Small projects

Some interesting project with synthesis

- New language
- Limited domain synthesizer
- New prosodic style
- Text analysis
- Translator

May be done in groups

New language

- New language:
  - current supported (or not)
- Diphone scheme and collected data
- Prosody by rule or from data
- Lexicon:
  - (don't collect your own)
  - letter to sound rules
- Orthographic processing:
  - maybe from roman characters

Doing all parts is probably too much work
Do one part well and rest with simple rules

Limited domain synthesizer

- Existing dialog system:
  - check its language output distribution
  - is it limited
- Possible domains:
  - Movieline, Banking
  - C-STAR
  - Eliza
  - Bus schedules.

New prosodic style

- Record data in one (or more) style:
  - build prosodic models for an existing voice
  - build prosodic models for an supported language
- Show it sounds ?better? than default

Information presentation

- Get data from resource and say it
- Three stages:
  - get data (from web)
  - massage it into useable form
  - modify synthesizer to say it well
- Possible areas:
  - mapquest.com
  - stock quotes
  - Dictionary lookup (m-w.com)
- Techniques:
  - text analysis
  - multiple voices, other noises
  - speed, style, etc
  - as Sable markup ?

Practicalities

- by this week:
  - paragraph description
  - three basic tasks in problem
  - identify resources (speakers, data, machines etc)
- week of 11th ?:
  - class presentations/demos
  - demos/short write-up

New voice for new language

- Phone set
- Token processing rules (numbers etc)
- Prosodic phrasing method
- Word pronunciation (lexicon and/or letter to sound rules)
- Intonation (accents and F0 contour)
- Durations
- Waveform synthesizer
- plus something hard you forgot about

http://festvox.org

Phoneset

- Can't do anything without this
- Choose an existing one:
  - note dialect of your speaker
- May be defined by your lexicon/diphone db
- may be other factors:
  - lexical stress (accents)
  - allophones (flaps, voicing etc)

Tokenization

- Start from native character sets:
  - its not a TTS engine if you don't
  - (maybe a romanized only form is useful)
  - may ASCII-ized version is required too
- Word boundaries:
  - Chinese, Japanese
  - complex morphology
- Numbers, symbols
- Homographs:
  - no vowels (?)
  - Kanji

Prosodic phrasing

- Part of speech tagging:
  - statistically trained phrasing model
- Hand written rule:
  - but test this
- Just use punctuation

Word pronunciation

- Find an existing lexicon:
  - note its copyright
- Write letter to sound rules:
  - often easy (though may need native speaker)
  - may be something missing (stress)
- Only have small tailored lexicon

Intonation

- Get data and train from it:
  - requires some framework
  - require labelling
- Simple rules do work well on simple sentences
- Use an existing model from other language:
  - often works better than writing rules

Durations

- Fixed or average
- Train from data
- Borrow models from other languages:
  - has some justification
  - often works quite well

Waveform Synthesizers

- Collect your own diphones
- Use existing MBROLA database:
  - Faculte Polytechnique de Mons
  - many diphone synthesizers
- Borrow diphones from another language:
  - often works, esp for minority languages
  - givens you something quickly

Other considerations

- Testing and evaluation:
  - synthesis in other languages always sounds better
  - get *native* speakers to evaluate it
- Ensure you have copyright:
  - if you want to use the voice, make sure
  - you have permission to use *all* parts
- What you think sound good:
  - still sounds awful to others

using festvox.org

- detailed documentation
- mailing list for similar projects
- example databases
- Scripts:
  - diphone_setup cmu nl jan
  - creates directory structure
  - diphone lists
  - basic .scm files

but diphones are boring ...

Building unit selection synthesizers

- Select text with rich phonetic coverage:
  - optimise for diphone coverage
  - or use acoustic modelling techniques
- record *very* carefully
- label *very* carefully
- tune and build clunit synthesizer

Building General Unit Selection Voice

- `SPPPDIR/src/festvox/src/clunits/setup_clunits`
  - `setup_clunits INST LANG VOX`
  - `setup_clunits cmu de hans`
- Instatiates language files:
  - Need to fill in some things by hand
  - `festvox/*_phoneset.scm`
  - `festvox/*_durdata.scm`
  - `festvox/*_lexicon.scm`

Acoustic based selecion of prompts

- Build a basic db for the language:
  - could be diphone or simply diphone rich text
- Build standard clunit synthesizer:
  - doesn't need to be good
  - will define the units that are acoustically distinct
- With regular text data:
  - count frequencies of use of each clunit type
- With regular text data:
  - select utterances with besty coverage

Can be used as compression technique too

Example data selection

- awb timit database:
  - build clunit cluster tree
- Use Alice in Wonderland to score distibution
- Use Alice in Wonderland for sample utterances:
  - i.e. which utterances in Alice can best be used to synth Alice

```
(txt_00645 "he added, turning to the Knave.")
(txt_00330 "It's always six o'clock now.")
(txt_00150 "How can you learn lessons in here.")
(txt_00151 "Fetch me my gloves this moment.")
```

...

257 utts out of 1913 utts

Phonetic based data selection

- From a large set text:
  - select "nice" utterances
  - 5 to 15 words long
  - all in lexicon, no homographs
  - text2utts -dbname txt_ text.txt -o text.data
- Convert text to phonemes:
  - text2utts -level Segment -itype data text.data -o text.seg.data
- Select utterances with maximal (di)phone coverage:
  - dataset_select text.seg.data
- Extract the selected utts from text.data:
  - dataset_subset text.data text.seg.data.selected

Selection Example

- alice30.txt (152k)
- `text2utts -dbname alice_ alice30.txt -o alice.data`
  - 1920 total utterances
  - 668 "nice" utterances

  (alice_00003 "THE MILLENNIUM FULCRUM EDITION 3.0")

  (alice_00011 "I shall be late!"')

  (alice_00025 "Would the fall NEVER come to an end!")
- `text2utts -level Segment -itype data alice.data -o alice.seg.data`

  (alice_00003 "pau dh ax m ax l eh n iy ax m f uh l k

  r ax m ax d ih sh ax n th r iy p oy n t z ih r ow pau")
- `data_select alice.seg.data` – 189 utterances
- `dataset_subset alice.data alice.seg.data.selected >alice1.data`

Prompt Selection

- Multiple-passes:
  - voice talent wont say exactly what you want
  - multiple passes will give you more varied data
- Acoustic selection:
  - slower to do, and more speaker dependent
- Phonetic selection:
  - faster to do, but ignores acoustics