The VUB Blizzard Challenge 2010 Entry: Towards Automatic Voice Building

Lukas Latacz, Wesley Mattheyses, Werner Verhelst

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

Abstract

In this paper we describe the voices we submitted to the 2010 Blizzard Challenge, a yearly challenge to evaluate auditory speech synthesis on common data. One of the goals of a data-driven synthesizer, such as ours, is to generalize the speech database in such a way that it allows a realistic rendition of unseen input text. The two main changes to our system, compared to previous submissions, are the inclusion of an HMM-based acoustic prosody model, and the automatic training of *context-dependent* target cost weights. These weights are estimated for each individual target during synthesis, and depend on the linguistic features of these targets which encompass their broader linguistic context. Another new aspect of our synthesizer is the ability to synthesize Mandarin Chinese speech. Its evaluation helps us assess the quality of our synthesizer for languages unfamiliar to the voice developers. Evaluation results and possible improvements to our synthesizer are also discussed.

Index Terms: speech synthesis, unit selection, weight training, evaluation

1. Introduction

The Blizzard Challenge is a yearly speech synthesis challenge to evaluate auditory speech synthesis on common data. For UK English, two speech databases were provided this year: a large database (RJS) consisting of about 4000 sentences, and a smaller one (Roger), which was also used in previous challenges and which is based on the ARCTIC corpus [3]. A version of the RJS database with a 48 kHz sampling rate was also provided by the Blizzard organizers. The other databases were only available with a 16 kHz sampling rate.

The VUB team has already participated in previous challenges with its DSSP synthesizer (2008 [1], 2009 [2]). This synthesizer is a unit selection-based synthesizer that uses non-uniform units (phrases, words, syllables and diphones or phones). The same UK English front-end as last year's challenge has been used for the UK English voices. In this paper, we focus mainly on the changes we made to our synthesizer. In contrast to our previous entries, this years' entry also includes Mandarin Chinese voices. Although this is a language that we are quite unfamiliar with, it was still possible to produce the two required Mandarin voices thanks to the supplied HTS labels for Mandarin, which provided both segmentation and linguistics features. This illustrates the ability of our synthesizer to support new languages relatively easily. Both Mandarin voices were based on the same database, except that only a subset of the database (800 sentences) was used for one of them.

This paper is structured as follows. In section 2 we give an overview of the voice building process of the English voices. In section 3 we describe the differences between the Mandarin and English voices and how the Mandarin voices were build. Results and discussion of the Blizzard Challenge evaluation are given in section 4. Section 5 concludes our paper.

2. Voice building

Data-driven speech synthesis, such as unit selection, requires a large speech database (at least 1 hour; up to 6 hours of speech and more) in order to produce high-quality synthesized speech. This large amount of high-quality speech from a single speaker is needed to train various aspects of the synthesizer, such as prosody models and acoustic models for segmentation. By using the speech database as effectively as possible, the synthesized speech can become closer to the original. This approach is also one of the main characteristics of the DSSP synthesizer. As most of the training algorithms require little or no human input, the construction of a new voice for our synthesizer is largely automated. Our voice building process can be summarized as follows. Firstly, the speech database is labeled and segmented, and acoustic features such as MFCC's, f0 and energy are calculated for each utterance of the database. These features are used, for example, to compute the join cost. Secondly, based on the labeled pauses in the database and linguistic features extracted from the text transcription of the utterances in the database, a silence and phrase break prediction model is trained [1] (this model is part of the synthesizer's front-end). Each segment (i.e. each phone) of the database is labeled with various linguistic features. These features are used to generate a multi-level segmentation of each utterance: each utterance is segmented into phrases, words, syllables and phones. The next step is to train prosody models and trainable target and join costs. Finally, the voice needs to be tuned by choosing appropriate synthesis parameters, such as the target and join cost weights. In the remainder of this section, we focus in more detail on some of these aspects.

2.1. Segmentation

Accurate segmentation is a crucial aspect of building a high quality voice. Our approach is to use the speech database and the corresponding text to train acoustic models for an HMM forced aligner, and to use these models to segment and label the database. For this year's challenge, the organizers provided also a manually corrected segmentation for the Roger voice. Although it would be interesting to know the difference in terms of synthesis quality between an automatically generated segmentation and a manually corrected version, due to timing constraints we were unable to build a version of the Roger voice using the manually corrected segmentation.

Like in last year's submission, we used the open-source speech recognition toolkit SPRAAK [4] to segment and label the utterances. A standard 5-state left-to-right context-independent phone model was used, with no skip states. Speech was divided into 25 ms frames with 5 ms frame-shift.

For each frame, 12 MFCC's and their first and second order derivatives were extracted. The acoustic models generated for last year's Roger submission, were used to bootstrap the acoustic model training. Based on the orthographic transcriptions and a lexicon, input files for the segmentation algorithm are created automatically. Utterances with out-of-vocabulary words are discarded and are not used in the rest of the voice building process. These words and their corresponding utterances are listed, so the voice developer can opt to manually add the missing entries, if necessary. Words having multiple pronunciations are supported, and the most likely pronunciation is automatically selected.

2.2. Prosody models

In previous versions of our synthesizer, the prosody was not predicted explicitly, but described in terms of several symbolic (linguistic) features, such as lexical stress, part-of-speech, etc. Acoustic prosody models use those features to predict the prosody in terms of acoustic parameters, such as duration and f0. Although current models are not perfect, the generated prosody is usually quite smooth and stable, and often corresponds to an *acceptable* rendition of the input text. Also, using predicted acoustic targets in unit selection might reduce the number of weights which needs to be set, as fewer acoustic target cost functions are needed to describe the prosody than when using symbolic target cost functions. Obviously, the accuracy of prosody models and the variability of natural speech should be taken into account when using the predicted prosody as the target prosody. Our system uses an HMM-based prosody model. We predict duration and f0, but the system can easily be extended to other acoustic features.

In order to train the prosody model, each phone of the database should be segmented into N states. In our case N=5 and the segmentation was obtained using speaker-dependent HMM models, generated by HTS [5]. State durations are predicted for each phone of the database using the symbolic features of the phone as input. The predicted durations for a single phone are scaled linearly to match the actual duration of the phone and each phone of the database is split into N states. The HMM forced-aligner which was used to segment and label the database could also have provided us with a segmentation into states, such that it would not have been necessary to train HTS models. However, since in SPRAAK only spectral information is taken into account, this segmentation might not be as suitable to train prosody models as HTS.

The questions that are used to split the decision trees of the prosody models are generated automatically based on the symbolic features and values present in the training database. The first step of the actual training process is to train duration models, which results in N decision trees, one for each state. Mean and standard deviation are calculated for each phone/state combination and z-scores are calculated for all states in the database. These z-scores and the linguistic features of the corresponding phone are used to train the decision trees. As these are standard regression trees [6], we use Wagon, part of the Edinburgh Speech Tools, for training. While synthesizing, the duration of each state can be predicted by obtaining the z-scores and the mean and standard deviation of the corresponding phone/state combination. The total duration of a particular phone is the sum of its state durations. The duration models are used to obtain a second segmentation of the database in states, which is used in further training.

The next part of the training process consists of training the f0 models. The database is analyzed using STRAIGHT [7] in order to calculate f0 and voiced/unvoiced information for each frame. A frame-shift of 5ms was used. While synthesizing, we need to determine whether a state is voiced or not. Therefore, classification trees [6] are being used. For each of the N states, a classification tree is trained with the voiced/unvoiced information of the frames of the training database as input. The voiced frames are used to train regression trees with Wagon, with *log* f0 and the corresponding linguistic features as input. The leaf nodes of the generated trees are expanded with the mean and standard deviation of the delta and delta-delta *log* f0 of the clustered states. The f0 contour is calculated in 5 ms steps in a similar fashion as in [8], using dynamic features (delta's and delta-delta's) and global variance.

	Roger		RJS	
	HTS	**Proposed**	**HTS**	**Proposed**
RMSE	0.0319	0.0257	0.0294	0.0246
MAE	0.0231	0.0185	0.0219	0.0179
Corr.	0.735	0.843	0.753	0.833

Table 1. *Evaluation of the duration models for the Roger and RJS voices.*

	Roger		RJS	
	HTS	**Proposed**	**HTS**	**Proposed**
RMSE	0.184	0.168	0.177	0.155
MAE	0.0935	0.0830	0.106	0.0966
Corr.	0.476	0.508	0.512	0.543
U/V	20.4 %	18.9 %	14.3 %	12.2 %

Table 2. *Evaluation of the f0 models for the Roger and RJS voices.*

The validity of our prosody model was tested in an initial evaluation and compared to the standard HTS training and prosody generation using hts_engine (both using default settings). The first 50 utterances of the RJS and Roger voices were used as a test set, and discarded from the training. The *stop* factor is a parameter that controls the size of the decision trees by indicating the minimum amount of training samples that are clustered in a single leaf node and was optimized by trial and error. A rather small stop size was found to give the best results for the duration models (*stop size* = 10). A much larger value was needed for the f0 models (*stop size* = 150 and 200 for the Roger and RJS voices, respectively). The same factor was used for the voiced/unvoiced classification trees as for the *log* f0 regression trees. Evaluation was performed by calculating the root mean square error (RMSE), mean absolute error (MAE), and correlation between the predicted values and the actual values found in the test set (Corr.). For duration, segmental duration (i.e. phone duration) was evaluated. For *log* f0, corresponding frames were found by linear scaling of the target and predicted contours. Only those instances where the predicted and target frames are both voiced were taken into account, but the percentage of frames which are incorrectly classified as voiced or unvoiced was also calculated (U/V error). Results are shown in tables 1 and 2. Although the proposed models are not that complex, an improvement over the baseline can be seen for both the duration and the f0 prediction. Similar improvements have been found with other voices we build (e.g. Mandarin Chinese voices, CMU Arctic voices …). The improvements in terms of duration are most likely caused by the use of z-scores, while the slightly better f0 contour prediction, compared to the multi-space probability distribution HMM used in HTS, is probably due to a slightly better voiced/unvoiced classification. Further improvements can probably be made by using different *stop factors* for the

training of the unvoiced/voiced classification and the *log* f0 regression trees and by incorporating dynamic features (delta's and delta-delta's) in the construction of the *log* f0 regression trees.

2.3. Target and join costs

Our system supports several types of target costs, such as costs based on symbolic features, acoustic features or statistical models. All of these types of costs were used in the construction of the submitted voices. The following *symbolic* target costs are used: next and previous phone, position of word in phrase, position of syllable in word, part-of-speech, punctuation and syllabic stress. The *acoustic* costs are based on the predicted segmental durations and f0 contour, using either an absolute Euclidean distance or a Euclidean distance between relative values (in terms of the predicted target values). Furthermore, two *statistical* costs are used, estimating the likelihood of segmental durations and f0 from the same HMM models that were used to determine the target prosody. As all these target costs have different ranges of values, a scaling factor is computed for each target cost. This scaling factor is the 95th percentile of the values for the non-symbolic costs; as the symbolic costs are either 0 or 1, they do not need scaling.

The same statistical join cost was used as last year's submission [4]; this cost models the likelihood of acoustic differences at join boundaries. The weight of this cost was tuned by trial and error, through iterative listening.

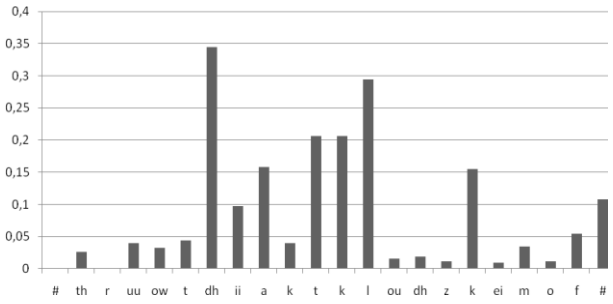

Figure 1: *Illustration of the context-dependent weight for the target cost function "duration" throughout the sentence "*Throughout the act clothes came off"*, in which the targets correspond to phones. Notice the different values of the weight for the different instances of the phones* /#/, /k/ *and* /t/.

2.4. Weight training

One of the key challenges of optimizing a unit selection voice is obtaining suitable target cost weights. Different weight values are needed because the cost functions are not all of equal importance, i.e. some may influence the synthesis quality more than others. Typically, these weights are trained manually (e.g. using expert knowledge or iterative listening experiments). One of the reasons that manual training is not that straight-forward and can be quite time-consuming, is that these weights not only depend on the particular target cost functions used, but also partly on the targets themselves. Some examples of these effects are given in [9], for example the influence of /r/ before and after vowels, and the increased sensitivity of sonorants to position and prominence compared to non-sonorants. Also, the quality of a particular target cost function might not be constant. For instance, some target costs are calculated using predicted acoustic parameters. When these predicted parameters are less reliable or accurate, the weight of the corresponding component of the target cost should be lowered. These examples suggest that, in order to yield the best possible synthesis quality, each individual target should receive a specific set of weights, optimized for that particular target. As targets are typically described in terms of linguistic features which do not only incorporate the phonemic sequence, but also the broader linguistic context of these features, we call this particular type of weights, *context-dependent* weights. An example of such weights for the target cost function that accounts for absolute duration differences, is shown in figure 1.

In order to obtain different sets of weights for different targets, we can cluster the targets and train a set of weights for each cluster, e.g. one for each individual (di)phone. As these clusters need to be specified beforehand, it is not clear how to choose an optimal set of clusters manually. A better solution is to generate these clusters dynamically as in [10] or [11]: the clustering is performed by constructing a decision tree containing phonetic questions about the targets. For each of the targets clusters - these correspond to the leaf nodes of the decision tree - a set of weights is trained. The broader context of a particular target, described in terms of linguistic features, is, however, ignored. Incorporating this broader context, or even adding additional linguistic information about the targets (e.g. lexical stress …) in the constructing of the decision tree increases the computation time tremendously.

This is one of the reasons why our approach works the other way around. Firstly, an automated weight tuning algorithm is used to generate optimal weight sets for each unit in the training database. Secondly, these weights are clustered using decision trees. This weight training framework is quite flexible in the sense that it allows experimenting with several weight training algorithms, clustering algorithms and objective acoustic distance measures. Objective acoustic distance measures are needed for fully automatic weight training, since we need to know which units are closer to each other. Although none existing measure matches human perception perfectly, using only objective measures has the advantage that more training material can be taken into account which increases the robustness of the weight training.

An optimized weight set for a particular speech unit can be calculated by treating this unit as the *target*, and selecting a set of units sharing the same phonemic sequence as candidate units. An objective distance measure can then used to calculate the acoustic distances between the target unit and the candidate units. These distances, in combination with the target costs between the target and the candidate units, can be used as input to an automated weight tuning algorithm to generate the optimal target cost weights.

As we needed to finish building the voices for this year's Challenge on time, our algorithms were adapted to speed up the training process. Firstly, as there are typically many instances of each phoneme[1] in the database, we only use a subset of these instances 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

[1] The smallest units that our system is able to select are demiphone-sized units. While synthesizing, target costs are calculated for each (demi)phone of a particular unit. As the two halves of a particular target phone share linguistic features, context-dependent weights have to be predicted for each phone of the target utterance. This implies that the target units used for the weight training will correspond to phone-sized units.

acoustic distances is obtained, which reduces the computational load significantly while maintaining the quality of the trained weights. In the remainder of this section, the term 'selected units' refers to this sampled subset. The optimal size of a subset is dependent on the amount of weights that need to be trained (in our case 100 units). Secondly, we also needed to limit the amount of units that were used as *target* unit. For each type of phoneme, at most *M* units were used as target units for which an optimal set of weights was trained. Obviously, using fewer target units for weight training may decrease the synthesis quality. For the Roger and RJS voices, this maximum was set to 200 and 1000, respectively, corresponding to 26.0% and 21.6% of the maximum amount of units available for training in the respective databases. Note that the first 50 utterances were discarded during the weight training for evaluation.

2.4.1. Minimum selection error training

An important aspect of our weight training algorithm is the algorithm that optimizes the weights for a given *target* unit. 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. Obviously, if the weighted sum of costs corresponds perfectly to a perceptual distance, an ideal comparison can be made, but in reality this is not the case. Errors that occur when matching costs to perceptual distances cause an incorrect ordering of the units, and thus in those cases a suboptimal sequence of units will be selected. In practice, an acoustic distance between two units is used instead of a perceptual distance. If a unit is acoustically closer to the target than another unit, this should be reflected in the weighted sums of costs of those two units. The quality of a particular set of weights *W* for a given target *t* can thus be measured by a selection error *serr(W,t)*, defined as

$$serr(W,t) = \sum_{i=1}^{N}\sum_{j=1}^{N} err(u_i,u_j,W,t) \quad (1)$$

with *N* the number of selected units and $err(u_i,u_j,W,t)$ an error measure between two units u_i and u_j. Let c_i be the weighted sum of target costs of unit u_i for the target *t* using weights *W* and $d_i = D(t, u_i)$ an acoustic distance between the target and unit u_i. Then the error measure $err(u_i, u_j, W)$ is set to 0 if $\text{sign}(c_i - c_j) = \text{sign}(d_i - d_j)$, and equal to $|d_i - d_j|$ 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, 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.

As the weighted sum of target costs can be seen as a prediction of the acoustic distance between a unit and its target, the weights are scaled by a constant scaling factor *f*, calculated by minimizing

$$\sum_{i=1}^{N}(f \cdot c_i - d_i)^2 \quad (2)$$

with *N* the number of selected units, and c_i and d_i as defined above.

2.4.2. Weights clustering

In order to generalize weights for unseen targets, the trained weights are clustered using decision trees. For each target cost function, a separate tree could be build and each of its leaf nodes would contains a single weight. The most straight-forward approach would be to train each tree individually, which can be done using standard regression tree training [12]. However, as the weights are not independent (since they are trained together), we propose to jointly cluster the weights. Standard techniques can be used to build the decision tree; at each split, the best question is chosen in order to minimize 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. In our case, this impurity is based on the selection error (Eq. 1) and can be calculated as follows. Let *L* be the number of *target* units in a cluster. Hence, each cluster contains *L* optimized sets of weights. The quality of a set of weights for a given target unit can be estimated by the selection error. Let W_{best} be the set of weights which has the lowest average selection error over all target units. The impurity can then be calculated as

$$L \cdot (\mu_{best} + \sigma_{best}\alpha) \quad (3)$$

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 a parameter that needs to be set (e.g. $\alpha = 2$). Thus, the impurity can be considered an estimate of the upper bound of the sums of errors. After constructing this 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. The following approaches are implemented:

- For each target cost, calculate the *average* weights over all weight sets of this node.
- Select the *best* set of weights W_{best}, that gives the lowest average selection error.

For the submitted voices, the following linguistic features were taken into account during the decision tree construction: current, previous and next phone, syllabic stress, part-of-speech, syllable position in word, word position in utterance and phrase, number of stressed and unstressed syllables of current word, punctuation, whether the current word is a *content* word, and whether the current word is capitalized.

In order to select the best clustering parameters, the 50 first sentences of each voice were used as a test set. These best settings and corresponding weight trees were used in the further construction of the voice. Table 3 lists the average selection errors on the test sets of the RJS voice, for the two clustering approaches and various *stop* sizes (that control the size of the decision trees). Selecting the *best* set of weights to represent the weights of a given cluster of target units yields better results compared to *averaging* the weights. A large stop size was necessary for the best results; this can probably be explained by the fact that we were unable to use all target units available in the database for training the weights. Although not listed here, we also tested whether training separate weight decision trees for each individual target cost yielded better performance compared to the joint clustering, but this resulted in errors roughly comparable to the ones obtained by averaging the weights in a single cluster.

2.4.3. Acoustic distances

When comparing two units acoustically, several distinct properties of the acoustic signals should be taken into account, such as duration, f0, spectrum, energy … As differences in

these properties can be measured by acoustic distance measures, they can be combined in a single distance measure using, for example, the weighted sum of the *individual* measures. As the construction and optimization of this single combined measure would require human input (e.g. subjective experiments), we choose to use a different approach that is fully automatic. We use *M* acoustic distance measures to train several sets of context-dependent weights, one set for each distance measure. We need to combine these different sets of "base" weights, as one single set of weights must be used during synthesis. We assume that the optimal set of weights for a specific target is a linear combination of its base weights sets. The latter weights are termed *combination* weights and are further explained at the end of this section.

Stop		5	10	20	50	100	200
Dur.	Avg.	400.4	397.0	396.1	397.1	397.4	398.8
	Best	390.6	385.0	379.0	373.0	372.3	383.2
F0	Avg.	258.2	255.9	255.2	255.0	255.4	257.4
	Best	235.6	230.7	228.1	225.9	224.3	224.2
Spec.	Avg.	381.9	382.4	383.5	384.8	385.2	385.7
	Best	379.9	378.5	376.7	375.9	376.0	376.1

Table 3. *Average selection errors for the test set of the RJS voice. In comparison, when default weights (all weights set to 1) are used, the average selection errors are* 512.2*,* 468.5 *and* 402.9*, for duration, f0 and spectrum, respectively.*

In our case, acoustic distance measures for duration, (log) f0 and spectrum were used. Let *t* be the unit corresponding to the target context and *u* a unit from the selected unit set. Let dur_t and dur_u be the durations of the target and the selected unit, respectively. The duration distance $D_{dur}(t, u)$ is then calculated as:

$$D_{dur}(t,u) = \left| \frac{dur_t - dur_u}{dur_t} \right| \quad (4)$$

The other distances $D_{f0}(t, u)$ and $D_{spec}(t, u)$ which take into account (log) f0 and spectrum, respectively, are calculated as the relative root mean squared error. Let f_j and g_j be the values of an acoustic feature for the j^{th} frame of the target unit and its corresponding frame in the selected unit, as found by straight-forward linear time scaling, respectively. $D_i(t,u)$ can then be calculated as

$$D_i(t,u) = \sqrt{\frac{1}{N}\sum_{j=1}^{N}\frac{\|f_j - g_j\|^2}{\|f_j\|^2}} \quad (5)$$

with N the number of frames of the target unit. In our system, features are calculated each 5 milliseconds. 12 MFCCs including the 0^{th} coefficient are used for the spectral distance measure. Unvoiced frames are discarded when calculating the log F0 distance. The quality of the weight training can be increased if a ceiling parameter is used for the distance measure. In general, units which have a large acoustic distance are all bad; in those cases, the exact value of the distance is of less importance. Currently, the ceiling parameter is calculated as the median of the distance of the selected units.

The following algorithm is used to find optimal *combination* weights c_i for *M* sets of base context-dependent weights W_i. As many equivalent solutions are possible, we add the constraints that the sum of these combination weights should be equal to 1 and that $0 \leq ci \leq 1$. We use a small number of held-out utterances from the database (e.g. 10 utterances) and generate *M* context-dependent weights sets for each phone-sized unit of these utterances, which are used to calculated M^2 *average* selection errors s_{ij} for each combination of distance measure D_i and weight set W_j. These average selection errors indicate the quality of a particular set of weights using a particular acoustic distance measure. We want that the optimal set of weights for a given target, result in better weights in terms of all acoustic properties, compared to, for example, setting all weights to 1. Therefore, a numeric minimization algorithm (e.g. a conjugate gradient algorithm) can be used to minimize the following function g_m while taking into account the previously defined constraints, resulting in optimized combination weights :

$$g_m = max(g_1, g_2, \ldots g_M) \quad (6)$$

$$g_i = \sum_{j=1}^{M}(c_j^2 \cdot s_{ij}) - f_i \quad (7)$$

with f_i the average selection errors calculated using all weights set to 1 using the i^{th} distance measure.

3. Mandarin Synthesis

Since a segmentation and HTS labels were provided for the Mandarin Chinese database, we used those to build our Mandarin voices. Therefore, we did not need to construct a complete Mandarin Chinese front-end. As such, we were able to adapt our system for Mandarin Chinese and to build our first Mandarin voices, within a single week. None of the members of our team was familiar with Mandarin speech.

Each utterance of the provided speech database is segmented into *initials* and *finals*, which correspond to the initial and final part of a Mandarin syllable: the initial part consists of the initial consonant of the syllable, while the final part consists of a combination of consonants and vowels. Similarly to the UK English voices, several types of units were used (words, syllables) but the basic units (i.e. smallest selectable units) differ: for Mandarin Chinese, these units correspond to *di-syllable parts* which start in the middle of an initial or final, and end in the middle of the next initial or final, similarly to diphones. Each initial and final was labeled with the linguistic features that were extracted from the HTS labels: pinyin, tone, part-of-speech, position in syllable and prosodic structure features. Information (pinyin, tone, part-of-speech) about the immediately neighboring initial or finals is also stored. The tone and part-of-speech features enabled us to group the initials and finals in syllables and words, necessary for multilevel unit selection. Each syllable was labeled in terms of pinyin + tone, and each word as the combination of its syllable labels.

A similar set of target and join costs were used as in the UK English voices, except for the symbolic target costs: costs using next/previous syllable part, next/current/previous tone and next/current/previous part-of-speech were included instead. The prosody models and weights were trained similarly as described in the previous section.

4. Evaluation and discussion

This year's challenge featured the same hub and spoke design as previous editions, in which the hub tasks correspond to the required voices, while the spoke tasks are optional. We submitted voices for all tasks, with the exception of the task in which voices needed to be build based on only 100 sentences (tasks ES1 and MS1). For the tasks in which noise was added

to the speech (tasks ES2 and MS2), we submitted the same voices as for tasks EH1 (using the RJS database) and MH1 (using the full Mandarin database). Our system is indicated with the letter *P*, on the overview plots on the *festvox* site [12].

As our system is based on unit selection without any signal modification, our voices sound quite similar to the target speaker. Overall, both UK English voices performed reasonably well, and are statistically similar to the Festival benchmark system which corresponds to CSTR's 2007 Blizzard entry. Mean naturalness scores of the two hub voices EH1 and EH2 are 3.0 and 3.1, respectively. Surprisingly, the use of a larger database did not seem to improve synthesis quality, but this may be explained by the use of relatively fewer target units during weight training (hence potentially suboptimal weights). Using the 48kHz version of the RJS database did not seem to improve quality (mean naturalness scores of voice ES3 was 2.8). Using a higher sampling rate could reveal some of artifacts (e.g. join artifacts) which were masked by using a lower sampling rate such as the *default* 16kHz.

The use of the same UK English Roger database as in the last two editions of the Blizzard Challenge, in combination with the benchmark voices, allows us to compare this year's results to previous versions of our system. Table 3 lists average naturalness scores and average word error rates of these voices: these results seem to suggest that (some) progress has been made to create more natural, easier to understand voices.

	Naturalness			**Word error rate (%)**		
Year	**DSSP**	**Fest.**	**HTS**	**DSSP**	**Fest.**	**HTS**
2008	2.9	3.1	3.0	45	40	31
2009	2.5	2.9	3.2	28	21	16
2010	3.1	2.9	2.7	29	23	20

Table 4. Average naturalness scores and average word error rate using the Roger voice of our system, and Festival (unit selection) and HTS (HMM synthesis) benchmark systems.

Performance of the Mandarin voices was almost similar to the performance of the UK English voices (e.g. the mean naturalness scores of the two hub Mandarin voices was 3.0 and 2.7, respectively). Manual tuning of the join cost weight proved to be quite challenging, and the value of this parameter can probably be improved. Furthermore, analysis of the synthesized sentences revealed that occasionally short repetitions or deletions were present at the join boundaries. This can be explained by the type of units we were using (*di-syllable parts*) of which the cut-points were set to the middle of an initial or final. Ideally, these would occur at the middle of the center phone, but this is not always the case due to variations of the phone durations within an initial or final. This may have had an effect on the intelligibility of the Mandarin voices.

Obviously, when noise is added to the speech, the intelligibility becomes smaller and the results of this evaluation (tasks ES2 and MH2) for our voice were consistent with the results for clean speech (without additive noise).

5. Conclusions

In this paper we gave an overview of our 2010 Blizzard Challenge submission, and gave some insights in the details of our current voice building and tuning procedures. Overall, our participation to the Blizzard challenge proved to be quite valuable, leading among other things to the development of our first Mandarin voices. The evaluation of the submitted UK English voices indicates similar quality as the Festival benchmark system (CSTR's 2007 Blizzard entry). The results of our Mandarin voices show reasonable quality, especially considering that this was our first attempt at building a Mandarin voice, and in view of our limited experience with Mandarin or other tonal languages. This confirms the ability of our system to support new languages quickly.

Our system still offers room for improvement. As the complexity of the voice building procedure increases due to the training of HMM prosody models and the target cost weight tuning, for example, the amount of time required to build a single voice becomes significantly longer. Therefore, in order to complete all voices on time, we had to cut some corners and were forced to use less optimal settings, e.g. during the training of target cost weights. We believe the quality of these weights can be further improved and we are currently experimenting with other distance measures and clustering techniques. The join cost weight still requires manual tuning and this proved to be quite difficult for the Mandarin voices. Also, the cut-points we used for these voices were not optimal, leading to occasional short repetitions or deletions at the join boundaries. A more detailed segmentation (i.e. using phones) or joining at initial/final boundaries only, could potentially solve these problems.

6. Acknowledgement

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).

7. References

[1] Latacz, L., Kong, Y. O., Mattheyses, W. and Verhelst W., "*An overview of the VUB Entry for the 2008 Blizzard Challenge*", Proc. of the 2008 Blizzard Challenge Workshop, 2008.

[2] Latacz, L., Mattheyses, W. and Verhelst W., "*The VUB Blizzard 2009 Entry*", Proc. of the 2009 Blizzard Challenge Workshop, 2009.

[3] Kominek, J. and Black, A.W., "*CMU ARCTIC databases for speech synthesis*," Tech. Rep. CMU-LTI-03-177, Carnegie Mellon University, 2003

[4] http://www.spraak.org

[5] Zen, H., Nose, T., Yamagishi, J., Sako, S., Masuko, T., Black, A.W. and Tokuda, K., "*The HMM-based speech synthesis system version 2.0*", Proc. of ISCA SSW6, Bonn, Germany, Aug. 2007.

[6] Breiman, L. et al., "*Classification and Regression Trees*" Wadsworth & Brooks, Monterey, California, 1984.

[7] 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

[8] Toda, T. and Tokuda, K., "*Speech parameter generation algorithm considering global variance for HMM-based speech synthesis*," in Proc. of Eurospeech, 2005

[9] 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

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

[11] 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

[12] http://festvox.org/blizzard/blizzard2010.html