An Application-Independent Speaker Adaptation Service
En tjänst för tillämpningsöberoende talaradaption

Henrik Söderquist
26 April 2002

Supervisor: Mats Blomberg
Department for Speech, Music and Hearing
KTH, Stockholm
Abstract
The primary goal of this master thesis work is to develop an application-independent speaker adaptation service. By using the speaker-adapted models performance in the applications will be increased compared to when using speaker independent models. Speaker-adapted models are not commonly used today since this would require that the user adapted a model for each service that he uses. The purpose of the service developed within this project is to make it possible for a user to perform one adaptation and use the resulting model in all the services that he desires. Thereby the user achieves a performance gain with only a small amount of extra work for him. This service records adaptation data, adapts and stores models. The stored models are made available for download to other voice-controlled applications. In order to verify this a small test is made and the result shows that there is a gain in recognition performance.

Sammanfattning
1 Introduction

1.1 Background
This master thesis project has risen from the CTT\textsuperscript{1}-project HörStöd [Johansson, 2002]. HörStöd is aimed as an aid during telephone conversation between normal hearing and hearing impaired persons. In HörStöd, the hearing impaired person is presented with a transcription of the other person’s utterance. In order to obtain a transcription of what the person said, speech recognition was used and the output from the recogniser was displayed to the hearing impaired person. Since the transcription had to be done in real-time it was done on the phoneme level, printing phoneme strings on a display for the hearing impaired person to read. This enabled the hearing impaired to read what the person said. As the project HörStöd showed, speaker adapted models were necessary to reach the required recognition performance, since speaker independent recognition was not accurate enough. Hence there is a need for an application that can easily perform speaker model adaptation. Since most voice controlled applications are accessed via telephone the best way to train the personal models would be to record a user’s voice over a telephone line and then perform the adaptation.

The aim of this project is to implement a stand-alone service that can perform speaker model adaptation over telephone. A user calls the service and records a number of utterances and when the user has read all the required utterances a personal speaker model is trained. The adapted models are made available for download to other voice operated telephony applications.

The advantage with this type of service is that it is application independent, which means that users only have to train their models once and use them in all, or almost all, voice controlled telephone services that they use. Another advantage compared to using a speaker independent model is that the computational requirements are reduced, yielding less complexity and costs for the service provider. Compared to using a speaker independent model recognition performance increases significantly, [Huang, Acero, Hon, 1999].

Speaker model adaptation is a process in which a speaker independent model is being adjusted to fit a certain speaker. In this application a global model, that has reasonable performance for all speakers, is adapted to a certain speaker and the resulting model is then expected to yield better performance for this speaker compared to using the global model.

1.2 Project outline
The aim of this master thesis project is to develop a system that performs adaptation after having verified the user utterances. This includes design and implementation of a speech-controlled telephony application and construction of a suitable set of sentences to be read. In order to show that the system actually can increase performance a small test will be made with different adaptation algorithms.

\textsuperscript{1} An abbreviation for “Centrum för Talteknologi”, which in English reads Centre for Speech Technology, see http://www.speech.kth.se/.
1.3 Purpose
The main purpose of this service is to supply telephony-based services with speaker adapted models. By storing the speaker models on a central server it is easier to access them for both the expected audience of voice-controlled applications that can use these models and the speakers for whom the models have been made.

When using personal models a gain in recognition performance is expected. Besides the performance gain it will also yield lower costs and reduction in system complexity. By performing the adaptation in a place common to several services it will save time for the user as well as the services that the user intends to use. When the user has adapted a personal model for a service the resulting model can be accessed by all the applications that the user uses. One limiting factor is however network performance since the models have to be downloaded from the central service to the application that wants to use the models. This could be solved, by simply downloading the model before the user’s first session starts, i.e. when the user registers. When speaker models are stored locally at each application, all the applications that have downloaded models must be informed of model updates.

A side effect of performing this speaker model training is that the speaker is getting acquainted to using voice controlled telephony applications. This could contribute to better performance with or without presence of the personal model.

1.4 Vital terms
Within this project a number of technical terms will be used and some of them might require some extra explanation. In the following sections a small introductory explanations to these terms are available.

1.4.1 Automatic speech recognition
If text-to-speech is the way the computer speaks, then speech recognition is the way the computer listens to what the user has to say. Automatic speech recognition represents a new way of computer interaction. Speech is the natural way for humans to communicate, and therefore gives a psychological advantage since communication takes place on human conditions not the machine’s conditions. Another advantage is that the user has her hands and eyes free and available for other tasks.

Automatic speech recognition is the process of transforming human speech into a form, which can be understood by the computer. The speech recognition system that is used in this project is structured in four blocks shown in Figure 1.1.

![Figure 1.1 The main building blocks of the Automatic Speech Recognition system. The functionality of the blocks are briefly described in section 1.4.1.](image)

The first block transforms the speech signal, represented by a waveform in the time domain, to the frequency domain. The waveform signal is sliced up into frames (usually of 10, 15 or 20 milliseconds) which are transformed into a short-time spectrum using Fast
Fourier Transform (FFT). From the resulting spectrum a set of relevant features are extracted which describe phonetically distinctive properties of the spectral information. The most common representation, which is also used in this system, is mel scale cepstrum coefficients (MFCC). The cepstrum is computed by taking the FFT inverse of the log magnitude of the FFT for the speech signal. The mel scale is based on the non-linear human perception of the frequency of sounds [Rabiner, Juang, 1993].

In the third block these features are analysed and the acoustic observations are mapped to phonetic classes using Hidden Markov Models (HMM), described in section 1.4.2. The phonetic sequences are matched with words in the vocabulary and text is presented. A more thorough description can be found in [Jurafsky, Martin, 2000].

1.4.2 Hidden Markov Models
A hidden Markov model (HMM) is a Markov chain, where each state generates an observation. A HMM is a hidden Markov model because we don’t see the states of the Markov chain, but just a function of them. You only see the observations, and the goal is to infer the hidden state sequence. HMMs are very useful for time-series modelling, since the discrete state-space can be used to approximate many non-linear, non-Gaussian systems. The parameters of the model are the transition and emission probabilities. These parameters are adjusted during training from speech data.

A hidden Markov model is defined as a pair of stochastic processes \((X, Y)\). The \(X\) process is a first-order Markov chain, and is not directly observable, while the \(Y\) process is a sequence of random variables taking values in the space of acoustic parameters, or observations. According to the first-order Markov Hypothesis the history has no influence on the chain’s evolution if the present is specified. The output independence hypothesis states that neither chain evolution nor past observations influence the present observation if the last chain transition is specified.

A useful tutorial on the topic can be found in [Rabiner, 1989]

A model can describe monophones, diphones or triphones. A model describing monophones describes each phoneme independent of context. Diphone or triphone models describe phonemes in their context. This means that these models describe both phonemes and the transition between phonemes. In this project monophone models will be used.

1.4.3 Adaptation
Rather than training speaker dependent models from scratch, which would require a large amount of training data from the speaker, adaptation techniques can be used. By using a small amount of data from a new speaker a good speaker independent model can be adapted to better fit the characteristics of the new speaker. During the adaptation phase the characteristics of the speaker’s voice are used to adjust a speaker independent model to fit the speaker. Potentially the size of the model can be reduced since the parameters’ representation will be more accurate. Apart from the size aspect the adapted model will also yield improved recognition performance, this means a smaller number of misinterpretations will occur.

There are two categories of adaptation, supervised and unsupervised adaptation.
Supervised adaptation means that someone controls the correctness of the transcriptions before they are used within the adaptation process. Unsupervised adaptation on the other hand uses transcriptions provided by a recogniser. Therefore if the recogniser output isn’t completely correct it will lead to incorrect training of the misinterpreted phonemes. Hence if the recogniser often makes mistakes the model will deviate more.

The adaptation process can be done in two different ways. Either by training with all data already available, static adaptation, or continuously as new data arrives, incremental adaptation.

In this thesis work a combined method will be used since the utterances that the user is supposed to read are known. The transcriptions are also available. To check whether the user read the correct utterance or not a recogniser checks the utterance. However small differences may slip through undetected and hence lead to deviations in the model.

1.4.3.1 Adaptation algorithms
In this thesis work two adaptation algorithms will be used:

- Maximum likelihood linear regression (MLLR)
- Maximum a posteriori (MAP)

Maximum likelihood linear regression (MLLR) computes a set of transformations that will reduce the mismatch between the initial model and the adaptation data,[Huang, Acero, Hon, 1999]. More specifically it is an adaptation technique that estimates a set of linear transformations for the mean and variance parameters of a Gaussian mixture HMM system. The effect of these transformations is to adjust the initial system so that it will be more likely to generate the adaptation data. The use of regression classes makes it possible to adapt models for phonemes not present in adaptation data.

Model adaptation can also be accomplished using the maximum a posteriori (MAP) approach,[Huang, Acero, Hon, 1999]. This adaptation method is sometimes referred to as Bayesian adaptation. MAP adaptation involves the use of prior knowledge of the model parameter distribution. Hence MAP can effectively deal with data-sparse problems and take advantage of prior information. Prior density prevents large deviations of parameters unless new training data provide strong evidence. However the MAP adaptation method requires more data than MLLR to yield a more accurate result, since it only adapts the phoneme models available in adaptation data. The MAP adaptation rate can be set, the adaptation rate indicates how much previous data influences the model.

1.4.4 Utterance
An utterance is the vocalization of a word or a sequence of words. Utterances can be a single word, a few words, a sentence, or even multiple sentences. In this application an utterance is a sentence that is read by the user.

1.4.5 TTS
TTS is short for Text-to-Speech and means that text is converted to speech. A text is put into the TTS module, which converts text to speech, in this case presented to the user via
the telephone. This is the way that the computer communicates with the user since there is no monitor available to the user. You could say that this is the computer’s way of reading a text aloud.

There are two ways of doing this. The first one is to simply concatenate isolated words or parts of sentences, denoted as Voice Response Systems. This method is only applicable when a limited vocabulary is required, typically a few hundred words, and when the sentences to be pronounced all follow a very restricted structure. As an example this type of systems could be used to announce train arrivals at a station. The second way of implementing a TTS synthesis is to perform a grapheme-to-phoneme transcription of the desired text. This can be obtained by simply concatenating elementary speech units, but to obtain high quality a set of rules have to be applied and signal processing performed for smoothing and adjustments in duration and prosody. For the interested reader a more detailed description of text-to-speech synthesis is available see [Dutoit, 1999]

![Diagram of text-to-speech synthesizer](image)

Figure 1.2 A schematic overview, showing the two main parts of a general text-to-speech synthesizer.

As displayed in Figure 1.2 a TTS synthesizer in general consists of two blocks. First the text is processed and intonation and phoneme information is extracted. In the second stage, the digital processing stage, the sounds are processed to smooth the prosody and to adjust phoneme durations.
2 Speech Technology Platforms

In this section a short introduction to existing speech technology software platforms and how adaptation is handled by these systems.

2.1 VoiceXML

The VoiceXML\(^2\) standard is developed by the World Wide Web Consortium\(^3\) (W3C). VoiceXML is a dialog mark-up language designed for telephony applications, where users are restricted to voice and touch tone (DTMF\(^4\)) input.

However VoiceXML isn't the same thing as HTML. HTML is designed for visual web pages and VoiceXML aims to bring web access by interacting with keypads, spoken commands, pre-recorded speech and synthetic speech. This allows for access to the web when there is no keyboard or mouse present and also keeping hands and eyes free for other things. It will also be a boon to people who are visually impaired. When working with speech content the user can only hear one thing at the time, unlike the visual web pages where the user can see more than one thing at the time, therefore the user and application take a dialog in turns.

However the VoiceXML specification does not include any specification of the underlying speech technology components, it only specifies the VoiceXML code and which actions that is to be performed by the code.

VoiceXML could have been used partly in this project since it can handle voice recording, speech recognition and text to speech conversion. Due to the absence of the necessary software development platform at the time of this work it was not chosen for this project. VoiceXML does not support low-level control of the speech technology components and hence models cannot be loaded dynamically at runtime. This makes this system less flexible and therefore less usable for this project.

2.2 SALT

SALT\(^5\) is short for “Speech Application Language Tags” and is a joint initiative between Cisco, Comverse, Intel, Microsoft, Philips and SpeechWorks. SALT is to be embedded in other markup languages such as HTML, xHTML and XML and enhance them with a speech interface. The objective of the group is to develop a royalty-free, platform-independent standard that will give multimodal and telephone access to information. SALT is designed to minimise authoring overhead by allowing maximum reuse of developers’ work.

\(^2\) http://www.w3c.org/Voice/
\(^3\) http://www.w3c.org/
\(^4\) http://www.shout.net/~wildixon/telecom/dtmf/dtmf.html
\(^5\) http://www.saltforum.org/
SALT is another standard that can perform similar things as VoiceXML, but not as flexible and powerful. Since SALT aims to extend existing HTML documents with speech interaction capabilities it does not allow any low-level access to the speech technology components. There is currently no implementation of SALT available. Hence SALT was not chosen for this project.

2.3 Nuance

Nuance\(^6\) delivers a self-titled speech recognition software. It’s software features an auto tuning system called “Listen & Learn” that automatically tunes the system to account for regional accents and filters out background noise. Except from this auto-tuning feature it also provides dynamic language detection and something that is called “Personalization Kit”. This enables automated system tuning and a tailored experience for each caller. The Nuance software has support for 26 languages, a user simply begins speaking in his preferred language and the system understands and can interact with the caller accordingly. With the “Personalization Kit” the software can detect gender, type of phone (wireless or landline) and determine noise levels. With this information at hand the application can prompt the speaker to speak louder when necessary.

To lower deployment efforts the Nuance software supports VoiceXML 2.0. Hence carriers and enterprises can leverage their existing expertise and investments in Web infrastructure to reduce cost and effort of deploying voice-driven services.

2.4 SpeechWorks

SpeechWorks\(^7\) provides a recognizer that goes by the name “SMARTRecognizer” and this recognition engine is self-learning, that means it uses the speech input from a session to retrain the models. This improves recognition accuracy in the system since the callers’ language patterns are adapted.

![Figure 2.1 A schematic view of the principle of work for SMARTRecognizer.](image)

To make it possible for a user to speed up the dialog they have added the possibility to barge-in and interrupt prompts.

\(^6\) http://www.nuance.com/

\(^7\) http://www.speechworks.com/
2.5 Summary

To summarize it all, there does not seem that the proposed application independent service is in use in existing systems. However there exist methods to improve recogniser performance within an application. The aim of this project is to increase performance in all applications by using centrally adapted and stored speaker models.
3 System description

The adaptation system developed in this thesis work consists of a number of co-operating pieces of software, most of them developed at CTT. The application is run on a set of PCs equipped with Linux as their operating system. This section will describe the main components around which the system is built.

3.1 Hardware

During the development and evaluation phase the software ran in a distributed mode on up to three PCs, the database server not included. The applications are designed to run under Linux, although some of the software is almost platform independent (file separators and other OS specific characters is chosen to fit Linux). During development and evaluation all applications ran under Red Hat Linux\(^8\).

As part of the main application there is a telephone interface, in this case it was an ISDN adapter. An AVM A1 passive Basic Rate Interface ISDN adapter in combination with software from the ISDN4Linux project was used. The ISDN-server communicates with the application via the broker described below.

3.2 Software platform

The software is built around a software platform known as Atlas [Melin, 2001a]. Atlas is an object-oriented API to several kinds of speech technology components. This software platform is described in more detail below.

3.2.1 The Broker

Atlas communicates with most of the speech technology components through an application known as the Broker\(^9\). The Broker is a system that handles interprocess communication, similar to a CORBA\(^{10}\) server. It dispatches requests for speech technology services between clients and service providers. The Broker’s protocol is text-based, which simplifies client construction using different programming languages and on different platforms.

3.2.2 Atlas

Atlas is the middleware that is used by high-level speech technology applications to connect to its service providers. It provides a multi-layered API allowing applications to interface with the speech technology components. Atlas allows direct access to low-level interfaces, when needed, as well as access to high-level dialog components, this is illustrated in Figure 3.1.

---

\(^8\) For more information on Red Hat Linux see [http://www.redhat.com/](http://www.redhat.com/).

\(^9\) The Broker (which goes by this name only), can be downloaded from [http://www.speech.kth.se/broker/](http://www.speech.kth.se/broker/).

\(^{10}\) The CORBA, Common Object Request Broker Architecture, has been published by the Object Management Group, [http://www.omg.org/](http://www.omg.org/), and is a standard for distributed object systems.
3.3 Speech technology components

In this section a short description of the speech technology components used in this project will be found. As a complementary to those components mentioned below there is also a module available that provides a graphical user interface to the Broker. Apart from this module and those mentioned below there exist some other modules that extend the functionality, such as audiovisual output. There are also a couple of components, which are not speech technology components, that are included here. However, they all use the Broker for interprocess communication.

3.3.1 Database connectivity

A MySQL\textsuperscript{11} database is used to keep track of registered users and to simplify the administration of their models. MySQL is a relational database management system that is available as Open Source Software. The database consists of only one table holding all user data.

3.3.2 Digitizer

This service provides desktop-based audio input and output. The service therefore runs on a computer fitted with a microphone and a pair of headphones or loudspeakers. Since the application is intended for use via telephone, the speaker models used will be trained using

---

\textsuperscript{11} For further information see http://www.mysql.com/.
material recorded via telephone. Using the computer microphone as speech input source yields unsatisfactory performance since the signal has a larger bandwidth than 4 kHz.

3.3.3 ISDN Server
The ISDN server is a module that operates in the Broker environment. It provides the application with telephony capabilities by interfacing to the Integrated Services Digital Network, also known as ISDN. The server interfaces to an ISDN terminal using device driver and modem emulator provided by the ISDN4Linux project. The server can be used both for incoming and outgoing calls. When a connection is established it signals the Broker which then initiates a session.

In this project the ISDN is used for voice input and speech output. Since ISDN is a digital telephone system, the signal that arrives is already available in digital form. The signal is a-law coded with a sample frequency of 8 kHz and with 8 bits/sample. This signal is then fed to the speech recognizer.

3.3.4 Recognizer
The speech recognition component is based on the StarLite ASR engine [Ström, 1997]. It was completed with acoustical models based on monophones trained on the SpeechDat project [Salvi, 1998]. StarLite is given a set of lexicon files containing all the utterances that is to be recognized by the application. In this project there exist three different lexicons, one for usernames, one for digits and one with the sentences that the user is supposed to read. These files contain all possible results that can come out of the recogniser. Since the recogniser isn’t able to suggest anything except the entries in these files, it will be unable to correctly recognise utterances that were not pronounced correctly according to the lexicon. In combination with the recognized string the recognizer also outputs a score value indicating how well the recognition result matched with user input. This score value could be used to help deciding whether or not the user utterance is correct or not, but since there is no easy way available to normalize these values this calls for string dependent target score values. There is no score-threshold function implemented within this project. A more accurate way to solve this would be to implement confidence, a statistical measure indicating the probability that the utterance is correct.

3.3.5 Sound coder
The sound coder encodes the recorded sound and can also transform sound streams between different formats if necessary. In this application the resulting sound files are stored in A-law format, which is a form of logarithmic quantization or companding. The encoding principles for this format is based on the observation that many signals are statistically more likely to be near a low level signal level than a high signal level. Therefore, it makes more sense to have more quantization points near a low level than a high level. The A-law encoding is a standard encoding scheme according to the International Telecommunication Union12 – Telecommunication Standardization Sector (ITU-T) Recommendation G.711.

12 http://www.itu.int/
3.3.6 Text to speech
This component, as the name indicates, converts text into speech, known as speech synthesis. It consists of two parts, a text-to-phoneme-string component and a phoneme-string-to-speech component. The former is a transcription engine known as RulSys [Carlsson, Granström, Hunnicutt, 1982], which transcribes sentences into phoneme strings. The second component is the synthetic voice framework MBROLA [Dutoit, Pagel, Pierret, Bataille, van der Vreken, 1996], which uses the output from the first component. It uses recorded diphones and concatenates them yielding an audio stream as output. There exist a couple of different voices that can be used to synthesize the phonemes. This can be set in the configuration file.
4 The telephony service

In this section the software design of the application independent speaker adaptation service will be described. The service is divided into three separate services; they can all operate independently from each other, although they share some vital components. The three parts are registration service, adaptation service and download service. The registration handles user registration. After having registered the user can call the adaptation service to create his/her own speaker model. Finally the third application handles speaker model distribution.

![Diagram](image.jpg)

**Figure 4.1** A schematic view of the entire service. The three common objects in the middle are the user database, the database with speaker-adapted models and the adaptation protocol. The abbreviation PSTN stands for “Public Service Telephone Network”.

4.1 Registration service

Before the user can phone in and perform a training session he/she must become a registered user in order to obtain username, password and protocol. This is done either thru a web browser or via an application run on a terminal server, e.g. Secure shell (SSH) or Telnet. It can also be executed from a local command line prompt. After the user has connected to the registering service, he/she is prompted for some data and is supplied with the session protocol. The user is then supposed to save this information so it can be used at a later time, since the user database has to generate a new user grammar. The user grammar contains the usernames of all registered users with their corresponding phoneme transcriptions. Once an hour is a check performed by the main application if an update has been made to the user database, if any changes have been made it automatically updates the user grammar.
Another way of implementing this service would be to use dynamic HTML to interact with the user through her web browser. When using an application like this one in full scale some security precautions might be considered, such as encrypted traffic.

A basic interface for registration via a web browser has been implemented using PHP\(^{13}\) embedded into normal HTML code. PHP is a freely available server side scripting language that easily allows database interactivity. The application is split in two parts, where the first part is a web page containing a form that is used to pass information to the PHP-script. When the information has been passed to the script, the server processes the PHP code and replies with an input dependent web page. The resulting web page contains user data and the protocol that is to be used during the training session. This method of registration is more comfortable to the user since user data and protocol are presented to the user in a known environment from which the user can easily make a hardcopy.

Apart from user information and training protocol the resulting web page also contains some instructions on how to use the service and a recommendation that the user saves it for future use, i.e. makes a hardcopy of the page. An important notice is that adding a bookmark pointing to this page is not sufficient since it is dynamically generated. It is not possible to retrieve the information at a later stage without supplying adequate arguments to the script. Another advantage of this type of application is that none of the passwords that are used to access the database is sent to the client side, since the server parses the script and only outputs plain HTML-code to the client.

### 4.2 Main application

#### 4.2.1 Session outline

This is the application that provides the actual telephone service; see Figure 4.3 for a schematic view of the application. The user dials a phone number, provided at registration, from any telephone and the application picks up the phone in the other end.

When a user has dialled in, the application creates a new session that handles the communication with the user. In the session a state machine handles the user dialog. The state machine works according to Figure 4.4.

When the application is started it will connect to the broker and all the required services. The administrator can obtain information about which services that are up and running. In the same window there is also a session monitor that provides information whether the application is waiting for a connection or is currently busy with a user session. By making the appropriate choice on the menu bar the service administrator can terminate the session or change resource defaults. The window is called “Resource centre”.

When a user has connected to the service via the telephone a message is displayed in the resource centre window. The user however does not see this window; he/she is just equipped with a telephone and an adaptation protocol. Upon connection a session is created and the state machine is initialised.

The state machine guides the user thru the session by keeping a dialog with the user. In the first state, *WAITING*, a welcome message is played for the user. This message informs the user to whom the call was placed, so that the user receives a verification that he has called the correct telephone number. After a short greeting the state machine proceeds to the next
Before the training session can take place the user is required to log in to the system by identifying him-/herself. This is handled by the **LOGIN** state. The user is first prompted for a username and then for a password. Since the user uses the voice for identification in this application the recogniser parses the audio input data and returns a hypothesis containing a username and a password. To ensure that the user is registered and has used the correct password, this is done by a simple database lookup. If the user data checks out to be legitimate then the user is accepted and can continue to the next phase. If the user was not accepted he/she gets another try, unless the maximum number (the default value is three) of attempts has been reached. If the user’s identity claim is accepted then he is greeted with his name. This way the user knows that he has logged in correctly.

If an error occurs during the login phase, the state machine proceeds to the **CRASH** state, which exists merely for the cause to catch errors and finish the user session safely. This state could be used to send a notification to the service administrator, in this implementation there is only a short message written to the standard output stream. The user is in this state informed by speech synthesis that an error has occurred, and then hangs up the phone. The **CRASH** state has been added so that the user will not sit and wait forever to hear a response to his actions.

After the login phase the dialog passes either of the two states **ENROLL** or **VERIFY**. Which of the two states the user will be passed thru depends on whether he/she is there for the first time or not, if it is the first time the user is passed to the **ENROLL** state. These two states have been put into the state machine for future developments. They could be used to let a new user enrol, i.e. train a personal model that could be used for speaker verification. Hence the verify state would let a returning user be verified before proceeding to the training state. This would be an extra security against adaptation to the wrong user ID. In this application however, these two states just passes the user on to the next state, the **TRAIN** state.

The **TRAIN** state is where the main action is performed. The user is prompted to read a sentence. When the user reads the sentence the voice is recorded. The audio data is parsed by the recogniser and controlled whether it could be the correct utterance or not. The recogniser chooses the sentence that matches the utterance best out of 70 sentences available in the lexicon. Accept from recognising an entire sentence some test were carried out using word recognition, but this turned out to yield less satisfactory results. If the recogniser has the correct sentence among the ten best hypotheses, sorted by score, it is accepted. Sometimes the recogniser doesn’t come up with as many as ten hypotheses to an utterance; ten is only the maximum number of hypotheses that will be taken into consideration. If this isn’t the case the user is re-prompted until the utterance is correct or the maximum number of attempts for the specific utterance is reached. This state also lets the **CRASH** state take care of errors that might occur. When the user has read all sentences and they have been verified by the recogniser the state machine lets the user proceed to the **ADAPT** phase. The user does not have to get all utterances accepted, they can also get rejected and if that is the case the application takes notice of this and the user can proceed anyway. The filenames of all utterances that have been accepted by the recogniser are stored in a file. This file is then used in the adaptation state to pass all the accepted files to the adaptation process.
In the *ADAPT* state a script is executed that runs the speaker model adaptation as a background process. When the script has been started the user is passed on to the next state, the *LOGOUT* state. The user is then told to hang up the phone and that the model will be available and ready to use after an amount of time specified by a configuration parameter.

### 4.3 Download service

This application acts as the bridge between the applications that uses the adapted models and the adaptation service. The speech application that wants to use one of the speaker models passes a request to this application containing a reference to a user who has told the requesting service that he/she has trained a personal speaker model and where it can be obtained. In this work the reference is the user’s email address, but in a real case it would most likely be the user’s personal identification number, since these are guaranteed to be unique and available for every user, at least in Sweden.

![Diagram](image)

**Figure 4.5** A schematic view of the download service. The leftmost part is a speech application that utilises the adapted speaker models available through this service.

The speech application sends a request and the download service checks if there is any available model for this person. If there is an available model, it transfers it to the speech application that made the request.

The administrators of the speech application that is going to use the speaker models can then choose for themselves whether to download them when they are needed, i.e. when the user is entering a session, or when the user registers in their service. Whether to choose the first or the second alternative depends on network and computer performance factors. More memory is required when several simultaneous users are using personal speaker models instead of one speaker independent models.

An alternative to this application would be to make all models available over the Internet using standard web protocols such as `ftp` or `http`. This would however require some smart naming of the files or some type of application that handles the translation between username and speaker model filename.

---

14 File Transfer Protocol.

15 Hypertext Transfer Protocol.
5 Test results and conclusions

In order to test whether the developed system increased recognition performance an evaluation test was set-up. The test included 50 adaptation sentences and 20 test sentences. These are listed in Appendix A, the first 50 were used for adaptation and the last 20 for testing. Every sentence was repeated three times. In this way we could simulate and test the effect of various usages of the recogniser to check the quality of the utterance recordings. The utterances were processed in order to sort out which one of the three tries for each utterance that was the best one. After this twelve models were made for each speaker. The twelve models were adapted by using three different methods:

- MAP with scale factor 0.0, no respect to previous knowledge of parameter presentation.
- MAP with scale factor 1.0, small respect to previous knowledge of parameter presentation.
- MLLR with 16 regression classes\(^{16}\).

For each of the three methods above four different sets of data were used:

- First repetition of each sentence, 50 utterances.
- Last repetition of each sentence, 50 utterances.
- Highest scoring repetition of each sentence, 50 utterances.
- All repetitions of each sentence, 150 utterances.

These model sets were tested with 20 test utterances. In total there were 84 model sets from seven speakers. All tests have been done with HTK, [Young, Kershaw, Odell, Ollason, Valtchev, Woodland, 1999]. The performance was measured for increasing number of adaptation utterances in steps of ten in order to display the supposed increase in performance correlated to the amount of training data. All seven user’s models have been tested and the average result has been calculated.

The models that were trained on the first of the three attempts and the models that were trained on the last of the three attempts showed equal performance, see Figure 5.1. As the figure shows the MLLR yields higher performance for smaller amounts of training data, which was expected.

The accuracy is computed using Eq. 5.1. In this equation \(N\) represents the total number of phonemes in the correct transcription, \(S\) the number of substituted phonemes, \(I\) the number of inserted phonemes and \(D\) the number of deleted phonemes.

\[
\text{Accuracy} = \frac{N - S - I - D}{N} \times 100\% \quad (5.1)
\]

\(^{16}\) Using regression classes similar states in the model will be tied together, hence cluster together components that are close in acoustic space. For a more detailed description of the combination of MLLR and regression classes see the HTK Book, which is part of the HTK documentation.
By adapting the models on all three attempts for each utterance a slight performance gain was noticed, Figure 5.2. However by adapting the models using only the best of the three attempts for each utterance further increased the recognition performance, Figure 5.3. The best utterance is the one that received the highest score from the recogniser.

Figure 5.1 Recognition results for the three adaptation algorithms using the first of the three attempts.

Figure 5.2 Recognition results for the three adaptation algorithms using the all three attempts for each utterance.
To answer the question whether a faulty utterance would influence the quality of the adapted models, a session was recorded during which the user read the wrong sentence once and also coughed and hawked during an utterance. This showed only slightly to decrease the recognition performance. However this test was only performed on one single session.

To summarize it, adapting a speaker-independent model to a specific speaker increases recognition performance. It is obviously better if the transcriptions used are correct and not merely guesses or assumptions of what the user said. A method to automate this with high accuracy would certainly increase recognition performance and hence smaller amount of training data could be used.
6 Future developments

To make it harder for people trying to act as another person in this system, impostors, speaker verification could be introduced. There is already a state in the state machine in the dialog handler to handle this. This is however not implemented, but if it were, an enrolment dialog could reside in the state machine. If the user is already enrolled he/she is sent through the verification phase where the identity claim is tested.

If the user has enrolled earlier and hence already has trained a model suitable for speaker verification his/her identity claim will be tested. If the user’s identity claim is accepted he/she will be able proceed to the training phase, else he/she will be denied access to the service.

Implementing this would make it harder, although not impossible, to log in using another users identity. If a person is logged in using another user’s identity, that user’s model will be trained incorrectly yielding in performance losses. However there is no big damage caused since there are no valuable things involved, which would have been the case in a voice-controlled bank service.

Another extension of this service would be to save the users’ verification models and let the models be available for download by other applications, in the same manner as the speaker models. This would save time, since the user doesn’t have to go through an enrolment procedure every time they want to start using a new voice controlled telephone service. The material that is recorded in the session could also be used to construct a model for text-independent speaker verification. If a text-dependent speaker verification model is wanted different training data is required.

Since the amount of training data in this application is rather limited, the models will fit better for some purposes than other. By gathering voice samples from the application that the users use and send them back to this application independent model training service and training the speaker models on this data they could be improved further. This would give more general models, unless a user uses only a few types of services. Nevertheless it can be assumed that this leads to a performance gain. Another method would be that all services trained local models for every user based upon voice samples gathered during user sessions, this would favour the user’s that use the service more frequently and also make the model more specialised. A more specialised model is expected to yield better performance than a general model, i.e. an application independent model.

To further improve the performance a wider variety of speaker independent models could be available. This would allow training of a speaker dependent model starting with a speaker independent model that is trained with material that is closer to the current user’s speech. As an example it could be based upon dialect, age, gender or any other factor that is known or can be estimated. This would yield better performance with a smaller amount of training data.
References


Johansson, M., 2002: Phoneme recognition as a hearing aid in telephone communication, Master of Art Thesis, Department of Linguistics, University of Uppsala, Centre for Speech Technology (CTT), Department of Speech, Music and Hearing (TMH), KTH, Stockholm, To be published.

Jurafsky, D., Martin, J., 2000: Speech and Language Processing – An introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition, Prentice-Hall; New Jersey, USA.


Sestoft, P., 2000: Java Precisely, http://www.dina.kvl.dk/~sestoft/javaprecisely/, IT University of Copenhagen, Denmark and Royal Veterinary and Agricultural University, Copenhagen Denmark.


Appendices

A. Adaptation protocol

1. Avsikten är att kartlägga var de ur beredskapssynpunkt största svagheterna finns.
2. Nej, skivan har väl inte sålt så bra, konstaterar Torgny Söderberg.
3. Där bodde något tjugotal familjer, irakisk-kurdiska bönder.
4. Det är ingen motsägelse i det, säger Anders Westerlund.
5. Där var trygghet och ljus.
7. Men dör fick de tji.
8. Jag har brustit som far.
9. Se upp med bordsdekorationer i form av små guldglittrande stjärnor.
10. Men känslolyttringarna behöver inte ha djupare innebörd.
11. Under 1993 prenumererade cirka 40 000 resenärer på säsongmärkena.
12. I Italien följdes den 4 oktober 1582 av den 15 oktober.
13. Fler ägare betyder ytterligare mångfald.
14. Så hemskt, så mossigt, bort med det!
15. Någon teknisk bevisning finns inte.
17. På måndag måste oenigheten vara löst, annars spricker den borgerliga fyrpartiregeringen.
18. Han brukade öva monologer om världens orättvisor framför badrumsspegeln.
20. Klara Johanson var litteraturkritiker, Lydia Wahlström filosofie doktor och rektor.
22. Men själva syns de sällan.
24. År du helt rökfri nu?
25. Men det gällde ju inte det.
27. Norge, Sverige och skidsport, det är numera en hemskt otäck kombination.
28. Vi kommer nu att införa ett varningssystem för att öka säkerheten.
29. När Kerstis narkotikamissbruk avslöjades för familjen rasade hela vår värld samman.
30. Gubbarna bjuder på härliga bananer.
31. Det är med god musik som med schack.
32. Nu ska han tjäna nya pengar.
33. Nää, det är ingen skillnad.
34. Alla skötte sig väl mot Schweiz.
35. Förnyelse av förbrukad medborgarstrategi genom att sammanföra de frivilliga till plutoner.
36. Placeringarna har gjorts på affärmässiga grunder, konstaterar han.
37. Franzen torde vara rätt person att reglera flödena i det rörsystemet.
38. President Jeltsin föreföll först inte angelägen att träffa Bildt.
39. Inte för ungdomarna själva, naturligtvis.
40. Fyra bilar och sju motorcyklar.
41. Så länge det låter lever vi!
42. Jag ska gå ensam på bio.
43. Vad var det som hände?
44. Utländska investerare, räntefall och svag kronkurs bakom rekordartad uppgång.
45. När krigen är över finns det djupa hatet kvar inom människorna.
46. Slutränsen, vad krävs för att verkligen lyckas sluta?
47. Siksten obegripligt kylig, säger att Anna missförstått hans avsikter.
48. Naturligtvis ska vi lägga fram fakta så bra som möjligt.
49. På den vägen är det.
50. Nej, nu är det slut.
51. I kväll läggs Disneyklubben ned.
52. Nu önskar jag mej bara en sak.
53. Han grep reservoarpennan och började skriva med stora, ivriga slängor.
54. Erfarenheterna från musikbranschen skulle väl närmast tala för kulturdepartementet.
55. Södersjukhuset fyller 50 år 21 till 23 april.
56. Däremot fick ett antal samlingsskivor högsta betyg.
57. Men vad hade jag egentligen förlorat?
58. Klocka räknar döda i New York.
59. Så här ligger det till.
60. Är du född i Stockholm?
61. En ny båt har anländt.
62. Eftersom jag automatiskt ifrågasätter självutnämnda auktoriteter, hade jag svårt med skolan.
63. Nu blev utfallet över förväntan, i storleksordningen 400 000 per klub.
64. Husläkarsystemet genomförts fullt ut den 1 mars.
65. Socialdemokraterna gick under väldiga vändor med på Österleden.
66. Nu får han sitt straff.
67. Men nu är allt bra.
68. Farligt med stjärnor på matbordet.
69. Repriser i P1 nyårssagan.
70. Jag bara tröstar mej med dej.
B. Administrator’s guide

B.1 System set-up
The system is configured thru configuration files. If the administrator wants to use several different set-ups it can easily be obtained thru multiple configuration files. Which configuration file to use is set at system execution, the configuration file is passed to the application on the command line. To simplify execution a set of useful scripts have been created.

B.2 System execution
To simplify a set of scripts have been created. These scripts takes the same command line arguments as when executing the application directly, in some cases even more arguments can be passed to the script.

B.3 Main application
The most usable scripts are the broker.csh and start.csh. As the name might reveal the script named broker start the broker, this is a somewhat inflexible script since it has to be modified if another broker set-up is wanted. This script however takes two arguments, broker host and port. It is executed according to the following example:

```bash
$> ./broker.csh host [port] [display]
```

As indicated by the brackets the port argument is optional and if left out it will default to 3000. The third argument is used when running the broker on a remote host, and allows for remote viewing of the GUI. This script can be used without modifications only when all broker services is run on the same computer.

When the broker starts it will prompt for a password, this password is required to stop the service. The first service to be started is the GUI that will display available services and allow monitoring.

The start.csh script is more flexible since it only starts the actual telephony service. This script takes argument, which is the name of the configuration file.

```bash
$> ./start.csh [file.cfg]
```

If the argument is left out the default configuration file will be used if available, if not default properties will be used.

When the system is up and running there should be two windows displaying broker resources and session status on the screen.
B.4 Registration and download

In order to allow for user registration and speaker model download the respective servers have to be started. Note that when the registration process is handled by a web interface there is no server required except the database server and of course a properly set-up web server.

$> ./regsrv.csh [file.cfg] [port]
$> ./dlsrv.csh [file.cfg] [port]

As stated the command line for the servers are quite similar, the only difference is the scripts names. The regsrv.csh script starts the registration server and likewise the dlsrv.csh script launches the download server. As denoted above both scripts can take two arguments, but they are both optional. The first argument specifies a configuration file and the second is used if a port number that differs from the one supplied within the configuration file is to be used. One of the major drawbacks of the registration server is shown here and that is the fact that if the default port number is not used the port number would have to be distributed to all the users. This drawback does not exist in the case of a web interface, where all the database connectivity data is stored in the script file on the server.

B.5 Database design and maintenance

The database consists of one table created according to the below example. Of course a more complex table could have been constructed, but in order to make a fully functional application this solves the task well. An observant reader might realize that the password is the only thing that identifies a user in this database. This may seem somewhat inconvenient. On the other hand if the primary key would consist of more than just the password the same password would apply to more than one user. This would be a drawback since the recognizer sometimes does not return the correct username and password, this could yield a larger amount of erroneous user logins.

```
CREATE TABLE users(
  fullname VARCHAR(40) NOT NULL,
  transcription VARCHAR(100),
  password INT(4) ZEROFILL NOT NULL AUTO_INCREMENT,
  email VARCHAR(40),
  changed TIMESTAMP(14),
  created TIMESTAMP(14),
  enrolled BOOL,
  modelfile VARCHAR(200),
  PRIMARY KEY(password));
```

The ultimate solution however is to use a personal identification number as primary key and let the user choose the password herself. This would probably be the case in the real world.
B.5.1 Database access

As the applications require different levels of access to the database a set of user accounts should be constructed to increase security. For example the registration service and the main application requires read and write capabilities, but the download service only requires read access to the database.

B.6 Tweaking the system

The system has been designed to be flexible and to allow changes in application behaviour. For example the application can be set to only record user utterances without trying to recognise them, the number of recognition tries per utterances can also be set via the properties file. There are many parameters that can be set in the properties file. To simplify administration it is also possible to use a properties file with another name, this makes it easier to keep different set-ups by having different configuration files. Which properties file to use, is specified by passing the file’s name as a command line argument when launching the application. All properties that can be set are specified below and in Javadoc.

B.6.1 Properties

This is a brief description of the properties that can be set in the properties file. For a more detailed description see Javadoc and the properties file.

- **adaption.log**: Specifies which log file to write adaptation output to.
- **adaption.script**: This property specifies which adaptation script to use.
- **broker.host**: Broker host.
- **broker.port**: Broker port.
- **country**: Country for locale setting.
- **debug**: This property can be set to either “yes” or “no”. Setting it to “yes” enables debug mode.
- **digitizer.host**: Digitizer host.
- **grammar.enable**: Enable dynamic generation of username grammar. (true/false).
- **grammar.lex.file**: Which grammar LEX-file to use when storing the username grammar..
- **grammar.lex.hmmfile**: Specifies which model to use when building username grammar.
Username grammar parameters

These can be set in order to use pruning on the username grammar:

- grammar.lex.param.backprune
- grammar.lex.param.backwidth
- grammar.lex.param.forwprune
- grammar.lex.param.forwwidth
- grammar.lex.param.penalty=

**grammar.slx.header**

Specify a header file for the dynamically generate username grammar.

**grammar.slx.output.file**

Specify the output filename for the username grammar.

**grammar.slx.symbols**

File containing the SLX-symbol set to be used in the username grammar.

**grammar.slx.symclasses**

File specifying the wordclasses for the username grammar.

**grammar.slx.trailer**

Specifies the trailer file for the username grammar.

**hmm.set**

HMM set.

**isdn.server.phone.number**

Phone number for the ISDN Server.

**language**

Language for locale setting.

**lexnet.file**

Lexnet file to use.

**local**

Whether to use a local terminal, i.e. microphone input or not. (yes/no)

**login.attempts**

Maximum number of login attempts.

**login.process.time**

The maximum process time to process a login utterance.

**login.record.time**

The maximum length of a login utterance.

**login.skip**

Set this to “yes” to skip login. All recorded data is stored in “test.test”-directory. (yes/no)

**model.directory**

Set model directory.

**output.directory**

Set model output directory.

**protocol.file**

Name of protocol file.

**protocol.length**

Number of sentences in protocol file, should be one per line and hence this is the number of lines that will be read in the corresponding file.
recognition.attempts: Maximum number of tries on each utterance before moving on to the next utterance.

recognition.skip: Set this to “yes” in order to just record user input without trying to verify it with the recogniser. (yes/no)

recognizer.base.tag: Path to StarLite.

recognizer.digits.lex.dir: The directory where the “4digits.LEX.gz” resides.

recognizer.digits.lex.set: This is the “4digits.LEX.gz” file. If any other digits lexicon is to be used this property and the above should be changed.

recognizer.hmm.file: The HMM to use, the model must be in StarLite-format.

recognizer.hmm.set: Which HMM set to use. The “mon8”-set was used in this project.

recognizer.hmm.tag: In which directory to find the HMM.

recognizer.hypothesis: Number of hypotheses to take in to consideration for each utterance.

recognizer.lex.dir: The path to the lexicon files.

recognizer.lex.set: Which lexicon set to use.

recognizer.lex.tag: Where the lexicon is located.

recognizer.name: The name of the recogniser.

recognizer.protocol.lex.dir: The path where the protocol lexicon is stored.

recognizer.protocol.lex.set: The protocol lexicon’s filename.

recognizer.protocol.mode: Recogniser mode, remember to change “lex.set” accordingly. (sentences/words)

recognizer.server: The server on which the recogniser resides.

recognizer.user.lex.dir: The path to the username lexicon.

recognizer.user.lex.set: The username grammar filename.

recording.path: Where to store the recorded data.

remote: Use a remote terminal, i.e. a telephone terminal. (yes/no)

service.name: The name of the service (Tilltalad)
soundcoder.host Soundcode host.
sql.database The name of the database.
sql.host Database server and port on the form “server:port”.
sql.password Database password.
sql.user Database user.
text.strings.file The file that contains the language specific utterances for the dialog. (“data/swedish”)
tts.factory TTS Factory name.
tts.name TTS engine.
tts.server TTS server.
tts.voice Which voice to use (default is ingmar)
utterance.process.time The maximum utterance process time, zero (0) means unlimited.
utterance.record.time.min The minimum time for a user utterance.

### B.6.2 How to create a lexicon.

The first step is to write a file containing the protocol for the user to read. After writing this file, the text must be transcribed to phonemes. This is made by parsing text through a speech synthesis program, that except voice output also supplies phoneme output. The phonemes are saved in a separate file.

Avsikten är att kartlägga var de ur beredskapssynpunkt största svagheterna finns.


Some of the phonemes have to be altered to make them more natural. This requires some manual work. The phoneme output now contains some unwanted characters, such as ’, ” and lower case letters, these have to be removed before making a lexicon file. This is done by simply removing them with an editor, such as Emacs. To generate a cleaner and more uniform phoneme transcription the file is parsed with the program STASplit which is part of the StarLite suite. The output from STASplit consists only of RULSYS phonemes and looks like the example below.
Before the lexicon can be built, all the parts must now be concatenated to one single file according to the definition of StarLex files (.slx). After this has been properly done it only remains to build the Lexicon file (.LEX) with the script BuildLex, see example below.

```bash
$> BuildLex -H hmmfile protocol.slx protocol.LEX
$> gzip protocol.LEX
```
C. User’s guide

C.1 Registering
Before placing a call to the adaptation service the user has to get registered in order to receive a username, password and the session protocol. This can be done thru a web browser or by using the registration client software. The easiest way to get registered is by surfing to a web page containing a registration form. The service provider must of course supply the web address where the form can be found. If the registration client is to be used the service provider has to provide the user with necessary information about which server and port to connect to. The web registration is preferable since the interface is more convenient and more flexible.

C.2 Placing a call
Once the user has registered her she has to wait a while for the service to update the name grammar. The amount of time the user has to wait between registration and the first call must of course be specified and sent to the user when she registers.

When the specified amount of time has elapsed the user can pick up the phone and dial the service, the service will then prompt the user for different actions in order to log in and start the training session. When the training session is over the user will be asked to hang up the phone and will also be notified when the adapted model is trained ready for use.

C.3 How to use the adapted model
The user has to notify the service provider that she has a personal model stored, and the service provider of the application in which the user wants to use her personal speaker model can download the model.
D. Model parameters

This appendix defines the model parameters that are used when adapting the personal speaker models. The following settings were used, they are all documented in the HTK Book. Accept these settings a filter called “alwfilter” is used in order to convert the A-law encoded speech data to a PCM-coded waveform.

```
# Input file format (headerless 8 kHz)
SOURCEKIND  = WAVEFORM
SOURCERATE   = 1250   # Sample frequency 8 kHz

# Mel-frequency cepstral coefficients with first order regression coefficients (delta coefficients) and second order regression coefficients (acceleration coefficients). 0th order cepstral coefficient is appended.
TARGETKIND   = MFCC_D_A_0
TARGETRATE   = 100000.0 # Sample period 10 ms
WINDOWSIZE   = 320000.0 # 256 samples * 1250 (100ns) per sample, 32 ms

USEHAMMING  = T     # Hamming window is used.
PREEMCOEF    = 0.97  # Pre-emphasis coeff.
NUMCHANS     = 24    # Number of filterbank channels
CEPLIFTER    = 22    # Cepstral liftering coefficient
NUMCEPS      = 12    # Number of cepstral parameters
ESCALE       = 1.0   # Scale log energy
LOFREQ       = 200   # Lower frequency cut-off
HIFREQ       = 3800  # Upper frequency cut-off

SAVEWITHCRC  = F     # Attach a checksum to output parameter file
USESILDET    = F     # Disable speech/silence detection.
MAXTRYOPEN   = 3     # Number of file open retries.
FORCEOUT     = T     # Output partial recognition results.
NONUMESCAPES = T     # Prevent string output using \012 format
ALLOWCXTEXP  = F     # No expansion of phone names is performed and each phone corresponds to the model of the same name.
```