Project 1 – Limited Domain TTS using Festival

CS4706 Spring 2010

Project 1

- Build a limited domain (LDOM) TTS system
  - Festival TTS
    - http://www.festvox.org/
    - Uses scheme as scripting language
    - Installed (with scripts) on speech lab linux machines
  - Design TTS based on design of your dialog systems. Do in same teams.
- Sign up for times in speech lab ASAP
  - http://www.cs.columbia.edu/~speech/sign-up/index.php
  - Choose main machines first. Use overflow machines if none available
  - CS linux account required
    - http://www.cs.columbia.edu/crf/

Limited Domain TTS project

- Part A (easy)
  - Set up and test a first small TTS system (talking clock) using Festival-provided scripts
  - Each team member should do this part with their own voices
- Part B&C
  - Do this in your teams
  - Design and implement your own LDOM TTS
  - Base this on your dialog system domain
  - Use some of the same procedures as in Part A

  - **Note**: these slides are an overview of the assignment. See full instructions at: http://www.cs.columbia.edu/~julia/courses/CS4706/hw/PROJ1.htm

Outline – Creating your TTS

- Create utterance list (text) to specify the words and phrases in your domain as said in context (to capture coarticulation effects).
- Record these utterances.
- For all recorded utterances, Festival aligns utterance audio to utterance text.
- Festival generates voice synthesizer for your domain (to cover words in your utterance list).
- Write script which translates an input code to the new utterance text to be synthesized. Synthesized utterance must use words found in utterance list.
- Run script to synthesize the given new utterance.

Part A – environment variables

- 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
- Note: after first setting these variables, you have to log out and back in for the changes to the ~/.bashrc file to take effect.
  - Or do *source ~/.bashrc* in a running shell
  - Check by running *echo $FESTVOXDIR*. The value specified above should be displayed.
- Festival documentation on the “telling the time” example:
  - http://festvox.org/bsv/x1003.html

Part A – build directories

- Each student should do Part A
- Create directory and cd to it
    - mkdir /proj/speech/users/cs4706/*USERNAME*
    - *cd /proj/speech/users/cs4706/USERNAME*
    - *mkdir time*
    - *cd time*
- Set-up directory
    - $FESTVOXDIR/src/ldom/setup_ldom SLP time xyz
- These two files (among others) are created:
    - **etc/time.data**, contains a set of utterances covering the possible word and phrase choices in the domain
    - **festvox/SLP_time_xyz.scm**, 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*.
- For part A of the homework you do not need to edit these files, but you will in Parts B and C.

Part A - build prompts and utterances

- Generate prompts from utterance list (outputs to prompt-wav/ subdir)
    - festival -b festvox/build_ldom.scm '(build_prompts "etc/time.data")’
- Record from prompts. The system will prompt you to say one utterance at a time. (see recording tips)
    - bin/prompt_them etc/time.data
    - Note: Recorded files go to wav/ subdirectory
- Create label files aligning utterance text to recorded utterances (outputs to lab/ subdir)
    - bin/make_labs prompt-wav/*.wav
- Build utterance structure
    - festival -b festvox/build_ldom.scm '(build_utts "etc/time.data")’
- Copy utterance list text file (used in next processing steps)
    - cp etc/time.data etc/txt.done.data

Part A – build and run speech synthesizer

- Extract pitchmarks and fix them (outputs to pm/ subdir)
  - bin/make_pm_wave wav/*.wav
  - bin/make_pm_fix pm/*.pm
- Power normalization (outputs to wavn/ subdir)
  - bin/simple_powernormalize wav/*.wav
- MCEP vectors (outputs to mcep/ subdir)
  - bin/make_mcep wav/*.wav
- Build synthesizer
  - festival -b festvox/build_ldom.scm '(build_clunits "etc/time.data")'
  - Note: it can only produce words that you have specifically given it in your utterance list.
- Run it
  - festival festvox/SLP_time_xyz_ldom.scm '(voice_SLP_time_xyz_ldom)'
  - (saytime)
  - (saythistime "07:57")
  - (saythistime "14:22")

Part A – final output

- Generate three wav files using your TTS as follows:
  - 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: Designing a limited domain TTS system

- Part B and C should be done in your groups
  - Base your TTS on a possible set of output utterances from your dialog system design.
  - In your actual dialog system, you will probably need to expand on this set.
- Include at least five degrees of freedom

Example: Time

- 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*}
- For example
  - *07:57 => The time is now, a little after five to eight, in the morning*
- Four degrees of freedom (EXACTNESS, MINUTE, INFO, DAYPART).
  - Number of possible sentences is approximately 4 x 12 x 12 x 2 = 1152.

Part B – preparing your limited domain TTS

- Create and set up directory
  - For teams, USERNAME (below) should be your unis concatenated
  - mkdir /proj/speech/users/cs4706/USERNAME/partc
  - cd /proj/speech/users/cs4706/USERNAME/partc
  - $FESTVOXDIR/src/ldom/setup_ldom SLP TOPIC xyz
- Design prompts
  - 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, instead, you will create a similar file for your domain, and save it as partc/etc/TOPIC.data

Tips for designing prompts

- In general you should
  - design your prompts to have at least 2 (and probably 5) examples of each word in you vocabulary.
  - Select utterances that maximize bi-gram coverage. That is try to ensure as many different word-word pairings over your corpus.
  - To have different voices usable within the same synthesizer, you can postfix your prompt words with a silent “h” and then modify your input accordingly
    - (prices0006 "one muffin will cost nineteen cents.")
    - (prices0027 "oneh muffinh willh costh nineteenh penceh.")
- More tips on designing prompts
  - http://www.festvox.org/bsv/c941.html#AEN952.

Part C – completing your limited domain TTS

- Record utterances and build synth as in Part A.
- **Write a script** to transform input code string into an English sentence for your domain. The script then invokes Festival to say the English sentence.
  - Eg "14:22" => *The time is now, a little after twenty past two, in the afternoon.*
- We provide a **perl script** to use as a **template**.
  - http://www.cs.columbia.edu/~julia/courses/CS4706/hw/hw4/tts.pl.txt
  - Update the variables **$USERNAME** and **$TOPIC**, complete the code where marked, and define the function **generate_sentence**, which does the input-output transformation.
  - Comment your code thoroughly.
  - The rest of the code in this script creates a temporary Festival script and runs it. That temporary script loads your limited domain and creates a wav file with the resulting synthesis. 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 Festival's **Scheme scripting language**.
  - If you want to do it that way, please check with the TA first.

Tips for Recording prompts

- Do not start without testing the microphone! To test it, try:
  - rec -s test.wav play test.wav
  - Or test recording with praat
- Change the settings in the Linux Volume Control (right click on the speaker in desktop panel) until your recorded speech is clear and loud enough, with as little background noise as possible. **Watch out for clipping!**
- Try to make the recordings in a silent environment.
  - Cooperate and take turns In the lab
- Make sure that your microphone is not directly in front of your mouth, to avoid noise produced by your breath.
- The script records all prompts in one sitting. If you make mistakes and want to re-record a few specific prompts, you don't need to run the script again**.** See how in hw assignment page.
- For audio setup problems (eg can't record anything), ask the speech lab monitor for help.

Submission

You must follow these conventions exactly.

- For each team, create a directory uni1-uni2-uni3-project1 (using your Unis) with three subdirectories: **parta**, **partb**, and **partc .**
- **For Part A**, save the following files in parta subdirectory:
  - For each user on your team, save the three wave files using their Unis:
    - Uni-time1.wav, Uni-time2.wav, and Uni-time3.wav
  - Make sure all your files under /proj/speech/users/cs4706/*USERNAME* have read permissions for everybody, and also exec permissions in the case of directories
- **For Part B**, save the following files in partb subdirectory:
  - A text document specifying:
    - a description of your limited domain (5 sentences at most)
    - 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.

Submission continued

- **For Part C**, save the following files in partc subfolder:
  - A document (plain text or PDF) specifying:
    - Path and name of your Perl script.
    - An example of how to run your perl script (must work on speech lab machines).
    - A description of any changes you made to your design in Part B, and an explanation of why you think they were necessary.
    - A short description of why you chose the voice you did among people on your team.
    - A description of any special features of your system, e.g. of your grammar, domain coverage, which you think are particularly interesting or sophisticated.
    - 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. Specifically discuss how it will need to expand in the context of your dialog system.
  - A shell script sample.sh that generates three wav files (from three different inputs you choose). Given three inputs, this script should generate a wav file for each (three in total). Choose inputs that best show off your system.
  - **Do not submit any wav files for this part**
- **Test your scripts** fully before submission (on a speech-lab machine). If your script doesn't work, you will lose lots of points. We won't debug your code.

Script file

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

Summary

- Get started early
  - Need CS Linux account
- Sign up for time in speech lab
- Design and implement your domain & TTS
    - Design prompts based on your domain
    - Record prompts carefully
      - Check audio before recording
    - Experiment with different prompts and grammars to see what works best. You don't have to stick to your initial plan.
- Follow submission instructions
  - Double check (on speech-lab machines) that your scripts all work.