BASELINE WSJ ACOUSTIC MODELS FOR HTK AND SPHINX: TRAINING RECIPES AND RECOGNITION EXPERIMENTS

Keith Vertanen

University of Cambridge, Cavendish Laboratory
Madingley Road, Cambridge, CB3 0HE, UK
kv227@cam.ac.uk

ABSTRACT

For speech recognition research, it is often necessary to start with a competent baseline acoustic model. But training and tuning a competent model using research recognizers such as Cambridge's HTK and CMU's Sphinx can be time-consuming. In an effort to minimize wasted effort, I have created recipes for HTK and Sphinx which utilize the standard Wall Street Journal training corpus. In this paper, these recipes are described. The word error rate (WER) and real-time performance of the models are evaluated for differing HMM topologies, number of tied states, number of Gaussians, and differing test sets. My goal is to provide practical advice and results to researchers who are thinking of using HTK or Sphinx for real-time recognition on dictation-like tasks.

1. RECIPE DETAILS

In this section, I give the details of the acoustic model training for HTK and Sphinx. Both acoustic models are trained for American English using utterances from the Wall Street Journal (WSJ) corpora. Both recipes use the CMU pronouncing dictionary and its corresponding set of 39 phones without stress markings. Each recipe consists of a set of Perl scripts, bash scripts, and configuration files appropriate for use on a Unix-type system.

1.1. HTK recipe

The HTK training recipe closely follows the steps in the tutorial in the HTKBook [1]. HTK version 3.4 was used for the experiments reported here. The recipe scripts support a number of different training variants:

- Amount of training data (SI-84, SI-284, or all WSJ training data)
- Word-internal or cross-word triphones
- Flat-start or TIMIT-bootstrapped monophones

The acoustic waveforms from WSJ and optionally TIMIT are first parametrized into a 39-dimensional feature vector consisting of 12 cepstra plus the 0th cepstral, deltas and delta deltas, normalized using cepstral mean subtraction (MFCC_D_A_Z). The recipe then initializes a set of 40 HMMs (39 monophones plus 1 silence model). Each HMM has three output states with a left-to-right topology with self-loops and no transitions which skip over states.

If TIMIT-bootstrapping is used, the phonetically aligned utterances in TIMIT are used to initialize the HMM parameters by cutting out the corresponding utterance segments for a particular phone model. The segments are then used in an iterative Viterbi-training scheme until the parameters converge. The Viterbi-estimated parameters and then refined by a further Baum-Welch training cycle.

If a flat-start initialization is used instead, models are initialized with the global mean and variance of the WSJ0 training data. Utterances are then uniformly segmented and their parameters re-estimated in several rounds of Baum-Welch training.

The silence (sil) model in the trained monophones is then duplicated to create a short pause (sp) model. The sp model adds a transition skipping over all output states. The output distributions of sp and sil have their parameters tied. The trained monophones are used to force-align the word-level training transcripts with the training utterances. Since a word my have multiple pronunciations in the dictionary, this realignment hopefully improves the phone-level accuracy of the training transcripts. The new transcripts are then used in 4 rounds of Baum-Welch training.

The phone-level transcripts are converted into triphone transcripts, generating a list of all triphones observed in the training data. The triphones may either be word-internal (blocking phone context across word boundaries) or cross-word. Each triphone is cloned from the monophone of its central phone. The state transition matrices of triphones sharing the same central phone are tied. The triphone models are then trained using 2 rounds of Baum-Welch training.

A list of all possible triphones given the pronunciation dictionary is created. This includes many triphones which may not have been observed in the training data but are in theory possible given a language model based on the words

in the dictionary. The states of these unseen triphones need to be tied to triphones we have training data for. In addition, such tying of triphone states reduces the number of model parameters, potentially leading to a more robust model. A decision tree process using a list of linguistically motivated questions about a triphone's context is used to cluster and tie triphone states together. The resulting tied-state triphones are re-estimated using 4 rounds of Baum-Welch training.

The number of Gaussians per-state is now increased by splitting each Gaussian and perturbing the mean of the two new Gaussians. The new models are trained with 4 rounds of Baum-Welch and the splitting/training process repeated as needed. Depending on the amount of training data, the recipe mixes up to 8 or 16 Gaussian per non-silence state. The silence models use double the number of output distributions, 16 or 32 Gaussians per silence state.

Throughout the recipe, the parallel mode feature of HERest is used during Baum-Welch parameter re-estimation. This causes the training data to be split into a fixed number of subsets. The accumulators for each subset are computed separately and subsequently combined. This improves the numerical accuracy and prevents errors when training on large amounts of data.

The HTK recipe is available from [2]. Many of the acoustic models used in the experiments described later in this paper are available for download.

1.2. Sphinx recipe

The Sphinx recipe is based on the tutorial from [3]. The recipe adapts the tutorial scripts to support training from WSJ data. The recipe allows flexible configuration of model training and decoding including:

- Amount of training data (SI-84, SI-284, or all WSJ training data)
- Continuous or semi-continuous models
- Number of senones
- Number of Gaussians per state
- HMM topology (number of states, skip transitions)
- Decoder (Sphinx-2, Sphinx-3, PocketSphinx)

The acoustic waveforms from WSJ are first parametrized into 13-dimensional cepstrum. For continuous models, these features along with computed delta and delta-deltas are used (1s_12c_12d_3p_12dd). For semi-continuous models, four feature streams are used: 12 cepstra, 12 delta cepstra, 3 power terms, and 12 double delta cepstra (c/1..L-1/,d/1..L-1/,c/0/d/0/dd/0/,dd/1..L-1/). Cepstral mean normalization is performed using the current utterance.

The Resource Management acoustic models (as trained using CMU's tutorial scripts [3]) are used to force-align the WSJ word-level training transcripts. This chooses the most probable pronunciation variant for each word and also inserts instances of the silence "word".

The recipe initializes a set of 40 HMMs (39 monophones plus 1 silence model). Each HMM has a left-to-right topology with self-loops. Depending on the configuration file, each HMM will have either 3 output states and no skip transitions, or 5 output states and transitions which skip states.

When training semi-continuous models, 2500 training utterances are used to initialize 256 codewords entries in each of 4 codebooks (corresponding to the 4 feature streams). Codewords are found using k-means clustering. The codebook Gaussians' mean and variance are initialized based on the 2500 utterances. The mixture weights and transition probabilities in each state of each HMM are initialized equiprobable.

For continuous models, the mean and variance of the Gaussian in each state of each HMM is set to the global mean and variance of the training data. Mixture weights and transition probabilities are initialized equiprobable.

The context-independent models are trained using multiple rounds of Baum-Welch training. Training iterations continue until the ratio of the likelihood of the training data using the current iteration's parameter to the last iteration's likelihood is less than some threshold (0.04 was used here).

Next, a list of all possible triphones given the pronunciation dictionary is generated. The number of times each possible triphone occurs in the training data is counted. Any triphone that did not occur in the training data is discarded. The context-dependent triphones are created, copying the model parameters in each state of the corresponding trained context-independent triphone. The context-dependent models are trained using Baum-Welch until convergence.

The states of all triphones (both seen and unseen in the training data), are now tied. The number of tied-states (or senones) is specified in the configuration file. A data-driven clustering algorithm is used to automatically derive a set of questions for building a decision tree. Note that similar to HTK, a linguistically motivated set of questions could be used, but this was not tested here. The decision tree is pruned until the desired number of tied-states is reached.

The tied-state context-dependent models are trained using Baum-Welch until convergence. After convergence, if the user-specified number of Gaussians per state has not been reached, each Gaussian is split into two with each Gaussian getting a slightly perturbed mean. Baum-Welch training is then performed and the splitting/training process repeated until the desired number of Gaussians is reached.

The Sphinx recipe is available from [4]. Many of the acoustic models used in the experiments described later in this paper are available for download.

Parameter	Value
Pruning beam width	200.0
Max model pruning	2048
Word end beam width	100.0
Tokens per state	4
Word insertion penalty	-4.0
Language model scale factor	15.0

Table 1. HTK decoding parameters used in experiments.

Parameter	Value
Pruning beam width	1e-60
Word beam width	1e-30
Phone beam width	1e-60
Fast GMM beam width	1e-80
Max active HMMs	20000
Max active words	20
Max history	100
Word insertion penalty	0.2
Language model scale factor	12.0

Table 2. Sphinx-3 decoding parameters used in experiments.

2. TUNING EXPERIMENTS

A series of acoustic models were trained using the previously described HTK and Sphinx recipes. Both recipes used the full set of WSJ0 and WSJ1 training data (211 hours). This included the long-term and journalist training data. I found this improved performance on held-out test sets as compared to training on just the short-term utterances (the SI-284 training set for example).

In addition, the HTK recipe used the TIMIT corpus for bootstrapping of monophones and as additional training data. The Sphinx recipe used the Resource Management corpus to train an initial acoustic model which was used to force align the WSJ training data.

The resulting acoustic models were evaluated on the November 1992 ARPA WSJ test set (Nov'92, 303 sentences) and the San Jose Mercury sentences from the WSJ1 Hub 2 test set (si_dt_s2, 207 sentences). Nov'92 was evaluated using the WSJ 5K non-verbalized 5k closed vocabulary set and the WSJ standard 5K non-verbalized closed bigram language model. si_dt_s2 was evaluated using a bigram language model trained on the English Gigaword corpus with a vocabulary of the top 60K words appearing in the corpus.

The real-time factor (xRT) was measured on a 2.8GHz Pentium 4 computer. Unless specified otherwise, the Sphinx-3 decoder was used for the Sphinx experiments. For HTK to be competitive with Sphinx with respect to xRT, the newly released large vocabulary decoder HDecode was used. In other work [5], I found that while HVite can provide comparable accuracy, it requires much more processing time (20.5% WER at 23 xRT on si_dt_s2).

HTK and Sphinx decoding parameters such as beam widths, language model scale factor, insertion penalty, etc. were tuned on the Nov'92 test set. In the case of computation related parameters such as beam widths, parameters were set to provide the lowest real-time factor while causing only a minor increase in word error rate. The specific values used are given in table 1 and 2.

RO	TB	Resulting tied-states
200	5500	4005
200	3000	5998
200	1850	8003
200	1250	9991

Table 3. Pruning parameters used for clustering in HTK.

2.1. Number of tied-states

HTK and Sphinx acoustic models were trained varying the number of tied-states (senones) between 4000, 6000, 8000 and 10000.

In the case of HTK, the exact number of tied-states cannot be specified, but instead thresholds are given to the phonetic decision tree state clustering step. The outlier threshold (RO) was held constant and the threshold controlling clustering termination (TB) was varied (see table 3).

On the "easy" 5K vocabulary Nov'92 task, there was little or no WER advantage in using more tied-states for either Sphinx (figure 1) or HTK (figure 2). On the "harder" 60K vocabulary si_dt_s2 task there appears to be a modest advantage to more tied-states using Sphinx (figure 7), but little difference using HTK (figure 8).

Of course having more tied-states requires the decoder to compute more Gaussian likelihoods per observation. This is shown by the increased xRT factor for the higher numbers of tied-states in figures 4, 5, 10, and 11.

2.2. Number of Gaussians

Recognition experiments where conducted on models with a varying number of Gaussians per state. Results for a single Gaussian per state were omitted from the graphs for clarity. In all cases the omitted single Gaussian model performed much worse than the semi-continuous or two Gaussian model.

Both Nov'92 (figure 1 and 2) and si_dt_s2 (figure 7 and 8) tasks show continued reductions in WER as exponentially

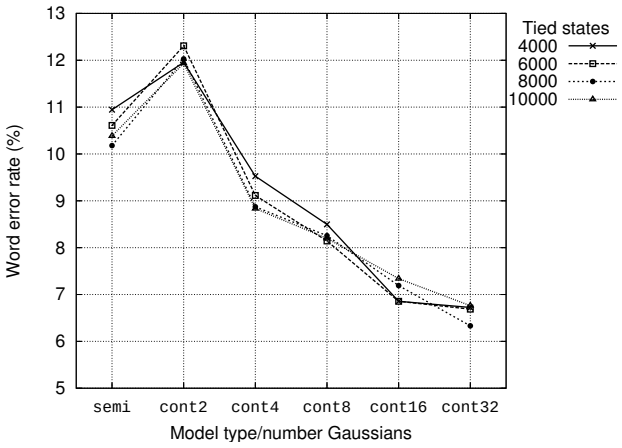

Fig. 1. Sphinx WER on Nov'92, 3 state HMM without skips.

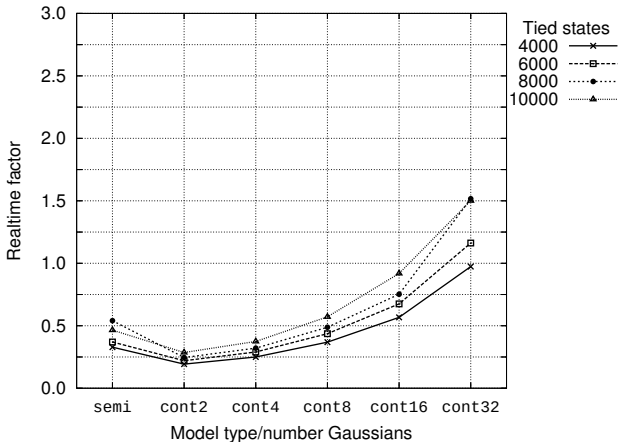

Fig. 4. Sphinx xRT on Nov'92, 3 state HMM without skips.

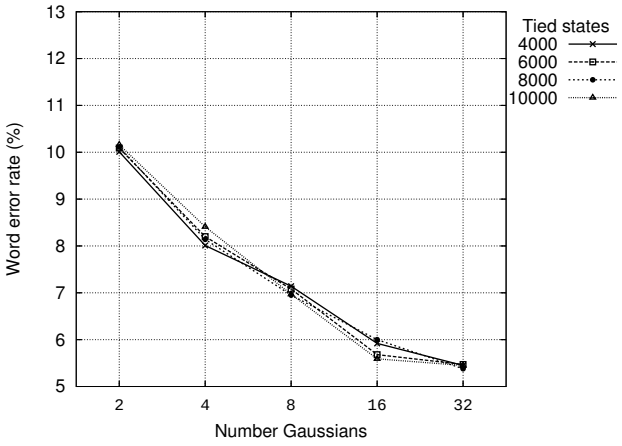

Fig. 2. HTK WER on Nov'92.

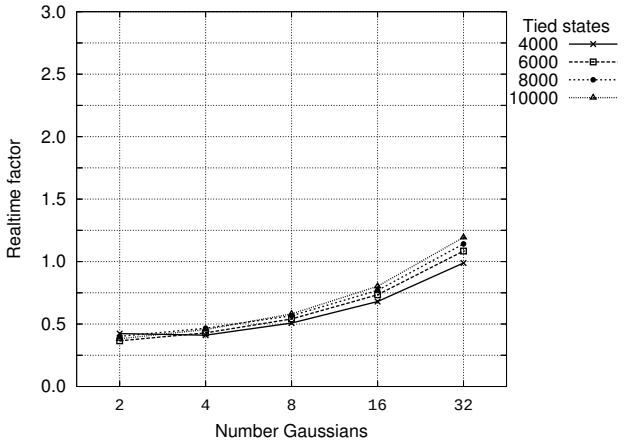

Fig. 5. HTK xRT on Nov'92.

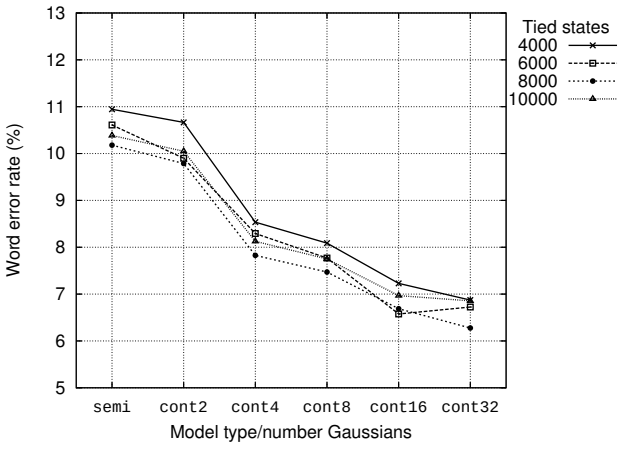

Fig. 3. Sphinx WER on Nov'92, 5 state HMM with skips.

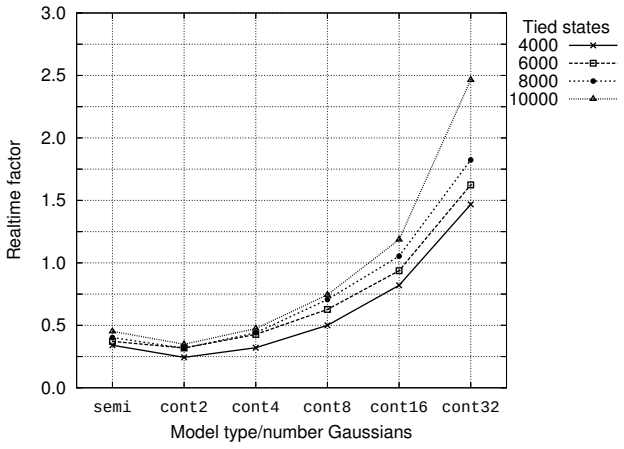

Fig. 6. Sphinx xRT on Nov'92, 5 state HMM with skips.

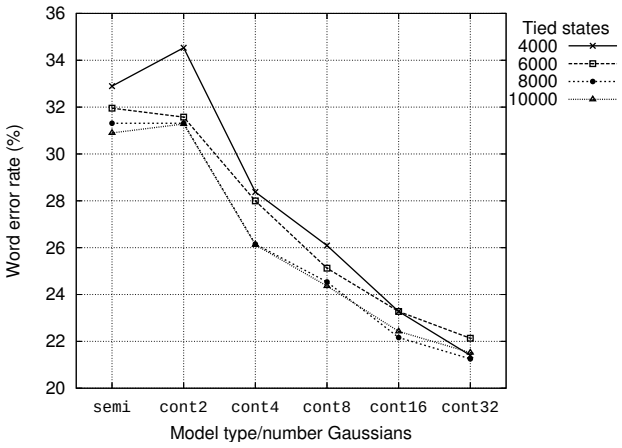

Fig. 7. Sphinx WER on si_dt_s2, 3 state HMM without skips.

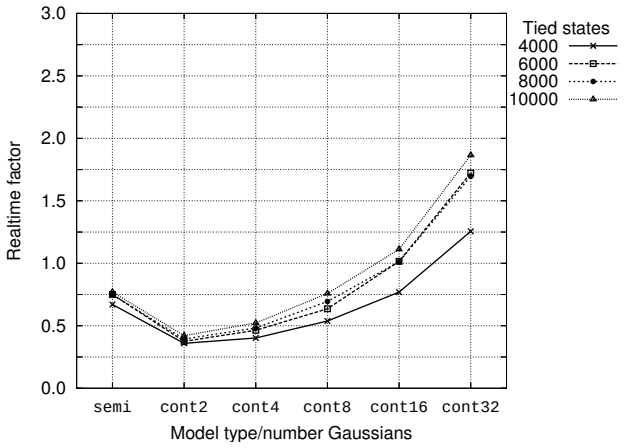

Fig. 10. Sphinx xRT on si_dt_s2, 3 state HMM without skips.

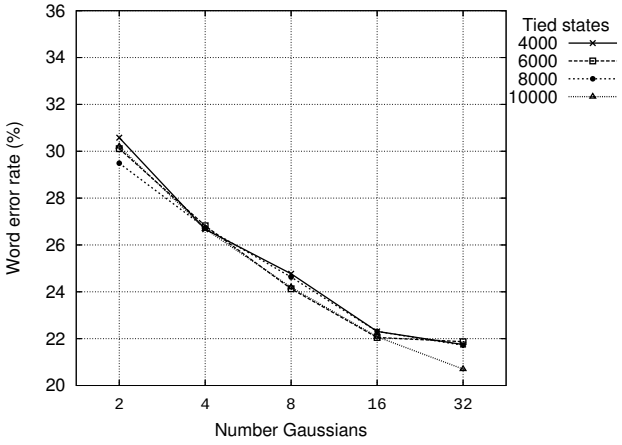

Fig. 8. HTK WER on si_dt_s2.

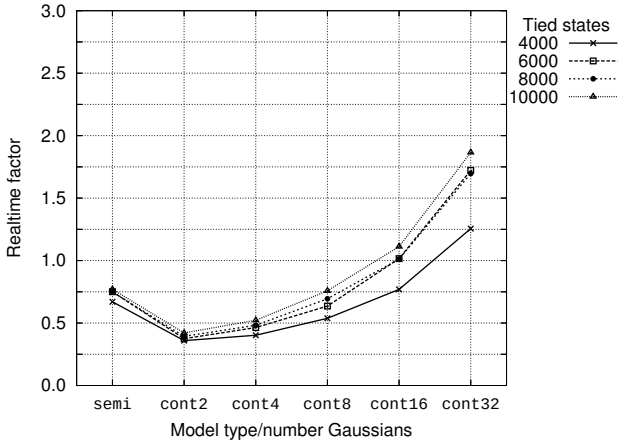

Fig. 11. HTK xRT on si_dt_s2.

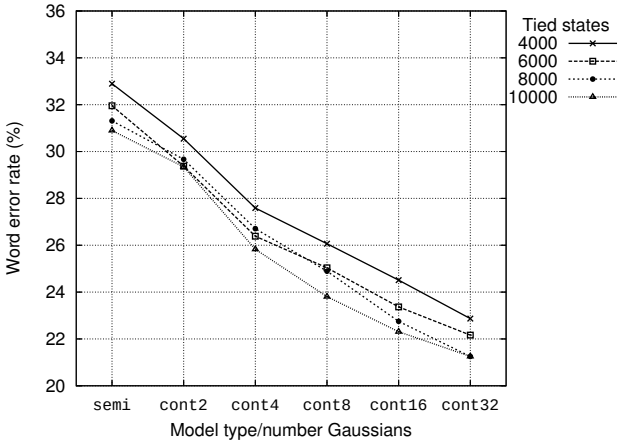

Fig. 9. Sphinx WER on si_dt_s2, 5 state HMM with skips.

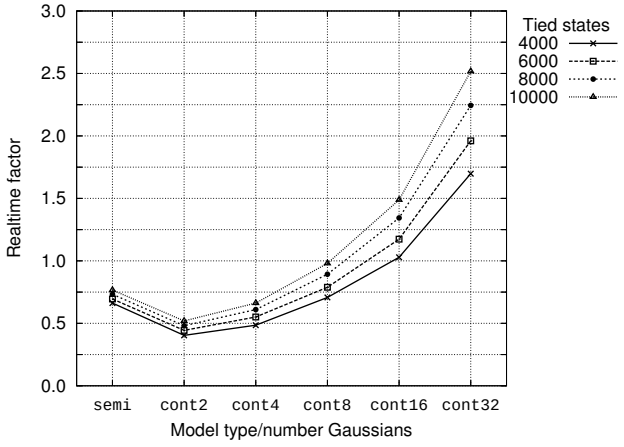

Fig. 12. Sphinx xRT on si_dt_s2, 5 state HMM with skips.

more Gaussians are added to the models. Noticeable gains were made even from 16 to 32 Gaussians suggesting even more Gaussians might prove advantageous.

The large number of Gaussians per state does not come for free, the real-time factor increases significantly as more Gaussians were added (figures 4, 5, 10 and 11).

Using the Sphinx recognizer, further tests were done on models with 64 and 128 Gaussians. As shown in figure 13, more Gaussians provided no additional benefit on either the Nov'92 or si_dt_s2 test sets. Using so many Gaussians also slows the recognizer to significantly below real-time (figure 14).

2.3. HMM topology

All models thus far have used the fairly standard HMM topology consisting of 3 output states, self-loops, and no skip transitions. Sphinx models were also trained using a 5-state HMM with skip transitions. The 5-state topology (along with semi-continuous models) allows decoding using the older but potentially faster Sphinx-2 decoder.

Using 8000 tied-states, on Nov'92 the five-state model provided a lower WER for models with up to 16 Gaussians (figure 3). However on si_dt_s2, the 5-state only did better for two Gaussians (figure 9).

The crossover point at which the 3-state model outperforms the 5-state model appears to depend both on the difficulty of the recognition task and the number of tied-states. This is likely due to the additional modeling flexibility offered by the 5-state topology. Eventually this flexibility is equaled or surpassed by the flexibility offered by a large number of Gaussians per state. This flexibility does not come for free, the 5-state models are more expensive to compute as shown in figure 6 and figure 12.

2.4. Semi-continuous versus Continuous

Using the Sphinx-3 decoder and the given tuning parameters, the semi-continuous models offered no advantages. The WER rate and real-time factors were lower for a continuous model with two or more Gaussians. To test if semi-continuous models might have an advantage with the Sphinx-2 or PocketSphinx decoders, experiments comparing the decoders were performed on a semi-continuous model with 5 states, skip transitions, and 8000 tied-states. The Sphinx-2 decoder parameters were set to provide similar WER to Sphinx-3 on Nov'92 test data (table 4). PocketSphinx used the same decoder parameters as Sphinx-3.

On the Nov'92 test set, PocketSphinx and Sphinx-2 provided a similar WER but were significantly faster than Sphinx-3 (table 5). On the si_dt_s2 test set, PocketSphinx and Sphinx-2 had a higher WER but were only marginally faster than Sphinx-3 (table 6).

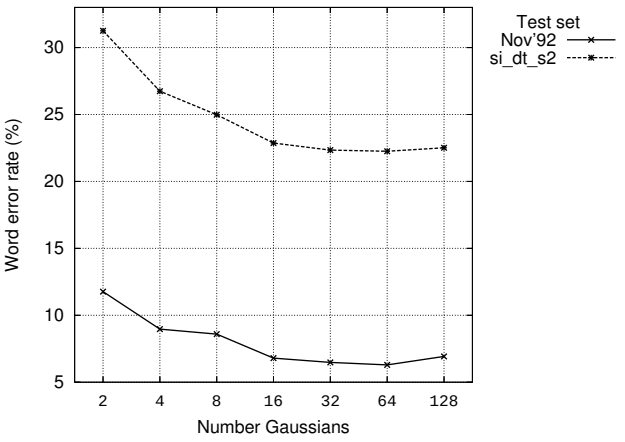

Fig. 13. Sphinx WER on Nov'92 and si_dt_s2.

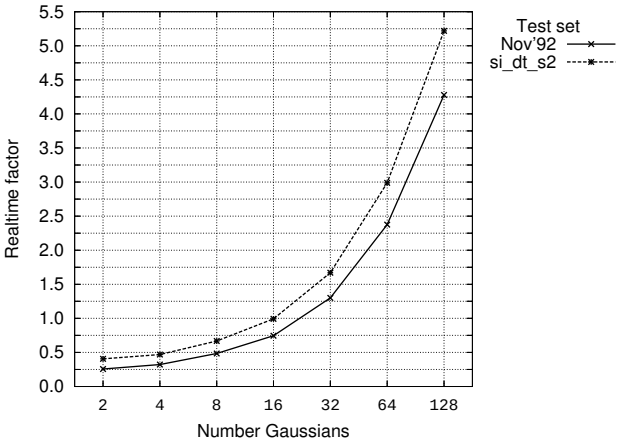

Fig. 14. Sphinx xRT on Nov'92 and si_dt_s2.

Parameter	Value
Pruning beam width	1e-7
Word beam width	3e-5
Phone beam width	1e-7
Last phone beam	1e-6
Last phone internal beam	3e-5
Word insertion penalty	0.2
Language model scale factor	12.0

Table 4. Sphinx-2 decoding parameters used in semi-continuous experiments.

Decoder	WER	xRT
PocketSphinx	10.2%	0.16
Sphinx-2	10.1%	0.26
Sphinx-3	10.2%	0.42

Table 5. WER and xRT on Nov'92 using different decoders and semi-continuous models.

Decoder	WER	xRT
PocketSphinx	38.8%	0.58
Sphinx-2	39.8%	0.70
Sphinx-3	31.1%	0.74

Table 6. WER and xRT on si_dt_s2 using different decoders and semi-continuous models.

3. CONCLUSIONS

In this paper, I described recipes which allow easy training and experimentation using the WSJ corpus and the Sphinx or HTK speech recognizers. I hope the recipes and the resulting trained acoustic models will be useful to other researchers.

Tuning experiments were conducted with both recognizers to test accuracy and real-time performance on small and large vocab recognition tasks. While I stress results may vary depending on your recognition task, real-time requirements, training and decoding parameters, etc., from my experiments I conclude the following:

- Sphinx-3 and HDecode provide comparable levels of WER and xRT.
- Increasing the numbers of tied-states beyond 4000 causes a significant increase in xRT with little reduction in WER.
- More Gaussians per state reduces WER (but not past 32 per state).
- HMM topology of 3-states with no skips is better than 5-states with skips.
- Continuous models with a small number of Gaussians have similar xRT but provide much lower WER than semi-continuous models.
- For semi-continuous models used on small vocabulary tasks, PocketSphinx and Sphinx-2 provide much better xRT with a similar WER.
- For large vocabulary tasks, continuous models with a large number of Gaussians are needed for sufficient accuracy.

4. REFERENCES

[1] S. Young, D. Ollason, V. Valtchev, and P. Woodland, *The HTK Book (for HTK Version 3.2)*, Cambridge University Engineering Department, 2002.

[2] Keith Vertanen, "HTK Wall Street Journal Training Recipe," http://www.inference.phy.cam.ac.uk/kv227/htk/, 2006.

[3] "Robust Group Tutorial," http://www.speech.cs.cmu.edu/sphinx/tutorial.html.

[4] Keith Vertanen, "CMU Sphinx Wall Street Journal Training Recipe," http://www.inference.phy.cam.ac.uk/kv227/sphinx/, 2006.

[5] Keith Vertanen, "Speech and speech recognition during dictation corrections," *Proceedings of Interspeech*, September 2006.