

Spoken Language Processing
Fall 2011
Homework 5
Speech Synthesis

Due Dec 12.

Part 0: Install Festival

NOTE: This section has changed from when the homework was first posted. The files that were pointed to previously were not internally consistent. The current files will compile correctly.

Download all files from here:

<http://www.speech.cs.cmu.edu/15-492/assignments/tts/packed2010/>

Download festival

<http://www.speech.cs.cmu.edu/15-492/assignments/tts/packed2010/festival-2.0.95-beta.tar.gz>

Download speech_tools

http://www.speech.cs.cmu.edu/15-492/assignments/tts/packed2010/speech_tools-2.0.96-beta.tar.gz

Download the lexicon and voice

http://www.speech.cs.cmu.edu/15-492/assignments/tts/packed2010/festvox_kallpc16k.tar.gz

http://www.speech.cs.cmu.edu/15-492/assignments/tts/packed2010/festlex_CMU.tar.gz

http://www.speech.cs.cmu.edu/15-492/assignments/tts/packed2010/festlex_POSLEX.tar.gz

Download festvox

<http://www.speech.cs.cmu.edu/15-492/assignments/tts/packed2010/festvox-2.4-current.tar.gz>

You may try different voices and lexicons. There are some available for download from:

<http://festvox.org/packed/festival/2.1/>

Download all of the files to a directory. We'll refer to it as: <INSTALL_DIR>

Install speech_tools

```
tar xzf *.tar.gz
cd speech_tools
./configure
```

```
make
cd ..
```

Install festival

```
tar xzf festival-2.1-release.tar.gz
tar xzf festvox_kallpc16k.tar.gz
tar xzf festlex_CMU.tar.gz
tar xzf festlex_POSLEX.tar.gz
cd festival
./configure
make
```

Install festvox

```
tar xzf festvox-2.1-release.tar.gz
cd festvox
./configure
make
```

setup environment variables

add the following lines to your ~/.profile file

```
export PATH=<INSTALL_DIR>/festival/bin:$PATH
export PATH=<INSTALL_DIR>/speech_tools/bin:$PATH
export FESTVOXDIR=<INSTALL_DIR>/festvox
export ESTDIR=<INSTALL_DIR>/speech_tools
```

Part 1: Input and output for your TTS system.

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

For example, for a clock application, 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}

For your submission, the sentences that you will generate will be the user turns in the dialog you produced in Homework 4.

The **input** contains the information that the system needs to convey. For example, {action: CONFIRM, field: DATE, value: TUESDAY} might produce "You're looking for a flight on Tuesday, is this correct?"

The **output** is the sentence that is generated.

Part 2: Configuration

- A. Create a directory: (In unix, mac osx or Cygwin)
 - a. mkdir tts
- B. Setup the directory
 - a. cd tts
 - b. \$FESTVOXDIR/src/lDom/setup_lDom SLP *TOPIC* xyz
 - i. Note that "TOPIC" should reflect the topic of your dialog system
 - ii. At this point, take a look at these two files:
 - etc/*topic*.data, which contains a set of utterances that should cover all the possible variations in the domain;
 - festvox/SLP_*topic*_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.

Part 3: Create prompts

Next, you need to design the prompts for your TTS system. A clock uses the prompts in 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 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>.

Part 4: Record Prompts.

* Read Tips for Recording (below. With thanks to Julia Hirschberg and Agustín Gravano)

1. `cd tts`
2. The file `etc/TOPIC.data` should contain the prompts you designed, check the format.
3. `festival -b festvox/build_ldom.scm '(build_prompts "etc/TOPIC.data")'`
4. `bin/prompt_them etc/TOPIC.data`

Note: on OSX, replace the line in `bin/prompt_them`

```
    $ESTDIR/bin/na_play prompt-wav/$f.wav
```

with

```
    play prompt-wav/$f.wav
```

and

```
    $ESTDIR/bin/na_record -f 16000 -time $duration -o wav/$f.wav
```

with

```
    rec wav/$f.wav trim 0 $duration
```

5. `bin/make_labs prompt-wav/*.wav`
6. `festival -b festvox/build_ldom.scm '(build_utts "etc/TOPIC.data")'`
7. `cp etc/TOPIC.data etc/txt.done.data`
8. `bin/make_pm_wave wav/*.wav`
9. `bin/make_pm_fix pm/*.pm`
10. `bin/simple_powernormalize wav/*.wav`
11. `bin/make_mcep wav/*.wav`
12. `festival -b festvox/build_ldom.scm '(build_clunits "etc/TOPIC.data")'`
13. `festival festvox/SLP_TOPIC_xyz_ldom.scm '(voice_SLP_TOPIC_xyz_ldom)'`
14. 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.

Part 5: Create a script that can be called from your Java dialog system to synthesize speech.

Festival Scripts can be written in a number of ways.

Here is how the script should look.

-- **Filename:** /tmp/tmp.scm

```
(load "BASEDIR/festvox/SLP_TOPIC_xyz_ldom.scm")
(voice_SLP_TOPIC_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 WAVEFILE")
(SayText "SENTENCE")
```

When you generate this file, set all the red variable names to appropriate values. **BASEDIR** is the location of your tts system. **TOPIC** is the topic you specified in Part 2. **WAVEFILE** is the name of the file that will hold the recorded material after festival synthesizes it. **SENTENCE** is a string holding the words that are to be read.

The java system call should execute the following

```
"cd BASEDIR; festival --batch FILENAME"
```

This script should be written and called by your java program for each dialog line. Then from java, read in the generated wave file and play it.

Part 6: Submission

Compress the main folder using zip, and email the directory to andrew@cs.qc.cuny.edu.

Part 7: Individual work

Write a document describing what each member of the team contributed to this submission.

Tips for recording

Do not start without testing the microphone! To test it, try:

```
rec -s test.wav
```

```
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 recording

- (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.
```