# Multilingual TTS System of Nokia Entry for Blizzard 2010

*Bufan Zhang, Jari Alhonen, Yong Guan and Jilei Tian*

Nokia Research Center, Beijing

{ext-bufan.zhang, jari.alhonen, ext-yong.guan,jilei.tian}@nokia.com

## Abstract

In Nokia's blizzard 2010 entry, we built the system with Nokia multilingual text to speech front end system and two high performance HTS backends. This MLTTS front end system describes the design and implementation designed for universal language coverage and a single code execution for them all based on the assumption that there are more features uniting world languages than differentiating them.

**Index Terms**: speech synthesis, multilingual TTS system, HMM based speech synthesis system.

## 1. Introduction

Nokia MLTTS system describes the design and implementation designed for universal language coverage (i.e. can support any language and it's easy to add languages) and a single code execution for them all based on the assumption that there are more features uniting world languages than differentiating them. The system has been tested with 12 languages belonging to five language families and four different writing systems, and the time required for adding new language support is down to 1-2 weeks work from one person, assuming the availability of some resources.

In Nokia's blizzard 2010 entry, we built the system with Nokia multilingual text to speech front end and EMIME project HTS back end and also the GlottHMM system by Helsinki University. Specifically, we use the Mandarin and British English front end in Multilingual TTS system for Blizzard 2010.

## 2. Multilingual TTS front-end

It was decided a very simple method of adding language support to the system would be to just drop a file with a certain name in a certain directory. The common code of the MLTTS system searches for such a language configuration file and finds in its instructions how to deal with text in that language. If the language is very regular, has low levels of morphology and no other complications, it requires very little in the way of configuration:

1. Character set so the language can be identified easier,
2. a lexicon file, containing abbreviations and basic numbers and any irregular pronunciations,
3. number rule that specifies how to expand larger numbers,
4. syllable rule that specifies what abbreviations should be read out letter by letter rather than as words,
5. list of allophones for the polyglot synthesis described later in this paper, and
6. voices trained for the language.

While languages with this high level of regularity are rare, most languages deviate only on subtle ways that can be solved with some additional settings in absolute values or regular expressions. For the more demanding cases like complicated conjugation in languages with high morphological demands the system allows for nearly any passage of code to be bypassed with an embedded scripting language. We chose the Lua scripting language as it has very low complexity and memory footprint [1], such as for Mandarin.

### 2.1. Text segmentation

For any processing of the text it needs to be segmented to smaller units: paragraphs, sentences, words, etc. An alternative here is formatted text such as SSML (Speech Synthesis Markup Language) which may already contain markers for at least some of these units.

Sentence segmentation is somewhat more complicated as different markings are used across languages. These markings are addressed in Unicode report #29 [2], which even lists rules where to break sentences and these rules have been coded into the MLTTS system.

A further challenge is the segmentation into phrases. One simple part-task follows any further markings in the text, such as commas or dashes to make major parts of sentence, but this is inadequate as a complete solution. Generally to find out where further phrase boundaries could be inserted one should know the parts-of-speech of the words in the text as well as the phrase structure of the language in question. This phrase structure can be defined in the language configuration file.

In the lack of very good phrasing it has been shown that even randomly placed silences can increase the naturalness of synthesized speech [3]. Hence as a temporary measure a random phrase-segmentator was included in the system, but the break it inserts was made extremely short and it is not reported to the voice-generating engine so that prosody would not suffer greatly. This has been noted most beneficial in platforms that process the synthesis slightly slower than the speech rate, such as on certain mobile phones, as processing-mandated pausing make long continuous speech seem out-of-place.

Much of the text processing also suggests Unicode normalization to clear up the differences between code points in accented letters, Korean letters, and double or half width letters and other such issues. Several tools are readily available for this task.

### 2.2. Text to phonemes

A main remaining task is to convert the text to a sequence of phonemes. This part is inherently language-specific, which is why our system is trained with data for the phonemization of each language. A common phoneme set similar to IPA but easier representable in computers was chosen. For polyglot synthesis it is important that the phoneme set must be the same across languages, wherefore e.g. SAMPA is not an adequate solution.

The phonemization module uses decision trees along the lines of Yvon [4], with fast language development described by Moberg et al. [5]. This has been strengthened for more generic use with regular expression support for special cases.

Lexicon-based exceptions are also supported to allow for easy additions of new words.

Further, the text should be segmented into syllables for the voice generating engine to take advantage of. We first employed a neural network-based algorithm, which created a high accuracy syllabification scheme that relied on language-specific data [6]. However, due to complexity issues slowing down the processing this was later replaced with an extremely simple universal syllabification scheme.

This so-called "fast syllabification" scheme merely considers consonants and vowels, and in one pass through the text string separates them into probable syllables. The syllable breaking is done at the phoneme level as languages differ greatly in the letter usage, and can be skipped for certain languages where the writing system already indicates syllable boundaries. A syllable is counted as consisting of at most one leading consonant, one vowel or diphthong, and one or two consonants at the end, using a phoneme set similar to IPA.

The fast syllabification is not quite as accurate as the trained scheme, but it uses less than 1% of the processing power needed for the neural network evaluation and the difference in the final speech quality is generally not audible.

### 2.3. Prosody and other features

The prosody module in place is still in its infancy, and only really deals with extremely simple intonation schemes and various break lengths. The intonation scheme looks into clear markers for questions and marks them as rising intonation, as this is common in many languages and in most of those where the intonation doesn't rise, raising the intonation is not an error, only not necessary. Improvements to the prosody module would require a tighter interface to the engines, which would mean less ease in integrating new engines. This does seem a worthy tradeoff, however. As it is, even the prosodic features marked in SSML, such as pitch and contour, are currently ignored. Fortunately HMM synthesis by its very nature is rather forgiving on prosody, and voices trained with a large enough database often sound surprisingly natural even without any prosodic processing.

### 2.4. Numbers and symbols

The expansion of numbers into their fully written-out forms is a rather more language-dependent task, but handled by the language-independent code in our TTS system. The expansion method was described in detail in our earlier paper [7].

## 3. Backend HTS System

EMIME HTS system: In English Hub1, Hub2 and Mandarin Hub2 task, the HTS system is built using the framework from the HTS-2005 system[8], and we built a speaker dependent system Following algorithms and technologies were applied in the speaker-dependent framework:

Speech analysis: A high quality speech vocoding method called STRAIGHT (Speech Transformation and Representation using Adaptive Interpolation of weiGHTed spectrum) [9] was used, in conjunction with the mixed excitation [10].

Training: To simultaneously model the duration for the spectral and excitation components of the model, the MSD hidden semi-Markov model (MSD-HSMM) was applied.

Speech generation: Generating smooth and natural parameter trajectories from HMMs considering the global variance (GV) .

GlottHMM system: In Mandarin Hub1 task, we built the system with the the GlottHMM system by Helsinki University. It has nearly same procedure as in EMIME system expect for the inverse glottal filtering instead of mixed excitation. [11]

## 4. Conclusion

Multilingual high quality TTS is capable of supporting nearly all languages of the world by using a single code executable and only straying from common code execution. The light resources needed to develop both the system and especially the new language support highlights the low cost value of the system. The system has also been optimized for low complexity for mobile devices. Polyglot synthesis further improves the experience of multilingual TTS but should be tuned to the task and ideally to the listeners preferences. These preferences can to a large extent be guessed based on their language abilities.

## 5. Acknowledgement

The research leading to these results was partly funded from the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement 213845 (the EMIME project (http://www.emime.org).

## 6. References

[1] Ierusalimschy, R, de Figueiredo, L.H. and Celes, W., "The implementation of Lua 5.0", Journal of Universal Computer Science 11 #7, 2005.

[2] Davis, M., "Unicode text segmentation", Unicode Standard Annex #29 for Unicode version 5.2.0, Unicode Inc. Online: http://unicode.org/reports/tr29/, accessed on 27 Apr 2010.

[3] Wang, X., Li, A. and Yuan, C., "A Preliminary Study on Silent Pauses in Mandarin Expressive Speech", ISCA, Proc. of Speech Prosody, 2008.

[4] Yvon, F., "Self-learning techniques for grapheme-to-phoneme conversion", Proc. of the 2nd Onomastica Research Colloquium, 1994.

[5] Moberg, M., Pärssinen, K. and Iso-Sipilä, J., "Cross-lingual phoneme mapping for multilingual synthesis systems", Proc. Of International Conference on Spoken Language Processing, pp. 1029-1032, 2004.

[6] Tian, J., "Data-Driven approaches for automatic detection of syllable boundaries", ICSLP, 2004.

[7] Alhonen, J., "Multilingual number expansion for TTS", IEEE, Proc. of Oriental COCOSDA 2009.

[8] H. Zen, T. Toda, M. Nakamura, and K. Tokuda. Details of Nitech HMM-based speech synthesis system for the Blizzard Challenge 2005. IEICE Trans. Inf. & Syst., E90-D(1):325–333, January 2007.

[9] H. Kawahara, I. Masuda-Katsuse, and A. 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, pp. 187–207, 1999.

[10]T. Yoshimura, K. Tokuda, T. Masuko, T. Kobayashi, and T. Kitamura, "Mixed excitation for HMM-based speech synthesis," in Proc. EUROSPEECH 2001, Sep. 2001

[11]T. Raitio. Hidden Markov Model Based Finnish Text-to-Speech System Utilizing Glottal Inverse Filtering. PhD thesis, Helsinki University OF Technology, Faculty of Electronics, Communications and Automation, Department of Signal Processing and Acoustics, Helsinki, 2008.