Creating a Voice for Festival Speech Synthesis System.

Submitted in partial fulfilment
of the requirements of the degree
Bachelor of Science Honours
of Rhodes University

By: Matthew Hood
Supervisors: S. Bangay, A. Lobb

8th November 2004

Abstract

One of the factors that limit the usefulness of Text to Speech Systems is the limited number of voices. This paper looks at Festival, a concatenative speech synthesis system, and how to create a voice for it. The process is fairly involved and human intensive. The paper gives a step by step approach to making a voice from scratch. As well as going into more detail on the areas of recording and labelling, both these have a big impact on the final voice quality. The auto labeller is looked at in more detail and results are obtained for its effectiveness, there is also an informal evaluation of the final voice.

Acknowledgements

I would like to acknowledge the financial and technical support for this project of Telkom SA, Business Connexion, Comverse SA, and Verso Technologies through the Telkom Centre of Excellence at Rhodes University.

I would also like to thank my two supervisors S. Bangay and A. Lobb who have helped on a number of issues and guided me through the year. Also thanks to all the other members of the VRSIG group whose weekly contributions have been a big help.

Finally a big thanks to my parents for getting me this far.

Contents

Chapter 1

Introduction

Much work has been done in recent years to try to make the process of voice creation easier and more automated. While it has helped, making a voice is still a complex process. This thesis aims to ease the reader through creating a voice, pointing out the pitfalls and explaining what can be a confusing process. It is obviously necessary to understand a little of how a text to speech system works and how the voice fits into that system. This is followed by an explanation of installing and using the software. Then a detailed step by step guide to building the voice. There is further discussion on recording and labelling. Finally the success of the auto labeller and overall voice quality is analysed.

Chapter 2

Text to Speech

You may be surprised to learn that synthetic speech systems have been around for over fifty years. Text to Speech systems as we know them today have been around for many years, but their application has been limited. This is for a number of reasons, intensive processing, poor quality output, limited application. However much of this is changing. With increased use of the internet and mobile devices new uses for text to speech systems are being found all the time. There is much research into digital actors, avatars and news readers.

One aspect that has always limited the usefulness of text to speech systems has been the limited number of voices and the difficulty in building new voices. In the past it has been a mammoth task to build a voice. [Black et al; 2003] report that in the early days of text to speech it could take up to a year for a voice to be built. Even today building a voice takes many hours and is a laborious process.

2.1 General Makeup of a Synthesis System.

In general a speech synthesis system can be broken into 3 parts. By doing this it means that there is a level of abstraction between the lower and upper layers. This means that voices can be built without having to worry about how they are to be used.

Text analysis

This is where the input text is analysed and words and utterances identified.

Linguistic analysis

Here the context of the words is analysed and hence its pronunciation can be worked out and utterances generated. This is a complex task, as often the same word is pronounced completely differently in different circumstances. E.g. 1990, is this nineteen ninety the date or one thousand nine hundred and ninety the number.

Waveform generation

This is the only layer that really concerns voice creation. Here the utterances generated after linguistic analysis are spoken. There are many different approaches, the most common and powerful approach today is to combine pre-recorded sound clips to generated entire spoken segments. This is the approach used by Festival and will be the one discussed throughout this paper.

2.2 Concatenative Speech Synthesis

A concatentative system works by combining sound clips. The easiest way of explaining it is with the talking clock example. To tell the time the system must have all the following clips saved.

"The time is" ; *"past"*; *"to"*; *"half"*; *"o'clock"*; and all the numbers.

Then by combining these it can speak the phrase

The time is half past three

This is the basic principle used in all concatenative systems. The system can only speak what it has available, and while this approach is often used for very specific systems (like any of the talking clocks), from the above example it should be obvious that recording such large sound clips is not practical for a more general synthesiser. The speech needs to be broken down into smaller components, ones that can be used to make any spoken word.

2.3 Phones and diphones.

Each language has a set of phonetic sounds associated with it. These are known as a phoneme set. This set comprises of every sound made when speaking any word in the language. The set of phonetic sounds varies from one language to another and this is why people not familiar with a language have such difficulty pronouncing its words. Think back to a case of pronouncing a foreigners name. See the appendix at the back of the paper for a full list of all 44 phones used by Festival for US English, as well as example pronunciations.

If every word in the language is made up by combining these phones then surely a concatenative system can work in the same way, and to a degree this is how they do work. Unfortunately it is not as easy as recording all the phones.

When we talk we do not always pronounce a phone in the same way. The neighbouring phones influence how we pronounce the desired phone. This is most often due to the changes in the mouth position as it moves from one phone to another and is known as the co-articulatory effect.

Festival relies on the simplifying assumption that a phone is only affected by its neighbouring phone, these phone pairs are known as diphones. So by recording the sound of a phone in context of its neighbour it is then possible to combine these diphones to sound like normal speech.

For example to say the word "JACK" a human would say the phones *jh-ae-k*.

Festival will require four diphones in order to say the same word.

pau-jh ; *jh-ae* ; *ae-k* ; *k-pau*

If you are unfamiliar with phonetic sounds *pau* indicates a silence. So by combining these four diphones the text to speech system can synthesise the word "JACK".

Chapter 3

Related Work

3.1 Current Speech Synthesis Systems

There are many uses for text to speech systems, these include: digital actors, avatars, digitised speech (in conjunction with speech recognition and possibly a translator), next generation chat applications. As such there is a lot of research in the area. The following are some of the better text to speech systems.

3.1.1 Commercial:

Whistler (Windows Highly Intelligent STochastic taLkER) was developed by Microsoft in 1998 and today is used in a number of their applications including Narrator as part of Windows 2000 and XP. The over all quality of the speech is not great and sounds like the typical talking computer. The project originally proposed a novel voice training system. Unfortunately this research appears to have been discontinued and no information about the training was released. The present system has no facilities to make new voices.

SoftVoice developed by SoftVoice, Inc was first written in 1979 and is now released as a set of Windows DLLs to enable programmers too easily add speech synthesis capabilities to their products. SoftVoice gives decent voice quality, but lacks any programs or documentation on how to create a new voice.

Lucent TTS engine was developed by Bell Laboratories. The engine is to be used in a number of Bells own telecommunication and messaging products as well as being available to other developers. The system works primarily within the context of a telephone system and is obviously written with that in mind.

Natural Voices is an ongoing research project at AT&T. The aim of the project is to get as natural as sounding speech for use in telephone systems. The quality of the output produced by the system is very impressive and has been used in a number of movies and recognised as a leading example by Discover Magazine in their article The Mathematics of . . . Artificial Speech [Discover Magazine, 2003]. At present there is no facility or documentation on new voice creation.

3.1.2 Academic:

Center for Speech Technology Research (CSTR) at the University of Endinburgh is one of the leading research groups in the field of Text-To-Speech. It released the Festival speech synthesis system, the TTS used in this project. The CSTR also have the FestVox project, which is a project looking at voice creation for Festival. Festival uses the concatenative approach, as do almost all of these systems. Its speech quality is good, but can be inexpressive. It is also not as clear as the Natural Voice project. Festival is designed to be easy to use as well as edit. It is fully open source and modules can easily be added to it. As such many other researchers use Festival in their own work. It also is one of the only products to actually look into creating voices.

Center for Spoken Language Understanding (CSLU) is a research group at OGI School of Science and Engineering. This research group looks at a number of areas related to speech and have a detailed section covering speech synthesis. The group use Festival as one of their major components, unfortunately their section on voice creation is not available at this time.

Speech Research Lab at the University of Delaware is a group interested in speech synthesis for use with disabled or ill patients, enabling them to better communicate with others. The group have developed the ModelTalker TTS system, the system produces output comparable to Festival, but is aimed more at the end user, and it does not have the same level of control and customisation. There is also the InvTool program which is designed to help create new voices for ModelTalker. The system works the user repeating words spoken by ModelTalker. This makes the process much easier for a novice user and also by analysing each sound when it is recorded, before continuing onto the next one it means a lot of the extra effort put into cleaning up the sounds is done away with.

3.2 Literature

As far as literature covering voice creation, there is only really the work done by the Center for Speech Technology Research. Their main documentation Building Synthetic Voices [Black et al; 2003] covers the process of creating a voice for Festival. Unfortunately this document still has chapters missing and is obviously still a work in progress. It also covers quite a large topic, dealing with different types of voice and languages. The document is invaluable to voice builders, but as it is not complete and fairly complex, a first time voice builder will struggle to build a voice of decent quality. In particular the section on labelling has not been written and this is one of the hardest parts of making the voice. It is unlikely this document will ever be finished as in the next year Festival 2.0 will be released and presumably chances will be made to how voices function and are built.

Another valuable resource is the FestVox homepage and in particular the forum area, where voice creation is discussed and questions can be posed to the professor in charge of the project. In my experience they have been very helpful.

Chapter 4

Software needed

Creating a voice needs very little software. One will, obviously, need Festival which relies on the Edinburgh Speech Tools. The Festvox tool kit will make the process much easier. When labelling, a valuable piece of software is EMU Label, which is part of the EMU tool kit. Each package is described below as well as basic instructions for its installation and use.

The basic system used in this project was Mandrake 9.0 and all instructions are aimed at that system. But because all programs are installed from source, it should be possible to use them on any Linux variant with a C/C++ compiler, this has been tested under Debian 3.0. Every program used is also available for Windows, but I have not used them and hence will not give any instructions.

4.1 Festival and Festvox

"Festival is an open source text to speech system developed at the University of Edinburgh. Festival offers a free, portable, language independent, run-time speech synthesis engine for various platforms under various APIs." [Black et al; 2003]. As this paper focuses on voice creation, there is little point in going into Festival in great detail, the information provided is not intended to be a Festival user manual and will only focus on the bits pertaining to creating a voice. While it is possible to make changes and improvements to the Festival system this is not needed for basic voice creation and Festival should be installed standard except where explicitly stated.

4.1.1 Installing Festival and Festvox.

Festival depends on the Edinburgh Speech Tools, and these are provided with Festival when it is downloaded. The three packages, Speech Tools, Festival and Festvox each need to be installed separately and in that order. When downloaded there are a number of compressed .gz files. It is recommended that you uncompress all of them before carrying on as some will add files to more than one of the three packages. To uncompress use

```
tar -xvzf thefile.tar.gz
```

Once uncompressed it is a simple case of changing to the relevant directory and using.

```
./configure
make
```

If there are errors while compiling check to see that, first you are doing this as super user (su). Also make sure everything is uncompressed properly and that you are compiling in the correct order. Lastly you can try using

```
make clean
```

Or

```
make all
```

These should sort out any linking problems. In my experience normally the process goes off without a hitch, but there have been a few cases that errors just keep happening, in these cases it is often best to delete all three packages and start again.

At this point it may be worth mentioning Linux sound drivers, the biggest problem I have had with getting all the software up and running is getting the sound drivers working properly. If you are using a Linux like Mandrake then it should set everything up for you, but often there are still problems. The most common one is that another process keeps forking the sound driver and Festival will sit and wait for it to be free. This results in no speech for up to a minute then the system seems to catch up. My best advice is to make sure you are running the most recent version of which ever Linux you are using and that you have the latest kernel. The newer kernels should sort out a lot of the problems as they use ALSA. Also choosing a sound card with good Linux driver support will also help a lot.

After Festival has been compiled there are a few environment variables that need to be set. The first is to add Festival to the PATH so it can be accessed from anywhere in the system, this can be achieved on most systems by

```
export PATH=$PATH:/home/...../Festival/bin/
```

(the ..... must be replaced with the user directory for that machine,

eg /home/miy59/work/Festival/bin/)

Before the FestVox tools can be used two more variables need to be set these can be done as follows

```
export ESTDIR=/home/...../speech_tools
export FESTVOXDIR=/home/...../FestVox
```

Everything should now be set to go. If there are still problems please refer to the relevant help files or look on the internet at `http://www.festvox.org`

4.1.2 Using Festival

You will never have to actually use the speech tools directly, and more information on using FestVox will be covered in Chapter 5. So for now, I will only give a basic overview of some of the commands in Festival that you will find useful when making and testing your voices.

```
festival
```

This will launch the Festival application and give you a prompt:

```
festival>
```

Any commands shown after such a prompt are meant to be typed once Festival is running. To exit Festival use (ctrl-d)

```
festival>(SayText "Hello World.")
#<Utterance 0x402a9ce8>
festival>
```

SayText is the most simple use of Festival, its takes a string argument and speaks it in whatever voice is currently selected. As standard this will be Kal.

```
festival>(SayPhones '(pau ta aa k a k aa pau))
#<Utterance 0x404b6ee1>
festival>
```

SayPhones is similar to SayText but takes a list of phonetic sounds. This is useful if you want to test that each diphone in the voice works properly.

```
festival>(voice_cmu_us_kal_diphone)
cmu_us_kal_diphone
festival>
```

The voice command changes the current speaking voice. Note that you can only load voices that are installed into Festival, one that you are still building and have not finished compiling cannot be loaded this way, more detail on loading unfinished voice is given in Chapter 5.

```
festival>(tts mybook.txt nil)
t
festival>
```

SayText generates the entire utterance before speaking it. This is not really suitable for longer pieces of text, so instead one should use the tts function which will speek one utterance while working on the next. The nil at the end denotes which mode tts is in, for more info on the tts nodes look in Building Synthetic Voices [Black et al; 2003].

Festival can be given arguments which it will execute one at a time, and if the b argument is used it will not go into interactive mode when it is finished, but rather exit back to the system.

```
text2wave -eval "(voice_ked_diphone)" mytext.txt -o output.wav
```

Text2wave takes in a text file and instead of speaking through the speakers will save the output to a .wav file. Optionally you can use -eval followed by a series of Festival commands to be executed before the file is spoken. This is very useful when making sound clips for later use.

4.2 EMU Label

Emu Label is a program that takes a list of sound files (normally .wav) and a list of label files (.lab) and shows graphically where in the sound file the labels are. It allows you to add delete or move any of the labels. EMU Label is part of the EMU speech tools 1.8, the other tools in the kit, while useful are not necessary for making a voice.

4.2.1 Installing EMU Speech Tools

EMU requires you to have previously installed the tcl and BWdiget packages, it will not install if these are not already present. There is supposed to be an automated installer, but I have never gotten it to work, instead I normally compile from source. This is not quite as easy as compiling Festival, but the approach is similar.

```
./configure --with-tcl=/usr/lib/tcl8.3/ \
--with-tk=/usr/lib/tk8.3/ \
--with-estools=/home/miy59/work/speech_tools/
```

Note the \ indicates a newline in the paper and should not be typed. The first two arguments tell it where to find tcl and tk, on some systems these can be left out, but generally it is a good idea to specify where they are. If you need to search for them on your machine use

```
locate tclconfig
locate tkconfig
```

The third argument tells emu where the Edinburgh speech tools are. This is optional, but recommended, as it adds extra file support to the emu set of tools. After that it is a simple case of.

```
make
make install
```

It is worth mentioning that version 1.7 had an error in one file that needed to be fixed before you could compile it. This problem seems to have been fixed in version 1.8, but keep as eye out.

4.2.2 Using Emu Label.

Emu Label has a simple GUI interface. Actually using it to label files will be covered in more detail in the Chapter 7.

Chapter 5

Building a voice.

This chapter covers the fundamental steps involved in building a voice for Festival using the FestVox toolkit. It is taken almost entirely from Chapter 19 of Building Synthetic Voices [Black et al; 2003]. These steps are for a voice built using an existing language, in this case US English, which has already been setup and diphone lists constructed.

There are three main phases to building a voice, recording, labelling and hand correction. These are covered in later chapters. This chapter should merely be a reference point when actually building the list, it is nothing more than a list of steps and a short explanation about what is happening at each stage. It is possible to build a voice by using nothing more than this chapter, but I would urge you to at least look over other parts, otherwise you will waste a lot of time and possibly end up with an inferior voice.

The process is fairly long, recording and labelling take a few hours each, and after that there may still be extensive hand correction. It also requires a fair amount of disk space so make sure you have at least 500 Meg free. The basic list of steps is as follows.

Create templates.

Generate prompts.

Record nonsense words.

Autolabel recorded words.

Generate diphone index Generate pitchmarks and LPC co-efficients.

Test Package for distribution.

Before continuing make sure that you have Speech tools, Festival, FestVox and EMU Label correctly installed. Also make sure the following environment variable are set.

```
export ESTDIR=/home/...../speech_tools
export FESTVOXDIR=/home/...../FestVox
export PATH = $PATH:/home/...../Festival/bin
```

1. Make a directory and change to it. By convention the directory is named
Insitution_language_name_type e.g.

```
mkdir ~/ru_us_matt_diphone
cd ~/ru_us_matt)diphone
```

2. FestVox provides a tool to build the basic directory structure. It takes institution, language and name as arguments. E.g.

```
$FESTVOXDIR/src/diphones/setup_diphone_ru_us_matt
```

The setup script also needs to copy in some language specific files. For US English the following packages will need to be part of Festival.

```
FestVox_kallpc16k
festlex_POSLEX
festlex_CMU
```

3. The nonsense word list must be generated.

```
festival -b FestVox/diphlist.scm FestVox/us_schema.scm \
'(diphone-gen-schema "us" "etc/usdiph.list")'
```

The \ signifies a new line in this paper and should not be typed.

4. The prompts must be synthesised so that Festival can prompt the user before recording the diphones.

```
festival -b FestVox/diphlist.scm FestVox/us_schema \
'(diphone-gen-waves "prompt-wav" "prompt-lab" \
"etc/usdiph.list")'
```

5. Now the actual diphones can be recorded.

```
bin/prompt_them etc/usdiph.list
```

If you want to start recording from a specific diphone, for example if you are recording in multiple sessions you can supply the diphone humber as an argument.

```
bin/prompt_them etc/usdiph.list 101
```

6. Once all the diphones have been recorded you can proceed to autolabel.

```
bin/make_labs prompt-wav/*.wav
```

7. If you wish to hand correct labels you can launch emulabel.

```
emulabel etc/emu_lab
```

8. Now the diphone index must be built.

```
bin/make_diph_index etc/usdiph.list dic/mattdiph.est
```

9. Next is pitchmark extraction, and then moving it to the nearest peak.

```
bin/make_pm_wave wav/*.wav
bin/make_pm_fix pm/*.pm
```

10. You can optionally match the power, first the files must be analysed and a mean factor extracted

```
bin/find_powerfactors lab/*.lab
```

And finally you can use this to build the pitch-synchronous LPC coefficients

```
bin/make_lpc wav/*.wav
```

The database should now be ready for initial testing.

```
festival FestVox/ru_us_matt_diphone.scm \
'(voice_ru_us_matt_diphone)'
```

It is worth repeating now, that this is merely a list of steps, many things could have gone wrong up till now, consider reviewing the relevant chapters if you have any troubles. Once you are happy with the voice it is ready for packaging and distribution.

11. A group file must be built that contains only the bits needed from the larger wave files.

```
festival (us_make_group_file “group/mattlpc.group” \
nil)
```

12. The new voice must be integrated so that Festival can find it. To do this change to Festivals voice directory

```
cd /home/...../Festival/lib/voices/English/
```

13. Finally add a symbolic link back to where the voice was built

```
ln s /home/./ru_us_matt_diphone
```

Now anyone using Festival on the machine will have access to the voice.

Chapter 6

Creating the voice

6.1 The Diphone List

Any concatenative text to speech system must have every sound it uses pre-recorded and labelled. It has a fixed choice of sounds to choose from and does not ever mix its own. So when making a voice for a diphone based system one needs to record every diphone pair spoken in the language the voice is intended for. In general this is a square of every phonetic sound used to speak the language. In reality there is often no need for some of the matches as they are never spoken, and there may, in fact, be extra phonetic sounds to consider. These extra sounds will normally result from strong accents.

It is worth considering adding these extra diphones to the list if you are making a voice of someone with a distinct accent that you do not want lost. There is also the problem that if words foreign to the language need to be spoken by the voice the diphones will be missing and hence the speech will not be able to pronounce them. It is then also important to consider what the voice is likely to be saying and if needed add foreign phonetic sounds to the list, the speaker being recorded may have trouble pronouncing them if they are from another language but at least the result will be better than having none at all.

6.2 Recording the Diphones

It is not enough to merely say the diphones, because of the co-articulatory effect the diphones must be in context of other phones, so there are two options for how you can record the diphones.

The first is to speak real words that contain the diphone required. This entails first finding all these words, but with the help of a linguist this is possible, and then extracting the diphones from the words. Because every word is different this means it will have to be done by hand, and even then the quality is not likely to be that great. It is almost impossible when speaking real words to pronounce each one the same , this is due to the humans adding emotional effects to the words.

The other option is to speak nonsense words. The diphone can then be pronounce in the context of a word, even if it is not a real word. This has the added advantage of making every diphone uniform and occur in the same place in the word, which makes labelling much easier later on. The general construction of a nonsense word follows a few basic principles

The desired diphone must be in the middle of the word encased in other phones which are uniform across all the nonsense words. In general we will use the t and aa sounds to make up the rest of the word.

You cannot have three soft or three hard sounds together, by soft I generally mean vowels and most hard sounds are consonants. This means that if you are recording a soft diphone pair (*ey-ae*) then before and after it should be a hard sound e.g. *t-ey-ae-t*. If you are recording a hard-soft (*k-ey*) combination then it is possible to record the soft-hard (*ey-k*) one at the same time, and encase these in other soft sounds e.g. *aa-k-ey-k-aa*.

Festival includes basic diphone lists for some languages, in this paper we will be using the standard US English diphone list produced by Festival. This list should suit most peoples needs, but can be extended. The format of a diphone list must match the following or the tools and scripts available to Festival will not be able to use the list. Each line of the file should contain a file id, a prompt, and a diphone name (or list of names if more than one diphone is being extracted from that file). E.g.

(us_0001 “pau t aa b aa b aa pau” (“b-aa aa-b”))

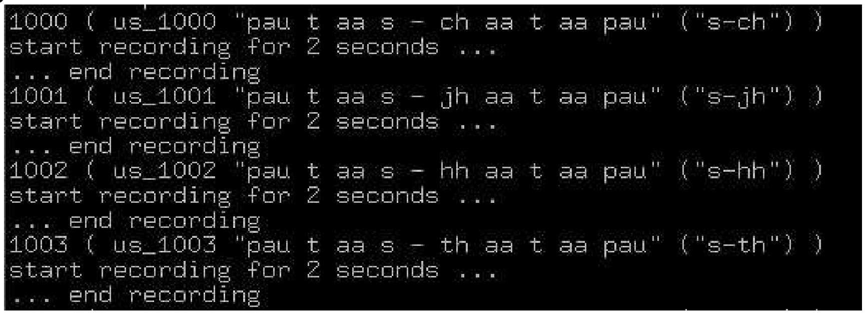
(us_0002 “pau t aa p aa p aa pau” (“p-aa aa-p”))

..^file ID... ^spoken prompt............ ^ name of diphones being recorded

6.3 The Speaker

It is preferable when making a voice, to use a trained speaker, or at least someone with experience at talking audibly for long periods. Trained speakers are able to talk in a constant even monotone fashion, which is preferable, with out loosing their individual accent. And also due to the number of diphones that need recording (anywhere from 1000 - 2500) it can take over one and a half hours and is fairly taxing on the speaker. It may be worth considering doing the recording over a couple of sessions, but then much care must be taken to ensure the recordings are consistent. This means setting up the venue the same, ensuring machine settings are constant and that the speaker is in a similar condition i.e. not tired, sick, hung over etc. Some even go as far as recommending that the recordings are done at the same time of day.

Pronouncing these nonsense words can be quite tricky, especially when you first try, so the speaker should have gone through the list of phones and their pronunciations found in the appendix. This will save a lot of time rerecording diphones later because their pronunciation was wrong. It is possible, and recommended, to then use the diphone list to prompt the speaker before they record the nonsense words.



```
1000 ( us_1000 "pau t aa s - ch aa t aa pau" ("s-ch") )
start recording for 2 seconds ...
... end recording
1001 ( us_1001 "pau t aa s - jh aa t aa pau" ("s-jh") )
start recording for 2 seconds ...
... end recording
1002 ( us_1002 "pau t aa s - hh aa t aa pau" ("s-hh") )
start recording for 2 seconds ...
... end recording
1003 ( us_1003 "pau t aa s - th aa t aa pau" ("s-th") )
start recording for 2 seconds ...
... end recording
```

Figure 1.

It works on a repeat after me basis, the computer speaks the word to the speaker who then repeats it back for recording. This has the advantage of making the speaker repeat the words in a consistent way and also helps with pronouncing, what can often be, fairly ambiguous sounds. Care must be taken to ensure that the speaker does not start imitating the voice being spoken and loose their own accent. It is of course not possible to do this for a new language which does not have a voice to speak in the first place. It has been recommended that one can boot strap diphones from other languages to use in such a situation, but I cannot comment on its effectiveness.

6.4 Recording Conditions

The environment in which the recording is done is very important, ideally it should be done in an isolation booth or recording studio. However with modern computer technology it is feasible to record straight onto the computer. All recordings I have made have been done this way in a normal room and the sound quality is acceptable for a basic voice. The biggest problem I have experienced is the poor sound support in most Linux distributions. To this end I strongly recommend upgrading to the latest release and the latest kernel version. Also if possible when setting up the machine consider buying a sound card that has good Linux driver support and is well documented, this will save you hours of trouble later.

If the work was recorded in a studio, probably on to DAT, then there is the problem of it being one long recording and not separate files. Oregon Graduate Institute use a system of introducing an identifiable noise between recordings so that they can be easily split once loaded onto the computer. One of the most important features is that the environment is quiet, there should be little or no background noise, and for this reason working at night is preferable.

The choice of microphone is crucial and should be of the highest possible quality. It is widely acknowledged that head mounted microphones are preferable when recording speech. They limit background noise and have the distinct advantage of always being the same distance from the speakers mouth. This makes the sound levels easy to control across sessions and allows the talker to move around, sit etc which will help as the recording takes a long time.

The other option mentioned in Building Synthetic Voices [Black et al, 2003] is the use of an electroglottograph (EGG), which instead of recording the sound coming out of the speakers mouth, uses electrodes on the speakers throat to pick up vibrations.

It is likely, even with trained speakers, that some of the recordings will be unsuitable. It may be possible that you do not need to re-record the sounds. Especially if there are only a few mistakes it is tedious to setup the environment and get the speaker back. In these cases one can either bootstrap the sounds from another wave file in the same voice, e.g. the b and p sounds are very similar, or use a file from another similar voice. Obviously doing either of these will lead to a drop in voice quality, but if there are only a few examples and if they are not very common sounds no one will notice.

Chapter 7

Labelling

Labelling is one of the most important, and difficult, parts to making a voice.

7.1 Label files

Each wave file needs a corresponding label file. Basically a label file is a set of times showing where in the wave file various phones are located. Below is an example of the first label file.

```
separator ;
nfields 1
#
     0.10000 26 pau
     0.72000 26 t
     0.84000 26 aa
     0.93000 26 b
     1.05000 26 aa
     1.18500 26 b
     1.42000 26 aa
     2.12000 26 pau
```

It is possible to create and edit these files manually, but that very tedious. The recommended approach is to use EMU Label.

7.2 EMU Label

EMU Label shows graphically where the phones are located in the wave form. For EMU Label to correctly link the label and wave file they must have the same name, and there must be a .tpl file telling EMU Label where to look. This file is found in /etc/ in the voice directory and is called emu_lab.tpl. Sometimes EMU Label cannot find the relevant files, in this case you need to edit the tpl file to reflect where the files are. In most cases all that need to be done is to change the path section. For example

```
path lab /lab
path hlb /tmp
path wav /wav
```

Should be changed to

```
path lab ../lab
path hlb /tmp
path wav ../wav
```

Below is an example of the first file loaded into EMU Label.

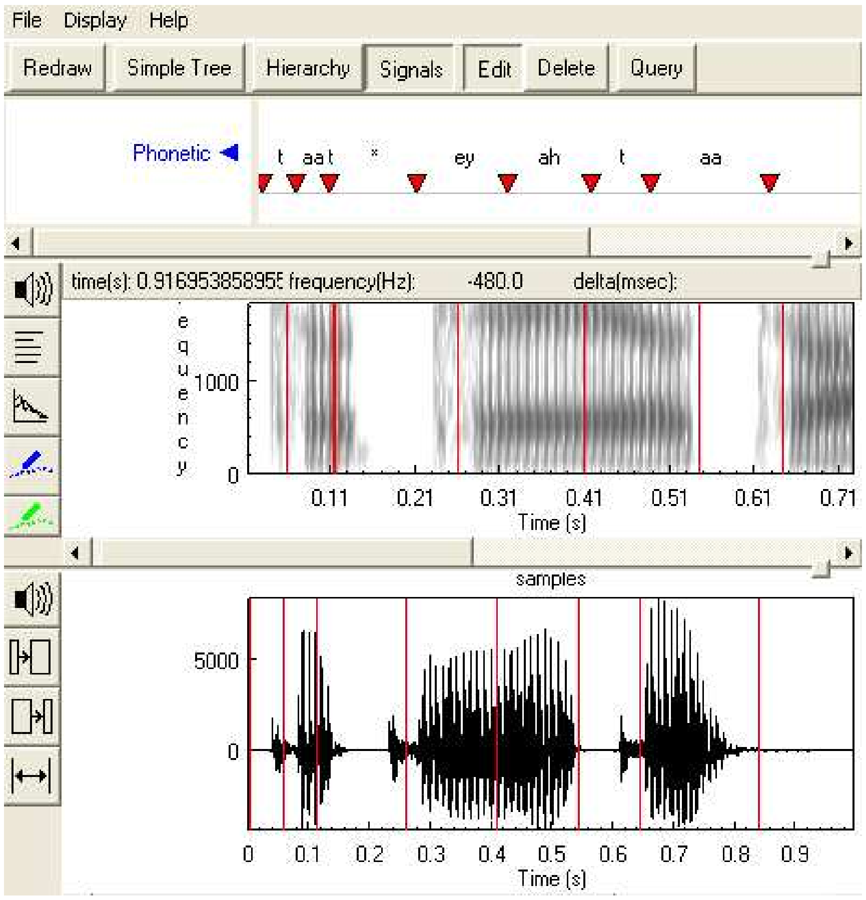


Figure 2.

The top segment shows the phones in the label. The middle piece shows the frequency and the bottom section the wave form.

7.3 Labelling advice

Festival extracts the sound from the mid point of the phone labels, this gives you a little bit of lea-way when positioning the labels. But extra time spent here greatly improves the quality of the voice. The previous example shows a file that has been hand labelled, it contains labels for all the phones in the file, even though only the diphone needs to be extracted, this is the *ey-ah* sound. It is always worth labelling all the phones.

The end point of one phone is always the starting point of the next, so in some cases you may consider inserting blank or useless phones to pad in between so as to help with alignment.

Phones can be added by clicking on the line in the top section. The position of the edge markers can be changed by dragging the red triangles. If you right click on the name you can delete or rename the phone.

If you click and drag on any area in the frequency or wave form parts they will be highlighted in yellow, this means that instead of playing the whole file only this bit will be played, this is very useful when testing your labels are right.

When recording a silence-phone diphone it is worth considering moving the silence marker further into the phone. If this is not done when the mid point is taken the volume will be at a lower level. This is because when we start or end words we gradually change volume, so you need to compensate for this when making your labels.

7.4 Auto Labelling

FestVox provides an auto labeller. Auto labelling is a fairly processor intensive exercise and can take a fairly long time if done on a slow machine. Below is an example of a file labelled by the auto labeller.

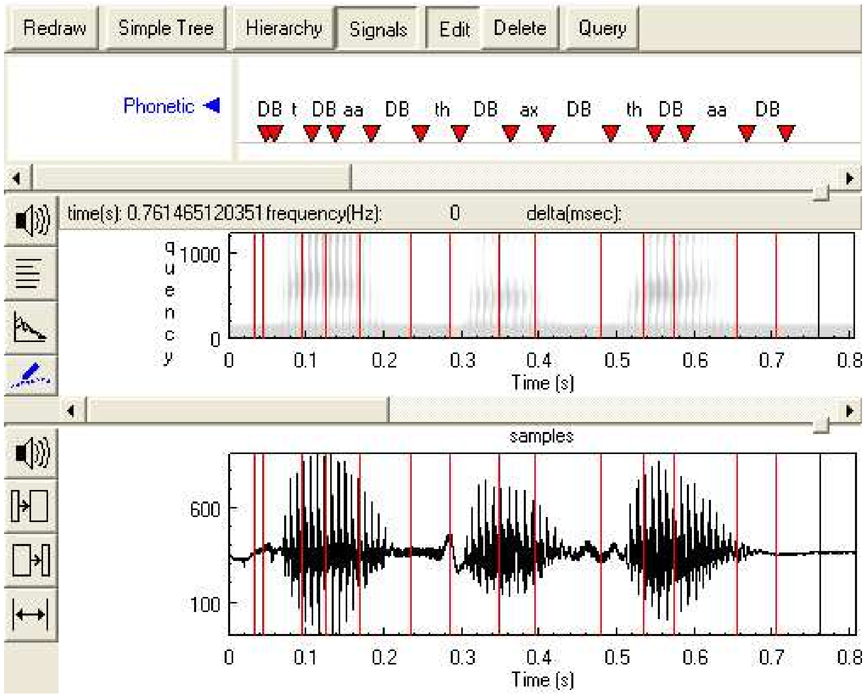


Figure 3.

You may notice that there are a lot of extra labels, these are used by the auto labeller to keep the phones apart. First thing I advise is to delete all of these, they will only get in the way, and while you can work around them, it is much easier without them.

When testing your voice you may notice that some cause Festival to give an error. By checking the name of the diphone back in the diphone list you can find which label file is causing the problem. The problem is most often caused by a previous label having the same or later time as the next phone, this means Festival cannot extract a mid point and will hense through an error. These labels will obviously have to be fixed before any further testing can go on.

On the CD accompanying this thesis is a file called spiltest.pl. This is a Perl program that will first take a diphone list and split it into its phones and then test these by speaking each one first in a working voice (Kal) and then in your voice. This tool, while very boring to listen to, can help you quickly identify which dihones are not being pronounced properly. The other alternative is to listen to various text examples typed into Festival.

Below is an example of a very badly labelled file.

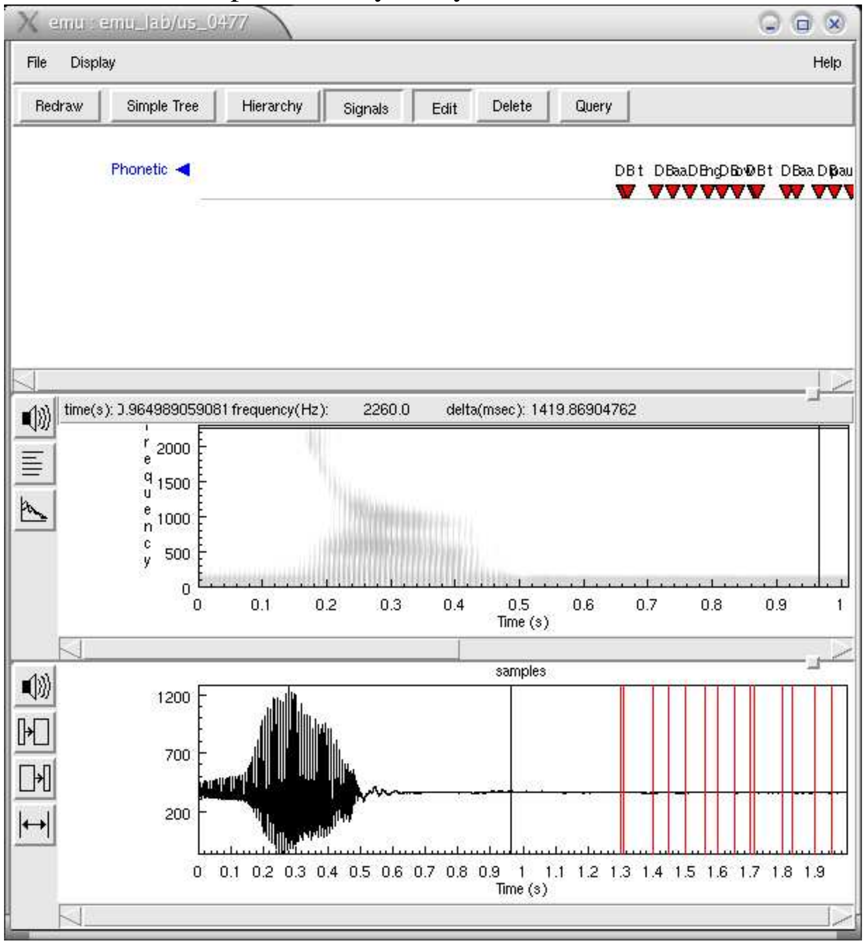


Figure 4.

This is due to the auto labeller not being able to identify all the sounds in the file. This is due to bad recording. You may notice that the sound is all bunched up at the beginning of the file and you cannot make out the different parts of the waveform. Such files are best re-recorded but you can fix them manually if necessary. Remember if you re-record files to backup all you label files first, else when you auto label the new files you will loose all the work that you have done up till now.

Chapter 8

Results

8.1 Setup

The machine specs are not important as there is no performance consideration. The sound card was an on-board Realtek AC97 Audio. The microphone was a standard desk mounted Creative Labs microphone. Further testing was done with a head mounted microphone. All input volume levels were initially set to 100%.

All tests were done under Mandrake 9.0 using the FestVox tools. US English was the standard language used for the voice. I did all my own recording in a normal room, most recordings were done at night to cut out background noise.

8.2 Auto Labeller

The first set of results test the effectiveness of the auto labeller. It was tested a number of times, varying the recorded input volume and quality (adding background noise etc). The following results are an average of all the tests.

1.6% failure rate.

8 15% error rate.

70% useable labels.

Failure rate is a term I use to describe a label which causes Festival to throw an error. These are caused by two phones having the same time, or the subsequent phone having an earlier time than the one before. These cause an error because Festival cannot extract a mid point. These errors are revealed when the voice is loaded into Festival. By looking by the name of the diphone in the diphone list you can find which label file has the error and correct it.

Error rate is when the label contains no actual errors, but the wrong part of the wave file is labelled. This means that when testing each phone these made no or the wrong sound. These most often require re-recording as often the wave file being labelled is of bad quality and this caused the problem.

Useable labels make the correct sound under phonetic testing and if used in speech the words are audible and understandable. There is approximately another 15% of unaccounted for labels, these can be best described as sort of working. Either they make approximately the right sound, or their volume is bad etc. With these still in the voice the over all quality, while useable, is not going to be of a high standard. If you are planning extensive use of the voice, it is worth re-labelling these files.

While the results are not bad, they still mean that on average 400+ labels will have to be re-recorded and labelled, probably by hand.

8.3 Voice quality

The following results were obtained by informal listening tests. There were 10 subjects who were asked to listen to 10 sentences and then write what they thought was said. The subjects were asked how well they understood the sentence being read and how similar they thought the voice was to the target speaker.

On average, subjects reckoned they understood 70% of what was said by the voice. Though results were better with the later sentences, this is probably due to the speaker getting used to the voice. Typically the first sentence was the hardest to understand, especially if it was long.

Most people said that the voice "sort" of sounded like its speaker. They commented it lost a lot of the natural tones, and sounded more robotic. This is due to two things. First the computer talks without emotion, and this is more noticeable with a poor quality voice. Second the voice took on a tinny effect. This was due to the recordings being very soft, which could not be improved on the test system. When the volume was increased the result was a loss of quality and a more robotic sound to the recordings. Under better recording conditions the diphones sound much more natural.

While most people did not identify the voice immediately with its speaker, it is worth mentioning that compared to other Festival voices all subjects said the new voice sounded the most like the speaker. This shows that while the voice may not be exactly like its speaker it does sound more like him than other voices, which means the voice at least sounds unique and hence is of some use.

Overall I was pleased with the quality of the voice, while I feel it is not good enough to be used in a text to speech application, it is at least a different sounding voice.

Chapter 9

Conclusion

9.1 Making the voice

The process of voice creation is something that requires reading and practice. While the process in itself is not very complex, it is also not very intuitive. While it is possible to make a voice from scratch within a couple of days, the quality is unlikely to be that good. In particular the process of labelling requires quite a bit of practice to get right.

Most problems are due to a lack of understanding. It is unlikely the reader will have much experience in the field and it can be confusing at first. This is not helped by a lack of documentation about voice creation. This thesis has laid out the steps involved in making the voice and will help a new voice creator through the process so that they can make a voice of decent quality without too much difficulty.

9.2 ru_us_matt_diphone

The voice was made and edited extensively over the year. It was used to test the auto labeller under varying conditions. As such the voice is not of as high a standard as it could be, but this editing helped greatly in finding what worked and what did not. While the voice may not be of much use to the text to speech community it showed that with a little knowledge and practice it is possible to produce and different sounding voice of decent quality.

9.3 Future Work in the area.

There is much work still being done in the field of voice creation. Much research is going into auto labelling and pitch mark extraction. There is also the possibility of extending the FestVox toolkit to make it easier to use or more automated. Also Festival 2.0 is to be released next year which will presumably have a number of differences in it voice creation process. There are two areas that could be considered a project in their own right

9.3.1 Languages

There is the option of adding extra languages to Festival, this will require a fair amount of linguistic knowledge. The process in theory is fairly simple. It consists of telling Festival how the language extracts and interprets utterances. Also diphone lists will have to be made and then a voice can be made for the new language. It is not possible to use another languages voice, although parts will be similar. I have been in contact with a Masters student who is attempting to implement Afrikaans into Festival.

9.3.2 Voice Adaptation.

During the course of this project I also looked at voice adaptation. But found the scope to be far greater than initially expected.

Voice adaptation, or transformation, is the process of adapting a voice to sound different after the waveform has been generated. The process of voice adaptation has been used before, but never in conjunction with a text to speech system. Voice adaptation works on the principle of separating speech into its source and resonator. This means that first the words are spoken (vocal cords), then they are adapted to give their unique sound (vocal tract and mouth). The diagram below illustrates how voice adaptation works.

The source voice and the target voice are analysed and their differences noted. From this a filter can be generated. This filter can then be used to convert any future speech from the source into the target. Think of a kidnapper speaking through a cloth to mask his voice. If this principle is to be used with text to speech then the source voice will be Festivals standard voice and Festival will speak in this voice, it will be adapted by a filter to sound like the intended voice.

Voice adaptation has a number of benefits over voice creation. For one there is not need to record every diphone. In Text-to-Speech Voice Adaptation from Sparse Training Data [Kain et al; 1998] the authors claim that with as little as a minutes training data Festival can be adapted to sound like its target. There is also the advantage of the filter being much smaller and taking up only a few hundred kilobytes, as opposed to over one hundred megabytes for a created voice. Finally, as the speaker will be training with normal words and only for a few minutes, it means that you do not need a trained speaker to produce a voice of good quality.

Oregon Graduate Institute has produced a plug-in for Festival that supposedly allows for voice adaptation. However, while I eventually was able to get the plug-in compiled and talking in an adapted voice, I was not able to successfully create an adapted voice of my own. The area is obviously still an on going research project and might be worth further investigation.

Chapter 10

Appendix

US phoneset

Below is the list of phones used by Festival in the US English diphone list. This list is taken from Chapter 29 of Builing Synthetic Voices [Black et al; 2003]. There are also words in which the phones appear to aid in pronunciation. This list was based on those phones that appear in the Boston University FM radio corpus with minor modifications.

aa	fAther, wAshington
ae	fAt, bAd
ah	bUt, hUsh
ao	lAWn, dOOr, mAll
aw	hOW, sOUth, brOWser
ax	About, cAnoe
ay	hIde, bIble
eh	gEt, fEAther
el	tabLE, usabLE
em	systEM, communisM
en	beatEN

er	fERtile, sEARch, makER
ey	gAte, Ate
ih	bIt, shIp
iy	bEAt, shEEp
ow	lOne, nOse
oy	tOY, OYster
uh	fUll, wOOd
uw	fOOl, fOOd
b	Book, aBrupt
ch	CHart, larCH
d	Done, baD
dh	THat, faTHer
f	Fat, lauGH
g	Good, biGGer
hh	Hello, loopHole
jh	diGit, Jack
k	Camera, jaCK, Kill
l	Late, fuLL
m	Man, gaMe
n	maN, New
ng	baNG, sittiNG
p	Pat, camPer
r	Reason, caR,

s	Sit, maSS
sh	SHip, claSH
t	Tap, baT
th	THeatre, baTH
v	Various, haVe
w	Water, cobWeb
y	Yellow, Yacht
z	Zero, quiZ, boyS
zh	viSion, caSual
pau	short silence

References

[Whistler] Microsoft Corporation, Whistler (Windows Highly Intelligent STochastic taLkER) <http://research.microsoft.com/srf/ssproject.aspx> , 1998.

[SoftVoice] SoftVoice, Inc, SoftVoice < http://text2speech.com/>

[Lucent] Bell Laboratories, Lucent TTS engine <http://www.lucent.com/press/0199/990128.cob.html>

[Natural Voices] AT&T Labs, Natural Voices < http://www.research.att.com/projects/tts/>

[Discover Magazine; 2003] Discover Magazine, The Mathematics of . . . Artificial Speech, <http://www.discover.com/issues/jan-03/departments/featmath/>, 2003

[CSTR] Edinburgh Univeristy, Center for Speech Technology Research, <http://www.cstr.ed.ac.uk/projects/festival/>

[CSLU] OGI School of Science and Engineering, Center for Spoken Language Understanding <http://cslu.cse.ogi.edu/>

[SLR] University of Delaware,
Speech Research Lab
<http://www.asel.udel.edu/speech/>

[Building Synthetic Voices; 2003] Alan W Black,
Kevin A. Lenzo, Building Synthetic Voices For FestVox
2.0 Edition, 1999-2003

[Text-to-Speech Synthesizer by Voice Adaptation; 1998]
Alexander Kain, Mike Macon, Text-to-Speech Voice
Adaptation from Sparse Training Data, CSLU, 1998.