The NTNU Concatenative Speech Synthesizer

Dyre Meen, Torbjørn Svendsen

Department of Electronics and Telecommunication, Norwegian University of Science and Technology

dyre@iet.ntnu.no, torbjorn@iet.ntnu.no

Abstract

This paper describes NTNU's entry for the Blizzard Challenge 2010. Our system is a conceptually simple variation of an HMM-based unit selection system, which uses diphones as the basic unit and employs a combined selection of units and their join points. The evaluation results of the Blizzard Challenge 2010 show that the system performs well when compared with the other systems.

1. Introduction

A key goal for a text-to-speech development system is to provide means for building voices having maximum quality while requiring minimum human effort. It should be able to produce new voices automatically from as little input as possible, typically a limited set of voice recordings accompanied by a manuscript. It should also ensure consistent high quality speech output even when the input data contains anomalies. HMM-based speech synthesis [1] has proven itself in terms of meeting the requirements just mentioned. It is inherently insensitive to the limited number of errors related to automatic text analysis or speaker artefacts you would expect to find in any training data set. Also, the output speech quality is highly consistent. Unit selection is on the other hand in general more sensitive to the quality of the source data. Errors present in metadata combined with the problem of joining speech segments that may have a poor perceptual fit may lead to unnatural or even incomprehensible speech output. In the best case however, the quality is as good as it gets.

What we would like is a system that provides the consistent quality of the HMM-based synthesis and the crisp best-case sound quality of the unit selection synthesis. The IBM Concatenative Speech Synthesis System based on work by Donovan et al. described in [2, 3] is interesting in that it is closely related to HMM-based synthesis. Decision trees, resulting from model training, are used to identify HMM state-sized segment candidate clusters, and basic unit selection principles are used to select one candidate from each cluster. In HMM-based synthesis, the statistics of the clusters are used to generate speech. Various other related schemes have been proposed and presented in previous work and Blizzard Challenge entries. Differences between systems include choice of basic unit, model and feature vector configuration and how the models are used in the selection algorithm. In the CMU entry[4] from 2007, a 3-state configuration is used with feature vectors composed of 25 MFCC-coefficients augmented with F_0. Diphones, with backoff to half-phones, is used as the basic unit. The system does unit selection based on target vectors generated by CLUSTERGEN. In the USTC system[5] from 2009, feature vectors consist of mel-cepstrum and F_0. A 5-state left-to-right model with no skip is used, but the unit selection is geared towards phone-sized units. This system also applies a Kullback-Leibler divergence based pre-selection of candidates. The MARY TTS system[6] presented in 2008, inspired by the USTC system, uses half-phones as the basic unit, and yet another system[7] focuses on using frame-sized units.

The work presented in this article results from experiments related to combining HMM-based synthesis with unit selection. As a starting point, we have focused on using HTS-models for selecting units, which relates our system to the mentioned systems.

The rest of the paper is organized as follows. First, a high-level system description is given, followed by section 3, which describes the data used by the synthesizer and how a voice is built. In section 4 the run-time synthesizer is described, followed by section 5 which discusses some system details, issues and specifics involved when used on the Blizzard Challenge 2010 dataset, along with evaluation results.

2. System overview

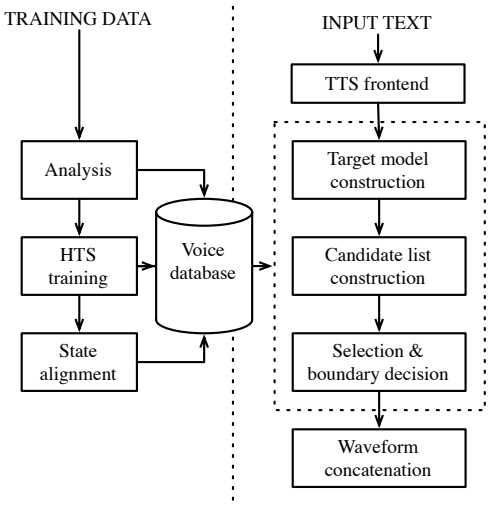

Figure 1: *System overview.*

Figure 1 illustrates the major components of the unit selection system. The task of the training stage is to prepare data, train models and align speech. During run-time synthesis, a frontend enriches the input text with features and uses this to extract a sequence of target models from the model set. For each target, a set of diphone-sized candidates is found and sent to the unit selection component. In addition to finding the specific candidate to use for each target, the selection component considers a number of alternative join positions and selects the one that results in the lowest cost. The final step of the synthesis stage is to produce a waveform output by extracting and concatenating the segments found.

3. Voice building

The voice database consists of an aligned and pitch marked waveform database, a set of Hidden Semi-Markov Models (HSMMs)[8] and the set of feature vectors used to train the models. Given a text-to-speech frontend, an orthographic manuscript and a corresponding set of recordings, the building process is fully automated.

The first step is to convert the manuscript into a richer representation using the frontend. An HMM-based system, similar to the one described in [9], is then used to phonemically align the speech. This is followed by a refinement step in which multiple pronunciation variants are considered.

Having found the most promising transcription and the corresponding alignment, a set of clustered HSMMs is trained using the `HTS-demo` scripts[10]. A conventional 5 state left-to-right model topology with no skip is used. Feature vectors consist of 25 mel-cepstral coefficients including the 0'th and $\log F_0$ along with delta and delta-delta parameters. Each vector is constructed from a 25ms Blackman-windowed frame every 5ms. Fundamental frequency and pitch marks are computed using `epochs` and `get_f0` from ESPS. The acoustic and F_0 coefficients are separated into different data streams and modelled by continuous and multi-space distributions[11] respectively. Clustering is done separately for each state, and the data streams and duration are clustered independently. The question set used for clustering is based on the one defined in HTS-demo, and includes questions about phonetic and intonational context.

The end result of the model training is a clustered HSMM set and decision trees for acoustic, F_0 and duration models, and are used heavily in the run-time synthesis.

A final forced state alignment is done using the trained HSMMs and `HVite` from the HTK toolset[12], which is modified in order to take into account the explicit duration models built into the models. This enables us to associate each state-sized segment with a specific model.

4. Run-time synthesis

The unit selection run-time system accepts an orthographic text, which is converted into an enriched label sequence by the front-end. This is used to construct a corresponding target sequence $\mathbf{t} = \{t_1, \dots t_L\}$ in which each target, t_i, is associated with a label, its corresponding HSMM, $\Phi(t_i)$ and an identifier, $\mathrm{name}(t_i)$, which is used to bootstrap the candidate search space for the given label. The model is found by applying the decision trees described in section 3, and in the current implementation the name is the diphone identity of the target, i.e. the current and previous phoneme.

The speech database can be considered to be a large contiguous array, $\mathbf{p} = \{p_1, \dots, p_N\}$, of N phone-sized units. Each unit is associated with a trained model, $\Phi(p_n)$, a waveform segment, and the corresponding feature vectors. A diphone candidate is thus associated with two phone-units: $u_n = (p_{n-1}, p_n)$, and the cost of selecting a sequence, $\mathbf{u_k}$, of candidate units is given by

$$C(\mathbf{u_k}, \mathbf{t}) = \sum_{i=1}^{L} C_{\text{target}}(u_{k_i}, t_i) + \sum_{i=1}^{L-1} C_{\text{join}}(u_{k_i}, u_{k_{i+1}}), \quad (1)$$

The well-known Viterbi algorithm is used to find the sequence, $\mathbf{u}^*_\mathbf{k}$, such that

$$\mathbf{u}^*_\mathbf{k} = \arg\min_{\mathbf{u_k}} C(\mathbf{u_k}, \mathbf{t}) \quad (2)$$

4.1. Building the candidate list

The candidate list $\hat{\mathbf{p}}^{(i)}$ for a given target, t_i is built by a two-stage process, according to

$$\hat{\mathbf{p}}^{(i)} = \mathrm{prune}(\mathrm{lookup}(t_i))). \quad (3)$$

First, all units matching the diphone identity of the current target is extracted from the unit database, as given by the function

$$\mathrm{lookup}(t_i) = \{p_n : \mathrm{name}(p_n) = \mathrm{name}(t_i)\}, \quad (4)$$

with a back-off to phone matching in the case of missing diphones. The resulting set is possibly very large, and so a pruning step is performed before going further. In order to do this, the candidate list is first ordered using a sorting criterion that tries to measure how well a candidate model, $\Phi(p_n)$, matches the target model, $\Phi(t_i)$. A simple and easy-to-compute measure is the number of clustered *streams* that are shared between the candidate and target models, as illustrated in figure 2. The sorted list is then truncated at a pre-defined maximum size or at the point where the computed value falls below some level relative to the best candidate.

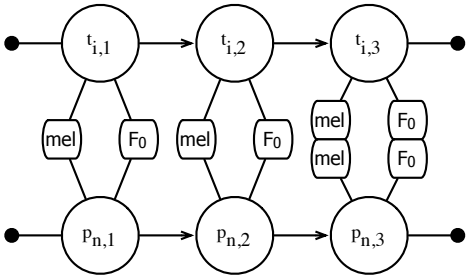

Figure 2: *Illustration of stream sharing for a 3-state/2-stream model configuration. Here, the two streams of the last state are not shared between the target and candidate models.*

For each of the remaining candidates a *target cost* is computed. Using the target model, we simply compute a weighted log-likelihood of the mean-vector of the candidate's model

$$C_{\text{target}}(\hat{p}^{(i)}_k, t_i) = -w_{\text{target}} \log P\left(\mu(\Phi(\hat{p}^{(i)}_k))|\Phi(t_i)\right), \quad (5)$$

where $P(\dots)$ represents a multi-space distribution[11] and w_{target} is a manually set weight. The target cost could have been used as a sorting criterion for the task of candidate pruning, but it is more computationally demanding. Experimentation also suggested that the simple stream-counting scheme results in a somewhat more lively speech output.

If a candidate's natural successor in the database matches the target sequence it is cached and considered as a possible candidate for the next target, t_{i+1}. For the current candidate list, the list of previously cached extension candidates will be added until a predefined maximum number of candidates is reached.

4.2. Computing the join cost and optimized join position

Two candidates may be joined at (automatically aligned) state boundaries, but only one join per diphone is allowed. Given models of S states and two units to be joined, $u_k = (p_{k-1}, p_k)$ and $u_{m+1} = (p_m, p_{m+1})$, the basic idea when considering a join at state s is first to concatenate the feature vectors corresponding to s first states of p_k and the last $S - s$ states of p_m, as

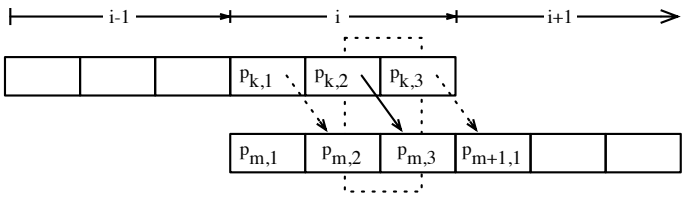

Figure 3: *The illustration shows two units, $u_k = (p_{k-1}, p_k)$ and $u_{m+1} = (p_m, p_{m+1})$, considered for joining, with a 3-state model configuration. A join is allowed only at one of the state boundaries of the overlapping phone segments. The solid arrow illustrates a selected join between state* 2 *and* 3*.*

illustrated by figure 3. The delta and delta-delta features at the join needs to be re-computed to reflect the discontinuity. Then, the target model can be used to compute a cost based on the likelihood of the combined feature vectors across all S states.

If the cost of joining the two units at state s is denoted $c(u_k, u_m; s)$, the unit join cost is given by

$$C(u_k, u_m) = c(u_k, u_m; s^*) \tag{6}$$

where s^* is the state index that minimizes the cost

$$s^* = \arg\min_s c(u_k, u_m; s) \tag{7}$$

Explorative testing revealed a need for fine-tuning and so the actual cost, $c(u_k, u_m; s)$, of joining the two units is a weighted sum of the following components:

- Normalized likelihood of all but the last frame of the s first states of p_k
- Normalized likelihood of all but the first frame of the $S - s$ last states of p_m
- Likelihood of the last frame of the s-th state of p_k. The delta and delta-delta coefficients for this frame are modified so that the join discontinuity is considered.
- Likelihood of the first frame of the $s + 1$-th state of p_m, modified to reflect the join discontinuity.
- Likelihood of the duration of the first s states of p_k
- Likelihood of the duration of the last $S - s$ states of p_m
- A simple penalty which considers the difference in duration of p_k and p_m before and after state s.
- Penalties for state durations if they are distant from the model mean, relative to standard deviation.

The two penalties are added in order to encourage joining units of similar durations and exclude outlier units, typically related to errors done by the automatic voice building process. If $d_{\text{left}}(p_k, s)$ denotes the duration of the first s states of p_k, then the duration difference cost is given by

$$w_{\text{durdiff}} \frac{|d_{\text{left}}(p_k, s) - d_{\text{left}}(p_m, s)|}{\min(d_{\text{left}}(p_k, s), d_{\text{left}}(p_m, s))} \tag{8}$$

where w_{durdiff} is a manually set weight. An equivalent cost is added for the duration of the last $S - s$ states. The duration difference cost indirectly encourages unit continuity, since p_k and p_m in that case are identical, which results in zero cost.

The duration model is multi-dimensional Gaussian distribution[13], and the state duration penalty is implemented by a simple modification to the log-likelihood computation so that each term, f_i is weighted by a function $g(f_i)$. If the i'th term, f_i is given by

$$f_i = \frac{(x_i - \mu_i)^2}{\sigma_i^2}, \tag{9}$$

then the weighted term becomes

$$f_i g(f_i) = f_i \left(1 + w_{\text{statedur}} \max\left(0, \sqrt{f_i} - b\right)\right)^2, \tag{10}$$

where w_{statedur} is the associated weight and b is a constant controlling the amount of variation accepted with zero cost.

4.3. Unit concatenation

The selection system finds the join position in terms of fixed rate frame boundaries and so the actual join point needs to be refined. For this purpose we search for the pitch mark closest to the given frame boundary and use a very simple time domain overlap-add scheme to join the units. No pitch modification is done, except for harmonizing a small number of pitch periods around the join position.

No other speech modification is performed.

5. Evaluation

The Blizzard Challenge 2010 participants were invited to solve a number of tasks, which differed in terms of speech database sizes, noise conditions, sampling rate and language. For British English, the following tasks were available:

- EH1: a 5 hour speech database of about 4000 sentences
- EH2: a 1 hour speech database of about 1000 sentences
- ES1: 100 sentences, a subset of EH2
- ES2: speech in noise task, using the same data as EH1
- ES3: higher sampling rate, using the same data as EH1

Ideally we would have liked to address all tasks except ES2 which, given the nature of our system, was out of reach. Unfortunately, due to time issues we were only able to solve the EH1 task.

5.1. Implementation details and run-time issues

The system has just recently been developed, and tested on a single Norwegian male speech database prior Blizzard. As described in section 4, the unit selection algorithm uses a number of heuristics and manually chosen weights. These were found and set by the developers based on trial-and-error experiments on the Norwegian database. A key goal in this process was to make the system accept natural variation, but reject outliers possibly related to training data and automatic training methods. No formal listening tests have been conducted.

Some experimentation with the Blizzard training data was done and we found that the heuristics and weights based on the Norwegian data were reasonable for the Blizzard data. This suggests that the system is quite robust with respect to input data, which is a very useful feature.

As we do not have a frontend for English, we used the already pre-processed utterances included in the Blizzard training and test data, with an automated transformation that merged multiple consecutive pause labels into a single label. No other work was done in order to improve the quality of the labels, and so any errors possibly present went directly into the training process. Another implication of using the Blizzard-labels were that it limited the set of label features used to construct HTS-labels, which in turn affects the model clustering. The labels were however quite rich with features and presumably well suited for the task.

Since the run-time system uses the complete set of feature vectors used for training the HSMM model set, the memory footprint is very large. For the Blizzard dataset, the feature vectors takes up about 1.2GB of data. At run-time, these data are loaded into a RAM drive and, when adding memory requirements for the model set and metadata, the memory footprint totals to about 2GB.

The unit selection algorithm, when applied on the 1597 utterances in the Blizzard test set, runs 4.97 times faster than real-time on a 2.66 GHz Mac Pro 1.1. When adding unit concatenation, the number is 3.79. This is not an impressive result, but it shows that it should be possible to achieve acceptable performance if optimizations are implemented, such as e.g. pre-computing costs and candidate sets.

If the system is to be targeted towards home-users, both memory footprint and real-time performance needs to be improved. This was however not a factor when the system was designed.

5.2. Evalutation results

The EH1 task was evaluated in terms of 1) similarity with original speaker, 2) MOS scores for naturalness and 3) word error rate, shown in figure 4, 5 and 6 respectively. Table 1 shows the systems that are judged as *not* significantly different from our system. System A identifies the natural speaker while B and C represent two benchmark systems. Our system is identified by T.

Similarity	B F J M P U
MOS	J
WER	B C F G J L M N O P Q R U V

Table 1: *Wilcoxon's signed rank tests judge these systems as* ***not*** *significantly different from our system T.*

It is not surprising that the system has a strong similarity with the natural speaker, as the box-plot in figure 4 indicates, since we are doing plain unit selection without any signal processing or compression. However, the significant difference when compared with the natural voice tells us that we did not succeed in making an indistinguishable synthetic replica of that speaker. It should be noted that this is not the most important design factor for us. We are more interested in a natural and comprehensible voice than a direct copy of one specific person.

The MOS-scores for naturalness show that our system performs well and, according to table 1, only system J is not significantly different from ours. This tells us that the system does

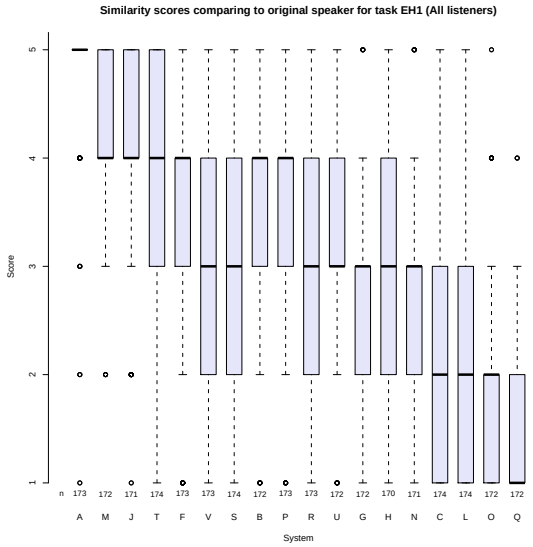

Figure 4: *Similarity with original speaker – all listeners.*

quite a good job at selecting units that fit well together. Again, there is a gap to the original speaker.

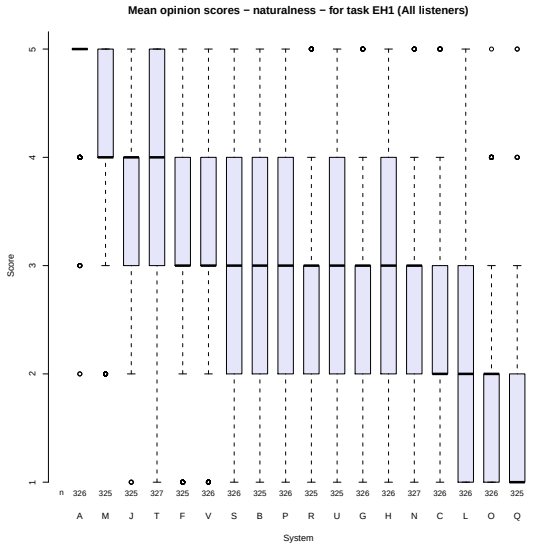

Figure 5: *MOS scores for naturalness – all listeners.*

In terms of word error rates, the system is judged not significantly different to most of the other systems, but significantly different when compared with the original speaker. This means that there is still issues that needs to be addressed. Our own impression is that, even though the quality in many cases is quite good, the generated speech still lack the consistency of the HMM-based synthesis. A part of this discrepancy may be attributed to sub-optimal cost functions, which for this system was found by trial and error. A re-examination of the source code revealed a bug related to the penalty added for outlier state-durations; it actually penalized good candidates. After fixing this we observed that some of the most audible errors vanished. However, there are still major issues that need to be investigated.

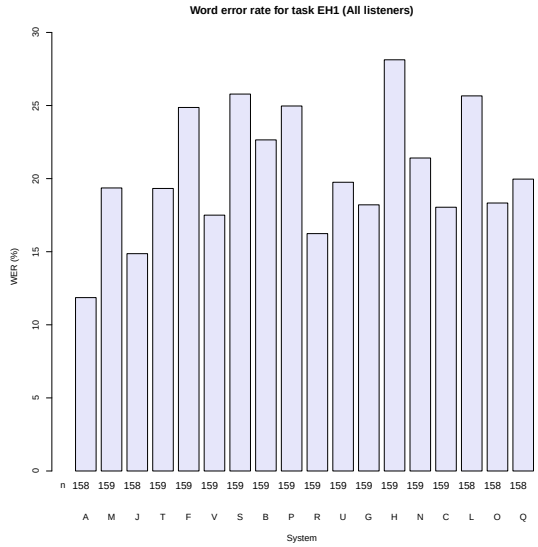

Figure 6: *SUS word error rates – all listeners.*

6. Conclusions

This paper described the NTNU entry to the Blizzard Challenge 2010. The system is a result of experiments related to combining HMM-based speech synthesis with unit selection, and is described as a conceptually simple HMM-based unit selection system, using flexible diphones with run-time optimized join point decisions.

The evaluation results show that the system performs well when compared with the other systems, but there is still a significant gap when compared with the original speaker. Thus improvements are needed in order to lessen the difference.

7. Acknowledgements

We would like to thank Lingit AS (www.lingit.no), a Norwegian company developing reading and writing support software, and the people there for valuable support and feedback during the process of making this system.

We would also like to thank the developers of publicly available tools including HTK, HTS, SPTK and Festival. Without these, it would have been practically impossible to develop our system.

8. References

[1] A. W. Black, H. Zen, and K. Tokuda, "Statistical parametric speech synthesis," in *Proc. ICASSP, 2007*, 2007, pp. 1229–1232.

[2] R. E. Donovan, A. Ittycheriah, M. Franz, B. Ramabhadran, E. Eide, M. Viswanathan, R. Bakis, W. Hamza, M. Picheny, P. Gleason, T. Rutherfoord, P. Cox, D. Green, E. Janke, S. Revelin, C. Waast, B. Zeller, C. Guenther, and J. Kunzmann, "Current status of the ibm trainable speech synthesis system," in *Proc. ICSLP*, 2001, pp. 1703–1706.

[3] R. E. Donovan, "A new distance measure for costing spectral discontinuities in concatenative speech synthesisers," 2001.

[4] A. W. Black, C. L. Bennett, B. C. Blanchard, J. Kominek, B. Langner, K. Prahallad, and A. Toth, "Cmu blizzard 2007: A hybrid acoustic unit selection system from statistically predicted parameters."

[5] H. Lu, Z.-H. Ling, M. Lei, C.-C. Wang, H.-H. Zhao, L.-H. Chen, Y. Hu, L.-R. Dai, and R.-H. Wang, "The ustc system for blizzard challenge 2009," in *Blizzard Challenge Workshop*, 2009.

[6] M. Schröder, M. Charfuelan, S. Pammi, and O. Türk, "The mary tts entry in the blizzard challenge 2008," in *Blizzard Challenge Workshop*, 2008.

[7] Z. Ling and R. Wang, "Hmm-based unit selection using frame sized speech segments," in *Proc. Interspeech*, 2006, pp. 2034–2037.

[8] H. Zen, K. Tokuda, T. Masuko, T. Kobayashi, and T. Kitamura, "Hidden semi-markov model based speech synthesis," in *in Proc. of ICSLP, 2004*, 2004.

[9] D. Meen, T. Svendsen, and J.-E. Natvig, "Improving phone label alignment accuracy by utilizing voicing information," in *Proc. SPEECOM*, 2005, pp. 683–686.

[10] HTS, http://hts.sp.nitech.ac.jp.

[11] K. Tokuda, T. Masuko, N. Miyazaki, and T. Kobayashi, "Multi-space probability distribution hmm," in *IEICE Trans.*, 2002.

[12] HTK, http://htk.eng.cam.ac.uk.

[13] T. Yoshimura, K. Tokuda, T. Masuko, T. Kobayashi, and T. Kitamura, "Duration modeling for hmm-based speech synthesis," in *Proc. ICSLP*, 1998.