Parameter Tuning for Fast Speech Recognition

Thomas Colthurst, Tresi Arvizo, Chia-Lin Kao, Owen Kimball,
Stephen Lowe, David R. H. Miller, Jim Van Sciver

BBN Technologies, Inc., 10 Moulton Street, Cambridge MA 02138, USA
thomasc@bbn.com

Abstract

We describe a novel method for tuning the decoding parameters of a speech-to-text system so as to minimize word error rate (WER) subject to an over-all time constraint. When applied to three sub-realtime systems for recognizing English conversational telephone speech, the method gave speed improvements of up to 21.1% while at the same time reducing WER by up to 6.7%.

Index Terms: speech recognition

1. Introduction

As computers have become faster over the past two decades, realtime speech recognition systems have become more complicated. A realtime system from the early 1990s might now run at 0.1 times realtime (xRT) or under, and can be used as the first part of a fast decoding architecture that does speaker adaptation, lattice rescoring and more.

This increase in system complexity comes with a corresponding increase in the number of system parameters. For example, Figures 1, 2 and 3 show three different speech-to-text architectures used in this study to achieve different speed / accuracy trade-offs. The simple architecture in Figure 1 already has 11 parameters; the architecture in Figure 2 has 63. We take the broad view that a parameter is any boolean or scalar value that affects performance and can be changed without retraining the system's models. These include

- Pruning parameters, which specify what fraction of the best theory's score a theory must exceed to be kept alive. Each search stage of our system uses several different pruning parameters, which are applied at different times (every frame, word endings, phone endings) to different scores (HMM activity, Viterbi score, combined forward and backwards score).
- Model weights, which are applied to the scores of the language, acoustic, and other models to produce a combined hypothesis score.
- Parameters of the fast Gaussian computation process [1], including the number of cells to divide the feature space into, the number of Gaussians to compute per cell, and the number of Gaussians to compute per Gaussian mixture model.
- Size and memory duration of the language model cache.
- Parameters of the speaker adaptation process, including pruning parameters for the conversion of the unadapted best transcript into "fuzzy labels" [2], the maximum number of iterations to perform in adaptation, and smoothing parameters, such as the minimum Gaussian variance.

At BBN, these parameters have historically been set and tuned by hand, and we are aware of no published literature on automatic methods for determining them, other than exhaustive search applied to individual parameters. This paper tackles the problem of jointly tuning all of the parameters of a speech recognition system so as to minimize word error rate under a given speed constraint. In the next section, we present a novel algorithm for doing so; this algorithm very efficiently searches for optimal parameter values by making some strong assumptions about the parameter tuning problem. The results in section 4 show the parameters produced by this method give very good performance when applied to three sub-realtime systems: decode time reductions of up to 21.1% along with WER reductions of 6.7% on a held out test set.

2. The Method

Let $\boldsymbol{\theta}$ be the vector of parameters of a speech recognition system, and consider the word error rate (WER) and speed of the system as measured on a fixed test set to be functions of $\boldsymbol{\theta}$. We seek to minimize $\text{WER}(\boldsymbol{\theta})$ subject to a constraint of the form $\text{speed}(\boldsymbol{\theta}) \leq S$ (note that our speeds are measured in multiples of real-time so that lower values of $\text{speed}(\boldsymbol{\theta})$ indicate faster systems). Because some of the parameters are discrete, we are in the category of a mixed-integer nonlinear programming (MINLP) problem [3]. A general MINLP solver would be much too slow, however, so we developed a method that exploited several special features of our situation:

- For most of the parameters, including all of the pruning parameters, both WER and speed are almost always monotonic. This in turn means that the problem has relatively few local optimums.
- From experience, we know for each parameter the magnitude of change required to see a nontrivial change in WER and speed.
- Also from experience, we observe that small changes in different parameters are, when combined, often very nearly additive in both WER and speed. While this is always true for smooth functions and infinitesimal changes, we observe it even when changing two discretely valued parameters.
- We have a large number of computers, so search methods that do many parallel evaluations are preferred.

Our algorithm is inspired by the theory of Lagrange multipliers, which proves that if the functions $\text{WER}(\boldsymbol{\theta})$ and $\text{speed}(\boldsymbol{\theta})$ were both smooth and WER was minimized at $\text{speed}(\boldsymbol{\theta}) = S$, then at any local solution $\boldsymbol{\theta}$ there would be a λ such that

$$\frac{\partial \text{WER}}{\partial \theta_i} - \lambda \frac{\partial \text{speed}}{\partial \theta_i} = 0 \qquad (1)$$

for all i. In our non-smooth case, we might hope that at a local solution $\boldsymbol{\theta}$ there exists a λ such that

$$\text{WER}(\boldsymbol{\theta}') - \text{WER}(\boldsymbol{\theta}) + \lambda(\text{speed}(\boldsymbol{\theta}') - \text{speed}(\boldsymbol{\theta})) \geq 0 \quad (2)$$

for all $\boldsymbol{\theta}'$ close to $\boldsymbol{\theta}$. To put it another way, at a local solution $\boldsymbol{\theta}$ there should be a WER / speed trade-off value λ such that all small changes to $\boldsymbol{\theta}$ degrade over-all performance, as measured by

$$\text{performance}(\lambda, \boldsymbol{\theta}) = -\text{WER}(\boldsymbol{\theta}) - \lambda\text{speed}(\boldsymbol{\theta}) \quad (3)$$

The following algorithm searches for the combination of a local maximum in performance and a λ value such that the local maximum satisfies the desired time constraint.

Algorithm 1.
Input: An initial set of parameters $\boldsymbol{\theta}$, an initial WER / speed trade-off value λ, a speed range $[S_1, S_2]$, and for each i, a size ϵ_i for small changes to the i-th parameter.
Output: A set of parameters $\boldsymbol{\theta}$ such that $S_1 \leq \text{speed}(\boldsymbol{\theta}) \leq S_2$ and a λ such that

$$\text{performance}(\lambda, \boldsymbol{\theta}') - \text{performance}(\lambda, \boldsymbol{\theta}) \quad (4)$$

is negative for all $\boldsymbol{\theta}' = \boldsymbol{\theta} \pm \epsilon_i \boldsymbol{e}^i$ for all i, where $\boldsymbol{e}^i$ is the vector of all zeros, except for a one in the i-th position.
Procedure:

1. Run a decode with the parameters $\boldsymbol{\theta}$.
2. For each parameter, run a decode which changes $\boldsymbol{\theta}$ in that parameter by a small amount in each direction. That is, run parallel decodes over the set $\{\boldsymbol{\theta} \pm \epsilon_i \boldsymbol{e}^i\}$.
3. For each decode, compute the performance difference (4). If this value is negative for both of the i-th parameter's decodes, set $c_i \leftarrow 0$. If it is positive for either decode, set $c_i \leftarrow \pm\epsilon_i$ with the sign taken from the decode with higher performance value.
4. If at least two of the c_i are non-zero, run a decode with the parameters $\boldsymbol{\theta} + \sum_i c_i \boldsymbol{e}^i$.
5. If any of the decodes done in step 2 or 4 improved performance, set $\boldsymbol{\theta}$ to be the parameters used in the decode that has the highest performance value and go to step 2.
6. If $S_1 \leq \text{speed}(\boldsymbol{\theta}) \leq S_2$, stop.
7. If $\text{speed}(\boldsymbol{\theta}) > S_2$, increase λ; otherwise decrease λ. In either case, change λ to the nearest value that makes (4) positive for at least one of the decodes done in step 2, and go to step 3.

As long as the input speed range $[S_1, S_2]$ is large enough that the $\pm\epsilon_i \boldsymbol{e}^i$ parameter changes don't jump over it, this algorithm will terminate. In practice, the "sum of the good parameter changes" decode done in step 4 almost always has much better performance than the decodes done in step 2, which greatly accelerates the search as compared to any local hill climbing method.

In our implementation, we first apply the above algorithm to each individual stage of the decode (except the VTL and Analysis stage, which has no parameters) before applying it to the decode as a whole. We also "freeze" parameters that remain constant for several iterations, and only try them again in a final iteration. This saves time from being wasted on parameters that converge quickly, while still guaranteeing a true local maximum of performance.

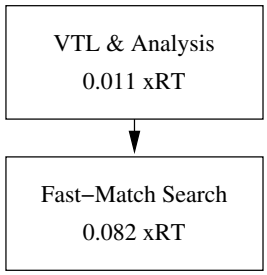

Figure 1: *Architecture of the 0.1xRT system. Time measurements are from decoding Eval01 using Swbd370 models.*

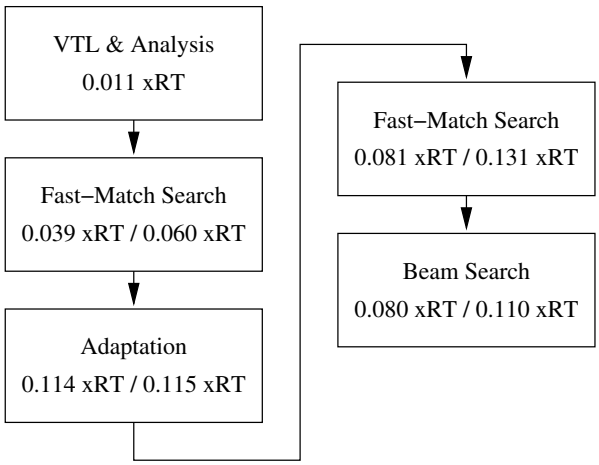

Figure 2: *Architecture of the 0.33xRT and 0.5xRT systems. Time measurements are from decoding Eval01 using Swbd370 models.*

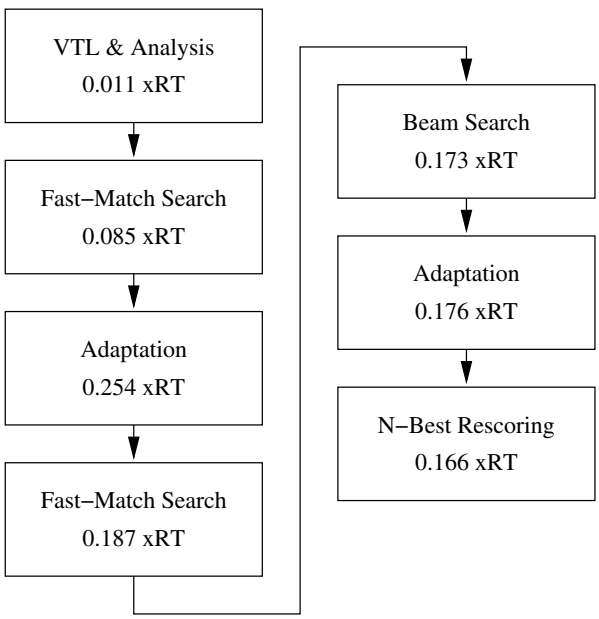

Figure 3: *Architecture of the 1xRT system. Time measurements are from decoding Eval01 using Swbd370 models.*

3. System Description

3.1. Training

The systems in this paper all used the same acoustic and language models. The acoustic model was trained on Swbd370, which consisted of 379 hours of data from the Switchboard 1 and Switchboard Cellular collections. The language model was trained from the Swbd370 transcripts plus 370 million words taken from a variety of broadcast news and web sources. Training was done using a procedure very similar to that described in [4], with the following differences:

1. Only PLP features were used.
2. All of the training corpora were re-segmented based upon a preliminary forced alignment so that all utterances were between 0.8 and 15 seconds long, with no internal silences of longer than 0.5 seconds, and endpoint silences of approximately 0.25 seconds.
3. All systems used a pre-HDA feature space obtained by concatenating nine frames (4 on each side) of the original 15 cepstra. These features were then projected down to a 46 dimensional space using HLDA-SAT [5].
4. The lattices used for MPE training were created directly from the decoding lattices, rather than from N-best lists.
5. No mixture exponents were used.

3.2. Decoding

Figures 1, 2, and 3 illustrate the three architectures considered in this paper, which are named after their target speeds of 0.1xRT, 0.33xRT and 0.5xRT, and 1xRT, respectively. The 0.33 and 0.5xRT systems share the same architecture and differ only in their parameters. The 1xRT system is based on the 1xRT system presented in [6], but uses N-best rather than lattice rescoring. The other systems were developed by simply truncating processing steps from the end of the 1xRT system.

All the systems begin by estimating vocal tract length (VTL) warps by scoring unwarped cepstra against Gaussian mixture models (GMMs) with 256 Gaussians, one GMM for each warp in 0.80, 0.82, 0.84 ... 1.20. Analysis is then performed on 25 ms windows to produce 100 frames of features per second, each containing 14 PLP smoothed cepstra plus normalized energy. Each frame is mean and covariance normalized per speaker.

The next step for each system is a fast-match search using a bigram language model and acoustic models that tie triphone, within-word contexts [7]. This fast-match search outputs a one best hypothesis which is the final output of the 0.1xRT system, but is used for unsupervised speaker adaptation in the 0.33, 0.5, and 1xRT systems.

After speaker adaptation, another fast-match search is performed using SAT models. The purpose of this fast-match is to produce a set of likely word-ends per frame, which is used to constrain the subsequent time-synchronous beam search [8]. This beam search uses an approximate trigram language model and linguistically-tied within-word quinphone SAT acoustic models. In the 0.3 and 0.5xRT systems, the beam search produces the final answer, while in the 1xRT system it produces both a 1-best and 100-best list. The 1-best is used for a second round of speaker adaptation, and these adaptation parameters are applied to the crossword quinphone acoustic models used to rescore the 100-best list.

We also experimented with two other decoding architectures: one that did only a fast-match search after adaptation, and one that removed the post-adaptation fast-match search from the 0.33/0.5xRT architecture and used for its final beam search the word-ends per frame produced by the pre-adaptation fast-match. Even with parameter tuning, both of these architectures had worse word error rates than the configuration in Figure 2 at the target speeds of 0.33xRT and 0.5xRT. (Our system cannot perform adaptation at much faster than 0.1xRT, so they were not even considered for that speed target.) It is possible they might have proved superior had the target speed been 0.2xRT or 0.25xRT.

Statistic	0.1xRT	0.33xRT	0.5xRT	1xRT
Mean	0.094	0.348	0.454	1.122
Std. Dev.	0.002	0.001	0.001	0.007

Table 1: *Mean and standard deviation of the speeds from ten runs of Swbd370 models on Dev04.*

3.3. Time Measurements

All time measurements were taken on computers running with dual 3.4 GHz Intel Xeon processors. Because our system is intended for batch mode speech processing, we treated each CPU as a separate computer and measured the sum of all kernel and user mode CPU times, as reported by the UNIX time command. The average CPU efficiency, defined as the ratio of this CPU time to wallclock time, was 96%.

The theory in section 2 treated speed($\boldsymbol{\theta}$) as a deterministic function. While we could have approximated deterministic times by running only on isolated machines with all models loaded onto ramdisk and all non-essential background processes disabled, we did not do so. Our time measurements are therefore noisy. (They would have been more noisy had we measured using elapsed wallclock time instead of CPU time.) To estimate the variability, we ran ten identical decodes on Dev04, a three hour test set of Fisher data. Table 1 lists the means and standard deviations of the speeds of these runs.

While the variances in table 1 are reassuringly small, they understate the true time variances because the decodes were all run on the same day and thus under sample the impact of network and RAID use by other researchers in our department. To be safe, we spaced parameter changes (the ϵ_i's) so that they all caused time differences of at least 0.005xRT. In a small number of cases, decodes with anomalous time measurements were rerun.

4. Results

We carried out parameter tuning on Eval01, the 6 hour test set of Switchboard 1 and 2 used for the 2001 NIST Hub-5 evaluation. For each of the target speeds of 0.1xRT, 0.33xRT and 0.5xRT, we started from the parameter values of the existing 1xRT system and manually tuned a handful of pruning parameters until the system was close (within 30% or so) to the speed target. Algorithm 1 was then applied to jointly tune the parameters subject to the speed constraint. (We had no mandate to jointly tune the 1xRT system's parameters in this work, but given the results, we hope to do so soon.)

Table 2 shows the error rate and speed of each system before and after the joint tuning, as well as their percent improvements. The amount of improvement obviously varies both on the quality of the initial parameter values and on the shape of the WER and speed curves between the initial and final parameter values, but it

Joint	0.1xRT Sys.		0.33xRT Sys.		0.5xRT Sys.	
Tuning	WER	xRT	WER	xRT	WER	xRT
Before	43.96	0.134	28.96	0.382	29.29	0.543
After	42.07	0.099	29.97	0.328	27.42	0.485
% Δ	4.3%	26.1%	-3.5%	14.1%	6.4%	10.7%

Table 2: *Effect of jointly tuning the decode parameters on Eval01.*

Sign(ΔT)	0.1xRT Sys.	0.33xRT Sys.	0.5xRT Sys.
$-$T	114	60	12.22
$+$T	26.5	11.25	1.76

Table 3: *Best values of* $\lambda = \frac{-\Delta WER}{\Delta xRT}$ *for parameter changes that increase (+T) or decrease (−T) the runtime of the three decoding systems tuned on Eval01.*

is noteworthy that joint parameter tuning was able to achieve the speed targets through 10% or greater speedups while at the same time noticeably reducing WER for two of the three systems. To put these results in context, the 1xRT system obtains a WER of 25.25 on Eval01 while running at 1.051xRT and the slow system we normally use for development work gets a WER of 22.42 while running at 12.633xRT.

Table 3 gives additional insight by listing the local speed / WER trade-offs for our jointly tuned systems. Two values of λ are given per system, one for the best parameter change that decreases ($-$T) the runtime of the system and one for the best change that increases ($+$T) the runtime. The λ values dramatically increase as the system becomes faster because fixed size improvements in speed cause greater and greater degradations in word error rate. It is interesting that the λ for increasing the speed of the 0.33xRT system is very close to the λ for decreasing the speed of the 0.5xRT system; this leads us to suspect that the graph of WER versus speed for optimally tuned systems with the Figure 2 architecture is essentially a straight line between 0.33xRT and 0.5xRT.

Table 4 summarizes the number of parameters examined, the number of parameters changed, and the number of decodes run by the joint parameter tuning procedure. Nearly five hundred decodes were run, but each was sub-realtime and many were only partial decodes that changed parameters in the later stages of the decoding pipeline and could reuse the beginning pipeline results of a previous decode. (We also cut corners by restricting the tuning of the 0.33xRT system to the parameters changed by the tuning of the 0.5xRT system.) In total, less than one thousand hours of compute time were used.

The real test of the joint tuning method is the extent to which the improvements it brings persist on data sets other than the

Quantity	0.1xRT Sys.	0.33xRT Sys.	0.5xRT Sys.
# Parameters	11	35	63
# Changed	5	12	35
# Decodes	86	108	304

Table 4: *The numbers of parameters examined, parameters changed, and decodes run by the joint parameter tuning procedure, per system.*

Joint	0.1xRT Sys.		0.33xRT Sys.		0.5xRT Sys.	
Tuning	WER	xRT	WER	xRT	WER	xRT
Before	41.87	0.128	27.09	0.399	27.03	0.505
After	39.04	0.101	28.31	0.344	26.05	0.481
% Δ	6.7%	21.1%	-4.5%	13.8%	3.6%	4.8%

Table 5: *Effect of parameters tuned on Eval01 on the held out Dev04 test set.*

Joint	0.1xRT Sys.		0.33xRT Sys.		0.5xRT Sys.	
Tuning	WER	xRT	WER	xRT	WER	xRT
Before	45.15	0.155	30.47	0.417	30.51	0.518
After	42.19	0.097	31.63	0.349	28.60	0.467
% Δ	6.6%	37.4%	-3.8%	16.3%	6.3%	5.1%

Table 6: *Effect of parameters tuned on Eval01 on the held out Eval03 test set.*

one used for the tuning. We used the parameters optimized on Eval01 in decodes on the Dev04 and Eval03 test sets, and tables 5 and 6 show the results. (Eval03 is the six hour corpus of Switchboard cellular and Fisher data used by NIST in the 2003 EARS evaluation.) For both test sets, the speed and word error rate improvements closely match those seen on Eval01.

5. Conclusions

The results from the previous section show that the novel algorithm we presented in this paper can be used to jointly tune the parameters of a fast speech recognition system and that doing so gives significant performance improvements. Decoding times were reduced by up to 26% and word error rates were improved by up to 6%, and the gains persisted when measured on two different held out test sets.

6. References

[1] D. B. Paul, "An investigation of Gaussian shortlists," ASRU 1999.

[2] R. Schwartz *et al.*, "Speech recognition in multiple languages and domains: the 2003 BBN/LIMSI EARS system," ICASSP 2004.

[3] M. R. Bussieck and A. Pruessner, "Mixed-Integer Nonlinear Programming," SIAG/OPT Newsletter: Views & News, 14(1), 2003.

[4] R. Prasad *et al.*, "The 2004 BBN/LIMSI 20xRT English conversational telephone speech system," Proc. Rich Transcription Workshop, Palisades, NY, Nov. 2004.

[5] S. Matsoukas and R. Schwartz, "Improved speaker adaptation using speaker dependent feature projections," ASRU 2003.

[6] S. Matsoukas *et al.*, "The 2004 BBN 1xRT recognition systems for English broadcast news and conversational telephone speech," INTERSPEECH 2005.

[7] L. Nguyen and R. Schwartz, "Single-tree method for grammar-directed search," ICASSP 1999.

[8] L. Nguyen and R. Schwartz, "Efficient 2-pass N-best decoder," Eurospeech 1997.