Joint Target and Join Cost Weight Training for Unit Selection Synthesis

Lukas Latacz, Wesley Mattheyses, Werner Verhelst

Department ETRO-DSSP, Vrije Universiteit Brussel, Belgium
Interdisciplinary Institute for Broadband Technology – IBBT, Belgium
{ llatacz, wmatthey, wverhels } @ etro.vub.ac.be

Abstract

One of the key challenges of optimizing a unit selection voice is obtaining suitable target and join cost weights. In this paper we investigate several strategies to train these weights automatically. Two training algorithms are tested, which are based on an acoustic distance that approximates human perception: a modified version of the well-known linear regression training and an iterative algorithm that tries to minimize a selection error. Since a single, global set of weights might not result in selecting all the time the best sequence of units, we investigate whether using multiple weight sets could improve the synthesis quality.

Index Terms: speech synthesis, unit selection, weight training

10.21437/Interspeech.2011-120

1. Introduction

Unit selection synthesizers [1] are able to generate high quality speech, which can be nearly indistinguishable from natural speech. The synthesized speech is created by concatenating units that were selected from a pre-recorded speech database. Unit sequences are evaluated on the basis of the weighted sum of target and join costs; these take into account the suitability of a unit in a given target context and the smoothness of a join, respectively. The target and join cost weights have a substantial effect on the synthesis quality. Typically, these weights are set manually, through several listening experiments and expert knowledge. Although training "by hand" can result in high quality synthesis, it is also quite labor-intensive and non-trivial. Furthermore, only a limited amount of sentences can be realistically used in these experiments, leading to potentially suboptimal weights.

Several algorithms have been proposed that involve human judgments [2] - [3] to automatically set the weights, typically by ranking or comparing synthesized words or sentences. Even though it might be easier for non-experts to tune a voice using these methods as they do not need to modify the weights directly, these methods face similar problems as manually tuning the weights such as being time-consuming and allowing only a limited amount of training sentences to be taken into account. Furthermore, it is still necessary that the people involved in the experiments have a certain amount of proficiency in the language of the synthetic voice.

Other approaches have also been proposed that allow fully automated voice-building based on the idea that the best set of weights generates utterances that are the closest to natural speech. Therefore, the weights can be automatically trained by minimizing the acoustic distance between the synthetic and natural speech [1], [4] - [6]. By calculating distances between acoustic targets and corresponding "pairs" of units [4], it becomes possible to train both the target and join cost weights.

In our experience with manual weight training, the optimal set of weights for a given sentence is rarely the optimal set of weights for another sentence. Obviously, it becomes harder to find a good set of weights when more sentences need to be taken into account. There are several reasons a single set of weights does not result in optimal synthesis for all sentences. For example, whether a join discontinuity at diphone boundaries can be perceived is partly influenced by the broad phonetic class [7] (e.g. unvoiced strong fricatives can be joined rather easily). This means that if only acoustic join costs are used, the join cost weights should be chosen dependent on the phonetic class. Also context-dependent effects can play a role, such as the influence of /r/ before and after vowels, and the increased sensitivity of sonorants to position and prominence compared to non-sonorants [8]. Furthermore, target costs often depend on predicted acoustic parameters. When the prediction of one of these parameters is less reliable, the weight of the corresponding component of the target cost should be lowered. Therefore, only with multiple sets of weights the synthesis quality can be maximized.

When using a weight training algorithm, we can train different sets of weights for different targets, e.g., one for each (di)phone. Thus, different sets of weights are trained for different target *clusters* which need to be specified beforehand. However, it is not clear how to define an optimal set of clusters a priori. A better solution is to generate these clusters dynamically, based on the speech database, as in [3] or [4]: the clustering is performed by constructing a decision tree containing phonetic questions about the targets. Targets that correspond to the same phonemic sequence share the same weights.

Recently, we have developed a weight training framework to automatically train *context-dependent* target cost weights [9]. As such, each target receives a custom set of weights. In this paper we extend this framework to also allow joint target and join cost weight training based on a perceptually optimized acoustic distance measure. The framework currently only supports joins at *diphone* boundaries but can also be extended for other types of joins. We investigate the quality of this training in both subjective and objective experiments, and compare it with manual weight training. The paper is structured as follows: the weight training framework is described in section 2 and section 3 describes our experiments. Finally, section 4 concludes the paper.

2. Weight training framework

In order to train a *single* set of weights the following steps are performed:

- For each unit of the database: treat this unit as the *target* unit.

Copyright © 2011 ISCA

- Select pairs of units from the speech database; the concatenation of the two units of a pair should result in the same phonemic sequence as the target.
- Calculate acoustic distances and target costs between the target and the selected pairs. For each pair, calculate join costs between the units of the pair. The smoothness of the join is objectively measured by calculating the acoustic distance between the last phone of the first unit, and the first phone of the second unit. The distance is set to 0 if the two units are adjacent in the speech database.
- Train weights using the calculated costs and the weighted sum of the *target* and *join* acoustic distances. Two training algorithms are described in section 2.2.

In our current set-up, the target units correspond to phone-sized units, and the selected pairs correspond to demiphone pairs.

2.1. Acoustic distance

The distances are calculated between the target unit and the unit pairs as follows. As a demiphone is a relatively small unit, for the acoustic distance calculation, *complete* phones are used instead. Since a unit is not uttered in isolation, the neighboring units are also partly taken into account. Let u_t be the target unit and $u_i - u_j$ a pair of (phone-sized) units. The distance $D(t,i,j)$ between these units is then calculated as:

$$D(t,i,j) = \frac{D_{\text{target}}(t,i) + D_{\text{target}}(t,j)}{2} + w_{join} D_{join}(i,j) \quad (1)$$

The weight w_{join} controls the importance of the joins relative to the target. The distances D_{target} and D_{join} are the acoustic target and join distances, respectively.

$$D_{\text{target}}(t,i) = D(t,i) + D(i,t) \quad (2)$$

$$D(t,i) = w_l D'(t-1,i-1) + D'(t,i) + w_r D'(t+1,i+1) \quad (3)$$

$$D'(t,i) = \frac{w_{dur} D_{dur}(t,i)}{s_{dur,t}} + \frac{w_{f0} D_{f0}(t,i)}{s_{f0,t}} + \frac{w_{spec} D_{spec}(t,i)}{s_{spec,t}} \quad (4)$$

The weights w_l and w_r take the left and right neighboring units into account, while w_{dur}, w_{f0} and w_{spec} correspond to the duration, f0 and spectral weights, respectively. The distances D_{dur}, D_{f0} and D_{spec} are normalized by a scaling factor which is calculated as the average distance between unit u_t and all units sharing the same phonemic sequence. Let dur_t and dur_i be the durations of units u_t and u_i, respectively. $D_{dur}(t,i)$ is then calculated as:

$$D_{dur}(t,i) = \left| \frac{dur_t - dur_i}{dur_t} \right| \quad (3)$$

The other two distances $D_k(t,i)$ are calculated as the relative root mean squared error. Let f_j and g_j be acoustic features for the j^{th} frame of unit u_t and its corresponding frame in unit u_i, as found by straight-forward linear time scaling, respectively. $D_k(t,i)$ can then be calculated as:

$$D_k(t,i) = \sqrt{\frac{1}{N} \sum_{k=1}^{N} \frac{\|f_k - g_k\|^2}{\|f_k\|^2}} \quad (4)$$

with N the number of frames of the target unit u_t. In our system, features are calculated each 5 milliseconds. 40 MFCCs extracted from the STRAIGHT spectrum [10], are used for the spectral distance measure. Unvoiced frames are discarded when calculating the *log f0* distance. The *join* distances $D_{join}(i,j)$ between units u_i and u_j are calculated similarly as the target distances, with the following two exceptions. As the neighboring units play a limited role the weights w_l and w_r are set to 0. Also, because we are mostly interested in the region were the actual join occurs, a Hann window is used for the f0 and spectral distances when calculating the join distances:

$$D_{k,join}(t,i) = \sqrt{\frac{1}{N} \sum_{k=1}^{N} \frac{1 - \cos(\frac{2\pi(k-1)}{N-1})}{2} \frac{\|f_k - g_k\|^2}{\|f_k\|^2}} \quad (5)$$

The weights of the acoustic distance measure were optimized perceptually, using the CMU Arctic SLT database. The quality of an acoustic distance measure can be measured by resynthesizing utterances from the database: the best sequence of units should correspond to speech close to the original utterance. In our case, we select diphones using a greedy algorithm: for each target diphone, the best matching diphone is selected from the database. Units from the original sentence are not allowed to be selected. In order to train the weights of the distance measure, we resynthesized a small number of utterances using different settings, and choose these settings which yielded the best performance: $w_l = 1$, $w_r = 1$, $w_{dur} = 4$, $w_{f0} = 2$, $w_{spec} = 10$, $w_{join} = 1$. An alternate version of the distance measure also included differences in energy, but this did not result in a better perceptual quality.

2.2. Training algorithms

Currently, two training algorithms are implemented in this framework: a modified version of the well-known linear regression training [1] and an iterative algorithm that tries to minimize a certain selection error.

2.2.1. Linear regression

The quality of a set of target and join cost weights can be determined by calculating the root mean square error (RMSE) between the weighted sum of calculated costs and the acoustic distances. Therefore, linear regression [1] can be applied to minimize the error between the predicted and calculated distances. However, this might result in negative weights; costs having negative weights do not contribute well to selecting better units: if a cost has a negative weight then this means that a unit which is actually *bad* in terms of this target cost will be preferred to a unit that is in fact performing *better* for this target cost. On the other hand, setting those weights to 0 might result in a suboptimal solution (in terms of least-square errors). Therefore, we propose the following iterative algorithm. After calculating the weights using linear regression, linear regression is performed again taking only those costs having non-zero weights into account and calculating the RMSE. If the error has decreased, iteration continues by again setting the negative weights to zero and performing linear regression using the costs that have non-zero weights. Iteration stops if the error increases and the set of weights with the lowest error is chosen as the optimal one.

2.2.2. Minimum selection error

In order to select the best sequence of units, it is important to know whether a unit is better or worse compared to another unit. If a unit is acoustically closer to the target than another unit, then this should be reflected in the weighted sums of costs of those two units, as these sums can be seen as an

approximation of the acoustic distances between the units and the target. Let M be the number of unit *pairs* u_i - u_i' which are available for a target t. The quality of a particular set of weights W for a given target t can then be measured by a selection error *serr(W,t)*, defined as

$$serr(W,t) = \sum_{i=1}^{M}\sum_{j=1}^{M} err(u_i,u_i',u_j,u_j',W,t) \quad (7)$$

with $err(u_i,u_i',u_j,u_j',W,t)$ an error measure between two units pairs. Let c_i be the weighted sum of the costs of the unit pair u_i - u_i' for the target t using weights W and $d_i = D(t,i,i')$ the acoustic distance between the target and the unit pair. Then the error measure $err(u_i,u_i',u_j,u_j',W,t)$ is set to 0 if sign(c_i - c_j) = sign(d_i -d_j), and equal to $\left| d_i - d_j \right|$ otherwise.

The n weights for a given target unit can be iteratively trained by greedy selection. The algorithm starts by using default weights, i.e. all weights set to 1. By using the selection error (7), we can measure the quality of a particular weight set. At each iteration, we evaluate various weights sets, and keep the one yielding the lowest error. Note that, each time only a single weight value is modified. Iteration will stop if no improvement can be found or if a predetermined maximum of iterations is exceeded. Finally, the weights are scaled in order to minimize the difference between the costs c_i and the acoustic distances d_i.

2.3. Multiple weight sets

2.3.1. Pre-defined target clusters

Different sets of weights can be trained by using different subsets of the available units in the speech database. These subsets need to be defined beforehand. Currently our system supports phone-dependent weights, separate weights for unvoiced and voiced phones, and global weights.

2.3.2. Context-dependent weights

It is also possible to train *separate* sets of weights for each unit in the speech database. In order to generalize weights for unseen targets, the trained weights are clustered using decision trees. As the weights are not independent (since they are trained together), we jointly cluster the weights. Standard techniques are used to build the decision tree: at each split, the best question is chosen in order to minimize an *impurity*. The splitting stops if no improvement can be found or if the number of elements in a cluster would become smaller than a given stop size. The optimal value of the stop size can be found using cross-validation. In our case, separate trees are constructed for each phoneme, which are merged afterwards in a single tree. This allows using different stop sizes for different phonemes, which is quite valuable since not all phonemes have the same distribution in the database.

In our case, the impurity is based on an error measure that measures how well the weights are suited for a particular combination of acoustic distances and costs. The error measure is related to the actual training algorithm used: for the linear regression we use the RMSE between the acoustic distances and the costs, while for the minimum selection error algorithm the selection error (7) is used. The impurity is calculated as follows. Let L be the number of weight sets in a tree node. As each set of weights was trained for a different target, this tree node has also L target units associated with it. The quality of a set of weights for a given target unit can be estimated by the error measure. Let W_{best} be the set of weights which has the lowest average error over all target units in the tree node. The impurity is then calculated as

$$L \cdot (\mu_{best} + \sigma_{best}\alpha) \quad (8)$$

with μ_{best} the mean and σ_{best} the standard deviation of the L selection errors calculated using W_{best} and the L target units. α is a parameter that needs to be tuned (e.g. $\alpha = 2$). Thus, the impurity can be considered an estimate of the upper bound of the sums of errors.

After constructing the tree, each leaf node needs to contain a single set of weights instead of L, as each target needs a single set of weights during synthesis. In our case, we select the set of weights W_{best} that gives the lowest average error. We also experimented with averaging the weight sets and building the separate trees using regression trees (one for each weight), but these resulted in lower performance in general.

2.4. Implementation

As there are typically many instances of each phoneme in the database, we only use a subset of these instances (with size N_{target}) in order to speed up the weight training. This subset is selected by uniformly sampling the phoneme instances based on their acoustic distance with the target unit. As such, a similar but smaller distribution of acoustic distances is obtained, which reduces the computational load significantly while maintaining the quality of the trained weights. By selecting N_{target} units from the database, $(N_{target})^2$ pairs can be constructed. The number of pairs used for training can be limited to N_{join} using a similar procedure when the acoustic distances between the target unit and each of the pairs are taken into account. Also, as the pairs $u_i - u_l$ and $u_l - u_i$ share the same acoustic distance to the target unit, we only use one of these pairs.

3. Experiments

3.1. Setup

The weight tuning framework is implemented as part of our unit selection synthesizer [9]. HMM models were trained using the procedure described in [9], and are used during synthesis to generate smooth parameter trajectories. Three target costs were used, taking spectrum, f0 and duration into account: the average Mahalanobis distance is used between the predicted and unit parameters. Corresponding frames are found using straight-forward linear time scaling and the variances needed for the distance are taken from the HMM models. Two join costs were used, taking spectrum and f0 at the join boundary into account using the Mahalanobis distance between the delta features resulting from the concatenation and the mean delta features of the Gaussian model at the join boundary, with the variances of this model. As the costs have different ranges of values, a scaling factor is computed for each cost, which corresponds to the 95th percentile of the values of the cost. The following settings were used to train the weights: N_{target} and N_{join} were both set to 100. In order to speed up the training, the number of target units for each phoneme was also limited (N_{max} = 1000). The same symbolic features as in [9] were used to construct the trees for context-dependent weight training. Possible stop sizes for cross-validation were 5, 10, 50, 100 and 500; 5 folds were used.

The following databases were used to test the training: the female US English CMU Arctic SLT database (50 min.), the male UK English "RJS" database used in the 2010 Blizzard Challenge (298 min.), and a subset of a female Flemish audio-visual database "Kaat" (129 min.).

3.2. Objective experiments

In an objective experiment we investigated the influence of the actual training algorithms using different *types* of weights on the synthesis quality. As a "baseline" we carefully tuned global weights by hand. 50 sentences of each database are used as a test set; these sentences were not used during weight or HMM model training. The test sentences were synthesized using different weight settings: manual trained weights and context-dependent, phone-dependent and global weights trained using the two training algorithms. In our objective test, the *naturalness* was measured using the acoustic distance (1) with $w_{join} = 0$, while the *smoothness* was measured by taking only the joins into account. The distances are calculated between natural and synthesized phones and averaged, excluding adjacent units for the smoothness measure. Results are shown in table 1. When training using the linear regression algorithm, using multiple sets of weights results in a better quality. Using phone-dependent or context-dependent weights for the female voices results in a better (objective) quality compared to manually set weights. The global weights trained using the minimum selection error algorithm are quite good. Even more, using more sets of weights in combination with this algorithm does not always result in better synthesis. This might be due to the greediness of the algorithm (stuck in local minima). Nevertheless, the use of context-dependent weights yields better objective quality for all tested voices.

Voice	*Manual*		*Global*		*Phone-dep.*		*Context-dep.*	
SLT	10.32	12.99	10.48	12.40	10.05	12.29	10.01	12.30
RJS	9.94	11.29	10.8	11.56	9.63	11.64	9.66	11.46
Kaat	8.45	11.28	8.66	11.63	8.33	10.81	8.31	10.79

Voice	*Manual*		*Global*		*Phone-dep.*		*Context-dep.*	
SLT	10.32	12.99	9.97	12.18	10.02	12.16	10.05	11.88
RJS	9.94	11.29	9.92	10.37	9.79	11.41	9.74	10.99
Kaat	8.45	11.28	8.51	11.28	8.42	11.04	8.40	10.48

Table 1. *Objective results using linear regression (top) and minimum selection error (bottom) weight training. Naturalness (left) and smoothness (right) are measured and smaller numbers meaning better results.*

	> Manual	*= Manual*	*< Manual*	*P-value*
CD – MSE	37%	23%	40%	0.51
CD – LR	23%	28%	48%	0.08
PHONE – MSE	30%	22%	48%	0.16
PHONE – LR	43%	23%	33%	0.68

Table 2. *Subjective results (voice: Kaat).*

3.3. Subjective experiments

In a subjective experiment we evaluated the quality of 4 types of automatically trained weights: context-dependent (CD) and phone-dependent (Phone) weights using the two training algorithms (LR, MSE). Due to timing constraints, only the Dutch voice was used. 10 sentences were synthesized using the different weight settings, including manually tuned weights. 6 listeners, all Dutch native speakers, participated in this experiment. They were given 100 pairs of synthesized sentences and were asked to compare the *overall quality* of the synthesized sentences using a comparative mean opinion score [-2, 2]. Conditions were the same for all listeners (i.e. quiet room, headphones). Statistical analysis was done using a Wilcoxon signed-rank test. For all comparisons between phone- and context-dependent sets of weights, we found a 38 % preference for the phone-dependent weights, versus a 29% preference for the context-dependent weights. This difference was found to be significant ($p = 0.013$). No difference was found between the two training algorithms. Although no significant differences could be found between the manually and automatically trained weights, a minor preference for manually trained weights was found to be a global tendency, as can be seen in table 2.

4. Conclusion

In this paper we presented algorithms to automatically train both target and join costs, which is useful to improve and speed-up the voice building process. Objective results show that training a global set of weights using the minimum selection error algorithm results in either an improvement over carefully tuned manual weights or quite close results; improvements for all voices could be found when context-dependent weights are used in combination with this algorithm. In the subjective test, the automatically trained weights were almost as good as manually training weights. Still, the automatic training algorithms have the advantage that no manual input is needed. These algorithms use an acoustic distance of which the parameters have been manually tuned, but the results demonstrate that these are fairly speaker-independent. Still, the differences between the subjective and objective experiments might be partly due to a mismatch between the objective acoustic distance and the perceptual distance. Also, the subjective test was limited in terms of sentences, so other factors (e.g. segmentation errors) could have played a role too.

5. Acknowledgements

The research reported on in this paper was partly supported by the projects IWT-SPACE, IBBT-SEGA and EC FP7 ALIZ-E (FP7-ICT-248116).

6. References

[1] Hunt, A. J., and Black, A. W. "Unit selection in a concatenative speech synthesis system using a large speech database", Proc. ICASSP '96, pp. 373–376, 1996.

[2] Peng, H., Zhao, Y., and Chu, M. "Perpetually optimizing the cost function for unit selection in a TTS system with one single run of MOS evaluation", Proc. ICSLP '02, 2002

[3] Alías, F. et al. "Perception-Guided and Phonetic Clustering Weight Tuning Based on Diphone Pairs for Unit Selection TTS", Proc. ICSLP '04, pp. 1333–1336, 2004

[4] Meron, Y., and Hirose, K., "Efficient weight training for selection based synthesis", Proc. Eurospeech '99, Vol. 5 pp. 2319–2322, 1999

[5] Alías, F., and Llorá, X. "Evolutionary weight tuning based on diphone pairs for unit selection speech synthesis", Proc. Eurospeech '03, Vol. 2, pp. 1333–1336., 2003

[6] Park, S. S., and Kim, N. S. "Discriminative weight training for unit-selection based speech synthesis". Proc. Eurospeech '03, 2003.

[7] Syrdal, A. K., and Conkie, A. D., "Perceptually-based data-driven join costs: comparing join types", Proc. Interspeech '05, Lisbon, 2005

[8] Coorman, G., Fackrell, J., Rutten, P., and Van Coile, B., "Segment selection in the L&h Realspeak laboratory TTS system", Proc. ICSLP-2000, vol.2, 395-398, 2000

[9] Latacz L., Mattheyses W., and Verhelst W., "The VUB Blizzard Challenge 2010 Entry: Towards Automatic Voice Building", Blizzard Challenge 2010, Kansai Science City, Japan, 2010

[10] Kawahara, H., Masuda-Katsuse, I., and Cheveigne, A., "Restructuring speech representations using a pitch-adaptive time-frequency smoothing and an instantaneous-frequency-based f0 extraction: possible role of a repetitive structure in sounds," Speech Communication, vol. 27, pp. 187–207, 1999