Random Forests for Statistical Speech Synthesis

Alan W Black, Prasanna Kumar Muthukumar

Language Technologies Institute, Carnegie Mellon Unversity, Pittsburgh, PA, USA.

awb@cs.cmu.edu, pmuthuku@cs.cmu.edu

Abstract

The world of statistical parametric speech synthesis continues to improve with recent investigations of different machine learning techniques to better model spectrum, F0 and duration from corpora of natural speech. Traditional techniques rely on decision trees alone. This paper shows the advantages of modeling with random forests of decision trees over single trees. Improvements equivalent to more than doubling the data can be achieved, offering end users significantly better synthesis from the same data size. These techniques give proportionally more improvements on smaller datasets, particularly with voices with only 30 minutes of speech. These techniques have been tested over a wide range of voices and languages of various sizes and quality, producing significant improvements in all cases. These techniques are documented, and robustly implemented for others to use through the Dec 2014 release of the Festvox voice building toolkit, thereby directly allowing these benefits to be used in standard voices build for the Festival Speech Synthesis System and CMU Flite.

Index Terms: random forests, acoustic modeling, statistical parameteric speech synthesis

1. Introduction

In spite of the continued improvements in data driven statistical parametric speech synthesis [1], until very recently we have mostly been focusing on aspects of the speech parameterization (e.g. STRAIGHT [2], MELCEP and LSP), and postfiltering techniques such as MLPG [3], Global Variance [4], and Modulation Spectrum [5] rather than addressing core modeling techniques. Both NITECH's HTS [6], and CMU's Clustergen [7] rely on traditional CART trees [8] as their core model for predicting spectral coefficients from linguistic/phonetic feature contexts. However it is well known that such decision tree modeling has the disadvantage of splitting the data with each question. With sufficient data this may not be a problem, but as we always have limited data in speech synthesis databases, other modeling techniques have the potential to use the data better.

Recently there has been active work in applying DNN technology to speech synthesis acoustic modeling [9, 10], which although certainly looks promising, is not yet giving the conclusive improvements found in speech recognition [11]. The work presented here is in a similar but parallel line of investigating better machine learning techniques to the problem of acoustic prediction from linguistic features. It has been well documented that random forests [12] can produce better predictions than a single decision tree over the same data. Although there are a few uses of random forests in the speech synthesis community (e.g. [13]) we believe there has not yet been work in the direct prediction in acoustic modeling.

In a single decision tree, optimal questions are greedily found that can best improve prediction. Typically some measure of impurity is used, such as reducing mean variance of the examples relevant to a particular node in the tree. However in splitting the data at each question, we introduce the problem of reducing the number of examples too fast. For example if a well-defined subset of the data can be identified by two questions thus (in a binary based decision tree system) we may be able to identify that subset, but we will have created three other subsets that may be better modeled together. Given that such splits may happen over multiple dimensions, it is in fact likely that even with carefully crafted features and questions about their values we will over-split the data and not have the most predictive models.

Random forests address this problem. Instead of a single tree, multiple decision trees can be built over the same data, and some combination can be used on each tree's predictions (such as averaging). When each tree is "good" (i.e. have good predictive properties), their combination can in part address the over splitting that will occur in the single tree case. The *randomness* in the random forest arises from the fact that each individual tree is built with a random set of predictors, predictees, or both.

There are examples of multiple models in the speech synthesis community, [14] used multiple models, splitting the prediction of static and delta MELCEPs, as did early versions of Clustergen [7]. [15] builds multiple trees by separating linguistic features based on context. But these were knowledge driven descriptions of splitting the data into multiple trees. Given the number of features used in prediction, and the number of coefficients predicted it would be prohibitively expensive to search the whole space (even if the search were directed by our knowledge of possible interactions). In other fields **random** selection of features has been a proven method to improve models.

2. The baseline

The work in this paper has been done with the Clustergen statistical parameteric framework [7]. Although it is not as popular as NITECH's HTS [6] system, we are much more familiar with it and its expected performance; hence it is easier for us to run our experiments within that system and understand their improvements. Clustergen and HTS fulfill the same basic task in statistical parametric speech synthesis, they both predict acoustic coefficients from linguistic/phonetic features. Both systems are free software, and can use the Festival Speech Synthesis System (or Flite) as their front ends. Both systems can use the same NITECH SPTK speech parameterization and signal reconstruction techniques. One of the few substantial differences between the two systems is that HTS predicts values per HMM state, while Clustergen predicts values per frame.

Although this paper does all of its experiments within the Clustergen system, the results are almost certainly likely to apply to HTS too. The HTS decision tree model uses the same basic Festival produced features, thus we would expect similar

gains in HTS to those presented here.

As much of our particular work in speech synthesis concentrates on voices (often recorded from low resource languages) with limited amounts of data we are particularly interested in techniques that can improve speech synthesis systems built from a limited amount of recorded data. Thus our main examples throughout this paper will be with 30 minutes and 60 minutes of speech. Later we will show how the techniques apply to a much wider range of voice databases sizes and languages. Again for familiarity to the community, our core tests are with the ARCTIC dataset [16] specifically the voices **rms** (a US English male speaker) and **slt** (a US English female speaker). We will call these datasets **rms_a** (30 minutes using the **arctic_a** set) and **rms_ab** (around 60 minutes, both **arctic_a** and **arctic_b**). Similarly with **slt_a** and **slt_ab**. As running listening tests for all possible combinations is too costly, we will initially use Mel Cepstral Distorion (MCD) as its proxy. MCD is a weighted Euclidean distortion measure that we derive from comparing synthesis of 10% held-out utterances against their original speech. Thus smaller values are better. [17] shows that doubling the size of a databases will typical decrease the MCD of a model by around 0.12 and that reductions of around 0.08 are typically perceived by human listeners as improvements.

Using our standard build process using SPTK MELCEP, with no mixed-excitation and using MLPG with a single decision tree we have the following results:

speaker	a set	ab set
rms	4.878	4.754
slt	4.864	4.707

Table 1: *Baseline results*

Remember that the absolute values on MCD are not, in themselves, good correlates of good sounding voices, but reductions in these numbers are good correlates of improved voices.

3. Random Forests

In order to test random forests, we first looked at the MELCEP prediction tree, and decided to replace it with multiple "random" trees. Our initial choice was to randomly pick features in the tree building. We first set a 50% chance for picking each of the 63 linguistic/phonetic features and built 20 trees with a different set of features in each tree. To combine the trees and form a forest, we simply average the predicted values (both the predicted means, and their variances). The following graph shows incrementally adding new random trees to the prediction from 1 to 20. The final number in the tree is the baseline (single tree with all features). Note the initial few combinations of trees are worse than the single tree as they use fewer features, but beginning with the combination of 4 trees, the results are better.

The MCD improvements are 0.074 (for both rms_a and slt_a) and 0.064 (for rms_ab) and 0.051 (for slt_ab). Thus numerical MCD improvements are clear, but not really at the significant level.

But there are a number of possible ways to address the "randomness" in the tree builds. We can randomize the choice of features, the choice of coefficients to predict, the choice of data to train on, or some combination of these.

3.1. Randomizing prediction features

We also need to consider the amount of randomness in our choices. If we consider randomly ignoring different amounts

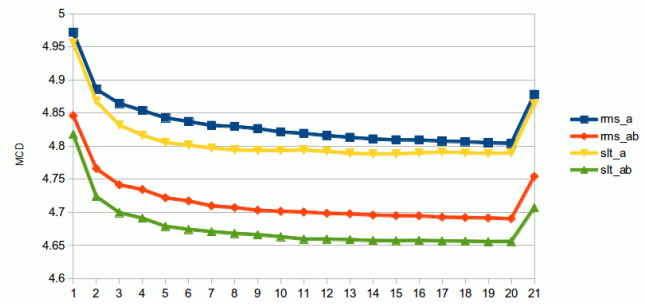

Figure 1: *Effect of adding random trees*

of features in prediction we can see the results in the following graph. We vary the forest by randomly ignoring 0% to 80% of prediction features, and build 20 trees with the same amount of randomly ignored features:

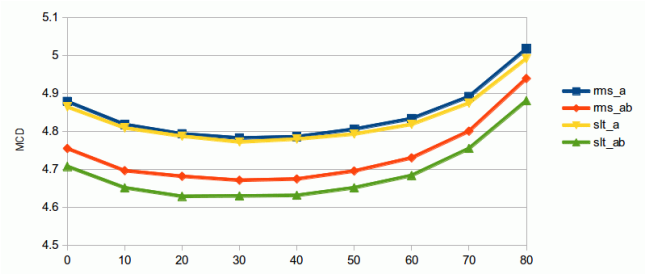

Figure 2: *Effect of ignoring features*

It appears that ignoring around 30% of features gives a better result. Over their baselines, rms_a improves by 0.096, rms_ab by 0.084, while slt_a improves by 0.092 and slt_ab by 0.079. These improvements are significant by our 0.08 standard.

3.2. Randomizing predictee coefficients

We can also vary the percentage of predicted coefficients. Following the known work of building different models for static and deltas [14], we chose to randomly ignore some coefficients in the prediction. When we ignore them, those coefficients will not take part in the impurity measure in the CART building algorithm. Here we ignore 30% of the prediction features, and vary the percentage of ignored predictee coefficients.

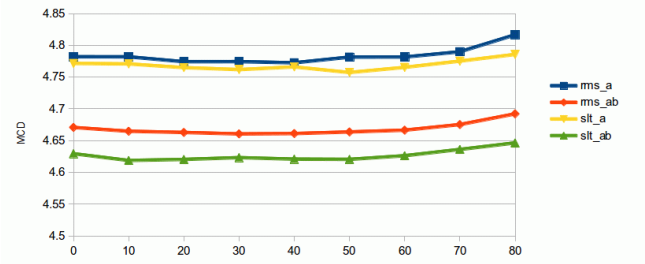

Figure 3: *Effect of randomizing predicted coefficients*

We again get an additional small gain over the baseline in Table 1; rms_a improved by 0.106, slt_a by 0.107 and rms_ab improves by 0.094 and slt_ab by 0.087 (randomly ignoring 30% of the

coefficients). These are all about the significance level.

3.3. Randomizing training data

Another method is to ramdomize the data that is being trained on. Although we did experiements in this space we did not find any obvious way to gain improvements. By using only a subset of the data we found that our general MCD dropped and combinations of different subsets never produced better results than using all the data. There were trends that might have given improvements when we trained on very large voices (>10 hours) but we have not continued that direction in this presentation.

3.4. Overfitting

One of the common techniques used in multi-model systems is to *intentionally* overfit each tree a little. Computing the average of these trees compensates for the overfitting while still providing improvements. We found this to be the case when we used separate models to predict static and delta MELCEPs. In our system we use a STOP value to influence the size of generated decision trees. For STOP=N there must be at least N examples before we will consider a further split. Thus a smaller STOP value will generated deeper trees. This serves the same purpose as minimal description length (MDL) in HTS. We varied the STOP size from 5 to 60 (50 is the baseline) to find the following results (randomly ignoring 30% of prediction features and randomly ignoring 30% for predicted coefficients):

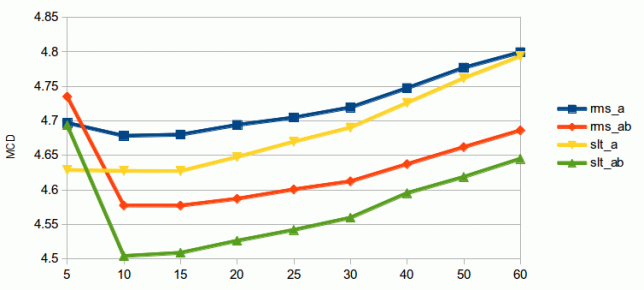

Figure 4: *Effect of overfitting individual trees*

The improvements here are now very clear. rms_a is 0.200, slt_a is 0.237, rms_ab is 0.177, and slt_ab is 0.203. These improvements of our four base voices are equivalent to doubling or more of the data.

3.5. Boosting

It is worth mentioning a further study we did at this time. Boosting [18] is another structurally similar technique for improving prediction using multiple models. The general technique is to build an initial model and predict the training set with that model. And then identify the error factors and build a second model to predict that error, and repeat, building more refined models that predict the residual from the previous. A second boosting technique is to increase the occurance of the more badly predicted examples. We tried both boosting techniques, and although both gave improvements none were close to the full random forest models. We did not however try combinations of these, for example building a 20 tree boosting model with random features, then build 20 such multimodels. The reason for this is that our ultimate goal is to produce a usable model not just the best model, and processing 400 tree models efficiently is hard.

4. Practical Models

On this point of producing usable models, we do care about the ultimate size of our models. With a larger stop size and 20 trees, our models now have some 40-60 times more parameters which make them less practical in a real-time synthesis system. For off-line generation of synthesis this may still be practical but we are interested in also finding more economic use of these new models.

We tested a greedy deselection of the least predictive trees from the random forest (keeping the best N-1 models based on a 50 utterance subset of the training set – so as not to tune toward the test set). The following graph shows the predictive value of deleting the worst tree at each stage.

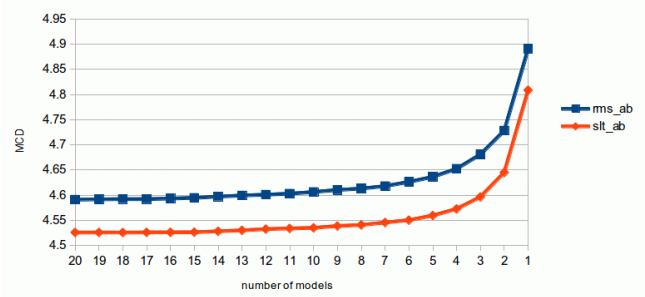

Figure 5: *Effect of greedily choosing best subset of trees*

We decided based on end voice size factors to only choose the best 3 trees. With stop values of 15 this gives us a model of approximately 10 times the size as the single tree model. Thus we do lose some of our gain, but we also have more practically sized models.

5. Checking our results

It is important to ensure that there are not external factors that might be giving the improvements. We have varied the STOP size of our trees and perhaps that alone is giving the improvements. The following graph shows the results when varying the stop size on a single tree.

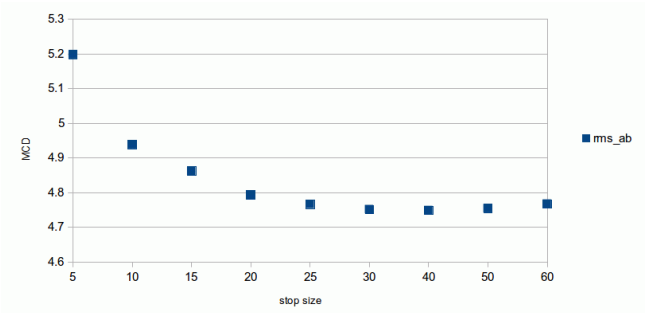

Figure 6: *Effect of varying stopsize on the baseline system*

Thus although there are slight improvements in using STOP=40 over the default (STOP=50) the best MCD (4.749) is substantially worse than the best random forest results (4.577).

Because these models are random, a valid question is what happens on different runs of the build process. We did our build process 20 times for each of the four base voices, including se-

lecting the best three trees. The tests show that the results are fairly stable, with a small standard deviation of <0.01.

speaker	a MCD	a STD	ab MCD	ab STD
rms	4.717	0.009	4.590	0.004
slt	4.656	0.005	4.525	0.004

Table 2: *The influence of randomness*

6. Is it General?

Although we get clear improvements on these base voices, it is important to find out if these improvements hold over a larger number of voices, and voices of different quality, size etc. Thus we applied this build scheme to 23 different voices. These vary in language (mostly English but also Indic Languages too) size (30 minutes to 20 hours) and quality (arctic, audio books, and mixed speaker (wsj)).

We used the same standard build script `build_cg_rfs_voice` for all of these voices that builds an initial single tree voice, then adds mixed-excitation [19], move_label [20], and then builds 20 random trees for spectral prediction and selects the best 3; then builds 20 duration prediction trees and selects the best 3 duration models.

Voice	size (mins)	base MCD	rf3 MCD	ΔMCD
ahw	30	4.795	4.566	0.228
aup	30	5.107	4.937	0.170
awb	60	4.347	4.187	0.160
bdl	60	5.290	5.054	0.236
axb	30	5.719	5.448	0.271
clb	60	4.167	4.019	0.149
cxb	1211	4.940	4.916	0.024
fem	30	5.035	4.804	0.231
gka	30	5.278	5.048	0.230
jmk	60	4.840	4.641	0.199
ksp	60	4.892	4.726	0.166
rms	60	4.754	4.587	0.167
rxr	30	4.763	4.587	0.173
slt	60	4.707	4.053	0.177
f1a	180	5.095	4.950	0.146
f3a	180	4.812	4.693	0.119
wsj	420	6.359	6.187	0.172
tats	367	5.080	4.987	0.093
emma	1000	4.982	4.940	0.043
iitmH	120	4.009	3.866	0.143
iiitT	180	4.472	4.316	0.156
axbH	135	4.989	4.829	0.160
sbH	120	5.456	5.225	0.231

Table 3: *"rf3" results over 23 voices*

Table 3 shows these results. ahw, aup, awb, bdl, axb, clb, fem, gka, ksp, jmk, rms, rxr, slt are English Arctic databases [16] with varying sizes and accents. cxb, tats and emma are audio book databases from Blizzard. f1a and f3a are US English news corpora [21]. wsj is a collection of different US male speakers [22]. axbH and sbH are CMU Hindi databases, while iitmH is Hindi [23], and iiitT is Telugu [23].

In the analysis of these results we can spot a trend in the improvements. If we look at the delta improvement from the base (single tree) voice to the "rf3" voice we see that in general smaller voices get a bigger improvement. This is elaborated in Table 4.

size	ΔMCD	STD
30 mins	0.218	0.039
60 mins	0.179	0.030
>120 mins	0.129	0.062

Table 4: *Improvements as a factor of voice size*

7. Listening Tests

MCD is not the ultimate measure, we also carried out listening tests for some of the voices. In all cases we compared our base build (with mixed-excitation) with an rf3 build using the versions of the voices described in Table 3 above. The listeners (recruited through Amazon Turk) were asked to listen to 20 pairs of utterances. Each sample was synthesized with either synthesizer (in random order) and listeners were asked to choose which they prefer.

voice	Preference base	none	rf3	num of listeners	num of pairs
rms_a	30.70%	8.37%	49.30%	12	215
rms_ab	24.08%	12.57%	52.88%	10	191
slt_a	30.99%	14.04%	44.44%	9	171
slt_ab	20.00%	17.00%	53.00%	10	200
aup	27.31%	12.78%	48.46%	13	227
bdl	29.69%	7.29%	52.08%	10	192

Table 5: *Results of listening tests*

Thus in all case people prefered rf3, or had no-preference in around 70% of the samples.

8. Conclusions

This technique is now built into the standard build process in the Festvox voice building tools release from Dec 2014. Thus both Festival (2.4) and Flite (2.0.0) built voices benefit from these results. We also use random forests to improve duration modeling (though not discussed here). However we have not yet applied this technique to F0 modeling. [24] reports positive results on using a gradient boosting approach[25] for modeling F0 trajectories. So applying a random forest approach is likely to yield useful results.

Although the voice models are around 10 times larger than the single tree versions, the run time speed is only about 1.5 times slower as tree traversal is not the most expensive part of the total synthesis time.

Voice building times however are, of course, longer. A single tree build of an Arctic voice (1 hour of speech) takes around 40 minutes on a 6 processor machine, while a full "rf3" build of the same voice will take around 6 hours.

We are still concerned about the increase in the size of the models and feel there is more to investigate here. However initial investigations using PCA to reduce the number of parameters, and sharing leaves of the different trees have not been successful.

Our results strongly show that random forests for spectral prediction can increase prediction accuracy equivalent to at least doubling the training data (and in some cases quadrupling it). Gains seem to be larger for smaller voices, something that we are particularly interested in, given that we often work with small amounts of data in less commonly processed languages.

9. References

[1] H. Zen, K. Tokuda, and A. Black, "Statistical parametric speech synthesis," *Speech Communication*, vol. 51, no. 11, pp. 1059–1064, 2009.

[2] H. Kawahara, M. Morise, T. Takahashi, R. Nisimura, T. Irino, and H. Banno, "Tandem-straight: A temporally stable power spectral representation for periodic signals and applications to interference-free spectrum, f0, and aperiodicity estimation," in *Acoustics, Speech and Signal Processing, 2008. ICASSP 2008. IEEE International Conference on*. IEEE, 2008, pp. 3933–3936.

[3] K. Tokuda, T. Yoshimura, T. Masuko, T. Kobayashi, and T. Kitamura, "Speech parameter generation algorithms for hmm-based speech synthesis," in *Acoustics, Speech, and Signal Processing, 2000. ICASSP'00. Proceedings. 2000 IEEE International Conference on*, vol. 3. IEEE, 2000, pp. 1315–1318.

[4] T. Toda, A. W. Black, and K. Tokuda, "Spectral conversion based on maximum likelihood estimation considering global variance of converted parameter." in *ICASSP (1)*, 2005, pp. 9–12.

[5] S. Takamichi, T. Toda, G. Neubig, S. Sakti, and S. Nakamura, "A postfilter to modify the modulation spectrum in hmm-based speech synthesis," in *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*. IEEE, 2014, pp. 290–294.

[6] K. Tokuda, H. Zen, J. Yamagishi, T. Masuko, S. Sako, T. Nose, and K. Oura, "HMM-based speech synthesis system (HTS)," 2008, `http://hts.sp.nitech.ac.jp/`.

[7] A. Black, "CLUSTERGEN: A statistical parametric synthesizer using trajectory modeling," in *Interspeech 2006*, Pittsburgh, PA., 2006.

[8] L. Breiman, J. Friedman, R. Olshen, and C. Stone, *Classification and Regression Trees*. Pacific Grove, CA.: Wadsworth & Brooks, 1984.

[9] H. Zen and A. Senior, "Deep mixture density networks for acoustic modeling in statistical parametric speech synthesis," in *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*. IEEE, 2014, pp. 3844–3848.

[10] P. Wang, Y. Qian, F. Soong, L. He, and H. Zhao, "Word embedding for recurrent neural network based TTS synthesis," in *ICASSP 2015*, Brisbane, Australia, 2015.

[11] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A.-r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath *et al.*, "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," *Signal Processing Magazine, IEEE*, vol. 29, no. 6, pp. 82–97, 2012.

[12] L. Breiman, "Random forests," *Machine Learning*, vol. 45(1), pp. 5–32, 2001.

[13] A. A. Raj, T. Sarkar, S. C. Pammi, S. Yuvaraj, M. Bansal, K. Prahallad, and A. W. Black, "Text processing for text-to-speech systems in Indian Languages." in *SSW*, 2007, pp. 188–193.

[14] Z.-P. Yu, Y.-J. Wu, H. Zen, Y. Nankaku, and K. Tokuda, "Analysis of stream-dependent tying structure for hmm-based speech synthesis," in *Signal Processing, 2008. ICSP 2008. 9th International Conference on*. IEEE, 2008, pp. 655–658.

[15] K. Yu, H. Zen, F. Mairesse, and S. Young, "Context adaptive training with factorized decision trees for hmm-based speech synthesis." in *INTERSPEECH*, 2010, pp. 414–417.

[16] J. Kominek and A. Black, "The CMU ARCTIC speech databases for speech synthesis research," Language Technologies Institute, Carnegie Mellon University, Pittsburgh, PA, Tech. Rep. CMU-LTI-03-177 `http://festvox.org/cmu_arctic/`, 2003.

[17] J. Kominek, T. Schultz, and A. Black, "Synthesizer voice quality on new languages calibrated with mel-cepstral distorion," in *SLTU 2008*, Hanoi, Viet Nam, 2008.

[18] Y. Freund, R. E. Schapire *et al.*, "Experiments with a new boosting algorithm," in *ICML*, vol. 96, 1996, pp. 148–156.

[19] T. Yoshimura, K. Tokuda, T. Masuko, T. Kobayashi, and T. Kitamura, "Mixed excitation for HMM-based speech synthesis." in *EUROSPEECH 2001, Aalborg, Denmark*, 2001, pp. 2263–2266.

[20] A. Black and J. Kominek, "Optimzing segment label boundaries for statistical speech synthesis," in *ICASSP 2009*, Taipei, Taiwan, 2009.

[21] M. Ostendorf, P. J. Price, and S. Shattuck-Hufnagel, "The Boston University radio news corpus," *Linguistic Data Consortium*, pp. 1–19, 1995.

[22] D. B. Paul and J. M. Baker, "The design for the wall street journal-based csr corpus," in *Proceedings of the workshop on Speech and Natural Language*. Association for Computational Linguistics, 1992, pp. 357–362.

[23] K. Prahallad, A. Vadapalli, S. Kesiraju, H. Murthy, S. Lata, T. Nagarajan, M. Prasanna, H. Patil, A. Sao, S. King, A. Black, and K. Tokuda, "The Blizzard Challenge 2014," in *The Blizzard Challenge 2014*, 2014. [Online]. Available: http://festvox.org/blizzard

[24] Y. Qian, H. Liang, and F. K. Soong, "Generating natural f0 trajectory with additive trees," in *Ninth Annual Conference of the International Speech Communication Association*, 2008.

[25] J. H. Friedman, "Greedy function approximation: a gradient boosting machine," *Annals of statistics*, pp. 1189–1232, 2001.