The CereProc Blizzard Entry 2009: Some dumb algorithms that don't work

Matthew P. Aylett[12]*, Christoper J. Pidcock*[2]

[1] CSTR, University of Edinburgh, UK
[2]CereProc Ltd, Edinburgh, UK
matthewa@cereproc.com

Abstract

Within unit selection systems there is a constant tension between data sparsity and quality. This limits the control possible in a unit selection system. The RP data used in Blizzard this year and last year is expressive and spoken in a spirited manner. Last years entry focused on maintaining expressiveness, this year we focused on two simple algorithms to restrain and control this prosodic variation. 1) Variable width valley floor pruning on duration and pitch (Applied to the full database entry EH1), 2) Bulking of data with average HTS data (Applied to small database entry EH2). Results for both techniques were disappointing. The full database system achieved an MOS of around 2 (compared to 4 for a similar system attempting to emphasise variation in 2008), while the small database entry achieved an MOS of also 2 (compared to 3 for a similar system, but with a difference voice, entered in 2007).
Index Terms: speech synthesis, unit selection.

1. Introduction

Dumb algorithms that work well are valuable. The simpler an algorithm the easier it is to understand, implement and debug. Arguably, dumb algorithms that don't work are too numerous to be of interest. In addition care must be taken in deciding the scope of what we mean by 'not working'. However, the Blizzard Challenge gives us an ideal opportunity to agree on a definition of *working*. A high naturalness (mean opinion score) MOS score is good, a low (word error rate) WER, is good. We are also able to compare systems directly with each other based on the same input data,and same evaluation. Given the limited resources CereProc®was able to apply to the Blizzard Challenge this year, and the desire to use Blizzard as an opportunity to investigate new approaches to synthesis, we decided to implement two simple algorithms over a short time span. Unfortunately, the results were not successful in terms of the results produced by the Blizzard evaluation. Within this paper we will describe these *dumb* algorithms, present some additional results, and discuss the problems in implementation and the reasons for their lack of success.

CereVoice®is a unit selection speech synthesis SDK produced by CereProc Ltd.[1], a company founded in late 2005 with a focus on creating characterful synthesis and massively increasing the efficiency of unit selection voice creation. CereProc enjoys a close relationship with the Centre of Speech Technology Research (CSTR) at Edinburgh University, and the CereVoice system is also made available for research use. CereVoice now also includes an HTS style implementation using STRAIGHT vocoding which allows a mix and match approach across these synthesis techniques.

The data used for the 2008 and 2009 Blizzard Challenge is prosodically rich. CereProc's 2008 entry focused on controlling and enhancing this richness in synthesis, this year's entry focused on controlling and removing this variation.

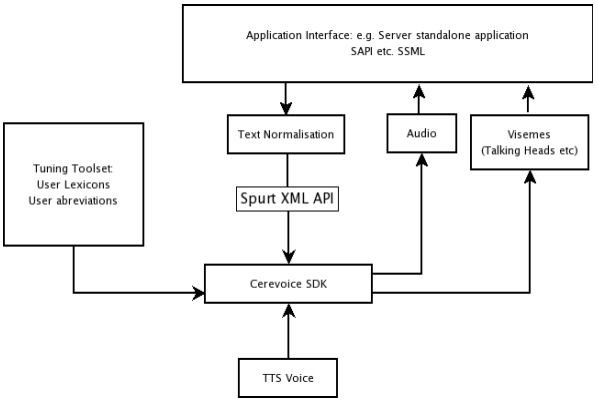

Figure 1: *Overview of the architecture of the CereVoice synthesis system. A key element in the architecture is the separation of text normalisation from the selection part of the system and the use of an XML API.*

2. Overview of the system

CereVoice is a faster-than-real-time 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 used 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.

In 2008 we added a parametric synthesis system based on the HTS2005[2] and HTS2007[3] systems. Voices can be generated using either a speaker specific approach or a voice adaptation technique. The same front end and feature extraction is used for both

10.21437/Blizzard.2009-4

systems and a voice can be switched between unit selection and parametric synthesis on a phrase by phrase basis.

3. The Database

The English (RP) database selected for the Blizzard Challenge contained a variety of speaking styles and acoustic properties that allowed for more expressive speech synthesis [4] [5], but our previous voices built with this data revealed that it can also cause considerable problems with inappropriate prosody and concatenation artifacts when different parts of the database are used within the same synthetic utterance. Last year this problem in concatenation was found to occur frequently in stressed syllables when trying to concatenate material from the the novel genre (Alice in Wonderland), with news (Herald and Post), and between emphasis data and either of these two genres. Thus in this years entry we limited the data to a sub set of relatively homogeneous news genres (Herald 2 and Herald 3).

The Arctic subset of the database, used for the small database voice, is in contrast more consistent with a fairly stable speaking style. However, results from previous Blizzard challenges have shown it possesses insufficient phonetic and prosodic coverage for a conventional unit selection system.

Our entry this year was severely limited by resources, with a team of two working for two days only on implementation, testing and evaluation. For this reason we focused on applying two simple algorithms with the objective of tightly controlling prosodic variation in the large EH1 voice, and bulking data with HTS generated waveforms to counter the sparsity issues in the EH2 voice.

4. Valley Floor Pre-Pruning

4.1. Introduction

Within unit selection, it is common to use an pitch (f0) target to guide the selection of units against an idealised pitch track generated for the sentence using a prosodic model. Unlike a classic diphone system, or a parametric system, this f0 target acts as a guideline only. Altering the target does not guarantee a change in the output. This is because the f0 target is one of many cost functions used in the unit selection search. In general, f0 models in unit selection are far from ideal, given that natural prosody resides in the database, it is common practise to give the f0 target a low cost so that errors in the model do not cause commensurate errors in the synthesised output.

The concept of valley floor pre-pruning is to:

1. Regard the target as a guideline only
2. Use the target to to strip away units that are far from the target before the search, without having a catastrophic impact on the sparsity of units.
3. Pass remaining units through to the Viterbi search, allowing, as with the conventional system, natural database prosody to overcome inappropriate f0 modelling.

4.2. Implementation

F0 values are calculated for all units (even if unvoiced), by interpolating across the initial utterances. For each diphone type percentile statistics are calculated giving the f0 values at the 95th, 80th, 65th, 50th, 35th, 20th, and 5th percentile. During synthesis a floor on the number of units to pass to the Viterbi search is set (in this system 50 units). The pre-pruning takes the f0 targets for each target diphone. It then finds the percentile covering this value (or the top or bottom percentile if the target is out of range). If a sufficient number of units is present in this percentile all other units are pruned out, otherwise the range is increased by a percentile on each side. The range is increase until enough units are found to pass to the Viterbi search stage. Thus if the diphone is rare all units may be passed, if it is very common only a small number close to the f0 target will be passed.

For example take the penultimate diphone of the word "still" $/i - l/$. In our EH1 voice we had 300 examples of this diphone. The f0 percentile statistics for the mid point of each half of this diphone type were as follows:

Percentile	Half phone /i-/ f0 Hz	Half phone /-l/ f0 Hz
p05	96	87
p20	107	103
p35	113	120
p50	120	115
p65	126	120
p80	137	163
p95	163	156

If the f0 target for the first half phone is 140Hz, this falls between the 80th and 95th percentile. There are 45 units within this range, which is less than the required 50. So the range is broadened to include units above the 65th percentile and above the 95th percentile. On this basis the minimum f0 value that will be passed by pre-pruning is 126Hz. On the same basis the minimum value for the second half phone will be 120Hz. As both targets are close to the top of the distribution no maximum limit would be set.

If the f0 targets are wildly different for each half of the diphone, this could result in maximum and minimum values that allowed no units to be passed. In this case the system back off passes units irrespective of f0 values.

4.3. Results

In terms of naturalness, the results were catastrophic. MOS dropped from 4, for a similar system without this approach entered in 2008, to 2. Intelligibility remained similar at around 35-38% WER. Possible reasons for the sharp drop in naturalness are:

More Joins The valley floor pruning raised the total number of non-contiguous joins in the test sentences from 50% to 58% of possible diphone boundaries.

Effect of Inappropriate F0 Target Although the overall variation in f0 was reduced as intended (SD 24Hz for valley floor voice, SD 26Hz for baseline voice. This reduction in variation was not consistent. At the start of phrases the variation was higher for the valley floor voice (SD 29Hz) than the baseline voice (SD 25Hz). In addition the f0 mean at the start of phrases was much higher (133Hz) than the baseline voice (124Hz) which allowed prosody to vary more freely. This suggests the model was requesting an overly high f0 for phrase initial contexts and the pruning was enforcing it when possible, causing inappropriate prosody and more joins in the synthesis.

5. Dumb HTS Merging

Taylor describes [6], and Pollet and Breen [7] demonstrate, a system where unit selection units are merged during synthesis with parametric style units. Thus despite phase mismatch, and potentially contrasting voice qualities, it is possible to acceptably concatenate HTS and unit selection data. Given the primary problem (for unit selection) in the small database entry is data sparsity, an obvious and simple solution might be to bulk out the data using HTS synthesis *before* voice building. This approach is attractive because it requires no fundamental change to the unit selection system. In addition, if text close to the original database is entered we can bias the system to prefer original units, and thus produce some output with a superior audio quality as well as with the more natural prosodic features of a unit selection system.

This is in contrast with approaches described in [6, 7], where generation of HTS segments is carried out at run-time, allowing for the characteristics of surrounding conventional units to be used during generation. As such it is a *dumb* algorithm, as we may cause concatenation errors between HTS units and more serious concatenation errors between HTS units and unit selection units unnecessarily. However, it is extremely easy to implement. In effect any HTS system can be used to generate extra material, which is then added to the unit selection database.

We used the CereVoice implementation of HTS to generate the data for bulking. This parametric system used a speaker specific model very similar to the HTS2005 entry to the Blizzard Challenge[2], but with the addition of global variance statistics to alter parameter generation. The feature set was also slightly different, very closely matching the feature set used in the CereVoice unit selection system.

The HTS voice was built only on the Arctic subset of the database. The text from a script created to produce good phonetic coverage for our larger voices was then used to generate more speech data. This data was then added to the Arctic subset and a new unit selection voice was created.

During synthesis, a genre feature was used to bias the selection of data to the original data, using HTS generated units only when data sparsity meant there were insufficient original units (less then 50).

Figure 2 shows an example of a section of speech where units from the original database and from HTS generated speech were concatenated together.

In one respect the dumb merged system was successful. A similar system (although using a different database) achieved a WER of around 40%. The mixed system in comparison managed a much more successful WER of around 24%. However, subjects did not appear to think it sounded natural (small voice MOS, 2007, around 3, mixed voice MOS, 2009, around 2), perhaps feeling that switching between the voice qualities was less natural than maintaining a consistent parametric speech quality. In addition concatenation errors were still present. Thus, our mixed system offered no advantage over a traditional HTS system - it didn't sound more natural, and it wasn't more intelligible.

6. Conclusion

The concatenative synthesis approach faces two main challenges: data sparsity, and joining non-homogeneous speech units. The algorithms described in this paper attempted to address these two issues in different ways, but did not perform well. It is difficult to be certain what the main cause of this failure was without a systematic analysis. It is possible that both approaches may have performed well if implemented in a different way. However, there are some key problems with both the valley floor pruning and the HTS merging which may mean neither approaches will be effective without significant alterations to the algorithms.

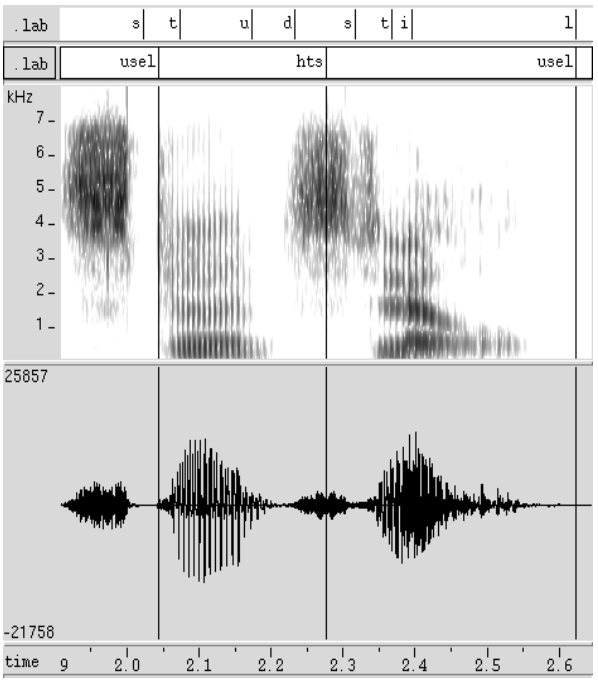

Figure 2: *Example of speech synthesised by merging HTS output with original data in the speech "stood still". The /s t/ is taken from original data, the nucleus of "stood" and onset of "still" using HTS generated data. Note also a poor concatenation in the nucleus of "still" between original database units. The more clearly repeating glottal pulses are visible in "stood" compared to the original vowel material in "still".*

6.1. The Problem with Pruning

Although the valley floor approach can be viewed as a soft pruning approach, in that only units very far in terms of f0 from the target units are removed, it suffers from the same central problem of all pre-pruning: units which may be optimal can be removed.

Given the increase in the number of joins in the synthesised data (which will inevitably increase the probability of a concatenation error), this does appear to have happened in our implementation.

Although the f0 target is not strictly adhered to, it does have a much bigger impact than in the conventional system. In our case, poor targets at the start and end of the phrase may have decreased the quality by choosing diphones from unusual and inappropriate contexts.

It would be interesting to generate the f0 model jointly with the percentile statistics that are being used to select the pruning thresholds. For example, by mapping Hertz values onto percentile ranges and building a statistical model with this morphed data.

6.2. The Problem with Merging

Figure 2 shows that the merging algorithm we used did not fulfil its purpose. The vowel/liquid sequence at the end of "stood still" shows a noticeable concatenation error. This is not surprising given the sparsity of phrase finality and the problem of concatenating vowels with liquid contexts in a small database unit selection system. However, it is precisely the sort of context that HTS excels at, producing smooth transitions. Why did we not use HTS generated data in this context?

There were 89 examples of the /i-l/ diphone in the Arctic data set. Given the HTS data is only used when the number of units falls below 50, few HTS units are likely to have been passed by pre-pruning. However in phrase final position there are only six examples. Thus the system has removed HTS units in a context where they are more likely to be required.

A second potential problem is the that the unit selection system biases selection to contiguous contexts. This is a very sensible approach with standard units. However the result of this is to select HTS units in longer sequences, this in turn may make the voice quality differences more noticeable, and potentially contributes to the lower naturalness score.

In contrast, two problems which we may have expected to occur did not seem to emerge. There did not appear to be any specific problems concatenating from HTS units to standard units. This is partly due to a preference to concatenate in unvoiced regions. The fact the HTS data was presynthesised, meaning it could not have the dynamic coverage of a run-time system, did not seem to mean coverage was insufficient. Both these points are supported by the intelligibility results, which although poor compared to standard HTS, were better than a standard unit selection small database system.

Overall we found the process of following these simple algorithms through to their conclusion in the Blizzard evaluation a useful process. Finding out what dumb algorithms don't work is an important step on finding out which ones do, and how a dumb algorithm could be made just a little bit smarter.

7. References

[1] Matthew P. Aylett and Christopher J. Pidcock, "The cerevoice characterful speech synthesiser sdk," in *AISB*, 2007, pp. 174–8.

[2] Heiga Zen and Tomoki Toda, "An overview of nitech HMM-based speech synthesis system for the blizzard challenge 2005," in *Interspeech*, 2005, pp. 93–96.

[3] Heiga Zen, Takashi Nose, Junichi Yamagishi, Shinji Sako, Takashi Masuko, Alan Black, and Keiichi Tokuda, "The HMM-based speech synthesis system (HTS) version 2.0," in *Proc. 6th ISCA Workshop on Speech Synthesis (SSW-6)*, Aug. 2007.

[4] Volker Strom, Robert Clark, and Simon King, "Expressive prosody for unit-selection speech synthesis," in *Interspeech*, Pittsburgh, U.S.A, 2006.

[5] Volker Strom, Ani Nenkova, Robert Clark, Yolanda Vazquez-Alvarez, Jason Brenier, Simon King, and Dan Jurafsky, "Modelling prominence and emphasis improves unit-selection synthesis," in *Interspeech*, Antwerp, Belgium, 2007.

[6] Paul Taylor, "Unifying unit selection and hidden markov model speech synthesis," in *Interspeech*, 2006, pp. 1758–61.

[7] Vincent Pollet and Andrew Breen, "Synthesis by generation and concatenation of multiform segments," in *Interspeech*, 2008, pp. 1825–8.