**CS4706 Spring 2011 - Project 1 - "Building a TTS System"**

Submission procedure explained below.

In this assignment, you are going to build your own limited-domain text-to-speech system. This is a complex task, so it is divided into three simpler subtasks. You will do the first subtask on your own. You will do the second and third with your project team.

**Part A: Setup and first small TTS system**

|  |
| --- |
| **IMPORTANT:** This part of the assignment cannot be done remotely. You have to do it on one of the Linux computers in the [Speech Lab](http://www.cs.columbia.edu/speech/lab.cgi), for which you need to [sign-up](http://www.cs.columbia.edu/speech/sign-up/index.php) first.**Note:** It might also be possible to do this locally on one of the Linux computers in the CLIC Lab. However, we have not tested the software there, and we cannot guarantee that it will work. So if you do this you will be on your own."  |

**A.1. Setup**

If your default shell is not bash, change it to bash. To find out what shell you currently have, use the command echo $SHELL. You will need to do this for each machine you use in the lab.

To change to bash, run the following command:

chsh –s /bin/bash

Next, add these lines to your ~/.bashrc file:

export PATH=/proj/speech/tools/festival/festival/bin:$PATH
export PATH=/proj/speech/tools/festival/speech\_tools/bin:$PATH
export FESTVOXDIR=/proj/speech/tools/festival/festvox
export ESTDIR=/proj/speech/tools/festival/speech\_tools

Remember that you have to log out and back in for the changes to the ~/.bashrc file to take effect. You can check if the changes are in effect by running echo $FESTVOXDIR. The value specified above should be displayed. If not, check .bashrc to make sure you have all quote marks, etc., just as above.

**A.2. Small TTS system: a talking clock**

*NB:  Everyone must do Part A.  Teams will turn in all 2-3 talking clocks with the team submission.*

Log on **locally** to a Linux computer in the Speech Lab. Open a Terminal window (*Applications » Accessories » Terminal*) and run the following commands, replacing *USERNAME* with the user name of your CS account (e.g. fb2175).

In <http://festvox.org/bsv/x1003.html> you can find a detailed explanation of each step.

|  |  |  |
| --- | --- | --- |
| **Step** | **Commands** | **Comments** |
| 1 | mkdir /proj/speech/users/cs4706/*USERNAME*cd /proj/speech/users/cs4706/*USERNAME*mkdir timecd time | Create a directory and cd into it. |
| 2 | $FESTVOXDIR/src/ldom/setup\_ldom SLP time xyz | Setup dir |
|   | At this point, take a look at these two files now in your “time” directory:• etc/time.data, which contains a set of utterances that should cover all the possible variations in the domain;• festvox/SLP\_time\_xyz.scm, which defines several functions (in Scheme) to convert a time like "07:57" into an utterance like "The time is now, a little after five to eight, in the morning".In order to build a new limited domain it is necessary to rewrite these files. For this part of the homework you do not need to edit these files, but you will in Parts B and C. |
| The following assume that you are in USERNAME/time |
| 3 | festival -b festvox/build\_ldom.scm '(build\_prompts "etc/time.data")' | Generate prompts |
| 4 | bin/prompt\_them etc/time.data | Record prompts.You need a microphone for this step. You will be asked to read out 24 prompts. **Read the**[**recording tips**](http://www.cs.columbia.edu/~julia/courses/CS4706/hw/hw4/#rectips)**before starting.** If you have trouble getting clean recordings even after following the tips, you can record your prompts as .wav files on other machines and import them; instructions are [here](http://www.cs.columbia.edu/~julia/courses/LSA11/import.htm) (http://www.cs.columbia.edu/~julia/courses/LSA11/import.htm). |
| 5 | bin/make\_labs prompt-wav/\*.wav | Autolabel prompts |
| 6 | festival -b festvox/build\_ldom.scm '(build\_utts "etc/time.data")' | Build utterances |
| 7 | cp etc/time.data etc/txt.done.data |   |
| 8 | bin/make\_pm\_wave wav/\*.wavbin/make\_pm\_fix pm/\*.pm | Extract pitchmarks & fix them |
| 9 | bin/simple\_powernormalize wav/\*.wav | Power normalization |
| 10 | bin/make\_mcep wav/\*.wav | MCEP vectors |
| 11 | festival -b festvox/build\_ldom.scm '(build\_clunits "etc/time.data")' | Build LDOM Synthesizer |
| 12 | festival festvox/SLP\_time\_xyz\_ldom.scm '(voice\_SLP\_time\_xyz\_ldom)' | Run your synthesizer |
| 13 | (saytime)(saythistime "07:57")(saythistime "14:22") | Once in Festival, use these commands to make your synthesizer say the time.Use CTRL+D to exit Festival. |

**Documentation**

* » [Festival & FestVox home page](http://www.festvox.org/)
* » [Building Synthetic Voices](http://www.festvox.org/bsv/)

**Requirements:**

1. To grade your work, we need to run your TTS system. Therefore, please double check that all your files under /proj/speech/users/cs4706/*USERNAME* have *read* permissions for everybody, and also *exec* permissions in the case of directories (that is the default setting, so if you have not *chmod*ed anything, you do not need to change anything).
2. You should generate three wav files using your TTS with the following procedure:

cd /proj/speech/users/cs4706/*USERNAME*/time
festival

(load "festvox/SLP\_time\_xyz\_ldom.scm")
(voice\_SLP\_time\_xyz\_ldom)

(Parameter.set 'Audio\_Method 'Audio\_Command)
(Parameter.set 'Audio\_Required\_Rate 16000)
(Parameter.set 'Audio\_Required\_Format 'wav)

(Parameter.set 'Audio\_Command "cp $FILE time1.wav")
(saytime)

(Parameter.set 'Audio\_Command "cp $FILE time2.wav")
(saythistime "07:57")

(Parameter.set 'Audio\_Command "cp $FILE time3.wav")
(saythistime "14:22")

**Part B: Preparing your limited-domain TTS**

|  |
| --- |
| **IMPORTANT:** All steps in Part B can be done remotely. |

**B.1. Choosing a limited domain:**

In Part A, you built a simple talking-clock TTS system. Now, you will build a TTS system for the Spoken Dialogue System application you have chosen. Also, instead of recording it using a neutral voice, you may want to choose a particular style or personality that you think is most appropriate for your domain and application.

**B.2. Designing the input and output of your TTS system**

Define as formally as possible what the input and output of your TTS system is going to look like.

* For example, in the talking clock the **input** is a string of the form HH:MM, and the **output** is a sentence of the form:

The time is now, EXACTNESS MINUTE INFO(, in the DAYPART).

where:

EXACTNESS = {exactly, just after, a little after, almost}
MINUTE = {-, five past, ten past, quarter past, twenty past, twenty-five past, half past, twenty-five to, twenty to, quarter to, ten to, five to}
INFO = {one, two, three, four, five, six, seven, eight, nine, ten, eleven, twelve, midnight}
DAYPART = {morning, afternoon, evening}

Every time the talking clock receives an input like 07:57, it will translate it into a sentence like The time is now, a little after five to eight, in the morning, and later synthesize it.

This TTS has four degrees of freedom (EXACTNESS, MINUTE, INFO, DAYPART). The number of possible sentences is approximately 4 x 12 x 12 x 2 = 1152.

Your limited domain must have **at least five degrees of freedom**, and you have to provide an estimate of the number of possible sentences that could be generated.

**B.3. Setting things up**

Log on to your CS account (NB: /proj/speech/ is only available from the following machines:  chat, felix, fluffy, veu, voce, voix, and lincoln) and run the following commands, where *USERNAME* is the user name of your CS account (e.g. fb2175), and *TOPIC* is a string such as 'number', 'weather', 'street', etc.:

|  |  |  |
| --- | --- | --- |
| **Step** | **Commands** | **Comments** |
| 1 | mkdir /proj/speech/users/cs4706/*USERNAME*/partc  | Create a directory for part c. |
| 2 | cd /proj/speech/users/cs4706/*USERNAME*/partc$FESTVOXDIR/src/ldom/setup\_ldom SLP *TOPIC* xyz | Setup the directory for part c. |

**B.4. Designing the prompts**

Next, you need to design the prompts for your TTS system. As you saw in Part A, the talking clock uses the prompts in time/etc/time.data:

( time0001 "The time is now, exactly five past one, in the morning." )
( time0002 "The time is now, just after ten past two, in the morning." )
( time0003 "The time is now, a little after quarter past three, in the morning." )
...

Now, you have to create a similar file for your domain, and save it as partc/etc/*TOPIC*.data.  NOTE:  The spaces after '(' and before ')'  in each line are ***critical***.

For an explanation on how to design the prompts, go to <http://www.festvox.org/bsv/c941.html#AEN952>.

**Documentation**

* » [Festival & FestVox home page](http://www.festvox.org/)
* » [Building Synthetic Voices](http://www.festvox.org/bsv/)
* » [Designing the Prompts](http://www.festvox.org/bsv/c941.html#AEN952)

**Part C: Completing your limited-domain TTS**

|  |
| --- |
| **IMPORTANT:** Section **C.2** of this part of the work cannot be done remotely. You have to do it on one of the Linux computers in the [Speech Lab](http://www.cs.columbia.edu/speech/lab.cgi), for which you need to [sign-up](http://www.cs.columbia.edu/speech//sign-up/index.php) first. |

**C.1. Introduction**

In Part B, you started building your limited-domain TTS system. You defined its input and output, designed the set of prompts you will record.

Now, in order to complete your TTS system you need to:
a) record the set of prompts (section C.2);
b) write a script that transforms an input string into an English sentence, and sends it to Festival to synthesize it (section C.3).

**C.2. Recording and processing the prompts**

Log on **locally** to a Linux computer in the Speech Lab. Open a Terminal window (*Applications » Accessories » Terminal*) and run the following commands, replacing *USERNAME* with the user name of your CS account (e.g. fb2175), and ***TOPIC*** with a string such as 'number', 'weather', 'street', etc.

Before starting, read the [tips for part C](http://www.cs.columbia.edu/~julia/courses/CS4706/hw/hw4/#rectipsC), which you may find useful.

|  |  |  |
| --- | --- | --- |
| **Step** | **Commands** | **Comments** |
| 1 | cd /proj/speech/users/cs4706/*USERNAME*/partc |   |
| 2 | The file etc/***TOPIC***.data should contain the prompts you designed. Make sure that its syntax is correct. |
| 3 | festival -b festvox/build\_ldom.scm '(build\_prompts "etc/***TOPIC***.data")' | Generate prompts |
| 4 | bin/prompt\_them etc/***TOPIC***.data | Record prompts.You need a microphone for this step. You will be asked to read out, one by one, the prompts you designed. **Before starting, review the**[**recording tips**](http://www.cs.columbia.edu/~julia/courses/CS4706/hw/hw4/#rectips)**.** |
| 5 | bin/make\_labs prompt-wav/\*.wav | Autolabel prompts |
| 6 | festival -b festvox/build\_ldom.scm '(build\_utts "etc/***TOPIC***.data")' | Build utterances |
| 7 | cp etc/***TOPIC***.data etc/txt.done.data |   |
| 8 | bin/make\_pm\_wave wav/\*.wavbin/make\_pm\_fix pm/\*.pm | Extract pitchmarks & fix them |
| 9 | bin/simple\_powernormalize wav/\*.wav | Power normalization |
| 10 | bin/make\_mcep wav/\*.wav | MCEP vectors |
| 11 | festival -b festvox/build\_ldom.scm '(build\_clunits "etc/***TOPIC***.data")' | Build LDOM Synthesizer |
| 12 | festival festvox/SLP\_***TOPIC***\_xyz\_ldom.scm '(voice\_SLP\_***TOPIC***\_xyz\_ldom)' | Run Festival (use CTRL+D to exit). |
| 13 | Now, you can make your synthesizer say sentences in your domain *by hand*. For example, in the *time* domain you could do that by running: (SayText "The time is now, a little after twenty past two, in the afternoon.") Warning: If you use words that do not belong to your domain, the synthesizer will default to a Festival voice.Note: If your prompts seem to be badly segmented when you synthesize a new sentence, you can edit the automatically assigned labels by converting .lab files to Praat .TextGrid files, hand correcting the alignments, and then converting the .TextGrid files back to .lab. See /proj/speech/tools/speechlab/lab\_to\_textgrid.py and textgrid\_to\_lab.py .  |

**C.3. Writing a script that transforms the input and synthesizes it**

Now, you need to write a script that receives an input string as you defined in Part B, and transforms it into an English sentence that your TTS system can synthesize. For example, in the *time* domain the script would transform a string like "14:22" into a sentence like "The time is now, a little after twenty past two, in the afternoon." A simple (and perhaps long) case or switch statement should be enough to achieve this.

You must test your script properly before submission (in one of the speech-lab machines). If you submit a nonworking script, you will lose a lot of points, and we will not debug your code.

Use [this Perl script](http://www.cs.columbia.edu/~julia/courses/CS4706/hw/hw4/tts.pl.txt) as a basis. Update the variables $USERNAME and $TOPIC, complete the code where marked, and define the function generate\_sentence, which does the input-output transformation. **Please comment your code thoroughly.**

The rest of the code in this script creates a temporary Festival script and runs it. That temporary Festival script loads your limited domain and creates a wav file with the resulting synthesis (you did exactly the same thing by hand in Part A for the *time* domain). You should not need to modify any of this.

**Note:** It is also possible to do the transformation part of the script (input string to English sentence) using the Scheme programming language, and later import your script into Festival. If you want to do it that way, please check with the TA first.

**Documentation**

* » [Festival & FestVox home page](http://www.festvox.org/)
* » [Building Synthetic Voices](http://www.festvox.org/bsv/)

**Submission**

Create a folder YourUni-PROJ1 (e.g., fb2175-PROJ1). Create three subdirectories, parta, partb, and partc -- YOU MUST FOLLOW THESE CONVENTIONS EXACTLY.

1. For Part A, save each team member's three wave files: time1.wav, time2.wav, and time3.wav (see part A) in a subfolder named <YourUni-parta>.  All 2 or 3 of these subfolders should be put into a subfolder named **parta.**So you should have a folder names **parta** which contains subfolders for each team members own clock.

2. For Part B, save the following files in **partb** subfolder:

* A text document specifying:
	1. a description of your limited domain as currently defined (5 sentences at most)
	2. the definition of the input and output of your TTS system, specifying the degrees of freedom and the number of possible sentences that could be generated.
* The file *TOPIC*.data which contains the prompts you have designed.

3. For Part C, save the following files in **partc** subfolder:

* **A plain text document** (not Word, PDF, etc.) specifying:
	1. path and name of the Perl script you did in section C.3.
	2. an example of how to run your perl script (must work on one of our speech lab machines).
	3. a description of any changes you made to your design in Part B, and an explanation of why you think they were necessary.
	4. a description of any special features of your system, e.g. of your grammar, domain coverage, which you think are particularly interesting or sophisticated.
	5. a paragraph or two on what it would require to turn your TTS system into a real application in your domain and the difficulties you might encounter in doing so.
* **Do not submit any wav files.**
* **A shell script sample.sh**that generates three wav files (from three different inputs you choose). That is, given three inputs *a*, *b* and *c*, this script should generate a wav file for each input. Three wav files in total.
Choose the inputs that you consider show your system to its best advantage.
The script should look like this:

#!/bin/bash

# path and name of the tts script (\*\*update\*\*)
SCRIPTDIR=/proj/speech/users/cs4706/*USERNAME*
SCRIPTNAME=tts.pl

# the resulting wav files are saved in the present directory
DESTDIR=`pwd`

# generate a wav file for input #a
$SCRIPTDIR/$SCRIPTNAME "03:39" $DESTDIR/utt1a.wav

# generate a wav file for input #b
$SCRIPTDIR/$SCRIPTNAME "14:47" $DESTDIR/utt1b.wav

# generate a wav file for input #c
$SCRIPTDIR/$SCRIPTNAME "23:55" $DESTDIR/utt1c.wav

|  |
| --- |
| Now, compress (in zip format only) the main folder to YourUni1-YourUni2-YourUni3-PROJ1.zip (e.g., fb2175-myy435-uma2129-PROJ1.zip). I.e., the file name should include the unis of all team members in alphabetical order.  Submit this zip file in Courseworks. |

**Tips for recording (for parts A and C only)**

* Do not start without testing the microphone! To test it, try:
rec -s test.wav or na\_record test.wav or use Praat for both
play test.wav or na\_play test.wav
Change the settings in the Volume Control (*Double click on the speaker in the upper-right corner*) until your recorded speech is clear and loud enough, with as little background noise as possible.
* Try to make the recordings in a silent environment.
* Make sure that your microphone is not directly in front of your mouth, to avoid noise produced by your breath.
* The script bin/prompt\_them in step 4 only lets us record all prompts in one sitting. If you make mistakes and want to record again a few specific prompts, you do not need to run that script again. Instead, follow these instructions (from the shell, not in Festival):

	1. Let *PROMPT* be the id of the prompt you want to record again (e.g. time0001).
	2. Run the command $ESTDIR/bin/ch\_wave -info wav/*PROMPT*.wav
	3. Let *DURATION* be the duration in seconds of the wav file, which is given in the first line of the output.
	4. Run the command $ESTDIR/bin/na\_record -f 16000 -time *DURATION* -o wav/*PROMPT*.wav -otype riff
	This will start recording immediately for the specified amount of seconds, and will replace the file wav/*PROMPT*.wav with the new recording.

**Further tips for part C**

* (This is very specific, most probably you will not need it.)
If you want to elicit the same word (or words) with a different intonation depending on its position in the sentence, you will find that Festival does not take word position into account, with one exception: a word immediately before a pause (comma, period) will be pronounced differently. For example, in "Pin number, one one one." Festival differentiates between the first two and the third "one", but not between the first and the second.
Therefore, we have to find a way to change the pronunciation of a word, depending on its position.
Suppose we have this set of prompts:
( pin0001 "Pin number, one twoh threehh." )
( pin0002 "Pin number, two threeh onehh." )
( pin0003 "Pin number, three oneh twohh." )
Numbers in the second position have one extra h, and numbers in the third position have two extra h's.
This lets us elicit each number differently, depending on its position. The h's might mess up the synthesizer that reads out the sentences before recording, but do not seem to affect the rest of the process.
Then, to access the different word instances when synthesizing speech, we have to use the correct version (e.g. one or oneh or onehh). Our prompts to the TTS system will look like:
( SayText "Pin number, two twoh onehh" )
( SayText "Pin number, three threeh twohh" )
etc.