The IBM Submission to the 2006 Blizzard Text-to-Speech Challenge

Ellen Eide[1], Raul Fernandez[1], Ron Hoory[2], Wael Hamza[1], Zvi Kons[2], Michael Picheny[1], Ariel Sagi[2], Slava Shechtman[2], Zhi Wei Shuang[3]

[1]IBM T.J. Watson Research Center
[2] IBM Haifa Research Center
[3] IBM China Research Lab

eeide@us.ibm.com

Abstract

In this paper, we present two concatenative text-to-speech systems built from the "Blizzard Challenge" speech databases. The two systems differ primarily in their segment selection cost function. One system has our baseline cost function, and the other has a cost function which has been altered to potentially better handle small datasets. Results indicate that both systems perform similarly in terms of MOS and intelligibility.

1. Introduction

One key factor in determining the quality of the output of a text-to-speech system is the dataset upon which it is modeled. Both the voice characteristics of the database speaker and the sheer quantity of data play important roles in shaping the goodness of the resulting speech output. The Blizzard Challenge arose in an effort to compare algorithm performance without the confounding effects of different datasets; various systems built using a common dataset are compared for overall quality as well as for intelligibility.

This paper describes two IBM submissions to the Blizzard Challenge 2006. The two systems differ in the cost functions used to select segments. One system is used primarily on large datasets, while the other uses the cost function which was developed to perform well in situations in which the amount of data available is much less. The build-time processing for both of these systems is the same; only the run-time synthesis engine is changed. We also explore the use of prosody models built from a pooled-speaker dataset [2].

The rest of the paper is organized as follows. After a brief overview of the IBM Concatenative Speech Synthesis system in Sections 2 and 3, we describe the application of these systems to the Blizzard Challenge databases and present the results of the listening tests for our systems in Section 4.

2. System build

The speech associated with each Blizzard dataset was used to build a speaker dependent Hidden Markov Model (HMM). In these models, each phone is represented by a typical left to right 3-state HMM. The resulting models are used to align the recorded sentences to their phonetic labels. Using the state alignments, a top-down likelihood-based acoustic tree is built for each HMM state using the standard acoustic tree growing algorithm used in speech recognition. The leaves of the resulting trees are considered to be the synthesis units, and all segments aligned to any particular leaf are candidates for this leaf. In addition, all words that occurred in the recorded sentences along with an index to the corresponding speech segments are kept in a dictionary [1] henceforth called the "alternates' dictionary." The alternates' dictionary is used during the segment search to allow segments that belong to a certain word to be considered during the search even if the front end predicts a different phonetic pronunciation for this word.

The recorded sentences are also used to grow statistical decision trees for energy, duration, and pitch. Rather than building prosody trees for predicting f0 and duration exclusively from the Blizzard dataset, we have chosen to build the prosody models from a dataset resulting from pooling the speech of several speakers together according to the method detailed in [2]. This choice enables us to gain more training data for our models, at the potential expense of losing speaker-specific characteristics of the Blizzard speakers. The form of the pitch model remains unchanged from our speaker-specific system. We use a decision tree with features derived from the text such as lexical stress, distances from phrase boundaries, etc. We predict one target pitch vector per syllable. The target pitch vector specifies the desired pitch at three points, corresponding to the beginning, middle, and end of the syllable's sonorant region. Although the form of the pitch model is the same in the pooled-speaker as the speaker-specific systems, the observation in each of these cases is different. Rather than modeling the pitch (in the log domain) as before, in the speaker-pooled model we subtract from the pitch the mean of the pitch for the speaker from whom the observation came. Finally, we divide by the standard deviation of the pitch for that speaker. Thus, our normalized observation is $(p-\mu_i)/\sigma_i$ where p is the log pitch, μ_i is the mean of the log pitch for speaker i, and σ_i is the standard deviation of the log pitch for speaker i.

In order to model durations, we combined the Blizzard datasets with in-house data collected from four professional speakers. All of the speakers who contributed data to the speaker-pooled prosody models spoke at roughly the same speaking rate. Thus,

we did not need to normalize the duration observations before pooling them.

3. Run-time Synthesis

3.1. IBM System 1

Our system [3] uses a rule-based front end to determine the sequence of units in synthesis. The candidate segments for the units determined by the front end are augmented with entries in the alternates' dictionary to address potential mismatches between the pronunciations predicted by the front end and the actual pronunciations by the database speaker. The sequence of candidate segment lists together with the prosodic targets generated by the pooled-speaker models described above drive a Viterbi beam search for the sequence of segments which minimize the cost function.

The search aims at selecting the token sequence with the least cost from among the candidate tokens. The target cost comprises a set of cost components: the f0 cost, which measures how far the f0 contour of the token is from that of the target, the duration cost, which measures how far the duration of the token is from the target, the energy cost, which measures how far the energy of the token is from that of the target (this component is often disabled, including in the experiments reported here). The transition cost comprises two components, one of which captures spectral smoothness across segment joins and another which captures pitch smoothness across the joins. The spectral smoothness component of this transition cost is based on the Euclidian distance between perceptually-modified Mel cepstral coefficients [6]. The target cost components and the transition cost components are added together using weights tuned by hand.

After the search, to remove unwanted "pitch warble," a new target contour is generated by convolving the concatenated f0(t) with a smoothing kernel, and the signal is then processed to match this new contour. While a smooth contour is, by and large, preferable to unintended rapid fluctuations, smoothing does have undesired side effects. Sometimes, f0 fluctuations are necessary, for example, to convey expressions such as excitement. Smoothing also reduces or obliterates other expressive voice qualities, such as creakiness. To preserve such effects whenever possible, we bypass signal processing completely for sufficiently long sequences of tokens which had been contiguous in the original corpus, except near the edges of the sequence where we impose a gradual transition to the f0 of the adjacent segments. While this "contiguous bypassing" removes prosody modification as a tool for expression manipulation, it allows maximally-faithful reproduction of expression present in the corpus.

3.2. IBM System 2

Our second submission to the Blizzard challenge explores the cost function developed for use in our embedded system, which is based on the same technology as our server voice, but is aimed to work in environments with memory limitations. This usually limits the size of the voice data, the executables and the runtime memory. Although the Blizzard Challenge does not pose any memory limitations, we have examined the possible usage of some of the techniques used in the embedded system. The motivation is that these techniques may be more suitable for small voices that comprise a small amount of speech data (the CMU-ARCTIC voice with only 1200 sentences).

The embedded system synthesis process is similar to the baseline system. The two main differences between the systems are in the transition cost functions and in segment concatenation. For the Blizzard challenge only the former has been found worthwhile to test.

3.2.1. Transition cost function

The baseline system can use its large database to generate long continuous utterances with only a small number of concatenation points. Since the embedded system typically works on considerably smaller voices than the baseline system, the number of splices is typically higher, and the role of the transition cost function becomes more important. This is also true for voices build for a small amount of voice data such as the ARCTIC voice.

The embedded system uses the speech modeling scheme described in [4], [5]. The speech is divided into overlapping frames. The length of each speech frame is 20ms and they are evenly spaced at 10ms intervals. Each speech frame is represented by its complex spectral envelope (magnitude and phase), by a degree of voicing value and by its pitch frequency. The complex spectral envelope is approximated by 32 magnitude coefficients and 32 phase coefficients, which are evenly spaced on a mel-scale axis.

This representation is computed and encoded during voice building. During synthesis, speech is reconstructed from the decoded parameters. The spectral envelope parameters (magnitude only) are also reused in the embedded transition cost function computation. Besides saving memory, this representation has been found to be more accurate than the MFCC representation used in the baseline system. We shall refer to the spectral envelope amplitude parameters for each frame as feature vectors.

When the embedded system calculates the transition cost between two segments it compares the feature vectors corresponding to two frames from each segment. From the first segment we pick the last frame that is inside the segment and the following frame. From the second segment we pick the first frame of the second segment and the preceding frame (Figure 1).

The pair of feature vectors from each pair of frames is converted to a *transition vector* (64 dimensional). The transition cost is then the Euclidian distance between the transition vectors.

Many studies have tried to find a transition cost that best matches human perception [6-9]. However, these studies have

not reached a consensus. One of the problems we have encountered is that a function that produces satisfactory results for voiced segments does not perform well for unvoiced segments. Hence, we have found it useful to process the feature vectors differently for voiced and unvoiced frames.

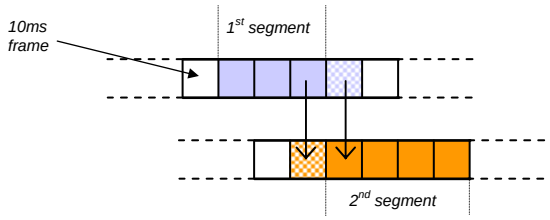

Figure 1: comparing frames of two segments in the transition cost function

For voiced frames we convert the spectral envelope features to loudness scale by raising them to the power of 0.5. We multiply them by weights that emphasize lower frequencies and then normalize the vector by its maximum. For unvoiced frames we compute the log of the spectral envelope amplitude features, multiply them by weights that emphasize higher frequencies and then normalize them by removing their mean value.

Clearly, for contiguous segments the cost is identically zero because the two transition vectors are identical. In case there is a voicing mismatch between the transition vectors compared, the distance between them is assigned a large value, and thus, undesirable voiced-unvoiced transitions are avoided.

3.2.2. Use in the Blizzard Challenge

One of the systems that were submitted to the Blizzard Challenge (IBM2) was the baseline system that was modified to work with the embedded transition cost. This was done by replacing the MFCC transition vectors with the embedded transition vectors and by changing the distance to be Euclidian instead of Mahalanobis [6].

4. Evaluation and Results

Two overlapping corpora were used to build the systems described in this paper. The main full corpus consists of approximately 5 hours of speech recorded by a native speaker of US-English. A second corpus, the Arctic dataset, is a subset of the full corpus containing approximately one hour of recordings of phonetically-balanced sentences chosen from publically-available novels. Any component of the voice building process that makes use of all the data available during its processing (e.g. aligning the text to the waveforms, where acoustic models are iteratively refined from all the relevant data available) was carried out separately for each of these two datasets to ensure that the smaller Arctic build was not influenced by any extra data present in the full set. Only algorithms that are applied on a single-file basis (e.g., pitch mark detection) were applied to the full dataset once and then the results recycled for the Artic build. Since we make use of multiple speakers' data when building the speaker-independent prosody models used here, additional data were combined with each of the two datasets to build the prosody trees. However, only the portion of the target speaker's data corresponding to the Artic build was used when building the prosody models for this corpus. In other words, the prosody models exploit data from several speakers, but only the portion of data from its target speaker contained in the dataset for which the prosody models are being built.

The synthesized samples were submitted to a total of three evaluation tasks, one aiming to evaluate the overall subjective quality of speech, and two functional tasks aiming to evaluate the speech intelligibility:

1. Overall quality judgments: Subjects rate on a 5-point scale the overall quality of the speech.

2. MRT Task: Subjects listen to carrier phrases containing a confusable monosyllabic word (chosen from the Modified Rhyme Test, a standard test for speech intelligibility), and are asked to transcribe this word.

3. Semantically Unpredictable Sentence: Subjects listen to sentences that are randomly generated according to a simple grammar (det adj noun verb det adj noun) and a medium-frequency vocabulary, and are asked to transcribe what they hear.

Three categories of listeners were used in the listening study: (i) undergraduate students (US-English speaking students who were paid for their participation in the study), (ii) volunteers (subjects who were available to take the test over a web interface), and (iii) speech experts (subjects who are knowledgeable about speech processing or synthesis and who were provided by the participants).

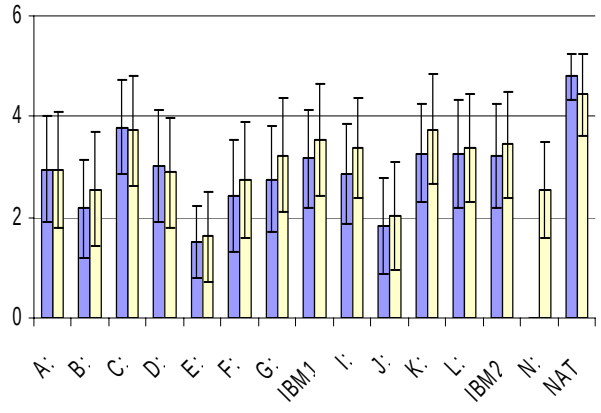

Figure 2. MOS across sites, from undergraduate listeners. Cream/light is the 5 hour dataset and blue/shaded is the 1 hour Arctic subset.

Figure 2 shows the overall MOS scores for all participants as determined by a group of undergraduate listeners. We have

identified by name the two IBM submissions presented in this paper.

Figure 3 shows the overall SUS results as determined by the undergraduate listeners, and figure 4 shows the overall MRT results, again as determined by the undergraduate listeners.
Similar results were obtained from both randomly selected listeners and also from speech expert listeners but are omitted here due to space limitations.

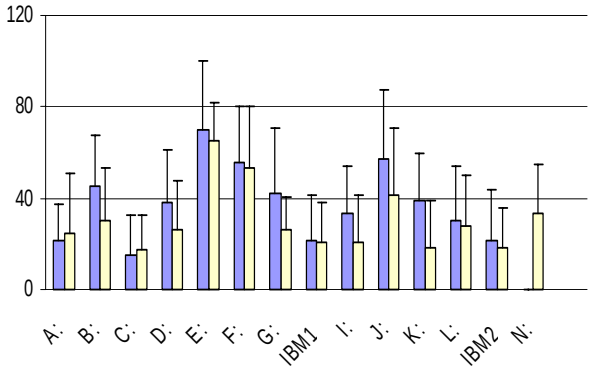

Figure 3. SUS results across sites, from undergraduate listeners. Cream/light is the 5 hour dataset, and blue/shaded is the 1 hour Arctic subset.

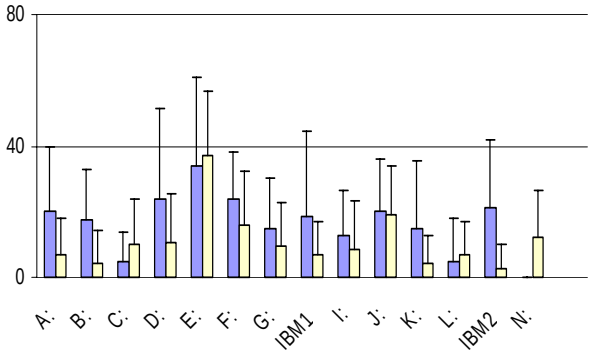

Figure 4. MRT results across sites, from undergraduate listeners. Cream/light is the 5 hour dataset, and blue/shaded is the 1 hour Arctic subset.

5. Discussion

Inherent to the design of a text-to-speech system is a choice of pleasantness and naturalness vs. intelligibility, as these qualities tend to trade off against one another. The IBM submissions to the Blizzard challenge show good performance in this two-dimensional space.

Another trend to note is the improvement of a system's performance in going from the small dataset to the large, as a means of gauging how the system may perform in very-large-dataset situations. The IBM submissions, especially IBM1, show good improvement in going from one hour to five hours of data, which is consistent with the fact that the system is designed to perform well on very large amounts of data.

6. References

[1] Hamza, W., Eide, E., Bakis, R., "Reconciling Pronunciation Differences between the Front-End and the Back-End in the IBM Speech Synthesis System," Proc. ICSLP 2004, Korea.

[2] Eide, E. and Picheny, M. A. "Towards Pooled-Speaker Concatenative Text-to-Speech," Proc. ICASSP 2006. Toulouse, France.

[3] Pitrelli, J. et al. "The IBM Expressive Text-to-Speech Synthesis System for American English," IEEE Transactions on Audio, Speech and Language Processing. Volume 14, Number 4. pp 1099-1108. July, 2006.

[4] Chazan, D., Hoory R., Sagi A., Shechtman S., Sorin A., Shuang, Z. and Bakis, R. "High quality sinusoidal modeling of wideband speech for the purpose of speech synthesis and modification", ICASSP 2006, Toulouse, May 2006.

[5] Chazan, D., Hoory, R., Kons, Z., Sagi A., Shechtman, S. and Sorin A., "Small footprint concatenative text-to-speech synthesis system using complex spectral envelope modeling", Eurospeech 2005, Lisbon, Sep 2005..

[6] Donovan, R. "A New Distance Measure for Costing Spectral Discontinuities in Concatenative Speech Synthesizers", Proc. 4th ISCA Tutorial and Research Workshop on Speech Synthesis, Scotland 2001.

[7] Vepa, J. and King, S., "Join Cost for Unit Selection Speech Synthesis" in "New Paradigms and Advances in Text to Speech Synthesis", Prentice Hall 2005.

[8] Vepa, J., King, S. and Taylor, P., "New objective distance measures for spectral discontinuities in concatenative speech synthesis", IEEE Workshop on Speech Synthesis 2002, NY 2002.

[9] Stylianou, Y. and Syrdal, A. K., "Perceptual and objective detection of discontinuities in concatenative speech synthesis", ICASSP 2001, Salt Lake City 2001.