The MaryTTS entry for the Blizzard Challenge 2016

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 architecture text-to-speech (TTS) system whose development started around 15 years ago. This paper presents the MaryTTS entry for the Blizzard Challenge 2016. For this entry, we used the default configuration of MaryTTS based on the unit selection paradigm.

However, the architecture is currently undergoing a massive refactoring process in order to provide a more fully modular system. This will allow researchers to focus only on some part of the synthesis process. The current participation objective includes assessing the current baseline quality in order to evaluate any future improvements. These can be achieved more easily thanks to a more flexible and robust architecture. The results obtained in this challenge prove that our system is not obsolete, but improvements need to be made to maintain it in the state of the art in the future.

Index Terms: MaryTTS, unit selection, Blizzard Challenge, modularity

1. Introduction

MaryTTS is a text-to-speech (TTS) system whose development began at DFKI and Saarland University more than 15 years ago [1]. A prominent design feature of MaryTTS is that it is based on a modular architecture to achieve the synthesis process. Such an architecture allows the user to modify or extend specific parts of the process. This is particularly useful for researchers as, generally, people who are focusing on one part of the process. Considering a baseline process, changing only one module will allow researchers to evaluate their work more accurately.

However, since the beginning of its development, the complexity of the MaryTTS system has grown significantly. Many developers have contributed to the project at different times, and under different circumstances. This has led the system to be more monolithic even if the base part relies on a modular architecture. This monolithic property of the system has made it increasingly difficult to maintain or extend. Also, and more problematically, for evaluation and analysis purposes several parts of the code have become more or less opaque.

Furthermore, due to technical limitations at the time the project was conceived, the modular paradigm was initially constrained to the runtime class architecture, while the build system (Ant) and source code management (SCM) – first CVS, then Subversion – were used in a monolithic way. Migrating from Ant to Maven allowed modular concepts to reach the build system level, but the full flexibility of modular software components was not unlocked until they could be individually published to central repositories such as Bintray [2]. And since migrating to Git, we can even modularize at the SCM level.

Consequently, a massive refactoring effort is currently underway on the MaryTTS code, in order to fully take advantage of the architecture's modularity. The objective is to offer a plug and play system with default configurations for unit selection, hidden Markov model (HMM) based, and, in the future, deep neural network (DNN) based synthesis. However, for now, we are focusing more on the software architecture than the evolution of the synthesis processes themselves.

The MaryTTS development team has, over the years, and with fluctuating composition, participated in several iterations of the Blizzard Challenge, including 2006 to 2009 and 2012 to 2013, and the corresponding papers document the implementation and evolution of the unit selection [3, 4], multilingual [5, 6], and HMM based synthesis [7, 8] capabilities.

This paper presents the MaryTTS unit selection system entered into the Blizzard Challenge 2016. This entry is based on the conventional unit selection synthesis process, first described in [4, 9], but with a significantly updated voice building workflow. Part of our motivation is to establish an evaluation baseline for the significant changes and improvements expected over the next two years.

The paper is organized as follows. Section 2 briefly presents a global description of the architecture of MaryTTS. Section 3 then focuses on the voice building process. Section 4 describes the configuration of our system for the 2016 Blizzard Challenge, and finally, Section 5 discusses the results.

2. Description of MaryTTS

MaryTTS is a pure Java system based on a modular architecture. We can currently distinguish the following main parts: the core, the runtime interface, the natural language processing (NLP) components for each language, the acoustic part (such as the unit selection implementation) and finally the specific voice components.

The core is the structuring part of the system and is composed mainly of the *module* concept. Classically, a module is a software component dedicated to one specific operation in the TTS process. Therefore, the goal of the module is only to enrich the data description. Finally, a module can just be a wrapper around an external dependency (such as *hts_engine* [10]) or be fully implemented in the system. This allows the user to precisely define the synthesis process. In order to communicate between modules, MaryTTS has been designed to use an XML representation ("MaryXML") throughout the TTS pipeline, including within each module's processing logic. However, this internal representation is currently scheduled to be replaced by a lightweight ROOTS [11] implementation. This will allow an easier and more explicit process implementation in the module.

The NLP and the acoustic parts implement, classically, the modules needed to achieve the descriptive feature extraction and the waveform synthesis. However, their specificity consists of proposing a set of modules. Therefore, these parts cannot

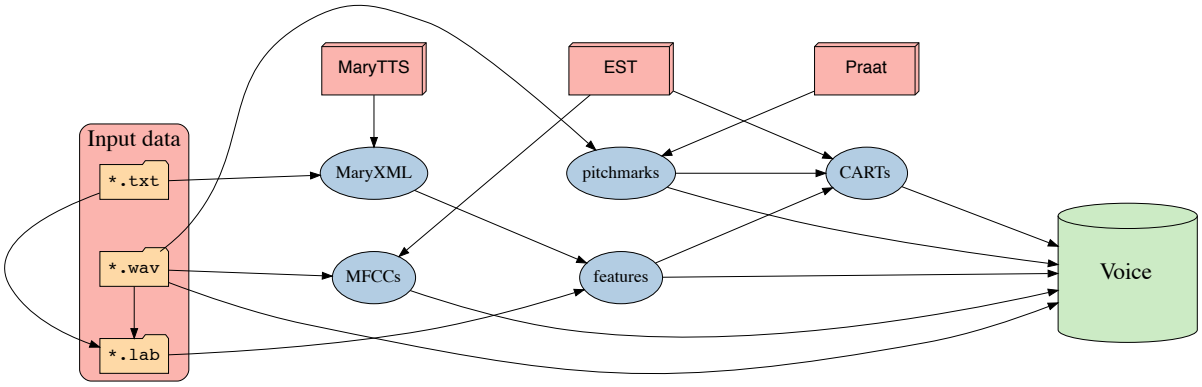

Figure 1: Overview of MaryTTS voice building process. See Section 3 for details.

be considered as autonomous as they do not architecture the sequence of modules.

The last part is the voice component, which not only contains the data needed for the synthesis itself, but it also describes the process to achieve it. For now, each voice is associated with a specific language and synthesis engine.

Finally, based on the available modules, the data and the process described in the selected voice, the runtime interface triggers the synthesis process. This is done by passing through the processing chain of modules defined in the voice configuration, in the proper sequence.

3. Voice building process

For this entry, we have completely redesigned the voice building process compared to the toolkit presented in [12]. The current toolchain is managed by a continuous delivery approach built on Gradle [13]. The general workflow is visualized in Figure 1.

The voice building process needs three matching sets of input files for the voice corpus: text, audio, and phonetic segmentation. The first stage is to extract linguistic/prosodic descriptive features from the text. This is done by using the MaryTTS runtime library and appropriate language components. The language of the voice is therefore available to the system in order to select the proper sequence of modules to run to get the features. Of course, as during the synthesis stage, it is possible to provide custom modules which are more accurate for the voice.

The second stage is to align the phonetic segmentation and the generated descriptive feature sequence. This is done by using a Levenshtein distance at the phone level. In order to be more robust, pauses are stripped out. It is of course possible to manually adapt the generated features to match the segmentation, or vice versa.

The third stage consists of extracting acoustic coefficients from the signal. This is done by calling Praat [14] for the F0, and the Edinburgh Speech Tools (EST) [15] for the mel-frequency cepstral coefficient (MFCC) extraction.

Finally, for a unit selection voice, the data and models are packaged as the runtime voice data, ready for distribution. If we want to build an HMM based voice, we only have to adapt this stage to use the HMM based speech synthesis system (HTS) [16] or an equivalent system.

3.1. Unit selection voice building

In MaryTTS, the unit selection voice database (i.e., the units and their feature vectors) is created during the voice building process. This information is provided at runtime to the unit selection algorithm. Likewise, the actual voice data is stored in "timelines", i.e., indexed sequences of pitch-synchronous data packets, which are then accessed by the synthesis engine to concatenate selected units.

The default feature weights applied to the computation of target and join cost coefficients (the latter consisting of 12 MFCCs, as well as log F0 and delta log F0) can be tuned, but this has to be done by hand.

Moreover, several classification and regression trees (CARTs) are trained from the units during the voice building process. One of these enables the unit selection algorithm to preselect candidate units. The others are used to predict prosodic parameters (viz., unit duration, F0) from the input text; these CARTs are trained with EST.

All of the publicly available unit selection voices for MaryTTS – such as *dfki-spike*[1] [12] – are built in this way, using open-source data and voice building projects wherever possible.

4. System configuration

The process of building the MaryTTS unit selection voice for the 2016 Blizzard Challenge was exactly the same as that of building all other unit selection voices, with the following special considerations. For future reference, we used MaryTTS version *5.2-beta3* with revision *eae3bcd* of our Gradle voice building plugin.

4.1. Data preparation

The provided data was unpacked, and all audio files were converted to 16 kHz, PCM WAV files, using FFmpeg [17]. The chapter-wise audio files, along with the corresponding orthographic text, were furthermore split into sentence-level utterances, using custom scripts.

[1] https://github.com/marytts/voice-dfki-spike

4.2. Phonetic segmentation

Phone-level segmentation was obtained in a forced alignment paradigm by processing the audio and orthographic text utterances with the online service WebMAUS [18]. After observing alignment imprecision and interoperability issues with the English language components in MaryTTS, we found that setting the language to "American English" in WebMAUS produced more reliable results than the "British English" setting.

4.3. Data processing and cleaning

After visually inspecting the phonetic forced alignment results, we decided to invest several hours into manual boundary and label corrections to alleviate the most egregious segmentation errors.

A small number of utterances and tokens in the voice data necessitated further attention. We manually prepared a small, auxiliary lexicon to handle three dozen proper names that were otherwise mispronounced by MaryTTS. Moreover, the alignment of eight utterances was manually adapted to avoid mismatches with the recorded audio.

4.4. 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 5 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:

- Halfphone:
  - is it left or right half?
- Segment
  - phoneme identity (PP, P, C, N, NN)
  - is it a pause? (P, N)
  - no. segments from/to start/end of syllable
  - no. segments from/to start/end of word
- Syllables
  - is it accented? (P, C, N)
  - is it stressed? (P, C, N)
  - position of segment in syllable (B/F)
  - syllable break (P, C)
  - no. syllables from/to prev/next stressed syllable
  - no. syllables from/to prev/next accented syllable
  - no. syllables from/to start/end of phrase
  - no. stressed syllables from/to start/end of phrase
  - no. accented syllables from/to start/end of phrase
  - no. syllables from/to start/end of word
  - no. segments in syllable
  - ToBI end tone (C, N, NN)
  - ToBI accent (C, N, NN)
- Words
  - no. syllables in word
  - no. segments in word
  - is it a punctuation word? (P, N)
  - no. words from/to prev/next punctuation
  - no. words from/to start/end of phrase
  - no. words from/to start/end of sentence
- Phrase
  - no. words in phrase
  - no. syllables in phrase
  - no. phrases from/to start/end of sentence
  - ToBI end tone of phrase (P, C)
- Sentence
  - no. phrases in sentence
  - no. words in sentence

These are the default descriptive features used for both the unit selection and the HMM synthesis in MaryTTS. Therefore, no custom descriptive features were introduced to be more specific to the task to achieve in the current challenge. The goal is to define a baseline of descriptive features that we will improve for the next challenges.

Finally, the number of features is particularly high considering a unit selection system. However, this can be justified by the use of decision trees for the prosody prediction.

4.5. Unit selection tuning

In the voice configuration's unit selection cost function, the weights for several prosodic and phonological target features were boosted slightly, while join cost feature weights were somewhat increased for delta log F0 and the first six MFCCs. However, the impact of this manual weight tuning was not systematically evaluated.

It should be noted that the experimental "sCost" coefficient [19] applied to the unit selection voice in the most recent MaryTTS Blizzard Challenge entry [8] was not used here.

4.6. Decision tree analysis

As indicated previously, to model the prosody, MaryTTS uses CARTs for F0 and duration. The previous descriptive features are used to construct the trees.

Considering the F0 decision trees, accent information is at the top. This is a positive finding as it indicates that the prosodic features are going to guide the F0 and therefore complete the unit selection process.

Considering the duration, we have found that pause/break information is also at the top of the decision tree. Once again, this is a positive finding as it allows the duration of the current segment to be controlled by the break context information.

5. Challenge 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 [20];
C HTS benchmark system;
D DNN benchmark system using the WORLD vocoder;
P our MaryTTS entry.

The 2016 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 (like intonation, stress, ...) at the paragraph level. In this section, we discuss some of these results.

5.1. Global results

First of all, Figure 2 and Figure 3 show the MOS for similarity and naturalness, respectively.

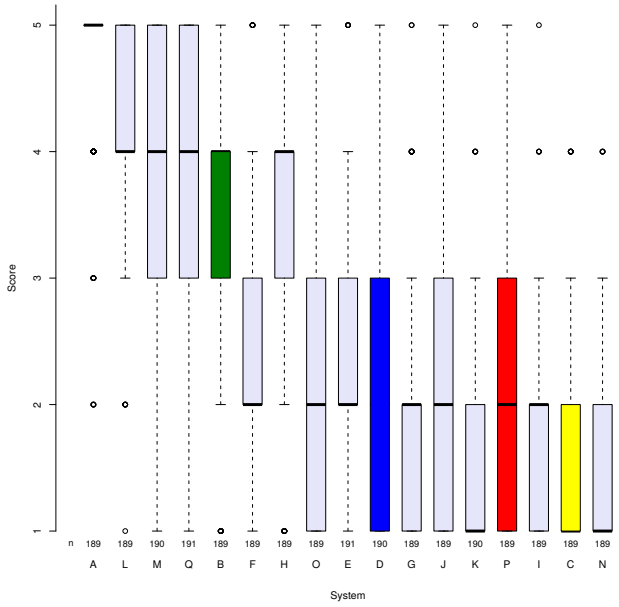

Figure 2: Similarity MOS results: Festival is shown in green, HTS in yellow, the DNN baseline in blue, and MaryTTS in red.

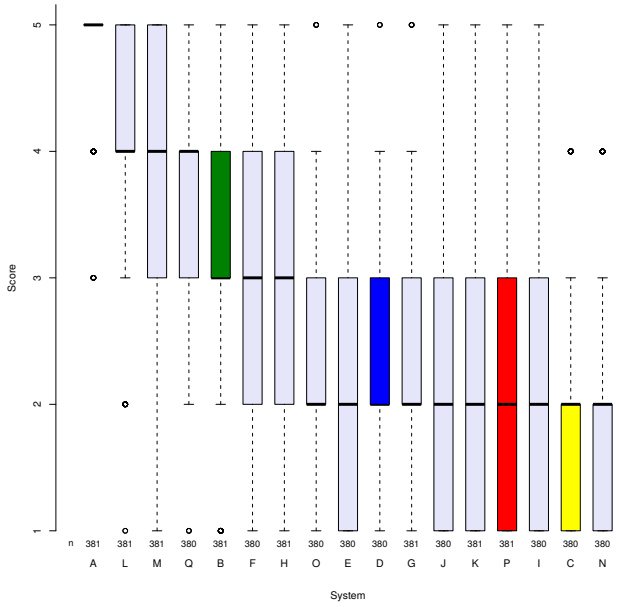

Figure 3: Naturalness MOS results: Festival is shown in green, HTS in yellow, the DNN baseline in blue, and MaryTTS in red.

These results show that our system is in the lowest range of the average-quality systems. If we also compare the quality of our system to the baseline unit selection system, Festival (system B), our system performs worse. However, we are at the same level as the parametric baseline, and slightly worse than the DNN systems.

For a unit selection methodology, our similarity score is low. This is due to several artifacts at the concatenation points and also some inconsistency in the F0 achieved by the system. Moreover, the phonetic segmentation was not thoroughly validated, and is far from perfect. The second part is confirmed by the low score achieved for the naturalness. The algorithm used is standard, and the weights applied to the descriptive features are not optimized. This statement is emphasized by the fact that the Festival system achieved a better score.

Considering also the fact that we did not include custom descriptive features appropriate for audiobooks (such as distinction between direct and narrative speech), the positive conclusion is that there is significant room for improvement.

5.2. SUS results

The second kind of analysis provided is the SUS WER results.

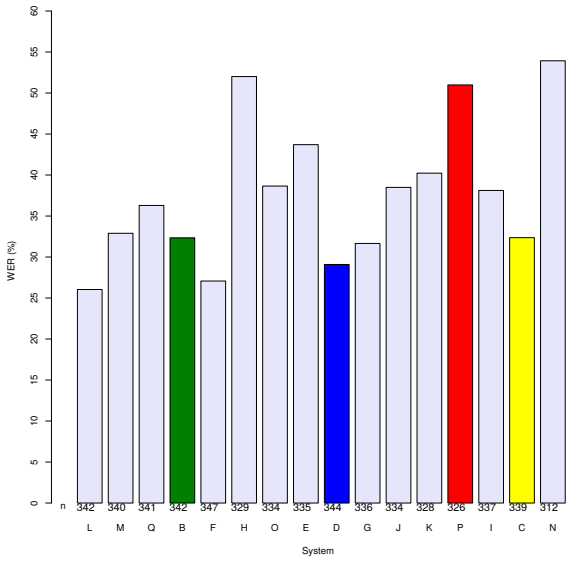

Figure 4: SUS WER results: Festival is shown in green, HTS in yellow, the DNN baseline in blue, and MaryTTS in red.

The results show that our system performance is among the worst. Based on the conclusion we drew before, we assume that the artifacts produced at the concatenation points are the main reason. We should therefore investigate more precisely which concatenated units produce these artifacts.

5.3. Focused analysis results

Specifically for this year's challenge, the organizers proposed a more focused analysis at the paragraph level. The following parameters were evaluated: emotion, intonation, listening effort, pleasantness, speech pauses, and stress.

First of all, Figure 5 presents the overall MOS for the synthesized paragraphs.

We achieved a synthesis quality around the HTS baseline, which is worse than the Festival and DNN baselines. However, we are at the same level as the majority of the other systems. This conclusion is the same for the majority of the parameters evaluated. Therefore, we are going to focus only on the part where the MaryTTS results differ from this trend. This concerns mainly two parameters where we achieve worse results than the majority of participants: speech pauses (Figure 6) and stress (Figure 7).

Considering the results, our system is predicting worse pauses than the majority. This is actually not surprising, as every pause duration is set to 400 ms. However, more surprisingly, using this default value we are not far away from the majority

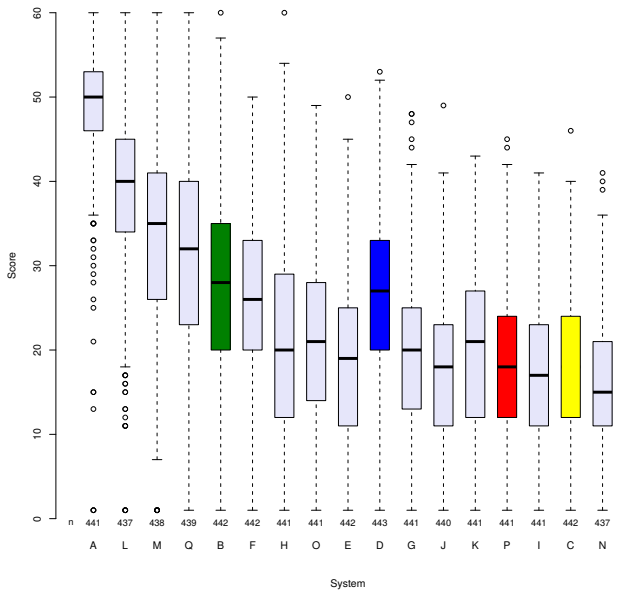

Figure 5: Overall paragraph MOS results: Festival is shown in green, HTS in yellow, the DNN baseline in blue, and MaryTTS in red.

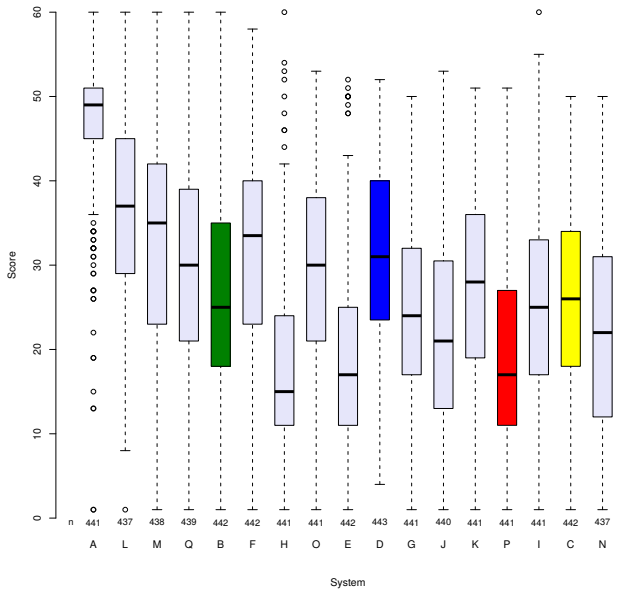

Figure 6: Speech pause MOS results: Festival is shown in green, HTS in yellow, the DNN baseline in blue, and MaryTTS in red.

of the other systems. In conclusion, by focusing a little bit on the pause modeling, we may improve the performance of our system.

Secondly, the stress modeling also performed slightly worse than the other systems. Therefore using descriptive features associated with stress is not enough. We may use other descriptive features, such as predictability, which worked for statistical speech synthesis [21]; we may also define other acoustic parameters to control the selection and the concatenation to improve the stress modeling.

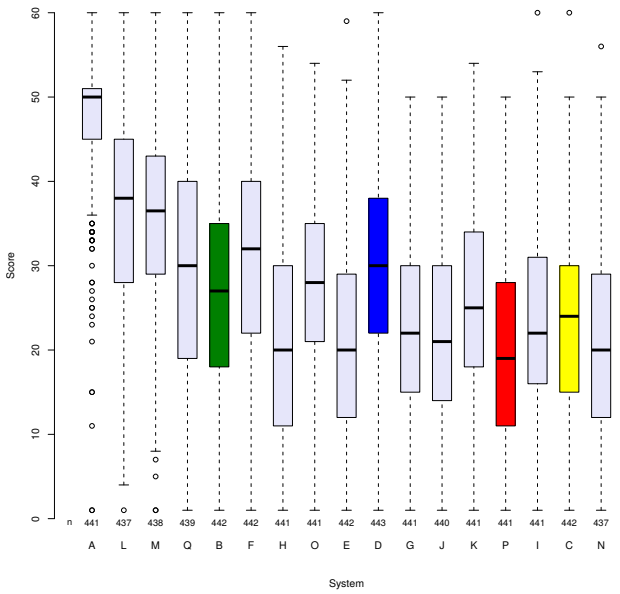

Figure 7: Stress MOS results: Festival is shown in green, HTS in yellow, the DNN baseline in blue, and MaryTTS in red.

6. Conclusion

In conclusion, we have presented the MaryTTS entry to the Blizzard Challenge 2016. This system can be considered a baseline in our case as it is currently in a massive refactoring process. This year's results achieved by the system indicate a big margin for improvement, especially if we compare our results to the Festival baseline.

In the next challenges, we plan to improve the descriptive feature generation and add more specific features adapted to the tasks. We also plan to update the unit selection module to close the gap to the state of the art in TTS synthesis.

7. Acknowledgements

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

8. References

[1] 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

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

[3] 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

[4] 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

[5] 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

[6] 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

[7] 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

[8] 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

[9] M. Schröder and A. Hunecke, "Creating German unit selection voices for the MARY TTS platform from the BITS corpora," in *Speech Synthesis Workshop*, Bonn, Germany, 2007, pp. 95–100. URL: http://www.isca-speech.org/archive_open/ssw6/ssw6_095.html

[10] "hts_engine API." URL: http://hts-engine.sourceforge.net/

[11] 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

[12] M. Schröder, M. Charfuelan, S. Pammi, and I. Steiner, "Open source voice creation toolkit for the MARY TTS platform," in *Interspeech*, Florence, Italy, 2011, pp. 3253–3256. URL: http://www.isca-speech.org/archive/interspeech_2011/i11_3253.html

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

[14] P. Boersma and D. Weenink, "Praat: doing phonetics by computer." URL: http://praat.org/

[15] S. King, A. W. Black, P. Taylor, R. Caley, and R. Clark, "Edinburgh Speech Tools library," 1994–2004. URL: http://www.cstr.ed.ac.uk/projects/speech_tools/

[16] H. Zen and T. Toda, "An overview of Nitech HMM-based speech synthesis system for Blizzard Challenge 2005," in *Interspeech*, 2005. URL: http://www.isca-speech.org/archive/interspeech_2005/i05_0093.html

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

[18] T. Kisler, U. Reichel, F. Schiel, C. Draxler, B. Jackl, and N. Pörner, "BAS speech science web services – an update of current developments," in *International Conference on Language Resources and Evaluation (LREC)*, Portorož, Slovenia, 2016, pp. 3880–3885. URL: http://lrec-conf.org/proceedings/lrec2016/summaries/668.html

[19] S. Pammi and M. Charfuelan, "HMM-based sCost quality control for unit selection speech synthesis," in *Speech Synthesis Workshop*, Barcelona, Spain, 2013, pp. 53–57. URL: http://www.isca-speech.org/archive/ssw8/ssw8_053.html

[20] 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

[21] S. L. Maguer, B. Möbius, and I. Steiner, "Toward the use of information density based descriptive features in HMM based speech synthesis," in *Speech Prosody*, Boston, MA, 2016, pp. 1029–1033. URL: http://www.isca-speech.org/archive/sp2016/abstracts.html#abs190