The "Uprooted" MaryTTS Entry for the Blizzard Challenge 2017

Sébastien Le Maguer[1], Ingmar Steiner[1,2]

[1]Saarland University, [2]DFKI GmbH
Saarbrücken, Germany
{slemaguer|steiner}@coli.uni-saarland.de

Abstract

The MaryTTS system is a modular text-to-speech (TTS) system which has been developed for nearly 20 years. This paper describes the MaryTTS entry for the Blizzard Challenge 2017. In contrast to last year's MaryTTS system, based on a unit selection baseline using the latest stable MaryTTS version, the basis for this year's system is a new, experimental version with a completely redesigned architecture.

Index Terms: MaryTTS, parametric synthesis, Blizzard Challenge, modularity

1. Introduction

This paper presents the MaryTTS system entered into the Blizzard Challenge 2017. This entry is centered on the use of a new process to extract the descriptive features. As a back-end, we used the hidden Markov model (HMM) based speech synthesis system (HTS) [1], which added support for deep neural network (DNN) modeling in v2.3.1.

Previous versions of the MaryTTS system have participated in the Blizzard Challenge, from 2006 to 2009 and 2012 to 2013, and the corresponding papers document the implementation and evolution of the unit selection [2, 3], multilingual [4, 5], and HMM based synthesis [6, 7] capabilities. Last year's submission [8] was focused on a baseline for a new voicebuilding process with the current stable MaryTTS architecture for unit selection as a back-end.

The paper is organized as follows. Section 2 briefly presents the evolution of the architecture of MaryTTS. Section 3 describes the configuration of our system for the 2017 Blizzard Challenge, Section 4 discusses the results, followed finally by a conclusion in Section 5.

2. Evolution of the MaryTTS architecture

MaryTTS is a text-to-speech (TTS) system which has been developed at DFKI and Saarland University for nearly 20 years [9]. A principal design feature of MaryTTS is its modular architecture, which allows easy modification or extension of the processing pipeline. This is particularly valuable for researchers focusing on one part of this pipeline. Given a baseline process, changing only one module will allow researchers to evaluate their work – and inspect the process – more accurately.

Over the history of its development, MaryTTS has undergone several major reorganizations in its software structure, and the complexity of the system has grown significantly. It was initially implemented as a collection of Perl scripts, then rewritten in Java, and first publicly released under a free, open source software (FOSS) license in 2006. In the scope of various projects, a number of developers with different backgrounds have contributed to the codebase at different times (cf. Figure 1), leading to heterogeneous code style and patterns.

Constrained by the technical landscape during the early phase of MaryTTS development, the modular paradigm was initially limited to a custom runtime class architecture, while the build system and centralized source code management (SCM) led to a monolithic codebase. Switching the build system from Ant to Maven, and more recently to Gradle [10], migrating the SCM to Git, and distributing individual software components via repositories such as Bintray [11], has allowed us to extend the modularity to the entire codebase.

The old architecture's implementation of modularity limited to the class level, combined with the complexity of the codebase, made the system increasingly difficult to maintain and extend in a flexible way. To solve these fundamental issues, we are currently in the process of refactoring the MaryTTS codebase, in order to align it with the design concepts at the software engineering level.

The main difference to the old architecture is the replacement of the module-internal XML data representation with a Java object oriented representation using a lightweight implementation of ROOTS [12, 13]. Another important feature is the introduction of serializers which allow the users to import/export any kind of data in the system. For the moment, it is possible to import text and MaryXML data, and we are able to export MaryXML, JSON, TextGrid, HTS labels (compatible with Festival), and custom HTS labels (using another set of separators which facilitate the extension of the property list). This redesigned architecture is described in detail in [14].

3. System configuration for Blizzard

For this year's Blizzard Challenge, we chose to use a new, experimental workflow based on pyHTS [15] and the DNN-enabled HTS v2.3.1 [1]. Portions of this workflow may serve as a prototype for future voicebuilding pipelines in MaryTTS.

The modules used for the English front-end processing are the same as last year. The remainder of the data processing and synthesis system are detailed below.

3.1. Data preparation

First of all, the data package provided by the Blizzard Challenge organizers was stored in an internal repository, and processed by an intermediate project, managed by the Gradle build automation platform [10].

The initial step in the data preparation consisted in patching the data, fixing spurious typos and formatting errors in the text files, problems with the segmentation (`*.lab`) files, and inconsistencies between the recorded and transcribed content. Any audio files in WMA format were also decoded to PCM WAV format during this step (using FFmpeg [16]), in order to avoid downstream compatibility issues.

Since not all audiobooks were accompanied by corresponding text files, we manually extracted the text from several PDFs

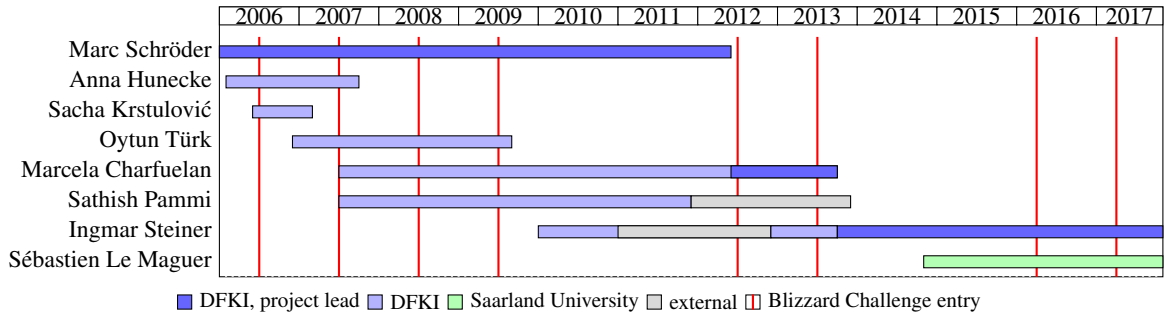


Figure 1: *Rough timeline of the core developer team for open-source MaryTTS at DFKI and Saarland University, with Blizzard Challenge participation*

and created segmentation files to align the text on each page with the audio. However, as our resources for this task were limited, we ended up using data from only 44 audiobooks:

```
AMidsummerNightsDream
AndroclesAndTheLion
AroundTheWorldIn80Days
Bears
BlackBeauty
BrerRabbitAndTheBlackberryBush
ChickenLicken
CleverRabbitAndTheLion
Dinosaurs
Elephants
GoldilocksAndTheThreeBears
HanselAndGretel
HowElephantsLostTheirWings
KingDonkeyEars
KnightsAndCastles
LittleRedRidingHood
Macbeth
OldMotherHubbard
OnAPirateShip
OnTheMoon
RailwayChildren
RainyDay
RobinHood_picturebook
RomeoAndJuliet
StoneSoup
StoryOfFootball
SunnyDay
TheBoyWhoCriedWolf
TheDaydreamer
TheDragonAndThePhoenix
TheEmperorAndTheNightingale
TheEnormousTurnip
TheGingerbreadMan
TheHareAndTheTortoise
TheInchPrince
TheLittleGiraffe
TheMousesWedding
TheMusiciansOfBremen
TheReluctantDragon
TheRunawayPancake
TheThreeLittlePigs
ThereWasACrookedMan
WhyTheSeaIsSalty
WindInTheWillows
  ↪  _picturebook
```

To bootstrap the data extraction for the available data, we assembled the orthographic text, along with the audio and transcribed segmentation, into a nested data structure to store *book*, *paragraph*, and *line* level alignment with the segmented audio. This data structure was serialized to JSON format as shown in Listing 1. In this way, each line from each book's text could be processed individually, but with line and paragraph number stored in the file basename.

The text files were processed to predict the syllable structure and phone sequence for each utterance, and serialized to MaryXML (see Listing 2). This processing was done using an internal snapshot build of MaryTTS 6 with a British English lexicon built from the Oxford Advanced Learners Dictionary provided for Festival v2.4. A total of 328 out-of-vocabulary tokens (including names of characters, places, and dinosaurs) were manually transcribed and provided as an auxiliary dictionary to ensure correct pronunciation.

The audio files were first transcoded to FLAC from their original compressed format, then upsampled to 48 kHz for voicebuilding, and downsampled to 16 kHz for automatic forced alignment. Since we extracted only the segmented utterances, concatenating them to one audio file per line of text, we ended up with a total of 3867 audio files containing 3 h, 57 min of speech.

We used a patched version of the Montreal Forced Aligner [17] (v0.8.1) running inside a Docker [18] container[1] to train and align the phonetic segments. The word-level pronunciation for all tokens was first extracted from the MaryXML files and compiled into a custom dictionary to prevent out-of-vocabulary issues during the forced alignment.

Finally, the upsampled audio, phonetic segmentation, and line-wise text files were packaged and deployed to our internal repository for consumption as data dependencies in the actual voicebuilding process.

3.2. Descriptive features

The descriptive features are used in two parts of the unit selection system: the preselection of the units and the prosody prediction. We distinguish a five-item context horizon: previous-previous (PP), previous (P), current (C), next (N), and next-next (NN). Based on this context, the descriptive features used are the following:

- Segment
  - phoneme identity (PP, P, C, N, NN)
  - no. segments from/to start/end of syllable
- Syllable
  - is it accented? (P, C, N)
  - is it stressed? (P, C, N)
  - no. segments in syllable (P, C, N)
  - no. syllables to end of phrase
  - no. syllables to end of word
  - no. segments in syllable
- Word
  - part-of-speech (POS) tag (P, C, N)
  - no. syllables in word (P, C, N)
  - no. segments in word
  - no. words to end of phrase
  - no. words to end of sentence
- Phrase
  - no. words in phrase (P, C, N)
  - no. syllables in phrase (P, C, N)
  - no. phrases to end of sentence
- Sentence
  - no. words in sentence
  - no. syllables in sentence
  - no. phrases in sentence
  - no. sentences to end of paragraph
- Paragraph
  - no. sentences in paragraph
- Speech turn

[1] https://hub.docker.com/r/psibre/kaldi-mfa/

```
{
 "AMidsummerNightsDream": {
  "paragraphs": [
   {
    "lines": [
     {
      "text": "A Midsummer Night's Dream",
      "basename": "AMidsummerNightsDream_001_001",
      "utts": [
       {
        "audioFile": "build/patched/blizzard_release_2017/enUK/fls/AMidsummerNights
          ↪  Dream/audio/01_Track_01.mp3",
        "start": 1.210204,
        "end": 3.044762,
        "text": "a midsummer night's dream"
       }
      ]
     }
    ]
   },
   {
    "lines": [
     {
      "text": "Based on the play by William Shakespeare.",
      "basename": "AMidsummerNightsDream_002_001",
      "utts": [
       {
        "audioFile": "build/patched/blizzard_release_2017/enUK/fls/AMidsummerNights
          ↪  Dream/audio/01_Track_01.mp3",
        "start": 3.044762,
        "end": 5.714739,
        "text": "based on the play by william shakespeare"
       }
      ]
     },
     {
      "text": "Adapted by Lesley Sims.",
      "basename": "AMidsummerNightsDream_002_002",
      "utts": [
       {
        "audioFile": "build/patched/blizzard_release_2017/enUK/fls/AMidsummerNights
          ↪  Dream/audio/01_Track_01.mp3",
        "start": 5.715011,
        "end": 7.885034,
        "text": "adapted by lesley sims"
```

```
    {
     "lines": [
      {
       "text": "WHAM!",
       "basename": "WindInTheWillows_picturebook_029_001",
       "utts": [
        {
         "audioFile": "build/patched/blizzard_release_2017/enUK/fls/WindInTheWillows
           ↪  _picturebook/audio/Wind_in_Willows_final01.mp3",
         "start": 243.095692,
         "end": 244.550658,
         "text": "wham"
        }
       ]
      }
     ]
    },
    {
     "lines": [
      {
       "text": "The stoats and the weasels were banished forever. Toad was so
         ↪  thrilled, he held a small party to celebrate.",
       "basename": "WindInTheWillows_picturebook_030_001",
       "utts": [
        {
         "audioFile": "build/patched/blizzard_release_2017/enUK/fls/WindInTheWillows
           ↪  _picturebook/audio/Wind_in_Willows_final01.mp3",
         "start": 247.240000,
         "end": 250.424762,
         "text": "the stoats and the weasels were banished forever"
        },
        {
         "audioFile": "build/patched/blizzard_release_2017/enUK/fls/WindInTheWillows
           ↪  _picturebook/audio/Wind_in_Willows_final01.mp3",
         "start": 250.424853,
         "end": 254.854649,
         "text": "toad was so thrilled he held a small party to celebrate"
        }
       ]
      }
     ]
    }
   }
}
```


Listing 1: *Head and foot of JSON data structure used to process the Blizzard data, with book, paragraph, and line level nesting*

– is it direct speech?

Therefore, compared to the previous year, the only noteworthy addition is the feature indicating that we are in a direct-speech turn or not. Furthermore, the punctuation information is not included as it is not available yet in the current MaryTTS version. This leads to an input feature vector of dimension 40 for the HMM duration training and of dimension 723 for the DNN training.

3.3. Back-end configuration

In order to achieve the synthesis, we use the pyHTS toolkit. This toolkit is a set of python scripts which decomposes the synthesis into two stages. The first stage is the acoustic parameter generation which can be HMM or DNN based. The second stage is the signal rendering using the parameters produced by the generator. Currently, the combination of STRAIGHT [19] and SPTK [20] is the main vocoder supported.

Therefore, pyHTS is mainly a wrapper around HTS and STRAIGHT which uses the same configuration.[2] For this entry, we used the default configuration as presented in Table 1. The duration is predicted using the HMM and then a feed-forward DNN is used to predict the other acoustic parameters. Unfortunately, during the training of the DNN part, performance bottlenecks on our department's shared GPU compute cluster prevented us from debugging our voice. Consequently, we found ourselves forced to submit our initial, "one shot" synthesis results.

Ac. param.	**configuration**
MGC	50 coef. + Δ + ΔΔ
F0	Linear interpolated = 1 coef. + Δ + ΔΔ 1 coef. V/UV property min. F0 = 50 Hz, max. F0 = 320 Hz
BAP	25 coef. + Δ + ΔΔ
Stage	**configuration**
HMM	5 states
DNN	3 hidden layers of 1024 nodes Sigmoid activation Adam optimizer

Table 1: *Configuration used for the DNN HTS/STRAIGHT back-end*

4. Results

During the Blizzard Challenge, 17 systems were evaluated. Among these systems, the following are particularly interesting in our case:

A natural speech;

B Festival benchmark system used for the CSTR entry in the Blizzard Challenge 2007 [21];

C HTS benchmark system;

D DNN benchmark system using the WORLD vocoder;

O our MaryTTS entry.

The 2017 Blizzard Challenge included three kinds of subjective evaluation: global mean opinion score (MOS), semantically unpredictable sentence (SUS) word error rate (WER) analysis, and MOS in a more focused analysis (such as intonation, stress, . . .) at the paragraph level.

In this section, we discuss some of the results.

[2] In contrast to HTS, the configuration is provided in JSON format.

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<maryxml xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="0.5"
  ↪    xml:lang="en-US" xmlns="http://mary.dfki.de/2002/MaryXML">
 <p>A Midsummer Night's Dream
  <s>A Midsummer Night 's Dream
   <prosody>
    <phrase>
     <t pos="DT">A
      <syllable stress="1">
       <ph p="ei">
        <features>
         <feature name="words_from_phrase_end" value="3"/>
         <feature name="ph_from_syl_start" value="0"/>
         <feature name="syls_from_word_end" value="0"/>
         <feature name="syl_stress" value="1"/>
         <feature name="next_syl_stress" value="2"/>
         <feature name="ph_from_syl_end" value="0"/>
         <feature name="next_word_pos" value="NNP"/>
         <feature name="phone" value="ei"/>
         <feature name="next_word_numsyls" value="3"/>
         <feature name="phrase_numsyls" value="6"/>
         <feature name="phrase_numwords" value="4"/>
         <feature name="sentence_numphrases" value="1"/>
         <feature name="syl_numph" value="1"/>
         <feature name="sentence_numwords" value="4"/>
         <feature name="inDirectSpeech" value="false"/>
         <feature name="next_syl_numph" value="2"/>
         <feature name="sentence_from_paragraph_end" value="0"/>
         <feature name="syls_from_phrase_end" value="5"/>
         <feature name="word_pos" value="DT"/>
         <feature name="phrases_from_sentence_end" value="0"/>
         <feature name="word_numsegs" value="1"/>
         <feature name="next_phone" value="m"/>
         <feature name="word_numsyls" value="1"/>
         <feature name="next_next_phone" value="ih"/>
         <feature name="syl_accent" value="false"/>
         <feature name="sentence_numsyllables" value="6"/>
         <feature name="next_syl_accent" value="false"/>
         <feature name="words_from_sentence_end" value="3"/>
         <feature name="paragraph_numsentences" value="1"/>
        </features>
       </ph>
      </syllable>
     </t>
     <t pos="NNP">Midsummer
      <syllable stress="2">
       <ph p="m">
        <features>
         <feature name="words_from_phrase_end" value="2"/>
         <feature name="ph_from_syl_start" value="0"/>
         <feature name="syls_from_word_end" value="2"/>
         <feature name="syl_stress" value="2"/>
         <feature name="next_syl_stress" value="1"/>
         <feature name="prev_word_numsyls" value="1"/>
         <feature name="ph_from_syl_end" value="1"/>
         <feature name="next_word_pos" value="NNP"/>
         <feature name="phone" value="m"/>
         <feature name="next_word_numsyls" value="1"/>
         <feature name="prev_phone" value="ei"/>
         <feature name="phrase_numsyls" value="6"/>
         <feature name="phrase_numwords" value="4"/>
         <feature name="sentence_numphrases" value="1"/>
         <feature name="syl_numph" value="2"/>

          [...]

        </features>
       </ph>
      </syllable>
     </t>
    </phrase>
   </prosody>
  </s>
 </p>
</maryxml>
```

Listing 2: *A MaryXML excerpt from `AMidsummerNights Dream_001_001.xml`, with serialized phone-level features*

4.1. Scoring evaluation results

The scoring evaluation results are presented in Figures 2a to 2c for the paragraph-level overall impression, sentence-level naturalness, and sentence-level similarity, respectively.

In all of the cases, our system was rated among the worst. Even more concerning is the fact that, even though we used a DNN approach with enriched labels, our system still scored lower than the HMM baseline. This can be attributed to two main factors.

First, the rendered audio sounds very buzzy. Secondly, and more problematically, some parts of the signals are just noise and completely unintelligible. But this was not unexpected, considering the aforementioned GPU issues, which prevented us from further investigating the source of these issues and fixing them in time for the submission.

4.2. SUS results

The second kind of analysis provided is the SUS WER results presented in Figure 2d. The results show that our system performance is once again among the worst. This confirms that the noisy audio is a significant problem, which needs to be resolved.

5. Conclusion

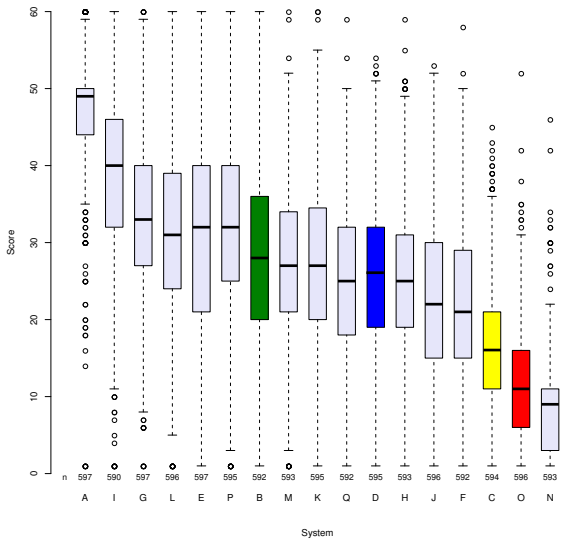
In conclusion, we have presented the MaryTTS entry to the Blizzard Challenge 2017. This system can be considered a first step in the refactoring process as we have used the new architecture to predict the descriptive features. We used the off-the-shelf HTS v2.3.1 as a back-end to produce the speech signals based on the descriptive features produced by HTS.

The results achieved by our system in this year's evaluation indicate a wide margin for improvement.

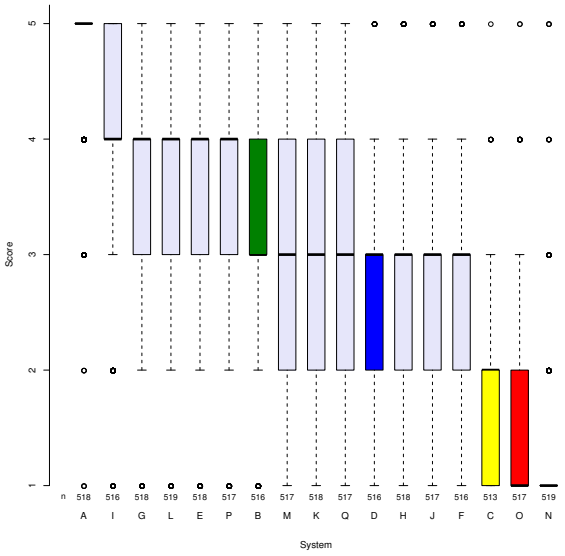
In the next challenge, we plan to focus on the back-end support in MaryTTS. This includes the introduction of unit selection, HMM, and DNN based synthesis into the system. Therefore, assuming these components work as expected, we would be able to focus on the core of the challenge: improving the synthesis.

6. Acknowledgements

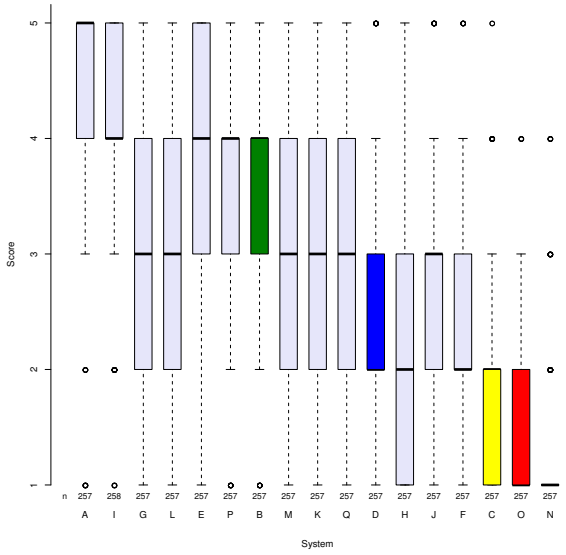
This research was funded by the German Research Foundation (DFG) as part of SFB 1102 "Information Density and Linguistic Encoding" at Saarland University.



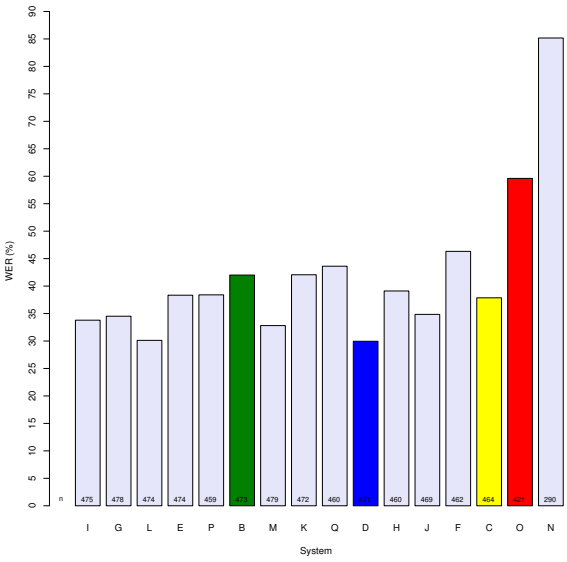
(a) *Overall paragraph MOS results.*



(b) *Naturalness MOS results.*



(c) *Similarity MOS results.*



(d) *SUS WER results.*

Figure 2: *Selected results from Blizzard Challenge 2017 evaluation. Festival is shown in green, HTS in yellow, the DNN baseline in blue, and MaryTTS in red.*

7. References

[1] "HMM-based speech synthesis system (HTS)." URL: http://hts.sp.nitech.ac.jp/

[2] M. Schröder, A. Hunecke, and S. Krstulović, "OpenMary – open source unit selection as the basis for research on expressive synthesis," in *Blizzard Challenge Workshop*, Pittsburgh, PA, 2006. URL: http://festvox.org/blizzard/bc2006/dfki_blizzard2006.pdf

[3] M. Schröder and A. Hunecke, "MARY TTS participation in the Blizzard Challenge 2007," in *Blizzard Challenge Workshop*, Bonn, Germany, 2007. URL: http://festvox.org/blizzard/bc2007/blizzard_2007/blz3_007.html

[4] M. Schröder, M. Charfuelan, S. Pammi, and O. Türk, "The MARY TTS entry in the Blizzard Challenge 2008," in *Blizzard Challenge Workshop*, Brisbane, Australia, 2008. URL: http://festvox.org/blizzard/bc2008/dfki_Blizzard2008.pdf

[5] M. Schröder, S. Pammi, and O. Türk, "Multilingual MARY TTS participation in the Blizzard Challenge 2009," in *Blizzard Challenge Workshop*, Edinburgh, Scotland, 2009. URL: http://festvox.org/blizzard/bc2009/dfki_Blizzard2009.pdf

[6] M. Charfuelan, "MARY TTS HMM-based voices for the Blizzard Challenge 2012," in *Blizzard Challenge Workshop*, Portland, OR, 2012. URL: http://festvox.org/blizzard/bc2012/DFKI_Blizzard2012.pdf

[7] M. Charfuelan, S. Pammi, and I. Steiner, "MARY TTS unit selection and HMM-based voices for the Blizzard Challenge 2013," in *Blizzard Challenge Workshop*, Barcelona, Spain, 2013. URL: http://festvox.org/blizzard/bc2013/DFKI_Blizzard2013.pdf

[8] S. Le Maguer and I. Steiner, "The MaryTTS entry for the Blizzard Challenge 2016," in *Blizzard Challenge*, Cupertino, CA, USA, 2016. URL: http://festvox.org/blizzard/bc2016/MARYTTS_Blizzard2016.pdf

[9] M. Schröder and J. Trouvain, "The German text-to-speech synthesis system MARY: A tool for research, development and teaching," in *Speech Synthesis Workshop*, Perthshire, Scotland, 2001. URL: http://www.isca-speech.org/archive_open/ssw4/ssw4_112.html

[10] "Gradle build tool: Modern open source build automation." URL: https://gradle.org/

[11] JFrog, "Bintray: Download center automation & distribution." URL: https://bintray.com/

[12] N. Barbot, V. Barreaud, O. Boeffard, L. Charonnat, A. Delhay, S. Le Maguer, and D. Lolive, "Towards a versatile multi-layered description of speech corpora using algebraic relations," in *Interspeech*, 2011, pp. 1501–1504.

[13] J. Chevelu, G. Lecorvé, and D. Lolive, "ROOTS: a toolkit for easy, fast and consistent processing of large sequential annotated data collections," in *International Conference on Language Resources and Evaluation (LREC)*, Reykjavik, Iceland, 2014. URL: http://lrec-conf.org/proceedings/lrec2014/summaries/338.html

[14] S. Le Maguer and I. Steiner, "Uprooting MaryTTS: Agile processing and voicebuilding," in *28th Conference on Electronic Speech Signal Processing (ESSV)*, Saarbrücken, Germany, 2017. URL: http://essv2017.coli.uni-saarland.de/pdfs/LeMaguer.pdf

[15] S. Le Maguer, "A python wrapper for HTS synthesis (pyHTS)." URL: https://github.com/seblemaguer/pyhts

[16] "FFmpeg." URL: http://ffmpeg.org/

[17] M. McAuliffe, M. Socolof, S. Mihuc, M. Wagner, and M. Sonderegger, "Montreal Forced Aligner: trainable text-speech alignment using Kaldi," in *Interspeech*, Stockholm, Sweden, 2017.

[18] "Docker – build, ship, and run any app, anywhere." URL: https://www.docker.com/

[19] H. Kawahara, I. Masuda-Katsuse, and A. de Cheveigné, "Restructuring speech representations using a pitch-adaptive time-frequency smoothing and an instantaneous-frequency-based F0 extraction: Possible role of a repetitive structure in sounds," *Speech Communication*, vol. 27, no. 3-4, pp. 187–207, 1999.

[20] T. Fukada, K. Tokuda, T. Kobayashi, and S. Imai, "An adaptative algorithm for mel-cepstral analysis of speech," in *International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, vol. 1, 1992, pp. 137–140.

[21] K. Richmond, V. Strom, R. A. Clark, J. Yamagishi, and S. Fitt, "Festival Multisyn voices for the 2007 Blizzard Challenge," in *Blizzard Challenge Workshop*, Bonn, Germany, 2007. URL: http://festvox.org/blizzard/bc2007/blizzard_2007/blz3_006.html