Introducing the Speect speech synthesis platform

Johannes A. Louw, Daniel R. van Niekerk, Georg I. Schlünz

Human Language Technologies Research Group
Meraka Institute, CSIR, Pretoria, South Africa

jalouw@csir.co.za, dvniekerk@csir.co.za, gschlunz@csir.co.za

Abstract

We introduce a new open source speech synthesis engine and related set of tools: Speect is designed to be a portable and flexible synthesis engine, equally relevant as a research platform and runtime synthesis system in multilingual environments. In this paper we document our approach to the rapid development of British English voices for the 2010 Blizzard Challenge using this platform and resources.

Index Terms: speech synthesis, multilingual, open source.

1. Introduction

This paper presents our first entry into the Blizzard Challenge [1], where different speech synthesis techniques are subjectively evaluated and can be directly compared due to the use of a common corpus of speech data.

The data given to the participants consisted of recorded and annotated speech, as well as pre-processed Festival [2] utterances and HMM-based Speech Synthesis System (HTS) [3] context labels. One of our goals was to test Speect on relatively large corpora as we have in the past only built South African voices with very small corpora. All the utterance and label generation modules were implemented in-house and only the supplied recordings and text annotations were used, thus excluding the other labels provided and limiting manual intervention.

The remainder of the paper is organised as follows, Section 2 introduces the Speect system, Section 3 describes the voice implementation, Section 4 provides details on voice building , Section 5 presents the results followed by a discussion and conclusion in Section 6.

2. The Speect TTS system

Speect is a multilingual text-to-speech (TTS) system that offers various application programming interfaces, as well as an environment for research and development of TTS systems and voices.

The use of the term "multilingual" text-to-speech in this paper, refers to *simple multilingual speech synthesis* [4] where language switching is usually accompanied by voice switching. Systems of this nature are especially important in countries with more than one official language as is the case in South Africa.

Speect was developed with maximum portability in mind. It is written in the C language, with a strict conformance to the ISO/IEC 9899:1990 standard. Platform specific system calls are abstracted to allow ports to new platforms. The architecture follows a modular object oriented approach with a plug-in system, aiming to separate the linguistic and acoustic dependencies from the run-time environment. A Python wrapper interface is also provided, allowing for rapid research and development of voices.

The system architecture can be divided into two distinct parts, namely: *engine* and *plug-ins*.

2.1. Engine

The engine is completely independent of any language or waveform generation modules, and is solely responsible for the loading of voices and their associated data and plug-ins, and controlling the synthesis process. The engine is not dependent on any external libraries, and provides lower level functionality to the plug-ins.

The engine consists of the base system and the object framework (see Figure 1). The base system provides a low level library to the following modules:

- abstract data types (lists, buffers, hash tables),
- utilities (memory allocation, byte-swapping, timing, fundamental types, versioning, math and system path functions),
- error handling, debugging and logging,
- platform independent concurrency abstraction,
- UTF-8 string handling (character and string level functions, printing functions and regular expressions),
- and an object system.

The object system allows an object-oriented programming approach to the higher level libraries implemented in the object framework. These higher level libraries provide the following modules:

- containers (map, list),
- data sources and data serialization,
- heterogeneous relation graphs (HRGs) [5] (for internal utterance representation),
- plug-in manager,
- and a voice manager.

2.2. Plug-ins

The plug-ins provide new object types and interfaces to voice data (linguistic and acoustic), as well as the processors that convert some form of textual input into a waveform. These processors, which are known as *utterance processors*, receive an utterance as input and transforms the utterance in some way.

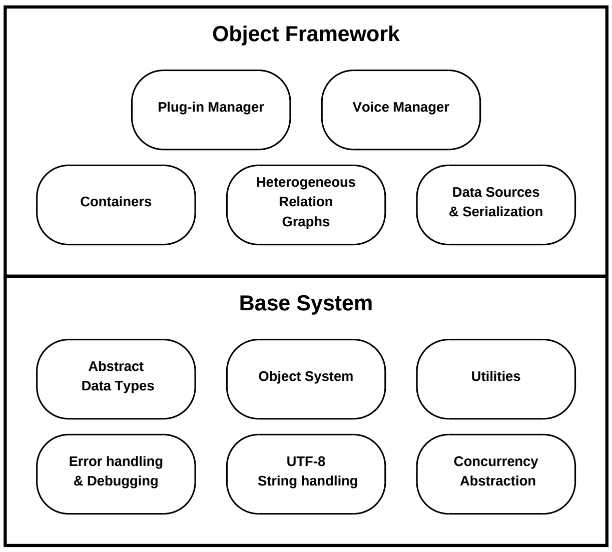

Figure 1: *Speect Engine architecture.*

2.2.1. *Utterance processors*

In Speect the *utterance* is the input and output of all utterance processors, even the waveform generating processors which generate the speech signal. Internally an utterance is represented as an HRG based on the work done in [6], and all utterance processors just add information to this structure through an interface provided by the engine.

Utterance processors manipulate the utterance based on knowledge of the:

- *input type*: an email message requires some extra processing when compared to a single line of text,
- *language*: phonetisation will, for example, be different for English when compared to isiZulu,
- and the specific *voice*: different voices will have different speaking rates, pitch contours and so on.

2.2.2. *Utterance type*

There is a pipeline of utterance processors doing transformations on the utterance, and producing the synthetic speech. Such a collection of utterance processors is known as an utterance type. This enables us to define a different pipeline of utterance processors, or order of utterance processors, for different input types.

2.2.3. *Feature processors*

Utterance processors also make use of *feature processors* (feature functions in [6]). A feature processor extracts features from individual units in an utterance; these features can then be used by the utterance processor. For example, we might have a feature processor that, when given a word, syllabifies the word and returns the syllables. Feature processors are defined in a key-value (name, processor implementation) mapping, and are called by utterance processors by their names. The real power of feature processors becomes apparent when doing multilingual TTS. We can, for example, reuse an utterance processor and just redefine the key-value pair of a feature processor (same name, different implementation) to do syllabification for a different language.

All of this comes together in the definition of a voice.

2.3. Voice

In Speect, a voice defines the utterance types that can be used for synthesis with the specific voice. Each of these utterance types defines a pipeline of utterance processors. The voice also defines the feature processors key-value mapping, connecting a named feature processor to a specific implementation, which the utterance processors can then use. Finally, the voice defines it's data, be that linguistic (phone sets, grapheme to phoneme rules, etc.) or acoustic (unit inventory, acoustic models, etc.). Note that everything that a voice defines can be shared between voices. This voice definition is in the JSON [7] format which can be easily edited to try different processor implementations and data sets. Figure 2 gives a representation of this voice definition.

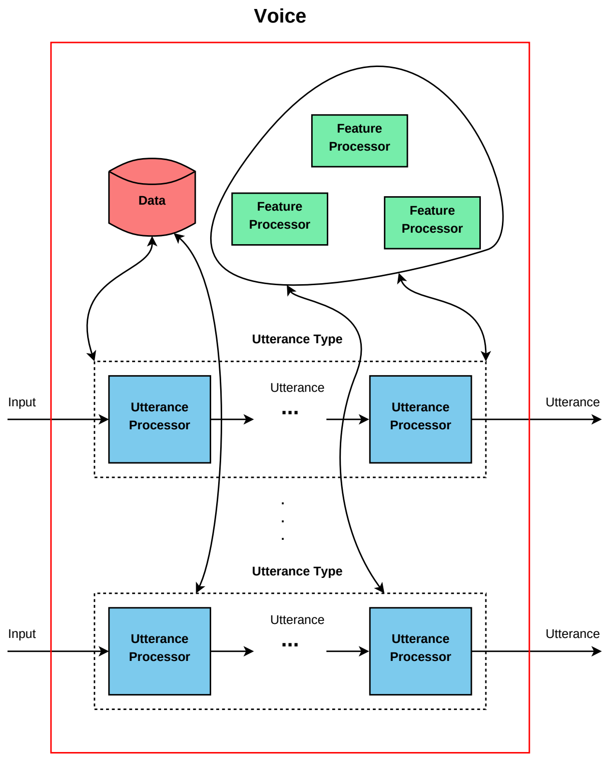

Figure 2: *Speect voice definition.*

Speect is released under an MIT license and is available for download at `http:\\speect.sourceforge.net`.

3. English voice implementation

For the implementation of the British English voices required for the challenge, we chose to start with our minimal set of currently available resources and implement modules through a process of rapid development using a number of freely available software packages and Speect's mechanism to incorporate Python code during synthesis.

As we planned to use HTS as the back-end synthesiser, natural language processing (NLP) modules were implemented according to the requirements of the set of linguistic features defined in the HTS label set developed in [8].

NLP and digital signal processing (DSP) utterance processors were implemented as described in the following sections.

3.1. NLP

3.1.1. Tokenisation

Sentence and word tokenisers were implemented based on Unicode Standard rule sets for detecting sentence and word boundaries [9]. The rule sets are regular expressions which operate on Unicode character *properties* instead of the characters themselves [10], allowing for a language-generic approach.

3.1.2. Normalisation

The classification and expansion of non-standard word tokens such as numbers, times and dates were inspired by the rule-based number spell-out approach introduced in [11]. The rule engine is separated from the rules for a modular implementation. Tokens are classified by regular expressions which perform substitutions recursively in a context-free grammar style. When the terminals in this "grammar" are reached, they are expanded into words. The number spell-out expands numbers by exploiting their inherently recursive verbal structure, through repeated division and modulo operations.

3.1.3. Lexical look-up

The lexical look-up follows the implementation in [2], and searches for each normalized word in the addendum (the addendum is a user definable lexicon) and then the lexicon. If not found, then a pronunciation prediction is done on the word.

3.1.4. Pronunciation prediction

For the pronunciation prediction module a pre-processed version of the CUVOALD pronunciation dictionary [12] containing around 63000 words and the MRPA phone set used in Festival was used as a starting point. This dictionary was modified to add explicit schwa's where syllabic consonants existed as well as a few other transformations to make it more appropriate in the case of South African English [13]. Using these resources the extraction of letter-to-sound rewrite rules was achieved using the Default&Refine algorithm [14], resulting in 19377 distinct rules. For the purposes of the challenge we also added 295 entries which did not exist in our source dictionary into a pronunciation addendum.

3.1.5. Syllabification

The pronunciation dictionaries (the addendum and lexicon) can define a word's syllabification. If it is not defined then the pronunciation is syllabified with an algorithm based on the Optimality-Theoretic analysis and markedness constraints in [15]. Some additional rules were added for syllabic consonants (e.g. bottle, button, etc.), which are not treated in [15]. As we deemed the output of this process to often provide better results than the syllables defined in CUVOALD, we chose to discard syllabification information provided in CUVOALD.

3.1.6. Part-of-speech tagging

A data-driven trigram part-of-speech (POS) tagger was built using the Natural Language Toolkit (NLTK) [16]. It was trained on the included CoNLL 2000 corpus of about 260K words [17].

3.1.7. Chunking

For chunking a data-driven approach was also considered and a noun, verb and prepositional phrase chunker was built with NLTK. The training data set was again the CoNLL 2000 corpus.

3.1.8. Phrasing

A baseline phrase break predictor inserts breaks based on punctuation. This was extended to use the chunk information by implementing a rule-based phrase tokeniser on the chunk sequences. The tokeniser uses the same engine as the sentence and word tokenisers. However, it was found that the developed rules did not behave consistently, so the extension was discarded (and, by implication, chunking as well).

3.1.9. Stress assignment

Rules based on syllable and POS information [18] assign word-level and sentence-level stress to the text. The categories for word-level stress are "unstressed", "primary" and "secondary". Sentence-level stress is binary. The rules are not exhaustive; they cover only words with a number of syllables up to three.

3.1.10. Tone assignment

The end of phrase ToBI [19] tone assignment was implemented by use of a CART [20] that is included as part of Festival [2] and trained on the Boston University FM Radio Data Corpus.

3.2. DSP

An utterance processor plug-in was written to provide an interface for Speect to the *hts_engine API* (version 1.02) [21]. A *feature processor* is executed on each phone segment (as determined by the *lexical look-up* process) to extract it's full context labels from it's utterance structure. These context labels are then passed on to the *hts_engine* which synthesises the speech waveform based on the labels and the trained models.

4. Voice building

During the voice building stage we use the text front-end as described in Section 3 in combination with the tools released as part of Speect to perform phonetic alignment and train HMM models for synthesis. These two aspects are described in the following two sections.

4.1. Phonetic alignment

We use a forced-alignment process based on HTK [22] implemented in a tool released as part of the Speect tool set. This tool allows one to perform model initialisation from manually aligned speech data transcribed in a different language or phone set by mapping to broad phonetic categories (see Figure 3 for an illustration of this process).

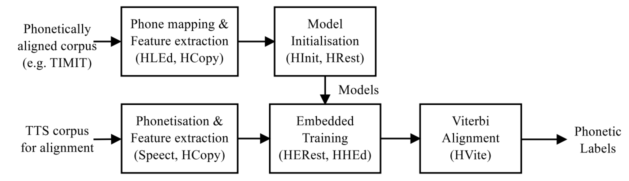
Phonetically aligned corpus (e.g. TIMIT)
Phone mapping & Feature extraction (HLEd, HCopy)
Model Initialisation (HInit, HRest)
Models
TTS corpus for alignment
Phonetisation & Feature extraction (Speect, HCopy)
Embedded Training (HERest, HHEd)
Viterbi Alignment (HVite)
Phonetic Labels

Figure 3: *Automatic phonetic alignment process.*

This approach has been shown to improve the accuracy of results consistently when aligning small corpora of various languages over flat start initialisation [23]. For the alignment of the two English corpora we follow this approach, using the multi-speaker TIMIT corpus [24] for model initialisation by applying the mappings detailed in Table 1.

Category	TIMIT	MRPA
long vowels	iy, ao, ow, uw, ux, er	@@, aa, uu
short vowels	ih, eh, ae, aa, ah, uh, ix	@, a, e, i, ii, o, oo, uh, u
diphthongs	ey, aw, ay, oy, ax	ai, au, e@, ei, i@, oi, ou, u@
approximants	l, r, w, y	l, r, w, y
nasals	m, n, ng, nx	m, n, ng
voiced plosives	b, d, g, dx, q	b, d, g
unvoiced plosives	p, t, k	p, t, k
voiced affricates	jh	jh
unvoiced affricates	ch	ch
voiced fricatives	z, zh, v, dh	z, zh, v, dh
unvoiced fricatives	s, sh, f, th, hh	s, sh, f, th, h
long pauses	pau, h#	pau
short pauses	epi, *cl	pau_cl

Table 1: Phone mappings for alignment purposes.

Informal inspection confirmed that alignments obtained were of a high quality and indeed of better quality than the flat start alternative, especially in the case of the "roger" corpus.

4.2. HMM training

Training of HMM models was done via the standard demonstration script available as part of HTS, incorporating global variance as described in [25] and modified to allow the use of Speect as a text analysis front-end. This made use of HTS version 2.1 to construct models for the hts_engine API (see Section 3.2).

For the model tying decision tree, we made use of the linguistic questions provided with the demo script and customised the phone set specific portions by generating questions based on phonetic categories defined in the phone set (e.g. categories such as plosives, nasals and vowels and voicing etc.).

For the HTS label feature set we experimented with labels of varying complexity, ranging from the full set used in the demo to reduced sets including only simple features based on word, syllable and segment positions. The final system used the full label set, however it was often difficult to distinguish between synthesis samples obtained using reduced label sets through informal listening tests.

Acoustic models for both tasks (EH1 and EH2) were trained on all the available utterances in the respective corpus.

5. Results

The Blizzard challenge consisted of a number of tasks; we participated in the tasks EH1 (build a voice from the UK English "rjs" database, 4014 utterances) and EH2 (build a voice from the ARCTIC portion of the UK English "roger" database, 1132 utterances). Our designated system identification letter is "O"; system "A" is natural speech, system "B" is a Festival unit-selection voice benchmark, system "C" is an HTS 2005 benchmark, system "D" is the same as "C" but with hand corrected labels and system "E" is a speaker-adaptive HTS 2007 benchmark. Systems "F" to "V" are other participants. In the challenge the submitted entries were evaluated in the following three categories:

- similarity to original speaker,
- mean opinion score (naturalness), and
- word error rate (intelligibility).

5.1. Similarity to original speaker

Figures 4 and 5 show the results of the perceptual tests from all the listeners for the "similarity to original speaker" evaluation. From the pairwise Wilcoxon signed rank tests we can see that our system is not significantly different from system L, but is different from the other systems for task EH1. For task EH2, our system is not significantly different from systems K, L and N, and is different from the rest.

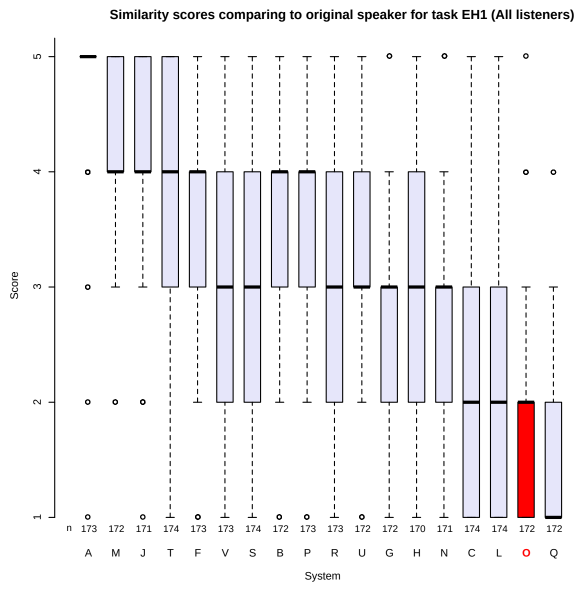

Figure 4: *Task EH1: Similarity to original speaker.*

5.2. Mean opinion score

Figures 6 and 7 gives the results from the perceptual tests from all the listeners for the mean opinion score (MOS) evaluation. The pairwise Wilcoxon signed rank tests return the same results for task EH1 as in the "similarity to original speaker" evaluation. In task EH2 our system is not significantly different from system systems K and L, but is different from the rest.

5.3. Word error rate

Figures 8 and 9 gives the results from the perceptual tests from all the listeners for the word error rate (WER) evaluation. The pairwise Wilcoxon signed rank tests results indicate that for task EH1 our system *is* significantly different from systems A, H and S, while there is no significant difference to the rest. For task EH2 our system *is* significantly different from systems A, C, D, G, J and V while it is not significantly different from the rest.

6. Discussion and conclusion

Considering the mean opinion scores and voice similarity scores, the trends for each task are consistent:

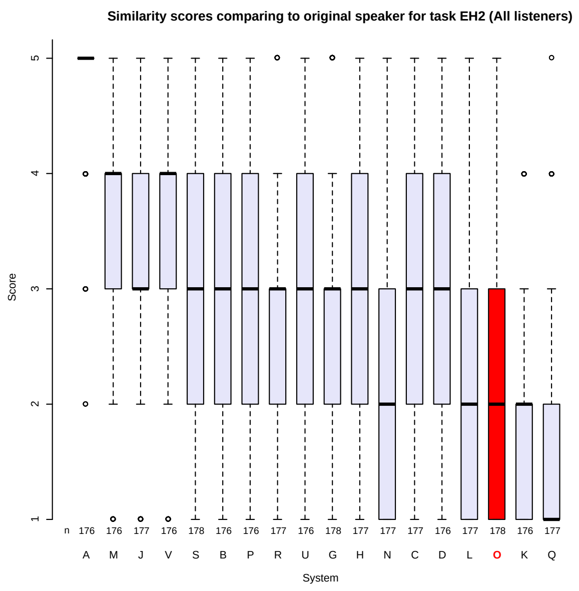

Figure 5: *Task EH2: Similarity to original speaker.*

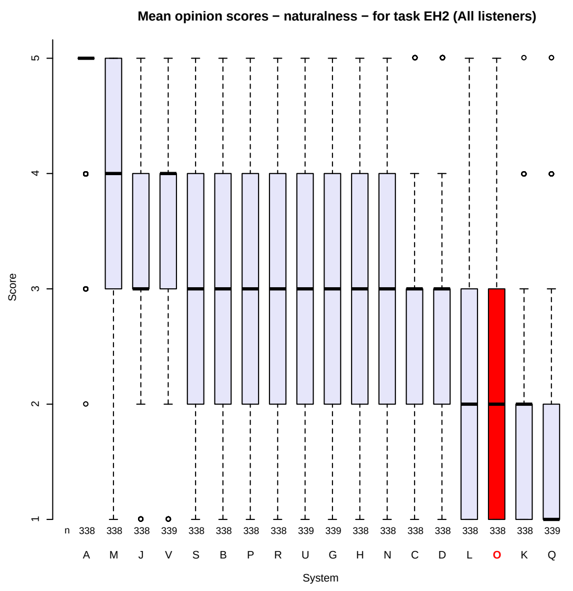

Figure 7: *Task EH2: Mean opinion score.*

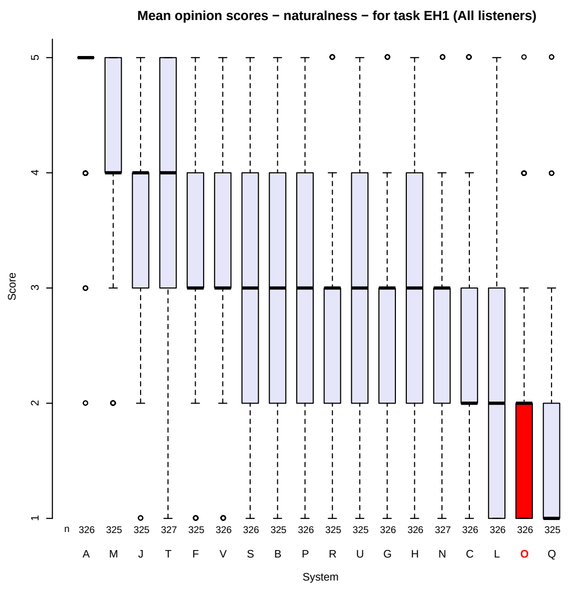

Figure 6: *Task EH1: Mean opinion score.*

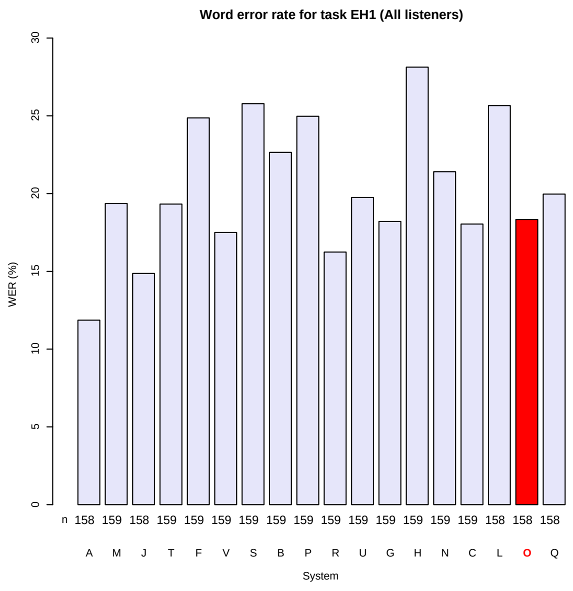

Figure 8: *Task EH1: Word error rate.*

1. scores are significantly lower than most entries, and
2. scores are generally higher on the smaller "roger" corpus.

Neither of these results are unexpected; the first result highlights the fact that our relatively new synthesiser implementation, containing only freely available components, does not yet employ advanced spectral representations (such as STRAIGHT [26]) or improved source excitations as proposed in recent works [27][28]. The second result suggests that it is easier to retain the original speaker's voice characteristics when a smaller number of well designed utterances are used for training; in the case of the "rjs" corpus, using all the available utterances probably resulted in "over averaged" acoustic models from this perspective.

Comparing results for intelligibility we were initially surprised by the relatively high word error rates on the EH2 task. Subsequent analysis suggested that some properties of our pronunciation dictionary with modifications for South African English might have compromised our efforts here. It is interesting that in the EH1 task this inefficiency seems to be less apparent; this is evident from the the results (and was also confirmed by listening to isolated speech samples).

These results suggest that future work should focus on an improved waveform generation back-end and more appropriate acoustic modeling to improve overall naturalness.

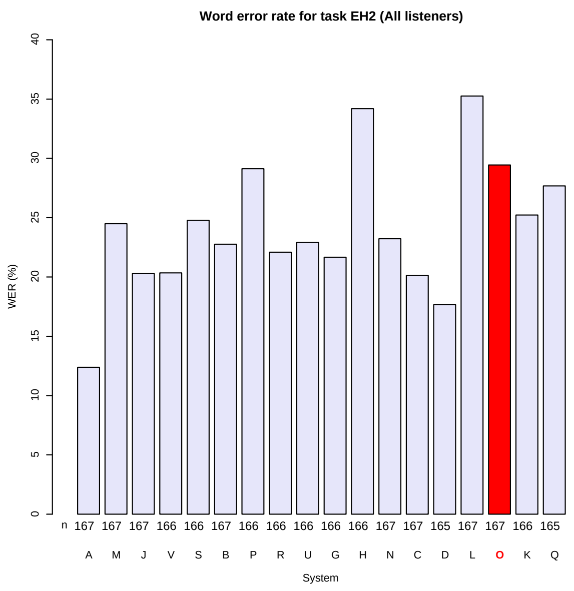

Figure 9: *Task EH2: Word error rate.*

7. References

[1] A. W. Black and K. Tokuda, "The Blizzard Challenge-2005: Evaluating corpus-based speech synthesis on common datasets," in *Interspeech*, Lisbon, Portugal, 2005.

[2] A. Black, P. Taylor, and R. Caley, "The festival speech synthesis system," 1999, Online: http://festvox.org/festival/.

[3] H. Zen, T. Nose, J. Yamagishi, S. Sako, T. Masuko, A. W Black, and K. Tokuda, "The HMM-based speech synthesis system (HTS) version 2.0," in *The 6th International Workshop on Speech Synthesis*, 2006.

[4] C. Traber, K. Huber, K. Nedir, B. Pfister, E. Keller, and B. Zellner, "From multilingual to polyglot speech synthesis," in *Proceedings of Eurospeech 99*, 1999, pp. 835–838.

[5] P. Taylor, A. W. Black, and R. Caley, "Heterogeneous relation graphs as a formalism for representing linguistic information," *Speech Communication*, vol. 33, no. 1-2, pp. 153–174, 2001.

[6] P. Taylor, A. W. Black, and R. Caley, "The Architecture of the Festival Speech Synthesis System," in *The Third ESCA Workshop in Speech Synthesis*, Jenolan Caves, Australia, 1998, pp. 147–151.

[7] D. Crockford, "RFC 4627: The application/json Media Type for JavaScript Object Notation (JSON)," .

[8] K. Tokuda, H. Zen, and A. W. Black, "An HMM-based speech synthesis system applied to English," in *IEEE Speech Synthesis Workshop*, 2002.

[9] Mark Davis, "Unicode standard annex #29: Unicode text segmentation," Technical Report 29, September 2009, http://www.unicode.org/reports/tr29/.

[10] Mark Davis and Ken Whistler, "Unicode standard annex #44: Unicode character database," Technical Report 44, September 2009, http://www.unicode.org/reports/tr44/.

[11] R. Gillam, "A rule-based approach to number spellout," 1998, Paper presented at the 12th International Unicode/ISO 10646 Conference.

[12] R. Mitten, "Computer-usable version of Oxford Advanced Learner's Dictionary of Current English," Tech. Rep., Oxford Text Archive, 1992.

[13] L. Loots, M. Davel, E. Barnard, and T. Niesler, "Comparing manually-developed and data-driven rules for p2p learning," in *PRASA*, South Africa, Nov 2009, pp. 35–40.

[14] M. Davel and E. Barnard, "Pronunciation prediction with Default&Refine," *Computer Speech and Language*, vol. 22, pp. 374–393, 2008.

[15] T. A. Hall, "English syllabification as the interaction of markedness constraints*," *Studia Linguistica*, vol. 60, no. 1, pp. 1–33, 2006.

[16] Steven Bird, Ewan Klein, and Edward Loper, *Natural Language Processing with Python: Analyzing Text with the Natural Language Toolkit*, O'Reilly, Beijing, 2009.

[17] Erik F. Tjong Kim Sang and Sabine Buchholz, "Introduction to the CoNLL-2000 shared task: Chunking," in *Proceedings of CoNLL-2000 and LLL-2000*, Claire Cardie, Walter Daelemans, Claire Nedellec, and Erik Tjong Kim Sang, Eds. 2000, pp. 127–132, Lisbon, Portugal.

[18] P. Roach, *English phonetics and phonology: a practical course*, Cambridge University Press, third edition, 2000.

[19] K. Silverman, M. Beckman, J. Pitrelli, M. Ostendorf, C. Wightman, P. Price, J. Pierrehumbert, and J. Hirschberg, "ToBI: a standard for labeling English prosody," in *Second International Conference on Spoken Language Processing*, 1992, vol. 2, p. 867870.

[20] L. Breiman, J. Friedman, R. Olshen, and C. Stone, *Classification and Regression Trees*, Wadsworth, Belmont, California, 1984.

[21] Keiichi Tokuda, Shinji Sako, Heiga Zen, Keiichiro Oura, Kazuhiro Nakamura, and Keijiro Saino, "The hts_engine API version 1.02," http://hts-engine.sourceforge.net/.

[22] S. Young, G. Evermann, M. Gales, T. Hain, D. Kershaw, G. Moore, J. Odell, D. Ollason, D. Povey, V. Veltchev, and P. Woodland, *The HTK Book (for HTK Version 3.3)*, Cambridge University Engineering Department, http://htk.eng.cam.ac.uk/, 2005.

[23] D.R. van Niekerk and E. Barnard, "Phonetic alignment for speech synthesis in under-resourced languages," in *INTERSPEECH*, Brighton, UK, 2009, pp. 880–883.

[24] J. S. Garofolo, L. F. Lamel, W. M. Fisher, J. G. Fiscus, D. S. Pallett, and N. L. Dahlgren, *Darpa Timit: Acoustic-phonetic Continuous Speech Corps CD-ROM*, US Dept. of Commerce, National Institute of Standards and Technology, 1993.

[25] T. Toda and K. Tokuda, "A Speech Parameter Generation Algorithm Considering Global Variance for HMM-Based Speech Synthesis," *IEICE Trans. Inf. & Syst.*, vol. E90-D, no. 5, pp. 816–824, 2007.

[26] H. Kawahara, "STRAIGHT, exploitation of the other aspect of VOCODER: perceptually isomorphic decomposition of speech sounds," *Acoustical Science and Technology*, vol. 27, no. 6, pp. 349–353, 2006.

[27] S. J Kim and M. Hahn, "Two-band excitation for HMM-based speech synthesis," *IEICE TRANSACTIONS on Information and Systems*, vol. 90, no. 1, pp. 378–381, 2007.

[28] T Drugman, G Wilfart, and T Dutoit, "A deterministic plus stochastic model of the residual signal for improved parametric speech synthesis," Brighton, UK, Sept., pp. 1463–1466.