The Cerevoice Blizzard Entry 2006: A Prototype Small Database Unit Selection Engine

Matthew P. Aylett, Christopher J. Pidcock, Mark E. Fraser

Cereproc Ltd.
Edinburgh, Great Britain
matthewa@cereproc.com

Abstract

Cerevoice®is a unit selection speech synthesis system produced by Cereproc Ltd. The system was used to build small and large unit selection databases using the data supplied by the Blizzard Challenge 2006. The large database system was used as a baseline system while two experimental approaches for improving the quality of the small database system were explored. 1) Synthetically generating diphones from sections of diphones in the database offline and then using them in synthesis, a process we term bulking. 2) Applying limited manual intervention based on negative feedback to improve quality, a process we term second-pass synthesis. Both techniques resulted in the small database system maintaining the quality of the larger system. We conclude that there is much room for improvement in the quality of small database systems using unit selection without the requirement for more data and that second-pass synthesis offers a potential means for training small database unit selection systems.

Index Terms: speech synthesis, unit selection.

1. Introduction

Cerevoice®is a unit selection speech synthesis SDK produced by Cereproc Ltd., a company founded in late 2005 with a focus on creating characterful synthesis and massively increasing the efficiency of unit selection voice creation. The system entered into the Blizzard Challenge 2006 was the result of Cereproc's initial 5 months development work. The system is designed with an open architecture, has a footprint of approximately 70Mb for a 16Khz voice and runs at approximately 10 channels realtime. The system is a diphone based unit selection system with pre-pruning and a Viterbi search for selecting candidates from the database similar to systems described in [1, 2, 3].

The main focus of early development work was to explore the extent to which databases can be reduced and still produce an acceptable quality of synthesis. Two approaches were explored. The first was to examine how far the degrees of freedom could be extended within small databases by using half phones. This differed from approaches such as [4] in that the required diphones were synthesised offline and used to *bulk-up* the small database. The second was to examine to what extent efficient manual intervention could be harnessed to improve synthesis quality, a process we term *second-pass synthesis*. The main objective for our Blizzard entry was to explore these techniques. With this in mind it is important to clearly distinguish the two Cerevoice entries to Blizzard. The full database entry used **no** small database techniques and was completely automatic, whereas the small (Arctic) entry did use these techniques and thus made use of some manual intervention.

1.1. Rationale behind second-pass synthesis

A vast proportion of speech audio currently used in computer applications is in the form of recorded prompts. This alone demonstrates that although fully automated synthesis is required for completely dynamic content, much content is, in fact, not that dynamic at all. Currently, users of speech synthesis have used markup such as SSML [5] to manually control exactly how synthesis is realised. However the format of much of this markup stems from earlier diphone based synthesis systems rather than database approaches. Cerevoice, however, will accept markup which allows users to control the inner working of the selection process. Such manual intervention is an effective stop-gap technique for competing with natural pre-recorded prompts.

In addition, investigating efficient manual methods for improving synthesis addresses a crucial research question; given the database, how good could the synthesis become if our search algorithms produced optimum quality speech? In order to supply synthesis for entertainment there is a requirement for building fast, good quality characterful voices, often within specific domains. It is currently unclear what the degrees of freedom are for minimising the size of collected databases. Previous work which has tried to improve the quality of voices made from small databases has made use of information from a different voice with a larger database, either by using voice-morphing e.g.[6] or the larger voices prosodic model e.g.[7]. In contrast, second-pass synthesis allows us to answer the question of whether critical errors in the synthesis are caused by the poverty of the search algorithm or whether they are caused by database sparsity.

2. Overview of the system

Cerevoice is a new faster-than-realtime diphone unit selection speech synthesis engine, available for academic and commercial use. The core Cerevoice engine is an enhanced synthesis 'back end', written in C for portability to a variety of platforms. The engine does not fit the classical definition of a synthesis back end, as it includes lexicon lookup and letter-to-sound rule modules, see Fig. 1. An XML API defines the input to the engine. The API is based on the principle of a 'spurt' of speech. A spurt is defined as a portion of speech between two pauses.

To simplify the creation of applications based on Cerevoice, the core engine is wrapped in higher level languages such as Python using Swig. For example, a simple Python/Tk GUI was

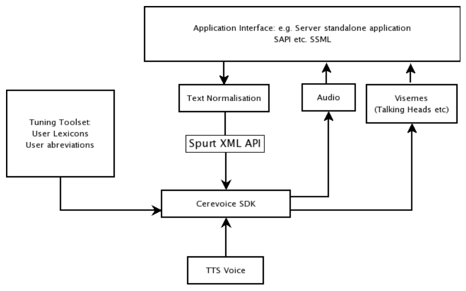

Figure 1: *Overview of the architecture of the Cerevoice synthesis system. A key element in the architecture is the separation of text normalisation from the selection part of the system and the use of an XML API.*

written to generate the test sentences for the Blizzard challenge.

The Cerevoice engine is agnostic about the ’front end’ used to generate spurt XML. Cereproc use a modular Python system for text processing. Spurt generation is carried out using a greedy incremental text normaliser. Spurts are subsequently marked up by reduction and homograph taggers to inform the engine of the correct lexical variant dependent on the spurt context.

3. Processing the Blizzard Data

A preliminary check of the text files provided by Blizzard suggested that most of the text had been pre-normalised. However, there were still problems with data ambiguity in some areas. The word ’corp’s’ is read once as ’corp’s’ and at other times as ’corporation’s’. Upper-case words were neither predictably spelt out as letters, or spoken as words. For example, ’GAP’ and ’LAN’ were read ’G A P’ and ’L A N’, with ’TOEIC’ read as a word. A hand created list of the latter type of word was created, and all other upper case strings split into individual letters. The one exception to this rule was ’II’, which was mapped to ’two’.

3.1. Voice building

Voice building in Cerevoice is a heavily automated, modular, dependency-driven process, consisting of two main types of component: speech parameterisation and segmentation. Speech data, text transcriptions, and a lexicon are the only required inputs to voice building. We decided to use our own pronunciation lexicon and segmenter to build the Blizzard entry. The American English lexicon is a modified version of the CMU pronunciation lexicon. Hence, of the Blizzard data, only audio files and their text transcriptions were used to build the Cerevoice entry. Due to resource constraints, some pronunciations from the full database that were missing from the lexicon were not added. It was decided to leave out 358 input sentences rather than include unchecked automatically generated pronunciations. Many of the missing words were foreign in origin. In fact, leaving out sentences that contain many Spanish, German, or Japanese words may improve the quality of the final system since attempts are often made to render sounds into a foreign phone set (for example, the word ’Kyoto’ in the Blizzard data).

Segmentation is carried out using the HTK hidden markov model toolkit in forced alignment mode. f0 and pitch mark parameters are generated using the ESPS tools ’epochs’ and ’get_f0’. Edinburgh Speech Tools’ ’sig2fv’ is used to generate cepstral parameters, which are used to generate Line Spectral Frequencies.

3.2. Voice Tuning

The system uses both symbolic (e.g. stress, break index information) and parametric target cost functions (e.g. f0 and duration). Transition costs are based on Line Spectral Frequencies, f0 smoothed over voiced plus unvoiced speech, and energy. Target and transition weights are set manually although setting for the Blizzard entry were very similar to values used for Cereproc’s other voices.

Tuning consisted mainly of modifying f0 and energy target parameters to suit the voice. In listening tests, the majority of problems were found to be due to wrong lexicon entries or missing pronunciation variants. Automatically generated segmentations were not modified in any way.

4. Improving Small Footprint Synthesis

Unit selection synthesis systems are often criticised for the large database size required to produce natural sounding speech across many contexts. A reduced database size can cause a sharp drop in the perceived quality of the speech output. The Arctic database portion of the challenge contains only a fifth of the audio data from the full challenge. For this reason, we decided to implement two novel means of improving the speech output: voice bulking and second-pass synthesis.

4.1. Voice Bulking

Voice bulking is the creation of new units from existing non-contiguous, but probably well matched, sections of speech. The result is a new diphone created from two demiphones (half-diphones). For example, taking an ’ao[1]-pause’ diphone from the end of the word ’thaw’ and a ’pause-p’ diphone from the word ’pat’, phone boundaries can be used to cut the diphones into demiphones and create a ao-p diphone.

All demiphones used for bulking were taken from word boundaries. This made it easy to create new utterances containing the bulked diphone, by simply splicing two words together to create a new diphone at the word boundary. All other diphones in the utterance are marked for removal from the database at build time. Fig. 2 shows an example of a bulked ’ao-p’ diphone made from concatenating ’thaw’ to ’pat’.

The flexible short pause model used during segmentation inserts a large number of pauses that can be used for bulking. Other word boundary locations for bulking were also selected, after creating some bulked diphones by hand to test the procedure.

Bulking Locations (either side can be used to create a new diphone):

- any phone - short pause - any phone
- fricative - vowel
- vowel - fricative
- any phone - plosive
- any phone - affricate

[1]We use the CMU style machine readable phone set to describe the phones here. (ao - vowel in ’pour’, aw - vowel in ’cow’, ay - vowel in ’pie’, er1 - stressed vowel in ’her’, er0 - unstressed second vowel in ’after’, oy - vowel in ’boy’, dh - first consonant in ’the’, zh - middle consonant in ’measure’)

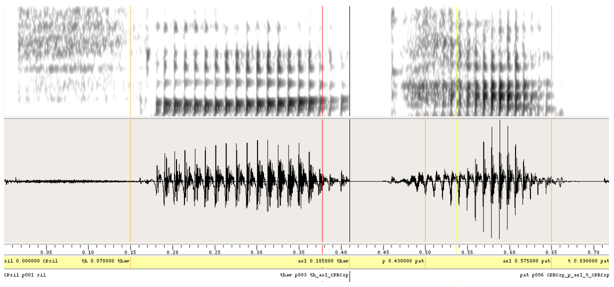

Figure 2: *Example of a bulked diphone constructed by concatenating the word 'thaw' to 'pat' offline. The resulting diphone is then added to the small database to reduce sparsity.*

Bulking was implemented as part of the Cerevoice voice building process. The bulking process added 102 missing diphones to the unit selection database for the Arctic task, raising the total number of diphones from 1320 to 1422. 6 diphones that appeared in the Blizzard test set were taken from this bulked set, 'ao-p', 'aw-ay', 'ax-er', 'd-oy', 'dh-z' and 'zh-p'. For this task, bulking was used only to improve coverage of missing diphones, but it could easily be extended to provide additional coverage of any diphone that could be bulked. Of the bulked units, all sounded natural, with the exception of 'dh-z'. A 'dh-z' diphone in the word 'clothes' contained a perceptual click due to the 'z' containing a portion of a previous vowel.

4.2. Second-pass synthesis

Second-pass synthesis is a post-hoc method of tuning the synthesis output to improve the perceived quality of the output. A Viterbi search is used to find the 'best' sequence of states. In Cerevoice it is possible to ask the engine to prune out a section of the best path found during the Viterbi search and to rerun the Viterbi over that section to find a less optimal alternative or *variant*. The next variant approach can be applied to a whole utterance or, more usefully, focus on a problem word or diphone. In the case of changing a single word or diphone in a larger utterance, units not within the the variant section are 'locked' to prevent modification of units that are considered acceptable. A new variant is selected by running the Viterbi search then pruning out the rejected selection of units. The pruning out of rejected units is cyclical, continuing until the requested variant number is found. Inside an XML spurt, a word can enclosed by a 'usel' tag containing a variant attribute to force this behaviour. For example <usel variant='0'> is equivalent to no tag, and <usel variant='6'> would be the sixth alternative according to the Viterbi search. Fig. 3 shows a schematic of this process.

In addition to adding variant tags to the XML, operators were also permitted to change lexical reduction. We altered reduction because our reduction rules are not sensitive to database size. For a large database this is acceptable but for smaller databases it is more important to reflect the speaker's produced reduction more closely.

It took approximately 1.5 hours for three operators to manually alter the synthesis for the small database entry. Alternative variants were selected only for words which were regarded as having a critical error and reduction was altered only when a clear concatenation error occurred in a reduced or reducible form.

Below is an example from the Blizzard Arctic database test

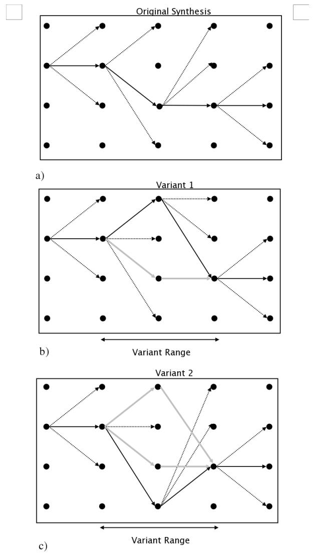

Figure 3: *Schematic diagram of the Cerevoice variant tag process. a) The best path chosen by the Viterbi is shown as a black line. b) The unit in row 3 column 3 is rejected and the variant tag requests the next alternative. The path going through the unit is pruned out and a second path marked in black is selected. c) The new unit at row 1 column 3 is also rejected, the process is run again, a final acceptable unit at row 4 column 3 is selected.*

marked up with variant tags.

```
The <usel variant='2'>Fruitto</usel> de
Mare featured, calamari served with <usel
variant='1'>tomatoes</usel>, peppers,
artichoke, avocado and, again, frisee.
```

It is interesting to note that a considerable improvement in the speech output can be made by massaging the output in this way. This suggests that considerable improvement could still be made to small-footprint unit selection engines, as it appears that although the 'right' data is there, it is not always selected.

5. Results

From a commercial point of view results from naive listeners are of particular interest to Cereproc. It is the reaction of these listeners to speech applications that will generally lead to commercial success or failure of a speech application. For this reason we will concentrate on the results from only the naive undergraduate group of listeners.

5.1. MOS scores

Using MOS (mean opinion score) scores to compare systems is difficult because of the large variation in results by both subjects and materials. However, even with the lack of confidence that results would remain consistent between subject groups or between trials, trends in the MOS scores should be taken very seriously. The question, after all, "How natural do you think it is?" is one of the two most important questions we should ask about synthetic speech. (The other is "Did you understand it?" see section 5.2).

Fig. 4a shows the results in ranked order. 'O' is the natural

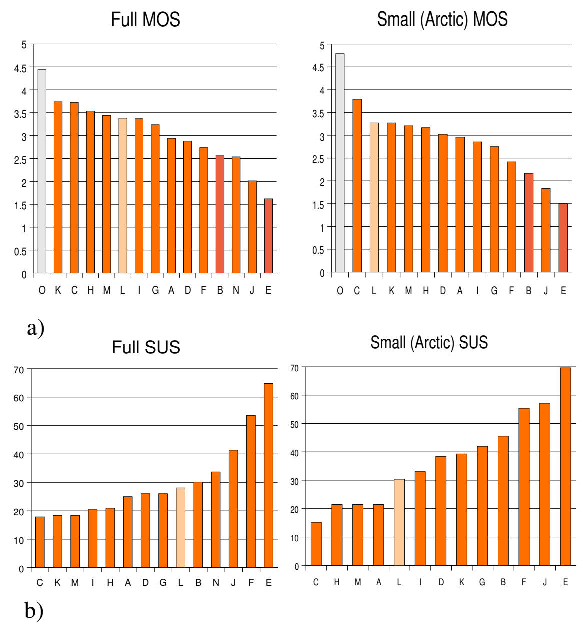

Figure 4: *a) Results for MOS scores across all systems. 'O' is the natural speaker, 'L' is the Cerevoice system. variance in MOS for most systems is approximately 1.0. b) Results for word error rate (WER) scores in the semantically unpredictable sentence (SUS) test across all systems. 'L' is the Cerevoice system. variance in the SUS WER scores varies more between systems, from 14 to 30.*

system and 'L' is the Cerevoice system. Firstly we should all be sobered by the clear difference between the natural speaker and all our systems. Even taking into account a 0.35 change between small and full systems in the naive subject group's judgement of the natural speaker, this shows there is a lot of work to do to improve text to speech synthesis.

The results also suggest that the strategies we used to improve the small database system worked. The system maintained its quality between the two databases while many other systems saw their quality degraded.

Given the fact that only critical errors were manually altered for the small synthesis system it also suggests that it is these critical errors which have the most fundamental effect on perceived naturalness.

5.2. WER scores

The word error rate (WER) is a second important measure of the quality of a synthesis system. As with MOS evaluation, intelligibility is hard to measure accurately. Factors such as short term memory, the subjects' typing speed, and word predictability have a major effect on intelligibility results. Typically words from normal contexts are too easy to recognise. For this reason various artificial neutral contexts are used, such as the modified rhyme test (where a word is held in a carrier phrase) and the SUS (semantically unpredictable sentence) test, where words are strung together into nonsense sentences.

Unfortunately, results for the intelligibility of the native speaker were not available for comparison with the synthesis systems. It was felt that collating results for the MRT and SUS tests was unwise given the very different nature of the tests. For this reason we include results for the SUS test only. This is arguably the harder test and reflects more closely the problem of recognising a word within a normal sentence. However, because the test is harder for the subjects to take there is also considerable variation.

Fig. 4b shows the WERs for the SUS text for the undergraduate (naive group) only. As with the MOS scores the small database techniques appear to have been successful. The WER of the full system is maintained. The Cerevoice system again moves up the ranking not because it does better but because several other systems show a considerable degradation in their WER.

6. Conclusion

Given the early stage of Cerevoice development at the competition deadline the overall results are positive. Cerevoice has competed effectively at an international level after a very short time in development and we expect the release system to do significantly better in the next competition. We have also demonstrated that small database techniques can make a significant impact on maintaining the synthesis quality of unit selection systems built with less data. Over the next year we expect to make use of bulking and second-pass synthesis both for competing with naturally recorded prompts but also to develop completely automatic means of improving the quality of small database unit selection systems.

7. References

[1] A.J. Hunt and A.W. Black, "Unit selection in concatanative speech synthesis using a large speech database," in *ICASSP*, 1996, vol. 1, pp. 192–252.

[2] Robert A.J. Clark, Korin Richmond, and Simon King, "Festival 2 - build your own general purpose unit selection speech synthesiser," in *5th ESCA Workshop in Speech Synthesis*, 2004, pp. 147–151.

[3] John Kominek, Christine L. Bennet, Brian Langer, and Arthur R. Toth, "The Blizzard challenge 2005 CMU entry - a method for improving speech synthesis systems," in *INTERSPEECH*, 2005, pp. 85–88.

[4] M. Beutnagel, A. Conkie, J. Schroeter, Y. Stylianou, and A. Syrdal, "The AT&T next-gen TTS system," in *Joint Meeting of ASA, EAA, and DAGA*, 1999.

[5] Paul Taylor and Amy Isard, "SSML: A speech synthesis markup language," *Speech Communication*, vol. 21, pp. 123–133, 1997.

[6] Wael Hamza, Raimo Bakis, Zhi Wei Shuang, and Heiga Zen, "On building a concatenative speech synthesis system from the blizzard challenge speech databases," in *INTERSPEECH*, 2005, pp. 97–100.

[7] M.P Aylett, J. Fackrell, and P. Rutten, "My voice your prosody: Sharing a speaker specific prosody model across speakers in unitselection TTS," in *Proceedings Eurospeech 03*, 2003.