The Blizzard Challenge 2005 CMU Entry
a method for improving speech synthesis systems

John Kominek, Christina Bennett, Brian Langner, Arthur Toth

Language Technologies Institute
Carnegie Mellon University, USA
{jkominek, cbennett, blangner, atoth}@cs.cmu.edu

Abstract

In CMU's Blizzard Challenge 2005 entry we investigated twelve ideas for improving Festival-based unit selection voices. We tracked progress by adopting a 3-tiered strategy in which candidate ideas must pass through three stages of listening tests to warrant inclusion in the final build. This allowed us to evaluate ideas consistently without us having large human resources at our disposal, and thereby improve upon our baseline system within a short amount of time.

1. Introduction

"Blizzard Challenge 2005" was conceived to fill a gap in the speech synthesis research community: namely, the need for a uniform mechanism for measuring the utility of competing techniques that may be applied to the problem of synthesis [1]. This need is met by providing CMU ARCTIC single speaker databases as a common speech corpus [2], by procuring a reasonably large base of subjects for listening evaluations of different systems, and by undertaking analysis of the results.

This paper describes the approach used for the CMU entry, divulging both successes and failures. Our focus here lies not in details of our system, but in how we attempted to improve it, our overall *approach*. We present a methodology that we found useful and hope is broad enough in scope that it may be adopted by other teams in future editions of the Blizzard Challenge. This is crucial because the circumstance we faced is endemic to many teams.

2. Problem and Proposed Solution

2.1. Problem Statement

The problem: how can a small team with limited resources improve a complex system (a speech synthesizer), operating under a tight deadline, when improvement is notoriously difficult to measure? To this one might also add: and when changes to the system take considerable effort to plan, execute, and evaluate.

In our case "small" means four students able to tackle the Challenge in a part-time capacity, plus one faculty member present in an advisory role – hence "limited resources."

2.2. Proposed Methodology

There are three major elements to the CMU approach: a research agenda, a testing strategy, and diagnostic tools. The crux of the research agenda consisted of a ranked list of candidate ideas allotted to different team members. Building an improved set of diagnostic tools received the least attention (unfortunately), amounting primarily to a more complete web server for conducting listening experiments. This supported our requirement for a lightweight approach to testing and evaluation.

a) Research agenda
- Identify constraints to minimize up-front effort.
- Assess past published work and current projects.
- Establish a baseline voice for point of comparison.
- Brainstorm a things-to-try list; divide up.
- One person integrates successes into the final entry**.**

b) **Testing strategy:** "propagate up 20"
- Individual implements a change and tests on a set of 20 sentences (using oneself as the test subject).
- If promising, put 20 A-B examples on web page and request other members to listen. If it is a statistically significant result, keep and define a new baseline.
- Finally, promote 20 test sentences to a larger pool of speech researchers for more reliable testing.
- Elements that pass the third stage make it to the official submission.

c) Diagnostic tools
- web-based server for sets of listening tests.
- unit selection tracing and waveform analysis

The form of our testing strategy derives from the challenging nature of human listening tests and is intended to balance speed with accuracy. Individual developers can test numerous small tasks quickly, but are not as reliable a judge of quality as someone working independently. (Practitioners of speech recognition, compression, and noise removal do not face such difficulties.) We used 20 in-domain (Gutenberg) sentences as our test set. As a rule of thumb, a score of 12-8 is desired to promote a candidate idea to the next level of evaluation. Validation by the development team promotes a candidate to the third level: testing on a larger group of colleagues. Ideally, only those ideas that pass through the three gates of evaluation are allowed into the final submission.

The baseline system is a standard Festvox build, a process well documented elsewhere [3]. It is comparable to the prebuilt voices released with the CMU ARCTIC databases.

2.3. Constraints

Our first decision resided in choosing Festival for our synthesizer framework, as opposed to adopting something unfamiliar or in developing an entirely new substrate. Other constraints derive from considerations of available time.

- Use the Festival "clunits" unit selection synthesizer, along with the EST speech tools.
- Employ only automatic techniques, i.e. no hand correction of the databases.
- Optimize one voice (arctic_rms) and then apply the parameter settings to the remaining three voices.
- Tune for in-domain sentences only. Let the other domains fall where they may.

We eschewed hand correction of the Arctic databases – even though this is mandatory in commercial-grade work – on the grounds that algorithmic solutions offer greater long range impact. This, plus a desire to explore as many design options as possible, acted to frame our endeavors.

3. Development

3.1. Candidate Ideas

Once we had established our working constraints we faced a basic choice of whether to be conservative or adventurous. In a conservative approach one would apply a list of known good techniques to the baseline system, then carefully tune the combination of parameters. Instead we opted to be more adventurous – exploring, in the time available, as many plausible but untested ideas as possible. The Challenge offered an appealing opportunity to explore a large chunk of design space, striving to answer what does and does not work.

Here is the list of ideas we set out to explore, with brief explanation. They are ordered from most promising on down, as we initially assessed it.

1. **Improved pitchmarks**. Pitch synchronous mel frequency cepstrum coefficients lie at the foundation of Festival's *clunits* clustering algorithm, and are used at runtime when computing unit join costs. In the baseline voice, pitchmarks are derived from the audio channel using the EST program *pda*. All the Arctic databases used in the Blizzard Challenge, however, are recorded with a second EGG channel, and it is a shame not to make use of this. Task number one, then, was to derive a second set of pitchmarks using the EGG as the primary source, adjusting each according to details of the audio channel.
2. **Improved label boundaries**. The baseline voice uses labels generated from the forced-alignment mode of Sphinx-2 acoustic model training. We've reported prior success in taking the average label set over multiple model runs [4] and thus it is worth applying again. Also, the current version of CMU's Sphinx engine is 3.5, so it is natural to want to use the latest version.
3. **Durational outlier removal**. According to the results of [5], removing unusually short or long units (durational outliers) from the database serves to purify the phone classes. There is also evidence that it indirectly detects and removes labels having the wrong identity.
4. **Corrected label identities**. The labels provided with Arctic databases are derived from CMU-Dict and hence represent standard American English. Fortunately, the databases selected for Blizzard were deliberately those of native Americans. Still, it remains that the transcript is predicted from the dictionary, and will not specify the actual acoustic realization in all cases. Correcting discrepancies helps the synthesizer avoid pronunciation errors.
5. **Power normalization**. Wavefiles in the Arctic database are normalized to a reference volume on a per-utterance basis. This doesn't guarantee that the sequence of units selected will have a natural power curve. Tighter control is possible by normalizing the selected segments, as part of the tail-end generation module.
6. **Join cost tuning**. The join cost function is an equation that predicts how natural one unit will sound concatenated after another. The default cost function doesn't accurately mimic human perception, however, and contains some rather *ad hoc* elements. A thorough re-engineering of the acoustic cost function is a huge task, but we did explore a set of heuristics that could plausibly improve synthesis.
7. **Unit cluster tuning**. A unit cluster is a phone class subset (allophonic variation) that is acoustically self-similar and can be predicted by surrounding phonetic context. The purpose is to constrain phone candidates to only those that are appropriate for the local context of words spoken, making the synthesizer more precise. Clusters thus provide a target cost, and serve to protect against flawed join cost functions. Variables open to optimization include cluster size, number and type of training features, and algorithmic search control.
8. **Unit name binding**. Name binding is a way of specifying unit cluster constraints directly from the surrounding phonetic context, without using acoustics (above, item 7). This technique was pioneered with success in limited domain voices [6], where phones are grouped according to the containing word. A general purpose synthesizer requires looser constraints. We explored aspects such as syllable position and consonant/vowel contexts.
9. **Better unit set**. The question of optimal unit size is an open issue. There is evidence that disyllables are superior to phonemes [7], but the complexity of changing this aspect led us to retain phoneme units at the foundation.
10. **Pronunciation selection**. One aspect of an individual's idiolect is their variant pronunciation patterns. The work of [8] attempts to predict variants of common function words such as *for*. Appropriate use of the reduced versus full word form can result in more natural synthesis. Conversely, using the reduced version /fer/ is sometimes bad, e.g. in a SUS test a reduced rendering of the word pair "in for" is confusable with the word "infer".
11. **Feature vector definition**. By default, the acoustic features are 12 dimensional melcep vectors. Including energy, Fo, and delta terms are typical alternatives. We explored their impact on concatenation join cost.

3.2. Idea Evaluation

Table 1 concisely encapsulates Team CMU's entire Blizzard Challenge effort. Rows refer to particular experiments; the middle columns follow an idea's progress through the testing ladder; and the rightmost column offers a *does it help* assessment. Some cases are clear, some unclear, and some never made it out of planning.

	Testing Level			Does it
Idea	Self	Team	Group	help?
1	67.5	62.8	bye	yes
2	40			no
3	maybe			maybe
4	fail			no
5	60	55	53.75	slight
6a	37.5			no
6b	47.5			no
6c	60	bye	47.5	maybe
7	70	bye	47.5	maybe
8	55	bye	37.5	no
9	skip			—
10	mixed	48.7		unclear
11	fail			no

Table 1. Testing procedure applied to the ranked list of candidate ideas. Highlighting indicates the stages an idea has successfully passed. Numbers reveal the preference ratio of a *new* voice versus *old*. Note that these numbers are relative to a moving baseline system, but are comparable along a single row.

The numbers in Table 1 depict preference ratios of the candidate voice over a moving baseline. During development the baseline gradually changed as improvements were incorporated. This will be clarified below in section 3.7, which discusses our build sequence.

Admittedly, we did not follow our own protocol to the letter, as revealed by the presences of testing *byes*. This was due to time constraints compounded by the delay in accumulating listening results. Plus, when seeking subjects, it is all too easy to lose customers, inducing a reaction of "oh no, not you again!". Thus, we saved the group tests for near the end of the development timeline.

3.3. Pronunciation Selection (details)

Before variant pronunciations can be predicted they must first be identified. We do this through iterative Sphinx-2 acoustic modeling. The process is initiated using the open source "6k" speaker-independent models. These are used to provide an initial pronunciation for each target function word. Using this initial transcript we build speaker-dependent models. These are run in forced alignment mode with all variant pronunciations permitted, the output of which yields a new transcript for another round of modeling. This continues until convergence.

With transcript in hand, a set of CART trees is trained to predict reduced forms of the words purely from the input text. Experiments in [8] report numbers equal to the performance of human prediction, equating to a major improvement over a naive choose-the-most-frequent predictor.

3.4. Unit Join Costs (details)

Given a sequence of candidate units for each utterance phone, Festival uses a Viterbi search to find the minimal cost path. Two consecutive units from a contiguous section of speech are assigned a join cost of zero. Otherwise, if a unit has a target cost c_{targ} and a join cost with the previous unit of c_{join} then $score = c_{targ} + w_{con} c_{join}$ where w_{con} provides a continuity weight. The target cost is the distance between a unit and the centroid of the cluster to which it belongs. The join cost measures the difference between adjacent frames and has three terms:

$$c_{join} = c_{pen} + w_{Fo} \times \Delta F_0 + \sum w_{cep,i}(x_i - y_i) \qquad (1)$$

where delta F_0 is the change in pitch at the join point and w_{Fo} is the pitch weighting term, **x** and **y** are the neighboring cepstral vectors at the join point and w_{cep} is a vector weighting the cepstral dimensions. The constant c_{pen} is a non-continuity penalty value.

Observe that these equations treat all phonemes equally, even though the human ear is probably more sensitive to joins in fast changing sonorants than in other phoneme classes such as fricatives. To explore this possibility we introduced a heuristic two-valued phone coupling weighting w_{coup} which has value 0 for joins we want to encourage and infinity for joins we want to prevent. The infinity condition could assume one of four cases.

1. joins not occurring at a silence.
2. joins not occurring at a stop consonant.
3. joins not occurring at a fricative.
4. joins not occurring at all of the above.

The net effect is to encourage joins at silences, stops, and fricatives individually, or at all three together.

3.5. Experimental Findings

1. **Pitchmarks** (idea 1). An improved set of pitch-marks derived from the EGG tracks was our first clear success. On our 20 sentence test set the developer reported a 67.5% preference (p=.002). When promoted to the full development team this result was confirmed with a 62.8% preference rate (p<<.001). We were sufficiently confident about it that we didn't bother to submit the change to stage 3 of the testing ladder. The new pitchmarks were adopted for all future integration tests.
2. **Label boundaries** (idea 2). Consistent with the advice of [4], when training Sphinx-2 models we set the number of tied triphone states to 3000. We expected the resulting labels to be good but not optimal, and so tried various things to improve them. None of the attempts offered anything significant.

   Attempts included a) averaging over multiple training runs where the number of senomes is varied, b) labeling from across multiple Arctic database and averaging various combinations of sets, and c) adding a time offset to compensate for the analysis window width (i.e. placing the boundary at the beginning, center, or end of the Hamming window, and places in between).

   As for Sphinx-3, we were unable to get it to work satisfactorily before the deadline. This was due to a couple of problems – one in the alignment program itself, and a second triggered by an undocumented change in the input file specification. These problems manifested by failing to produce labels for 40% of the utterances, and performing unreliably on the remainder.
3. **Durational Outliers** (idea 3). In this procedure, phones are excluded from the unit catalog if their purported duration is unusually long or unusually short (zscore > 3). Because of the encouraging results published in [5], we had another team member reimplement the technique for independent verification. The verdict was that it "probably helps but the effect is not large". Without a strong verdict we moved onto other agenda items.
4. **Label Identity** (idea 4). For this problem again we tried iterative acoustic modeling. Briefly: using speaker-dependent models run the decoder on the database. Utterances where the transcript disagrees significantly from the decoder output are discarded. The remaining utterances are then used as raw material for new models, and the procedure is repeated until convergence. While this should result in tighter acoustic models by eliminating unsuitable utterances, the final result was that it made the synthesizer worse. Presumably, the trimmed database lost too much phonetic coverage. We considered implementing the approach of [9] but lacked time to undertake such an effort.
5. **Power normalization** (idea 5). This relatively simple technique offered a slight gain (53.7%, p=0.3), improving some wavefiles but not all consistently. This was the one idea to traverse all three steps of the testing ladder and did make it to the final submission.
6. **Feature vector definition** (idea 11). Here we addressed the frame-based feature vectors. In particular, the question of whether an energy term should be added to the 12-dimensional cepstral coefficients. This alteration seemed to make no difference to the final synthesized voice.
7. **Pronunciation selection** (idea 10). This idea, though it didn't make it past the second testing step, had intriguingly mixed results. One of the voices (bdl) improved significantly, while another degraded by almost the same amount (slt). Further tests are needed.

8. **Join cost tuning** (idea 6a). Wavefiles generated from the four modified join costs achieved an aggregate preference of 30% ($p << .001$), and a top preference of 37.5% ($p=.02$). This last was for the fourth case of section 3.4.
9. **Optimal couple lookahead** (idea 6b). When looking for an optimal join point, Festival will search partway into the preceding phone. The default value is 33% of the preceding phone's length. We tested this number at values of 10, 33, 50, 67, and 90%. The default worked best.
10. **Fo join weighting** (idea 6c). In the default, pitch change weighting is turned off, i.e. $w_{Fo}=0$. We tested a broad range and found that a value of 2000 performed better (60%, $p=.17$). However, when subjected to larger listening tests the advantage disappeared (47.5%).
11. **Unit clustering** (idea 7). The "stop size" parameter of Festival's CART tool controls the size of decision tree leaf nodes. The default value is 20. We could find no better value, as the following table of listening scores reveals. Adding phone duration as a selection feature in the tree building process also resulted in no improvement.

Compare	Score	Better?
5 vs 20	27.5	no
10 vs 20	20	no
15 vs 20	42.5	no
25 vs 20	30	no
30 vs 20	27.5	no
40 vs 20	25	no
80 vs 20	20	no

12. **Unit name binding** (idea 8). This attempt became a frustrating case of vanishing gain. The situation: developer A discovers a name binding scheme that passes stage 1 tests. Developer B incorporates the code into the current build – and can't detect a difference. Undaunted, B searches for variant that does seem to offer gain (55%, $p=.54$). When results return from larger listening tests the verdict is firmly negative (37.5%, $p=.05$).

3.6. Developer's Delusion

In every instance of Table 1, ideas that promoted out of first stage testing received decreasing preference scores when presented to a larger populace. We call this trend *Developer's Delusion* (alternatively: *Developer's Optimism* or *Developer's Bias* for differing connotations). Why does this happen, despite randomized wavefile presentation?

First, every person has a slightly different internal measure of quality. For some, clarity dominates continuity; for others, the reverse, etc. When working alone a developer has no choice but to optimize for their own sense of goodness. Problem is, this sense might be idiosyncratic.

Second, the developer is so familiar with the effect of their changes that during tests one is highly attuned to familiar details. A-B choices then become an intertwined mixture of quality judgments and system recognition. A gap of several weeks is needed to shake the biasing influence of system recognition.

3.7. Build Sequence Testing

Ideally we would have methodically followed our testing protocol but found we had to save time with a short circuit. We selected the best wavefiles from the initial baseline and the results of improved pitchmarks, and from this generated a new artificial baseline. Then the promising ideas of first phase testing were organized into a ladder structure for third phase testing. With this we presented four tests.

1. Test 1. Idea 5 versus the new combined baseline
2. Test 2. Idea 6c versus idea 5.
3. Test 3. Idea 7 versus ideas 6c and 5.
4. Test 4. Idea 8 versus ideas 7 and 6c and 5.

Sadly, this was a tactical mistake. Recruited subjects often would do the first test and then stop, requiring encouragement to continue. Results for tests 3 and 4 were especially needed but were fewer in number and delayed. When the negative results came back it was too late to pull those features from the submitted system. Instead of adopting the natural order, we should have presented the tests in reversed chronology.

3.8. Before and After Comparison

In the end, the primary question is: after all this effort, did we manage to improve from our baseline at all? To find out we ran one more second stage listening test to compare our baseline with the final submission. For this we allowed three months to pass from previous tests and used the Blizzard utterances of the *conv* and *guten* domains (100 in total). The 500 A-B decisions revealed a significant – but not dramatic – preference of 60% for the new voice over the old ($p<<.001$).

4. Conclusions

Our entry is identified in the Blizzard report as system C; the official results place us in the middle of the pack. No doubt, good method does not guarantee first place. What it did do is allow us to structure our efforts and make progress towards the goal of better synthetic voices. We believe this experience will serve us well in Blizzard Challenge 2006.

5. Acknowledgments

We thank Alan W Black and Keiichi Tokuda for spear-heading the first Blizzard Challenge, and again to Alan W Black for offering advice and impetus to Team CMU. Also, we owe a big debt of gratitude to all those who took part in our listening experiments.

This work is in part supported by the US National Science Foundation under grant number 0219687 "ITR/CIS Evaluation and Personalization of Synthetic Voices," and 0205731 "ITR: Prosody Generation for Child Oriented Speech Synthesis." Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

6. References

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

[2] Kominek, J. and Black, A. "The CMU ARCTIC Speech Databases," *5th ISCA Speech Synthesis Workshop*, Pittsburgh, USA, 2004, pp. 223-24.

[3] Black, A., Lenzo, K., "Building voices in the Festival speech synthesis system," 2000. www.festvox.org/bsv.

[4] Kominek, J., Black A. "A Family-of-Models Approach to HMM-based Segmentation for Unit Selection Speech Synthesis," *ICSLP-2004*, Jeju, Korea.

[5] Kominek, J. and Black, A. "Impact of Durational Outlier Removal from Unit Selection Catalogs," *5th ISCA Speech Synthesis Workshop*, Pittsburgh, USA, 2004, pp. 155-60.

[6] Black, A, Lenzo, K. "Limited Domain Synthesis," *ICSLP 2000*, Beijing, China.

[7] Kishore, S. and Black, A. "Unit Size in Unit Selection Speech Synthesis," *Eurospeech 2003*, Geneva Switzerland.

[8] Bennett, C., Black, Alan W. "Prediction of Pronunciation Variations for Speech Synthesis: a data-driven approach," *ICAASP 2005*, Philadelphia, USA.

[9] Kim, Y-J., Syrdal, A., Conkie, A. "Pronunciation Lexicon Adaptation for TTS Voice Building," *ICSLP-2004*, Jeju, Korea.