speech and Audio Processing*, Vol. 2, No. 2, pp. 291-298, April, 1994

[37] R. Hariharan, J. Häkkinen and K. Laurila, "Robust End-of-utterance Detection for Real-time Speech Recognition Applications", in *Proceedings of the International Conference on Acoustics, Speech and Signal Processing*, pp. 249-252, Salt Lake City, UT, USA, 2001

[38] M. Harju, P. Salmela, J. Leppänen, O. Viikki and J. Saarinen, "Comparing Parameter Tying Methods for Multilingual Acoustic Modeling", in *Proceedings of Eurospeech*, pp. 2729-2732, Aalborg, Denmark, 2001

[39] S. Haykin, *Neural Networks - A Comprehensive Foundation*, Prentice Hall, 2nd ed., 1998

[40] H. Hermansky, "Perceptual Linear Prediction (PLP) Analysis of Speech", *Journal of Acoustic Society of America*, Vol. 87, No. 4, pp. 1738-1752, 1990

[41] H. Hermansky, N. Morgan, A. Bayya and P. Kohn, "RASTA-PLP speech analysis technique", in *Proceedings of the International Conference on Acoustics, Speech and Signal Processing*, Vol. 1, pp. 121-124, 1992

[42] Hidden Markov Model toolkit, http://htk.eng.cam.ac.uk/

[43] J. N. Holmes, *Speech Synthesis and Recognition*, Van Nostrand Reinhold (UK), 1988

[44] D. Huffman, "A Method for the Construction of Minimum-Redundancy Codes", *Proceedings of the IRE*", pp.1098-1101, Vol. 40, Issue 9, 1952

[45] D. Huggins-Daines and A. Rudnicky, "A Constrained Baum-Welch Algorithm for Improved Phoneme Segmentation and Efficient Training", in *Proceedings of Interspeech*, Pittsburgh, USA, 2006

[46] A. Hunt and A. Black, "Unit Selection in a Concatenative Speech Synthesis System Using a Large Speech Database", in *Proceedings of the International Conference on Acoustics, Speech and Signal Processing*, Vol. 1, pp. 373-376, Atlanta, USA, 1996

[47] E. Ifeachor and B. Jervis, *Digital Signal Processing - a Practical Approach*, pp. 491-540, Addison-Wesley Publishing Company, 1993

[48] B. Imperl, Z. Kačič, B. Horvat and A. Žgank, "Clustering of triphones using phoneme similarity estimation for the definition of a multilingual set of triphones", *Speech Communication*, Vol. 39, Issue 3-4, pp. 353-366, 2003

[49] The International Phonetic Association, *Handbook of the International Phonetic Association (IPA)*, Cambridge University Press, Cambridge, UK, 1999

[50] J. Iso-Sipilä, K. Laurila and P. Haavisto, "Optimal Adaptive Garbage Modeling in Speech Recognition", in *Proceedings of the Nordic Signal processing Symposium (NORSIG '96)*, Helsinki, Finland, 1996

[51] J. Iso-Sipilä, "Data-driven Filtering of Cepstral Time Trajectories for Robust Speech Recognition", United States Patent 7035797, 2006

[52] J. Iso-Sipilä, K. Laurila, R. Hariharan and O. Viikki, "Hands-Free Voice Activation in Noisy Car Environment", in *Proceedings of Eurospeech*, pp. 2375-2378, Budapest, Hungary, 1999

[53] J. Iso-Sipilä, "Speech Recognition Complexity Reduction Using Decimation of Cepstral Time Trajectories", in *Proceedings of the European Signal Processing Conference (EUSIPCO 2000)*, Tampere, Finland, September, 2000

[54] J. Iso-Sipilä, M. Moberg and O. Viikki, "Multi-Lingual Speaker-Independent Voice User Interface for Mobile Devices", in *Proceedings of International Conference on Acoustics, Speech and Signal Processing (ICASSP 2006)*, pp. 1081-1084, Toulouse, France, 2006

[55] ISO 639-1:2002, *Codes for the representation of names of languages -- Part 1: Alpha-2 code*, International Organization for Standardization, http://www.infoterm.info/standardization/iso_639_1_2002.php, 2002

[56] ISO 7098:1991, *Romanization of Chinese*, International Organization for Standardization, 2002

[57] F. Itakura, "Minimum Prediction Residual Principle Applied to Speech Recognition", in *IEEE Transactions on Acoustics, Speech and Signal Processing*, Vol. 23, No. 1, pp. 67-72, February 1975

[58] C. Jankowski, H.-D.H. Vo and R. Lippmann, "A Comparison of Signal Processing Front Ends for Automatic Word Recognition", in *IEEE Transactions on Speech and Audio Processing*, Vol. 4, No. 4, pp. 286-293, July 19995

[59] F. Jelinek, *Statistical Methods for Speech Recognition*, MIT Press, 1998

[60] K. J. Jensen and S. Riis, "Self-Organizing Letter Code-Book for Text-to-Phoneme Neural Network Model", in *Proceedings of International Conference on Spoken Language Processing*, Vol. 3, pp. 318-321, Beijing, China, 2000

[61] Heng Ji and Zhensheng Luo, "A Chinese Name Identifying System Based on Inverse Name Frequency Model and Rules", in *Proceedings of 2001 IEEE Symposium on Natural Language Processing and Knowledge Engineering (NLPKE01)*, Arizona, USA, 2001

[62] T. Joachims, "Text Categorization with Support Vector Machines: Learning with Many Relevant Features", in *Proceedings of the 10th European Conference on Machine Learning*, pp. 137-142, 1998

[63] B-H Juang and S. Katagiri, "Discriminative Learning for Minimum Error Classification Pattern Recognition", *IEEE Transactions on Signal Processing*, Vol. 40, No. 12, pp. 3043-3054, 1992

[64] J. Junkawitsch and H. Höge, "Keyword Verification Considering the Correlation of Succeeding Feature Vectors", in *Proceedings of International Conference on Acoustics, Speech and Signal Processing*, pp. 221-224, Seattle, USA, 1998

[65] J.C. Junqua, "The Lombard reflex and its role on human listeners and automatic speech recognizer", Journal of Acoustical Society of America, Vol. 93, pp. 510-524, 1993

[66] N. Kanedera, T. Arai, H. Hermansky and M. Pavel, "On the Importance of Various Modulation Frequencies for Speech Recognition", in *Proceedings of Eurospeech*, pp. 1079-1082, Rhodes, Greece, 1997

[67] DS Kim, SY Lee and R. Kil, "Auditory Processing of Speech Signals for Robust Speech Recognition in Real-World Noisy Environments", in *IEEE Transactions on Speech and Audio Processing*, Vol. 7, No. 1, pp. 55-69, January 1999

[68] I. Kiss and M. Vasilache, "Low Complexity Techniques for Embedded ASR Systems", in *Proceedings of the International Conference on Spoken Language Processing*, pp. 1593-1596, Denver, USA, 2002

[69] E. Klabbers and K. Stöber, "Creation of Speech Corpora for the Multilingual Bonn Open Synthesis System", *4th ISCA Tutorial and Research Workshop on Speech Synthesis*, pp. 23-27, Pitlochry, Scotland, 2001

[70] D. Klatt, "Software for a Cascade/Parallel Formant Synthesizer", in *Journal of the Acoustical Society of America*, Vol. 67, No. 3, pp. 971-995, 1980

[71] D.Klatt and L. Klatt. "Analysis, Synthesis, and Perception of Voice Quality Variations Among Female and Male Talkers", in *Journal of the Acoustical Society of America*, Vol. 87, No. 2, pp. 820-857, 1990

[72] S. Kunzmann, V. Fischer, J. Gonzalez, O. Emam, C. Günther and E. Janke, "Multilingual Acoustic Models for Speech Recognition and Synthesis", in *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 745-748, Montreal, Canada, 2004

[73] T. Köhler, C. Fügen, S. Stüker and A. Waibel, "Rapid Porting of ASR-Systems to Mobile Devices", in *Proceedings of the Interspeech*, pp. 233-236, Lisbon, Portugal, 2005

[74] K. Laurila, "Noise Robust Speech Recognition with State Duration Constraints", in *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 871-874, Munich, Germany, 1997

[75] K. Laurila, M. Vasilache and O. Viikki, "A Combination of Discriminative and Maximum Likelihood Techniques for Noise Robust Speech Recognition", in *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing*, Vol. 1, pp. 85-88, Seattle, USA, 1998

[76] C.-H. Lee, B.-H. Juang, F. Soong and L. Rabiner, "Word Recognition Using Whole Word and Subword Models", in *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing*, Vol. 1, pp. 683-686, UK, May 1989

[77] C.J. Legetter and P. Woodland, "Maximum Likelihood Linear Regression for Speaker Adaptation of Continuous Density Hidden Markov Models", *Computer speech & Language*, Vol. 9, No. 2, pp. 171-185, 1995

[78] Linguistic data consortium, Hosted by University of Pennsylvania, www.ldc.upenn.edu

[79] *List of Various Families Surname*, http://cn.netor.com

[80] P. Lockwood and J. Boudy, "Experiments with a Nonlinear Spectral Subtractor (NSS), Hidden Markov Models and the Projection, for Robust Speech Recognition in Cars", *Speech Communication*, Vol. 11, Issue 2-3, pp. 215-228, 1992

[81] F. Malfrère, O. Deroo, T. Dutoit and C. Ris, "Phonetic alignment: speech synthesis-based vs. Viterbi-based", *Speech Communication*, Vol. 40, pp. 503-515, 2003

[82] R. Mammone, X. Zhang and R. Ramachandran, "Robust Speaker Recognition: a Feature-Based Approach", *IEEE Signal Processing Magazine*, Vol. 13, Issue 5, September 1996

[83] C. Manning and H. Schütze, *Foundations of Statistical Natural Language Processing*, The MIT Press, 1999

[84] M. Moberg and O. Viikki, "Optimizing Speech Synthesizer Memory Footprint through Phoneme Set Reduction", in *Proceedings of IEEE 2002 Workshop on Speech Synthesis*, pp. 171-174, Santa Monica, 2002

[85] M. Moberg, K. Pärssinen and J. Iso-Sipilä, "Cross-Lingual Phoneme Mapping for Multilingual Synthesis System", in *Proceedings of the Interspeech*, pp. 1029-1032, Jeju Island, South Korea, 2004

[86] M. Moberg, "Conversion of Number into Text and Speech", International Patent Application PCT/FI2006/050345, 2006

[87] M. Mohri, F. Pereira and M. Riley, "Weighted Finite State Transducers in Speech Recognition", *Computer Speech and Language*, Vol. 16, No. 1, pp 69-88, January 2002

[88] S. Narayanan and A. Alwan, *Text to Speech Synthesis: New Paradigms and Advances*, Prentice Hall, 2004

[89] P. Nguyen, P. Gelin, J-C. Junqua and J-T. Chien, "N-best Based Supervised and Unsupervised Adaptation for Native and Non-native Speakers in Cars", in *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 173-176, Vol. 1, 1999

[90] T. Niesler, "Language-dependent state clustering for multilingual acoustic modeling", *Speech Communication*, Vol. 49, pp. 453-63, 2007

[91] D. Palmer, "A Trainable Rule-based Algorithm for Word Segmentation", in *Proceedings of the 35th Annual Meeting on Association For Computational Linguistics*, pp. 321-328, Madrid, Spain, July, 1997

[92] K. Pärssinen and M. Moberg, "Multilingual Data Configurable Text-to-Speech System for Embedded Devices", in *ISCA Workshop on Multilingual Speech and Language Processing*, Stellenbosch, South Africa, April 9-11, 2006

[93] L. R. Rabiner, "A tutorial on hidden Markov models and selected applications in speech recognition", in *Proceedings of the IEEE*, Vol. 77, No. 2, pp. 257-285, February, 1989

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

[95] K. Rao and P. Yip, *Discrete Cosine Transform: Algorithms, Advantages, Applications*, Academic Press Professional, Inc., 1990

[96] R. Rose and D. Paul, "A Hidden Markov Model Based Keyword Recognition System", in *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 129-132, Albuquerque, USA, 1990

[97] SAMPA (Speech Assessment Methods Phonetic Alphabet) homepage. http://www.phon.ucl.ac.uk/home /sampa/home.htm

[98] *SAMPA-C definition developed by Chinese Academy of Social Sciences, http://ling.cass.cn/yuyin/english/sampac/sampac.htm*

[99] S. Seneff, "A Joint Synchrony/Mean-rate Model of Auditory Speech Processing", *Readings in Speech Recognition*, pp. 101-111, Morgan Kaufmann Publishers Inc., USA, 1990

[100] T. Schultz and K. Kirchhoff, *Multilingual Speech Processing*, Elsevier, 2006

[101] H. Sheikhzadeh, E. Cornu, R. Brennan and T. Schneider, "Real-Time Speech Synthesis on an Ultra Low-Resource, Programmable System", in *Proceedings of the International Conference in Acoustics, Speech and Signal Processing*, Vol. 1, pp. 433-436, Orlando, Florida, USA, 2002

[102] R. Siemund, H. Höge, S. Kunzmann and K. Marasek, "SPEECON - Speech Data for Consumer Devices", in *Proceedings of the Second International Conference on Language Resources and Evaluation*, Vol. 2, pp. 883-886, Athens, Greece, 2000

[103] *Speech Application Language Tags*, SALT Forum, http://www.saltforum.org

[104] *SpeechDat website*, http://www.speechdat.org/

[105] R. Sproat, C. Shih, W. Gale and N. Chang, "A Stochastic Finite-State Word-Segmentation Algorithm for Chinese", *Computational Linguistics*, Vol. 22, No. 3, pp. 377-404, 1996

[106] J. Suontausta and J. Häkkinen, "Decision Tree Based Text-to-Phoneme Mapping for Speech Recognition", in *Proceedings of the International Conference on Spoken Language Processing*, Vol. 2, pp. 831-834, Beijing, China, 2000

[107] J. Suontausta and J. Tian, "Low Memory Decision Tree Method for Text-to-Phoneme Mapping", in *Proceedings of IEEE Workshop on Automatic Speech Recognition and Understanding*, pp. 135-140, USA, 2003

[108] J. Suontausta, J. Iso-Sipilä and J. Tian, "Handling of Acronyms and Digits in a Speech Recognition and Text-to-Speech Engine", US Patent application, 20050267757, 2005

[109] Yuezhong Tang, Xia Wang, Yang Cao and Feng Ding, "Feature Masking in an Embedded Mandarin Speech Recognition System", in *Proceedings of 2004 International Symposium on Chinese Spoken Language Processing*, pp. 245-248, Beijing, China, 2004

[110] P. Taylor, A. Black and R. Caley, "The Architecture of the Festival Speech Synthesis System", In *Proceedings of the 3rd ESCA Workshop on Speech Synthesis*, pp. 147-151, Jenolan Caves, Australia, 1998

[111] D. L. Thomson and R. Chengalvarayan, "Use of Periodicity and Jitter as Speech Recognition Features", in *Proceedings of the International Conference in Acoustics, Speech and Signal Processing*, Vol. 1, pp. 21-24, Seattle, WA, USA, 1998

[112] J. Tian, J. Häkkinen, S. Riis and K. Jensen, "On Text-based Language Identification for Multilingual Speech Recognition Systems", in *Proceedings of the International Conference on Spoken Language Processing*, pp. 501-504, Denver, Colorado, USA, 2002

[113] J. Tian, J. Häkkinen and O. Viikki, "Multilingual Pronunciation Modeling for Improving Multilingual Speech Recognition", in *Proceedings of the International Conference on Spoken Language Processing*, pp. 497-500, Denver, Colorado, USA, 2002

[114] J. Tian and J. Suontausta, "Scalable Neural Network Based Language Identification from Written Text", in *Proceedings of the International Conference in Acoustics, Speech and Signal Processing*, Vol. 1, pp. 48-51, Hong Kong, China, 2003

[115] J. Tian, "Method for Compressing Dictionary Data", US Patent, 7181388, 2007

[116] J. Tian, "Efficient Compression Method for Pronunciation Dictionaries", in *Proceedings of the International Conference on Spoken Language Processing*, pp. 2733-2736, Jeju Island, South Korea, 2004

[117] K. Tokuda, T. Masuko, N. Miyazaki and T. Kobayashi, "Hidden Markov Models Based on Multi-space Probability Distribution for Pitch Pattern Modeling", in *Proceedings of the International Conference in Acoustics, Speech and Signal Processing*, Vol. 1, pp. 229-232, Phoenix, Arizona, USA, 1999

[118] *The Unicode Standard, Version 5.0*, Addison-Wesley Professional, 5th Edition November, 2006

[119] M. Vasilache, "Speech Recognition Using HMMs with Quantized Parameters", in *Proceedings of the International Conference on Spoken Language Processing*, Vol. 1, pp. 441-443, Beijing, China, 2000

[120] M. Vasilache and O. Viikki, "Speaker Adaptation of Quantized Parameter HMMs", in *Proceedings of Eurospeech*, Vol. 2, pp. 1265-1268, Aalborg, Denmark, 2001

[121] M. Vasilache, J. Iso-Sipilä and O. Viikki, "On a Practical Design of a Low Complexity Speech Recognition Engine", in *Proceedings of the International Conference in Acoustics, Speech and Signal Processing*, pp. 113-16, Montreal, Canada, 2004

[122] Viikki, K. Laurila and P. Haavisto, "A Confidence Measure for Detecting Recognition Errors in Isolated Word Recognition", in *Proceedings of SST*, pp. 67-72, Adelaide, Australia, 1996

[123] O. Viikki and K. Laurila, "Cepstral Domain Segmental Feature Vector Normalization for Noise Robust Speech Recognition", *Speech Communication*, Vol. 25, pp. 133-147, 1998

[124] O. Viikki, D. Bye and K. Laurila, "A Recursive Feature Vector Normalization Approach for Robust Speech Recognition in Noise", in *Proceedings of the International Conference in Acoustics, Speech and Signal Processing*, Vol. 2, pp. 733-736, Seattle, WA, USA, 1998

[125] J. Viterbi, "Error Bounds for Convolutional Codes and an Asymptotically Optimal Decoding Algorithm", in *IEEE Transactions on Information Theory*, Vol. 13, Issue 2, pp. 260-269, April 1967

[126] *VoiceXML Forum*, http://www.voicexml.org/

[127] X. Wang, Y. Dong, J. Iso-Sipilä and O. Viikki, "On Integrating Tonal Information into Chinese Speech Recognition", in *Proceedings of International Symposium on Chinese Spoken Language Processing*, pp. 187-190, Beijing, China, 2000

[128] Xia Wang and J. Iso-Sipilä, "Low Complexity Mandarin Speaker-Independent Isolated Word Recognition", in *Proceedings of the International Conference on Spoken Language Processing*, pp. 1589-1592, Denver, Colorado, USA, 2002

[129] X. Wang, Y. Dong, J. Häkkinen and O. Viikki, "Noise Robust Chinese Speech Recognition Using Feature Vector Normalization and Higher-Order Cepstral Coefficients", in *Proceedings of International Conference on Signal Processing*, Vol. 2, pp. 738-741, Beijing, China, 2000

[130] J. C. Wells, "SAMPA Computer Readable Phonetic Alphabet", in *D. Gibbon, R. Moore and R. Winski (eds.), Handbook of Standards and Resources for Spoken Language Systems*. Mouton de Gruyter. 1997

[131] J. Wilpon, L. Rabiner, C.-H. Lee and E. Goldman, "Automatic Recognition of Keywords in Unconstrained Speech Using Hidden Markov Models", *IEEE Transactions on Acoustics, Speech and Signal Processing*, Vol. 38, No. 11, pp. 1870-1878, 1990

[132] R. Yates, "Fixed-Point Arithmetic: an Introduction", *Technical reference from Digital Signal Labs*, http://www.digitalsignallabs.com, March 29, 2007

[133] T. Yoshimura, K. Tokuda, T. Masuko, T. Kobayashi and T. Kitamura, "Simultaneous Modeling of Spectrum, Pitch and Duration in HMM-Based Speech Synthesis", in *Proceedings of Eurospeech*, Vo1. 5, pp. 2347-2350, 1999.

[134] S. Young, N. Russel and J. Thornton, "Token Passing: a Conseptual Model for Connected Speech Recognition Systems", *Technical report CUED/F-INFENG/TR.38*, Cambridge University, 1989

[135] S. Young, J. Odell and P. Woodland, "Tree-based State Tying for High Accuracy Acoustic Modelling", in *Proceedings of the Workshop on Human Language Technology*, pp. 307-312, Plainsboro, NJ, USA, 1994

[136] S. van Vuuren and H. Hermansky, "On the Importance of Components of the Modulation Spectrum for Speaker Verification", in *Proceedings of the International Conference on Spoken Language Processing*, pp. 631-634, Sydney, Australia, 1998

[137] J.-S. Zhang and K. Hirose, "A Robust Tone Recognition Method of Chinese Based on Sub-syllabic F0 Contours", in *Proceedings of the International Conference on Spoken Language Processing*, pp. 703-706, Sydney, Australia, 1998

[138] M. Zissman and K. Berkling, "Automatic Language Identification", *Speech Communication*, Vol. 35, No. 1, pp. 115-124, August 2001