11-752: Speech Synthesis

Objectives

- *Understand basic processing in speech synthesis*
- *Understand relative complexity of implementing solutions to problems*
- *Become familiar with Festival's architecture and know what is can and cannot do*
- *After the course you will*
  - *Be able to make Festival speak what you want*
  - *Be able to influence the way it does it*
  - *Be able to adapt it for your applications*
  - *Be able to explain how the system works*
  - *Be able to build simple voices within the system*

Text to Speech

- *Four major topics in speech synthesis*
- *Architecture*
  - *Objects and processes required*
- *Text processing*
  - *From text to tokens to utterances to words*
- *Linguistic processing*
  - *Lexicons, phrasing, intonation duration*
- *Waveform generation*
  - *Diphone, unit selection, parametric synthesis*

Course Outline

- *March*
  - *History, basic Festival use*
  - *TTS, Utterance structure, processes*
  - *Text Analysis, Lexicons and LTS*
  - *Prosody: phrasing, intonation, duration*
- *April*
  - *Large projects*
  - *Waveform synthesis: diphones, unit selection, SPS*
  - *Limited Domain synthesis*
- *May*
  - *Project time*
  - *Voice conversion*
  - *Evaluation*
  - *Concept to speech*

Course Evaluation

- *(approximately) Weekly homeworks*
  - *Best 4 contribute to grade*
- *Large project*
  - *Set beginning of April*
  - *E.g. build a new voice*
  - *Requires presentation (demo) and write up*
- *No exam*

Important Web Links

- *Course notes*
  - *http://www.cs.cmu.edu/~awb/11752.html*
- *Building Voices in Festival*
  - *http://www.festvox.org*

Physical Models

- Blowing air through tubes…
  - von Kemplen's synthesizer 1791

Homer Dudley's Voder

- Bell Labs 1939
  - Controlled keys and foot pedals

  - Picture courtsey of "Talking Chips" Morgan 1984. Audio from Klatt record 1987.

More Computation – More Data

- *Formant synthesis (60s-80s)*
  - *Waveform construction from components*
- *Diphone synthesis (80s-90s)*
  - *Waveform by concatenation of small number of instances of speech*
- *Unit selection (90s-00s)*
  - *Waveform by concatenation of very large number of instances of speech*
- *Statistical Parametric Synthesis (00s-..)*
  - *Waveform construction from parametric models*

Waveform Generation

- Formant synthesis 
- Random word/phrase concatenation 
- Phone concatenation 
- Diphone concatenation 
- Sub-word unit selection 
- Cluster based unit selection 
- Statistical Parametric Synthesis 

Festival: a generic speech synthesis system

Multi-lingual text-to-speech

Synthesis for language systems

Synthesis development environment

Festival Speech Synthesis System

http://festvox.org/festival
General system for multi-lingual TTS
C/C++ code with Scheme scripting language
General replaceable modules
    lexicons, LTS, duration, intonation, phrasing,
    POS tagging tokenizing, diphone/unit selection
General Tools
    intonation analysis (F0, Tilt), signal processing
    CART building, n-grams, SCFG, WFST, OLS
No fixed theories
New languages without new C++ code
Multiplatform (Unix, Windows, OSX)
Full sources in distribution
Free Software

CMU FestVox Project

http://festvox.org
“I want it to speak like me!”
-Festival is an engine, how do you make voices
- Building Synthetic Voices
  - Tools, scripts, documentation
  - Discussion and examples for building voices
  - Example voice databases
  - Step by Step walkthroughs of processes
-Support for English and other languages
-Support for different waveform techniques:
  - diphone, unit selection, SPS, limit domain
- Other support: lexicon, prosody, text analysers

The CMU Flite project

http://cmuflite.org

"But I want it to run on my phone!"

- FLITE a fast, small, portable run-time synthesizer
- C based (no loaded files)
- Basic FestVox voices compiled into C/data
- Thread safe
- Suitable for embedded devices
  - Ipaq, Linux, WinCE, PalmOS, Symbian
- Scalable:
  - quality/size/speed trade offs
  - frequency based lexicon pruning
- Sizes:
  - 2.4Meg footprint (code+data+runtime RAM)
  - < 0.025 secs "time-to-speak"

Synthesis Tools

- I want my computer to talk
  - Festival Speech Synthesis System
- I want my computer to talk in my voice
  - FestVox Project
- I want it to be fast and efficient
  - Flite

Getting your machine to talk

- *Installing the software*
  - *You need*
    - *Edinburgh Speech Tools*
    - *Festival*
    - *Festvox*
    - *(and Flite)*
  - *http://www.cs.cmu.edu/~awb/11752/progs.html*
- *Works under*
  - *Linux*
  - *Windows (with cygwin)*
  - *OSX*

Using Festival

- *How to get Festival to talk*
- *Scheme (Festival's scripting language)*
- *Basic Festival commands*
- *Exercise*

Getting it to talk

- *Say a file*
  - *festival –tts file.txt*
- *Command line interpreter*
  - *festival> (SayText “Hello World”)*

Scheme – Festival's Scripting Language

- *Why:*
  - *Too many options*
  - *Need flexibility*
  - *Easy to add functionality*
    - *New languages with no new C++ code*
- *Why Scheme*
  - *Very simple language*
  - *Very powerful*
  - *Well established*
  - *No external dependencies on other libraries*
  - *Authors are familiar with it*

Bluffer's Guide to Scheme

- *Scheme is a dialect of Lisp*
- *Expressions are*
  - *Atoms: a bcd "hello world" 3.14 42*
  - *Lists: (a b c) (a b (d e)) () ((a b c)) (3.2 (seven))*
- *Expressions can be evaluated*
  - *(+ 2 3) => 5*
  - *6 => 6*
  - *"hello world" => "hello world"*
  - *'(a b) => (a b)*
  - *(list 'a 'b) => (a b)*

Bluffer's Guide to Scheme

- *Setting values*
  - *(set! a 3.14)*
  - *(set! x '(a b c))*
- *Defining functions*
  - *(define (timestwo n) (* 2 n))*
- *Calling functions*
  - *(timestwo a) => 6.28*

Scheme: Lists

festival> (set! alist '(apples pears bananas))
(apples pears bananas)
festival> (car alist)
apples
festival> (cdr alist)
(pears bananas)
festival> (set! blist (cons 'oranges alist)
(oranges apples pears bananas)
festival> (append alist blist)
(apples pears bananas oranges apples pears bananas)
festival> (length alist)
3
festival> (length (append alist blist))
7

Scheme: speech

- *Make an utterance of type text*

  festival> (set! Utt1 (Utterance Text “hello”))

  #<utt 96754>
- *Synthesize an utterance*

  festival> (utt.synth utt1)

  #<utt 96754>
- *Play the synthesized utterance*

  festival> (utt.play utt1)

  #<utt 96754>
- *Do all together*

  festival> (SayText “This is an example.”)

  #<utt 96854>

Scheme: speech

- *In a file*

```
(define (SpeechPlus a b)
   (SayText
      (format nil "%d plus %d equals %d"
                  a b (+ a b))))
festival> (load "file.scm")
t
festival> (SpeechPlus 3 4)
#<utt 54329>
```

Scheme: speech

```
(define sp_time hour minute)
 (cond
    ((< hour 12)
      (SayText (format nil “Its %d %d in the morning”
         hour minute))
    ((< hour 18)
      (SayText (format nil “Its %d %d in the afternoon”
          (- hour 12) minute)))
    (t
      (SayText (format nil “Its %d %d in the evening”
          (- hour 12) minute)))))
```

Getting help

- *Online manual at http://festvox.org/*
- *Example code in*
  - *festival/examples and festival/lib/*
- *Alt-h on symbol displays help*
- *Alt-s speaks the help*
- *Use TAB key for completion*

Lexicons and Lexical Entries

- *Festival will make errors in pronunciations*
  - *It only has an 86K lexicons (and statistical pronunciation of unknown words)*
- *Lexical entry format*
  - *(WORD POS (SYL0 SYL1 …)*
  - *Syllable is ((PHONE0 PHONE1 …) STRESS)*
- *You can add new pronunciations*

  (lex.add.entry '("barak n (((b ax) 0) ((r aa k) 1))))

Exercises

*This exercise is *not* optional*

1. *Install the festival tools*
2. *Saying Names*
    1. *Make festival say your name*
    2. *Make festival say the names of everyone in class*
    3. *Add a lexical entries if required*
3. *Find ten things festival does not say properly*
4. *How long does it take for Festival to say “Alice in Wonderland”*