# CircumReality functionality delta: Blizzard Challenge 2007 to 2008

*Mike Rozak*

μXac, Darwin, NT, Australia

Mike@mXac.com.au, http://www.CircumReality.com

## Abstract

Although performing poorly in the Blizzard Challenge 2008, the CircumReality text-to-speech engine improved significantly from the Blizzard 2007 test. The engine's acoustic model, prosody model, and acoustic synthesis were improved between tests. This paper discusses the CircumReality engine's test results and reasons why it did poorly. The paper provides a list of improvements that resulted in higher test scores in 2008, as well as implementation details one change: objectively-calculated target costs.

**Index Terms**: speech synthesis, games, Blizzard Challenge

## 1. Introduction

The Blizzard Challenge was devised "*to better understand and compare research techniques in building corpus-based speech synthesizers on the same data. The basic challenge is to take the released speech database, build a synthetic voice from the data and synthesize a prescribed set of text sentences. The sentences from each synthesizer will then be evaluated through listening tests.*" [1] Participants then write a paper discussing their results. Over the course of years, the intent of the competition and publication cycle is to improve the quality of TTS engines.

The CircumReality TTS engine is designed for the CircumReality multiplayer online game. [2] The engine uses concatenative synthesis with a trained prosody model. Half-phone units are used with a triphone context.

The engine was first entered in the Blizzard 2007 challenge and did poorly, ranking last on nearly all the tests.[3] Although the CircumReality engine ranked poorly in the latest Blizzard 2008 challenge, its scores improved significantly from 2007. This paper discusses why the CircumReality engine did poorly, what changes to the engine significantly improved the quality, and implementation details of one change: objectively calculated target and join costs.

## 2. Blizzard challenge 2008 results compared to 2007

Although CircumReality TTS engine did poorly in the 2007 and 2008 challenges, it showed significant improvement.

CircumReality's mean-opinion score (MOS) rose 0.7, from 1.3 for the "A" voice in 2007, to 2.0 in 2008. (See Figures 1a and 1b.) The average of all other engines' MOS (excluding the original speaker and two 2008 benchmark engines, Fest and HTS) was 2.95 in 2007 and 2.92 in 2008, down slightly. Of course, the 2007 and 2008 voices were different, so only large changes in score or relative to other engines are meaningful. Nor are the participants in 2008 the same as 2007.

In contradiction, the Festival benchmark engine's MOS improved from 3.0 to 3.3 despite the average engines' MOS declining slightly. Since participants in both years are largely the same, I suspect this represents either an engine bias towards American English, or voices recorded with 2007's "news presenter" prosody.

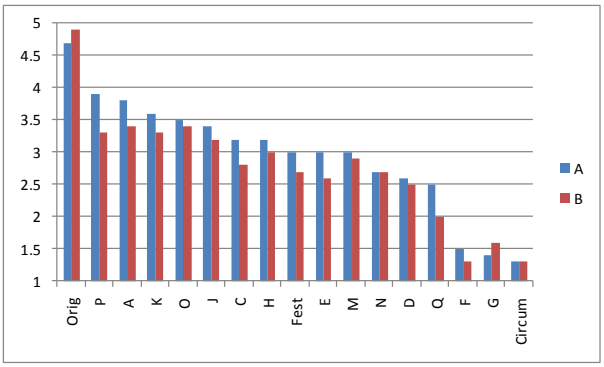


Figure 1a: Blizzard 2007 MOS

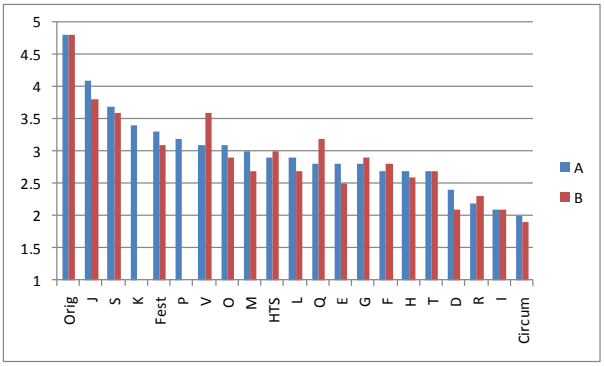


Figure 1b: Blizzard 2008 MOS

CircumReality performed relatively better with the word-error-rate test (WER), both in 2007 and 2008:

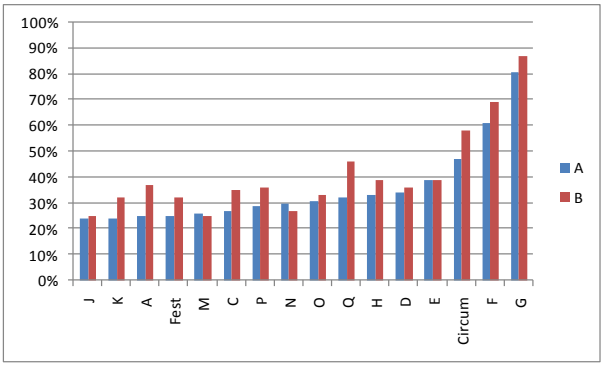


Figure 2a: Blizzard 2007 WER

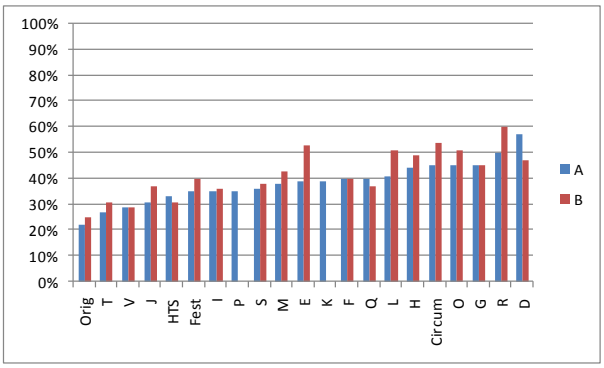


Figure 2b: Blizzard 2008 WER

CircumReality's mean WER only dropped 2%, from 47% in Blizzard 2007 to 45% in Blizzard 2008. The mean WER for all other voices (excluding the original speaker and two 2008 benchmark voices, Fest and HTS) was 35% in 2007, and 40% in 2008, an increase of 5% due to the 2008 voice's ebullient British prosody. The benchmark engine, Festival, also had an increased WER, increasing from 25% in 2007 to 35% in 2008. Despite the aggregate WER increasing, CircumReality's WER decreased marginally.

# 3. Failure analysis

## 3.1. Acoustic model

As discussed in CircumReality's Blizzard 2007 paper [3], the CircumReality engine uses an acoustic feature set consisting of a voiced and unvoiced spectrum. The feature set was chosen to enable easy voice transformations, important for games. Unfortunately, the feature set introduces vocoder-like artifacts, more prominent in some voices than others. The 2007 CircumReality TTS engine (CR2007) used additive sine-wave synthesis to synthesize the wave.

The Blizzard 2007 voice exposed many problems with the acoustic feature extraction. Between Blizzard Challenges, the feature extraction algorithms were improved, but the Blizzard 2008 voice still exhibited significant artifacts.

The feature set had pitch-synchronous PCM added. A full wavelength was included with each frame, and time-stretched to the required wavelength when synthesized. PCM functionality was originally included in the feature set for testing and debugging purposes only. Its functionally was kept to a minimum since PCM isn't flexible enough for voice transformation, an important feature for game synthesis. Consequently, TD-PSOLA was not implemented to save development time, despite TD-PSOLA sounding better.

The 2008 CircumReality TTS engine (CR2008) could synthesize with either PCM or additive sine-wave synthesis. PCM synthesis improved acoustic naturalness, but introduced other errors:

Overlapping pitch-synchronous PCM with a Hanning window reduces unvoiced energy at high frequencies, particularly impacting the "brightness" of the "s" phoneme. Energy at high frequencies was amplified to counteract this effect, improving the intelligibility of "s". Unfortunately, the high-frequency energy boost changed the voice's quality, impacting the "similarity" score.

PCM is a proverbial double-edged sword. PCM produces high fidelity speech synthesis, but introduces large artifacts when pitch shifted. The target costs for pitch-shifting PCM were calculated and included in the unit-selection search. (See section 5.). The costs, as expected, were large.

Even using large F0 target costs, the unit-selection search would occasionally select a unit requiring a substantial pitch shift, producing "hiccups" in the speech synthesis. The hiccups undoubtedly lowered the voice's MOS, counteracting some of the MOS improvements gained by using PCM.

To minimize the hiccups, units' original F0 contours were averaged into the synthesized-prosody's F0 contours, producing higher acoustic quality at the expense of lower prosody quality.

Extremely high F0 target costs outweighed all other join and target costs: duration, energy, and phoneme context. Consequently, the use of PCM forced less well-fitting units to be substituted, reducing the voice's quality.

At the time of the 2008 test, PCM sounded marginally better than additive sine-wave synthesis, despite all its negative consequences. PCM was used for the tests.

## 3.2. Prosody model

For computer games, personality is often more important than intelligibility. CircumReality's synthesized prosody is designed to try and reproduced the prosody of the training voice, often at the expense of intelligibility. CR2008's prosody algorithms produce lower-quality prosody than hand-generated rules.

The 2008 speaker spoke in an "ebullient" manner that, even before synthesis, was more difficult to understand than the "news presenter" prosody spoken by the 2007 Blizzard Challenge voice.

Ebullient speech exposed weaknesses in CR2008's prosody algorithms that weren't as obvious in the "news presenter" voice from 2007. CR2008 wasn't able to synthesize ebullient prosody that well, certainly less well than it could synthesize "news presenter" prosody.

CR2008 did manage to approximate ebullient prosody. Unfortunately, in partially succeeding, CR2008's prosody modeling made the voice more difficult to understand because ebullient prosody is inherently more difficult to understand than "news presenter" prosody, even when spoken by a real person.

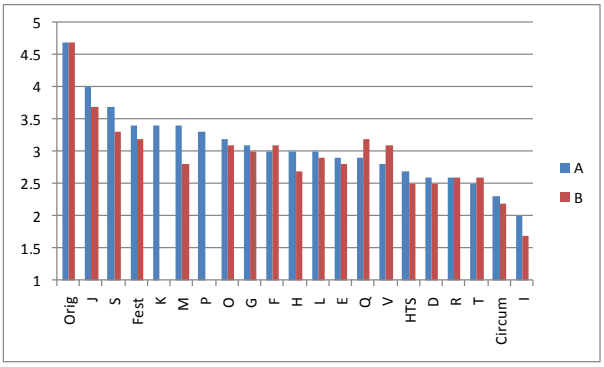


Figure 3: Blizzard 2008 similarity scores

When listening to the test sentences, I thought CircumReality mimicked the voice's prosody better than many of the other engines. I expected CircumReality's similarity score (see Figure 3) to be relatively higher than its MOS score. This didn't happen; both the MOS and similarity scores, and their positions relative to other engines, were approximately the same. Either my perception of how well CircumReality mimics prosody is incorrect, or prosody is only a very minor part of how people perceive voice similarity. Acoustic similarity seems to be a much larger component, at least when listeners are presented with an

unfamiliar voice. If the voice were Winston Churchill's, with its own unique prosody, would prosody modeling count for more?

CR2008's prosody model failed in other ways:

As already stated, the prosody model was hindered by the need for PCM to minimize F0 changes.

Synthesized prosody was further impaired by problems with F0 detection. Pitch doubling would occasionally happen, particularly at the end of utterances. Units that are pitch-doubled are normally eliminated from the acoustic model because an F0-mismatch results in distorted features that produce a low ASR score; low-scored units are automatically discarded. The prosody model doesn't have any equivalent F0 integrity checks, so a pitch-doubled word results in synthesized prosody that suddenly doubles F0 for a word or two, usually at the end of a sentence.

# 4. Changes between Blizzard 2007 and Blizzard 2008

Below is a list of major changes between CR2007 and CR2008. All of these changes produced at least minor improvements to voice quality. Some will be discussed in detail, in section 5.

## 4.1. General changes

- Bugs that produced minor reductions to speech quality were found and fixed.

## 4.2. Acoustic model

- F0 detection was improved by assuming that F0 stayed near to the median F0 throughout the sentence.
- Acoustic feature extraction was improved. The same features were used, but new algorithms more-accurately extracted the features from the training utterances.
- Pitch-synchronous PCM was stored, allowing CR2008 to synthesize using PCM.

## 4.3. Unit selection

- CR2007's voice-construction tool ran out of 32-bit memory when building the Blizzard 2007 voice, limiting the voice to 20,000 units. CR2008's tool was rebuilt with 64-bit pointers, easily allowing a 265,000 unit voice.
- When building a voice, all units were scored by a combination of ASR and target costs based on F0, duration, and energy. CR2008 automatically discarded the bottom 25% of all units to minimize bad units.
- In CR2007's voice building tool, an ASR model for each triphone was trained and used to compute the unit's score. In CR2008's tool, nine ASR models for each triphone model were calculated as a two-dimensional matrix of low, medium, and high F0 by low, medium, and high energy. This improved the voice's clarity and eliminated some "muffled" units.
- In CR2007's voice building tool, ASR was used to test how similar the unit sounded compared to its triphone model. In CR2008, this value was modified based on how differently the unit compared to other phonemes, comparing the unit against ASR models for similar phonemes. This discouraged units that were in-between two phonemes, producing a voice that was easier to understand.
- CR2008 differentiates between triphones at the start, middle, or end of a word. CR2007 did not.

## 4.4. Prosody model

- The prosody model was refined. The same basic principles were used.

## 4.5. Prosody synthesis

- In CR2008, F0 and duration of synthesized units are now affected by the original unit's F0 and duration. When PCM acoustic synthesis is enabled, F0 from the original unit is weighted much more strongly than the F0 from the synthesized prosody model.

## 4.6. Acoustic synthesis

- The acoustic-unit-selection Viterbi-search was refined.
- The acoustic unit search for CR2007 voice used ad-hoc target and join costs. For CR2008, these were calculated using ASR. See section 5.
- F0, duration, and energy of the prior unit are included in the target cost to encourage smooth transitions when non-contiguous units are used.
- As stated earlier, CR2008 can synthesize using PCM instead of additive sine-wave synthesis.

# 5. Objectively calculating target and join costs

Once change between CR2007 and CR2008 warrants further discussion.

In CR2007, target costs were based on ad-hoc guestimates. For example: Left/right context substitutions were assigned a very high target penalty, around ten times higher than F0, energy, and duration target costs.

The MARY-TTS Blizzard 2007 paper [4] implied that objective target costs for concatenative synthesis hadn't been calculated before. This challenge intrigued me so I decided to calculate the costs. I later learned that the USTC/iFlytek Blizzard 2007 paper [5] discussed target cost calculations for HMM synthesis. I'll discuss the differences later.

The target-cost calculating tool was created, and values were calculated from 9000 sentence-length recordings of my own voice. What follows is a list of the calculated target-cost values, and the algorithms used to calculate them.

## 5.1. F0 target cost

To calculate F0 target costs, an ASR model was trained for every triphone. (To reduce computation time and memory, the left and right content phonemes of the triphone were grouped into one of 17 groups. For example: "m", "n", and "ng" were placed in the same group.) Importantly, not all versions of the triphone unit were included in the ASR model; only phonemes whose F0 fell near the median F0 for the triphone were trained.

In a second pass, all of the phonemes in the training data were compared against the F0-limited triphone ASR models. The ASR scores were graphed on a scatter plot.

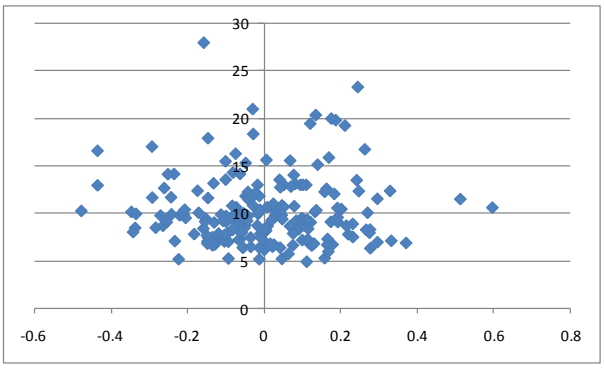


Figure 4: F0 target costs for (l, r) – eh1 – (m, n, ng). The vertical axis shows the ASR score, with large values being poor matches. The horizontal axis shows the number of octaves that F0 was above or below the triphone's median F0.

To ensure that enough data existed to produce an accurate linear fit, the data points were combined into four sets based on broad phoneme categorization. Phonemes were categorized into voiced (V) or unvoiced (U), and plosive (P) or non-plosive (N). For example: The phoneme, "m", is voiced non-plosive (VN), while "t" is unvoiced plosive (UP).

| | Per octave higher | Per octave lower |
|---|---|---|
| **UN** | *3.26* | *6.97* |
| **UP** | *2.53* | *1.13* |
| **VN** | *5.43* | *6.52* |
| **VP** | *3.76* | *3.71* |

Table 1: F0 target costs per octave the target is higher or lower than the original data.

The calculated F0 target costs, although lower than expected, make intuitive sense; F0 target costs for unvoiced plosives (UP) are much lower than costs for voiced non-plosives (VN).

### 5.2. F0 target costs with PCM

The F0 target costs in 5.1 were calculated assuming that additive sine-wave synthesis would be used. Synthesizing with PCM requires additional target costs since even small F0 shifts in PCM produce extreme artifacts.

To calculate the PCM F0 target costs, the voiced and unvoiced spectrums were shifted up and down by half an octave, simulating a PCM F0 shift of half an octave. The shifted spectrums were compared against the triphone ASR models. The ASR score for the original un-shifted unit was also calculated, and subtracted from the two shifted ASR scores. All the shifted scores were averaged based on UN, UP, VN, and VP.

| | Per octave higher | Per octave lower |
|---|---|---|
| **UN** | *18.28* | *18.82* |
| **UP** | *11.06* | *7.44* |
| **VN** | *20.94* | *19.8* |
| **VP** | *10.9* | *8.8* |

Table 2: PCM F0 target costs per octave the target is higher or lower than the original data.

PCM F0 target-cost values are very high, especially for non-plosives (UN and VN). Because TD-PSOLA has fewer acoustic artifacts than the simplistic PCM synthesis I used, I suspect that TD-PSOLA would have produced lower F0 target-costs, although still significant.

### 5.3. Energy target costs

Energy target cost was calculated using the same basic approach as F0 target costs. Instead of training an ASR model with F0's near a target, only units with an energy-value near the target energy were trained.

| | Energy doubled | Energy halved |
|---|---|---|
| **UN** | *5.95* | *4.94* |
| **UP** | *6.58* | *1.62* |
| **VN** | *4.04* | *6.37* |
| **VP** | *7.11* | *3.13* |

Table 3: Energy target costs based on the target's energy relative to the unit's original energy.

### 5.4. Duration target costs

Duration target costs were calculated in the same way that F0 and energy target costs were calculated.

| | Duration doubled | Duration halved |
|---|---|---|
| **UN** | *0.05* | *1.5* |
| **UP** | *3.02* | *15.53* |
| **VN** | *1.04* | *5.04* |
| **VP** | *4.45* | *4.81* |

Table 4: Duration target costs based on the target's duration relative to the unit's original duration.

### 5.5. Start/end word target costs

CR2008 differentiates units occurring at the start, middle, or end of a word, and applies a target-cost penalty if there's a mismatch.

To calculate the target cost, four ASR models were trained per triphone: At the start of a word, middle of a word, and end of a word, and triphones that were the entire word.

A second pass compared every phoneme in the training data against each of the four ASR models for the triphone. The ASR score from the correctly-matched word-position model was subtracted from all the others so the target-cost penalty for an exact match is always 0.

All of the ASR scores were averaged:

| | Target-cost penalty |
|---|---|
| **No mismatch** | *0.00* |
| **Start-of-word mismatch** | *3.56* |
| **End-of-word mismatch** | *3.38* |
| **Start and end-of-word mismatch** | *5.61* |

Table 5: Target-cost penalty for word-position mismatches.

### 5.6. Mismatched left/right context target costs

CircumReality's unit selection search can substitute in a different triphone with the same centre phoneme. For example: The "a" in "cat" might be used to synthesize the words "cap" or "bat" even though the "a" in "cap" and "bat" should use different triphones.

Calculating the context target costs requires an enormous number of ASR models to be trained:

- **Exact match ASR models** – An ASR model was trained for every triphone, using the left and right phonemes as the left and right triphone context.
- **Narrow-group ASR models** – The left and right phonemes were categorized onto one of 17 groups, and the triphone trained based on the left and right phonemes' groups.
- **Broad-group ASR models** – The left and right phonemes were categorized into one of 5 groups, and the triphone was trained based on the left and right phonemes' groups.

In a second pass, every phoneme in the training database was compared against numerous "exact match", "narrow group" and "broad group" ASR models:

1. **"Exact-match" target cost**– The phoneme was compared against the appropriate "exact match" ASR model.
2. **"Mismatched-stress" target cost** – If the right context was a stressed or unstressed phoneme, then the phoneme was compared against the "exact match" ASR model of the opposite stress-context. For example: If the right context was "eh1", the phoneme was compared against the model with "eh0" as the right context.
3. **"Mismatched-phoneme in narrow group" target cost** – The phoneme was compared against all the "exact match" ASR triphone models with varied right contexts, such that: (a) the right context phoneme was part of the true right-phoneme's "narrow group", and (b) the right context's phoneme was not a stressed or unstressed version of the true right-context phoneme. The ASR scores were then averaged.
4. **"Mismatched-phoneme in broad group" target cost** – The phoneme was compared against all possible right-context variations of the "narrow group" triphone ASR models, except the phoneme's true "narrow group" ASR model. The results were averaged.
5. **"Mismatched-phoneme not in broad group" target cost** – The phoneme was compared against all the "broad group" triphone ASR models whose right-context did not match the phoneme's true right context. The results were averaged.
6. To ensure that an exact left/right context match would have 0 target cost, the "exact match" score from step 1 was subtracted from all the other scores (steps 2 through 5).
7. Steps 1 through 6 were repeated, but the left context was varied instead of the right.

The resulting target costs are:

| | Left context mismatched stress | Left context mismatched phoneme in narrow group | Left context mismatched phoneme in broad group | Left context mismatched phoneme not in broad group |
|---|---|---|---|---|
| **UN** | *0.67* | *1.30* | *1.67* | *2.90* |
| **UP** | *0.3* | *1.47* | *2.13* | *4.84* |
| **VN** | *0.43* | *0.99* | *1.51* | *2.22* |
| **VP** | *0.51* | *1.20* | *1.60* | *5.76* |

Table 6a: Left context target costs.

| | Right context mismatched stress | Right context mismatched phoneme in narrow group | Right context mismatched phoneme in broad group | Right context mismatched phoneme not in broad group |
|---|---|---|---|---|
| **UN** | *1.28* | *7.41* | *7.44* | *5.87* |
| **UP** | *2.16* | *4.83* | *5.29* | *11.21* |
| **VN** | *2.71* | *3.30* | *2.62* | *4.36* |
| **VP** | *3.02* | *7.46* | *5.61* | *8.54* |

Table 6b: Right context target costs.

As anticipated, mismatched plosives (P) tend to incur a higher target cost then non-plosives (N). Unexpectedly, right-context-substitution target costs are much higher than left-context costs.

Context target costs were much lower than anticipated, less than one tenth the ad-hoc values used in CR2007.

The data also illustrates some noise. Theoretically, all values should be monotonically increasing from left to right. Values don't always do this, as in the case of the right context being an unvoiced non-plosive.

## 5.7. Non-contiguous join costs estimates

Join costs between two units were calculated using a distance measure between the spectrums of the two boundary frames. With thousands of possible joins considered by the unit-selection Viterbi search, this process can be very slow. Pre-calculating all the join costs is not an option for CR2008 due to the memory and file size requirements of games.

As an optimization, the "mean join costs" based on diphones are calculated and used to reduce the number of candidates for which accurate boundary scores must be calculated. The unit-selection search's hypothesized units are narrowed down to the top 100 candidates by sorting based on their anticipated scores, including all the target costs and the mean join costs. Join-costs are then accurately calculated between the existing hypothesis and 100 new candidates. (See section 5.8.)

To calculate the mean join costs, a diphone ASR database is trained. Unlike the other ASR training, only the last frame in the first unit is trained – where the join occurs. CR2008's ASR comparison for a single frame is exactly the same mathematics as used to calculate join costs.

ASR scores for the diphones are calculated and averaged into a database. As with other target cost calculations, phonemes are categorized into one of four groups:

| | Right context is UN | Right context is UP | Right context is VN | Right context is VP |
|---|---|---|---|---|
| **Left context is UN** | *20.98* | *17.38* | *17.88* | *13.75* |
| **Left context is UP** | *23.42* | *25.67* | *16.51* | *22.20* |
| **Left context is NV** | *21.42* | *22.52* | *11.80* | *15.89* |
| **Left context is VP** | *24.71* | *31.30* | *12.59* | *13.46* |

Table 7: Mean join-costs given the left and right contexts.

## 5.8. Join costs

Join costs between non-contiguous units were calculated using a distance measure between their adjacent spectrums.

As shown earlier (Section 5.7), the mean join cost was around 20, with higher values indicating poorer joins.

The unit's score and all the target costs have a "per second" connotation. When the unit-selection Viterbi search includes them in the hypothesis score, the ASR scores are scaled by the unit's duration.

Join costs are different because they are the calculation of an instantaneous value, over one frame, not the duration of the phoneme. Because join cost is instantaneous, the join cost value should theoretically be scaled by one frame (5 milliseconds) and added to the Viterbi search score.

This doesn't work well in practice. A scaling value of 25-50 milliseconds produces better-sounding results. I haven't yet determined why the theoretical value doesn't work well.

### 5.9. USTC/iFlytek

I hadn't noticed the Blizzard 2007 USTC/iFlytek paper [5] discussing their target and join cost calculation methods until long after implementing my own algorithms.

CR2008's target/join-cost algorithms differ from those iFlytek's in a number of ways:

- CR2008 uses a different acoustic feature set than iFlytek, so per-frame acoustic distance calculations are handled differently, but with the same intent.
- iFlytek uses F0 as a join cost, probably because the iFlytek's acoustic synthesis employs a PCM acoustic representation. Since PCM doesn't handle pitch bending well, little to no pitch bending would be applied to iFlytek's synthesized units, leaving only F0 mismatches at joins. CR2008 is designed for games, where transplanted prosody is required. F0 is dictated by the transplanted prosody or prosody model. Synthesized F0 is never the same as the original unit's F0. Consequently, F0 must be part of the "per second" target cost instead of the instantaneous join cost.
- iFlytek's duration model is built into the same framework as its acoustic and concatenation models. F0 and energy are also included in that framework. CR2008 separates F0, duration, and energy into a separate model. They're either provided by a prosody model or transplanted prosody. In CR2008, the prosody model drives F0, duration, and energy, in turn driving the unit selection. Conversely, iFlytek's HMM synthesis, with no explicit prosody model, appears to let unit selection drive F0, duration, and energy, which in turn drives prosody.
- Context mismatch costs are implicitly handled by iFlytek's HMM acoustic distance measures. CR2008 must explicitly include them.

## 6. Conclusions and future work

The CircumReality TTS engine improved significantly between the 2007 and 2008 Blizzard Challenges. This was achieved through a variety of changes in the acoustic model, unit selection, prosody model, and acoustic synthesis.

Using PCM in CR2008 was a mistake; although it improved the acoustic quality of the voice, PCM's F0 inflexibility hurt many other TTS subsystems. TD-PSOLA is expected to sound better, but isn't ideal for games, so it may not be worth the experimental effort. CR2008's acoustic feature extraction algorithms have been improved in the months since submitting synthesis results for the 2008 Blizzard Challenge. The quality of the additive sine-wave synthesis voice now matches or exceeds the PCM voice.

I plan to improve acoustic synthesis in a number of ways:

1. More-accurate ASR, since ASR is the foundation of good unit selection.
2. More-accurate unit scores, such as different scores for each half of the unit.
3. More-accurate target and join costs, using more than the four groups (UN, UP, VN, VP) discussed here.
4. The join-cost scaling problem from 5.8 needs to be solved.
5. Smoother unit joins are needed, the exact location of the join determined by ASR.
6. Prosody tradeoffs that improve the aggregate (unit + target + join) scores for a synthesized utterance.

The prosody model needs to be improved too, although the tradeoffs between intelligibility and mimicking the original voice's prosody will continue to be an issue.

## 7. References


[1] Anonymous, "The Blizzard Challenge 2008", CMU. Online: http://festvox.org/blizzard/, accessed on 4 July 2008.

[2] Rozak, M., "What is CircumReality?", mXac. Online: http://www.CircumReality.com, accessed on 16 July 2008.

[3] Rozak, M., "Text-to-speech Designed for a Massively Multiplayer Online Role-Playing Game (MMORPG)", in *The Blizzard Challenge 2007*, Bonn, Germany. mXac. Online: http://festvox.org/blizzard/bc2007/index.html, accessed on 16 July 2008

[4] Schroder, M. and Hunecke, A., "MARY TTS Participation in the Blizzard Challenge 2007", in *The Blizzard Challenge 2007*, Bonn, Germany. Online: http://festvox.org/blizzard/bc2007/index.html, accessed on 16 July 2008

[5] Zhen-Hua Ling, Long Qin, Heng Lu, Yu Gao, Li-Rong Dai, Ren-Hua Wang, Yuan Jiang, Zhi-Wei Zhao, Jin-Hui Yang, Jie Chen, Guo-Ping Hu, "The USTC and iFlytek Speech Synthesis Systems for Blizzard Challenge 2007", in *The Blizzard Challenge 2007*, Bonn, Germany. Online: http://festvox.org/blizzard/bc2007/index.html, accessed on 16 July 2008