# Seminar: Voicebuilding for TTS Synthesis

Ingmar Steiner

WS 2014/15

# Formalities

This course is a **project seminar** (for LST/CoLi students), or a regular **seminar** (for CS/VC students).

*Successful* participation of the lecture "Text-to-Speech Synthesis" (Prof. Möbius) is a **mandatory prerequisite**.

To pass *this* course, you will need to build TTS voices and **submit them, along with a written report**. The report must explain the entire process, including any problems encountered, and their resolution (5 to 10 pp.). This report is due *two weeks* after the end of the seminar (**Friday 20$^{\text{th}}$ March, 2015**).

Register through LSF/HISPOS by **Friday 27$^{\text{th}}$ February, 2015**.

Mailing list for questions, discussion:
`voicebuildingsem@ml.coli.uni-saarland.de`

# Course Overview

- Split into 5 to 6 groups of 3 to 4 people each
- Design prompt list
- Record speech corpus in studio
- Process recordings (including automatic phonetic segmentation)
- Build TTS voices (unit-selection and maybe HMM-based variants)
- Use MaryTTS[1] (*invented here*)

[1] http://mary.dfki.de/

# Prompt list creation

0. Prerequisites: TeX Live, SoX (with MP3 support)
1. Surf to `https://github.com/psibre/arctic-prompts`
2. Download or clone it
3. Run `gradlew`
4. Postrequisites: Adobe Reader, Adobe Flash

Next: MaryTTS installation

# MaryTTS

Open-source, multilingual TTS platform implemented in Java

`http://mary.dfki.de/`

Development hosted at

`https://github.com/marytts/marytts`

# Download and install MaryTTS

0. Prerequisite: Java 7 or later
1. Surf to `https://github.com/marytts/marytts/releases`
2. Download latest installer package
3. Unzip it
4. Run `marytts`
5. Surf to `http://localhost:59125/`

# Install MaryTTS *from source*

0. Prequisites: JDK 7 or later, Git, Maven
1. Clone the source repository:

```
git clone https://github.com/marytts/marytts.git
```

2. Enter your repository and install:

```
mvn install
```

# Debugging in Eclipse

See `https://github.com/marytts/marytts/wiki/Eclipse`

Next: Voicebuilding Done Quick

# Get the data

1. Clone CMU SLT Arctic database

```
git clone git@bitbucket.org:psibre/cmu-slt-arctic-data.git
```

2. Enter directory and unpack with Gradle

```
./gradlew generateTxt
```

# Prepare voicebuilding directory

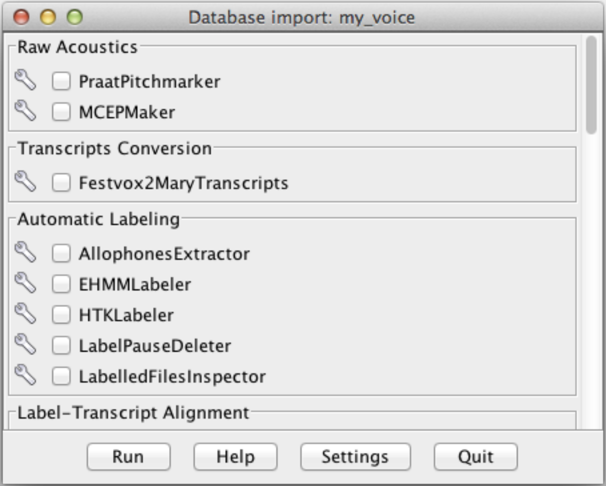
1. Move directories into place

```
cd build
mv ../wav ../text ../pm ../mcep .
```

2. Convert labels

```
ln -s ../lab lab_raw
mkdir lab
curl -fsSL https://gist.githubusercontent.com/
psibre/abf0d2ac833046af17cb/raw/1866078
b549510095 0392ccb253ae19fc156d099/convert_labels.
pl | perl
```

# Initialize voicebuilding



# Run voicebuilding components (1/2)

1. FeatureSelection
2. AllophonesExtractor
3. PhoneUnitComputer
4. HalfPhoneUnitComputer
5. TranscriptionAligner
6. PhonUnitFeatureComputer
7. HalfPhonUnitFeatureComputer
8. PhoneLabelFeatureAligner
9. HalfPhoneLabelFeatureAligner
10. WaveTimelineMaker
11. BasenameTimelineMaker
12. MCepTimelineMaker
13. PhoneUnitfileWriter
14. PhoneFeatureFileWriter

# Install Edinburgh Speech Tools

## OSX

Prerequisite: Homebrew

```
brew install speech-tools
```

## Ubuntu

```
sudo apt-get install speech-tools
```

# Run voicebuilding components (2/2)

Prerequisite: configure `estDir` as *parent* directory of where EST was installed

1. DurationCARTTrainer
2. F0CARTTrainer
3. HalfPhoneUnitfileWriter
4. HalfPhoneFeatureFileWriter
5. F0PolynomialFeatureFileWriter
6. AcousticFeatureFileWriter
7. JoinCostFileMaker
8. CARTBuilder
9. VoiceCompiler

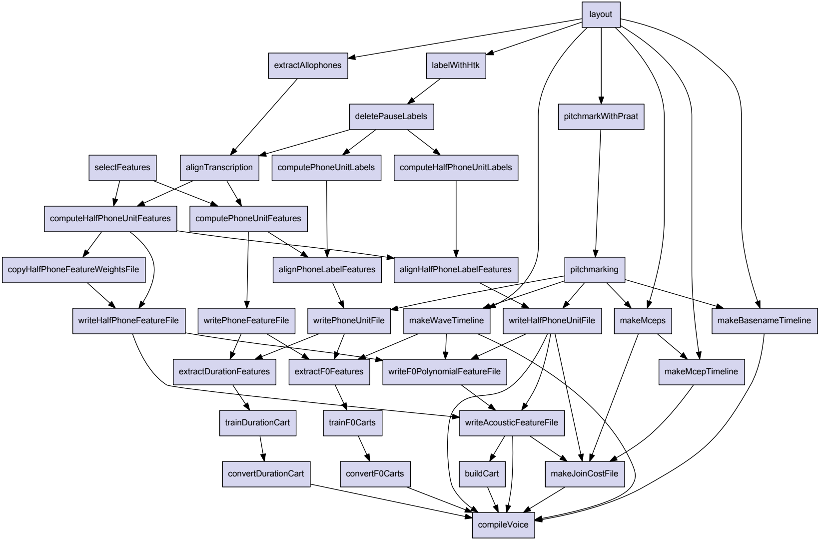
# Install new voice

1. Copy zip package for voice to MaryTTS installation directory
2. Unzip under `installed`
3. Move jar file from `installed/lib` to `installed`, so that

```
tree installed
installed/
+-- lib
|   \-- voices
|       \-- my_voice
|           +-- halfphoneFeatures_ac.mry
|           +-- halfphoneUnits.mry
|           +-- joinCostFeatures.mry
|           +-- timeline_basenames.mry
|           \-- timeline_waveforms.mry
\-- voice-my_voice-5.1.2.jar
```

Next: Instant Replay

# Voicebuilding Task Execution Graph



# Raw acoustics

1. Pitchmarking (using Praat)
   **input** `wav/*.wav`
   **output** `pm/*.pm`
2. MCEP coefficient extraction (using EST)
   **input** `wav/*.wav`
   **output** `mcep/*.mcep`

# G2P and labeling

1. Predict phone sequence from text (using MaryTTS)
   **input** `text/*.txt`
   **output** `prompt_allophones/*.xml`
2. Phone-level segmentation
   **input** `text/*.txt`, `wav/*.wav`
   **output** `lab/*.lab`
3. Check alignment
   **input** `prompt_allophones/*.xml`, `lab/*.lab`
   **output** `allophones/*.xml`

# Unit features

1. Select feature set

   **output** `mary/features.txt`

2. Compute and assign feature vector to each unit (using MaryTTS)

   **input** `allophones/*.xml`, `mary/features.txt`

   **output** `phonefeatures/*.pfeats`,
   `halfphonefeatures/*.hpfeats`

# Data files

Compile "timeline" files for

- Audio samples
  - **input** `wav/*.wav`, `pm/*.pm`
  - **output** `mary/timeline_waveforms.mry`
- Utterances
  - **input** `wav/*.wav`, `pm/*.pm`
  - **output** `mary/timeline_basenames.mry`
- MCeps
  - **input** `wav/*.wav`, `mcep/*.mcep`
  - **output** `mary/timeline_mcep.mry`

These contain the actual data from the `wav` and `mcep` files, in pitch-synchronous "datagram" packets.

# Acoustic models

Phone-level unit file

**input** `pm/*.pm`, `phonelab/*.lab`

**output** `mary/phoneUnits.mry`

Phone-level feature file

**input** `phonefeatures/*.pfeats`,

**output** `mary/phoneFeatures.mry`,
`mary/phoneUnitFeatureDefinition.txt`

CARTs for duration and F0

**input** `mary/phoneUnits.mry`, `mary/phoneFeatures.mry`,
`mary/timeline_waveforms.mry`

**output** `mary/dur.tree`, `mary/f0.left.tree`,
`mary/f0.mid.tree`, `mary/f0.right.tree`

# Unit selection files (1/3)

Halfphone-level unit file

**input** `pm/*.pm`, `halfphonelab/*.hplab`

**output** `mary/halfphoneUnits.mry`

Halfphone-level feature file

**input** `halfphonefeatures/*.hpfeats`,

**output** `mary/halfphoneFeatures.mry`,
`mary/halfphoneUnitFeatureDefinition.txt`

# Unit selection files (2/3)

F0 contour file

**input** `mary/halfphoneUnits.mry`,
`mary/timeline_waveforms.mry`,
`mary/halfphoneFeatures.mry`

**output** `mary/syllableF0Polynomials.mry`

Acoustic feature file

**input** `mary/halfphoneUnits.mry`,
`mary/syllableF0Polynomials.mry`,
`mary/halfphoneFeatures.mry`

**output** `mary/halfphoneFeatures_ac.mry`,
`mary/halfphoneUnitFeatureDefinition_ac.txt`

# Unit selection files (3/3)

Join cost file

**input** `mcep/*.mcep`, `mary/timeline_mcep.mry`,
`mary/halfphoneUnits.mry`,
`mary/halfphoneFeatures_ac.mry`

**output** `mary/joinCostFeatures.mry`,
`mary/joinCostWeights.txt`

Top-level CART

**input** `mary/halfphoneFeatures_ac.mry`

**output** `mary/cart.mry`, `featureSequence.txt`

# Distributable voice package

Collect, filter resources, generate descriptor using Maven

**input** `mary/cart.mry`, `featureSequence.txt`, `mary/dur.tree`, `mary/f0.left.tree`, `mary/f0.mid.tree`, `mary/f0.right.tree`, `mary/halfphoneFeatures_ac.mry`, `mary/joinCostFeatures.mry`, `mary/joinCostWeights.txt`, `mary/halfphoneUnits.mry`, `mary/timeline_basenames.mry`, `mary/timeline_waveforms.mry`

**output** `my_voice.zip`, `my_voice-component.xml`

Next: your turn

# Grouping

Work in small groups of 3 to 4 people.

Each group should have at least

- one native English speaker, and
- one programmer/hacker, and
- one phonetician

# Speech recording

Each group plans and carries out recordings for $\sim$1 h of speech data

Use a phonetically balanced prompt set, e.g., TIMIT or ARCTIC

Use collaborative versioning tools to share this data in the team, e.g., Dropbox, `git-annex`, etc.

# Phonetic segmentation

Use forced alignment for automatic segmentation

- EHMM,
- HTK,
- MAUS,
- CMU Sphinx,
- Julius,
- Kaldi,
- ...

and let's not forget: *manual labor!*

# Software dependencies

MaryTTS

- Java JDK (7 or higher)
- Maven 3
- GitHub

Acoustic analysis

- Praat (or WaveSurfer or ESPS)
- Edinburgh Speech Tools
- SoX

HMM-based voicebuilding

- HTK (with HDecode and HTS patch)
- HTS_engine
- SPTK
- Tcl (with SNACK library!)

At least some of this must be built from source, so GCC 4.5 (or so) is a must

Have fun!