COMPLEMENTARY SYSTEM GENERATION USING DIRECTED DECISION TREES

C. Breslin and M.J.F. Gales

Cambridge University Engineering Department
Trumpington Street, Cambridge, CB2 1PZ, UK
{cb404,mjfg}@eng.cam.ac.uk

ABSTRACT

Large Vocabulary Continuous Speech Recognition (LVCSR) systems often use a multi-pass decoding strategy with a combination of multiple systems in the final stage. To reduce the error rate, these models must be complementary, i.e. make different errors. Previously, complementary systems have been generated by independently training a number of models, explicitly performing all combinations and picking the best performance. This method becomes infeasible as the potential number of systems increases, and does not guarantee that any of the models will be complementary. This paper presents an algorithm for generating complementary systems by altering the decision tree generation. Confusions made by a baseline system are resolved by separating confusable states, which might previously have been clustered together using the standard decision tree algorithm. Experimental results presented on a Broadcast News Mandarin task show gains when combining the baseline with a complementary directed decision tree system.

Index Terms— Speech recognition, Complementary systems, System combination

1. INTRODUCTION

Large Vocabulary Continuous Speech Recognition systems typically use a multi-pass decoding strategy where a number of systems are combined in the final stage. Schemes such as ROVER [1] and CNC [2] are used to combine the outputs from the multiple systems to give the final hypothesis. One example of such a system is that built for Broadcast News transcription at Cambridge university [3, 4]. Improvements can only be obtained from the final combination stage if the individual systems are *complementary*. That is, they must make different errors. The standard approach to generating complementary systems is to independently train a number of different systems, perform all possible combinations and select the best result. For example, the individual systems might use different segmentations [4], frontends [3], or dictionaries [4]. It is not possible to predict which systems are complementary based on individual performance alone, hence the need to perform all possible combinations. Furthermore, it is not guaranteed that any independently trained systems will be complementary, and it becomes increasingly difficult to perform all combinations as the number of potential systems grows.

Previous work has also shown that the combination of independent systems with very different error rates often yields no gain, and so it is desirable to combine independent systems with comparable error rates [4]. However, this conflicts with the requirement that complementary systems must make different errors, and hence limits the type of systems that can be combined in practice.

Ensembles of complementary classifiers have successfully been used in the machine learning community for many years. It has been found that, for both theoretical and practical reasons, they often outperform the individual constituent classifiers [5]. Several methods exist for training complementary classifiers but many of these are not directly applicable to the task of automatic speech recognition.

For these reasons, recent work has begun to look at algorithms for explicitly training complementary systems in ASR. These have focused mainly on altering the training algorithm. For example, boosting [6, 7] and Minimum Bayes Risk Leveraging [8] both weight the training data set to focus later classifiers on portions of training data which are poorly modelled by previous systems. These algorithms no longer focus on individually training each system to have the best performance, but rather on training an ensemble of models so that their combination has the best performance.

Another approach to generating complementary systems is to introduce randomness into the training. This is done in [9] by randomly selecting portions of the training data, and in [10] via the decision tree algorithm. Introducing randomness into a number of systems does lead to variety, but has the same problems as the standard approach to generating complementary systems; specifically that the optimal order of combination is not predictable.

This paper presents a new algorithm for generating complementary systems by separating confusable states in the decision tree. The following three sections describe the standard decision tree algorithm, random decision trees and the directed decision tree algorithm. Next, experimental results are presented on a Broadcast News Mandarin task, and conclusions are presented.

2. DECISION TREE CLUSTERING

Decision trees are binary trees that are used to cluster states of HMMs for parameter tying [11]. They contain questions, typically concerning triphone context, at their nodes, and states are clustered at their leaves. Decision trees are widely used as they provide an elegant way to cluster unseen contexts, they allow expert knowledge to be incorporated via the questions, and their size can be dependent on the amount of training data available. There are three stages to building a decision tree:

1. **Statistics**
   - Obtain statistics for *seen triphone* contexts using the forward-backward algorithm

2. **Question Selection**
   - Recursively build the tree by selecting the question which gives the highest data likelihood

3. **Stopping criterion**
   - Stop building the tree when the data likelihood falls below a threshold

The decision tree algorithm does not consider whether states are confusable or not when performing the clustering, and so it is possible that confusable states will be clustered. This is undesirable as clustered states share the same parameters, and hence the information available to distinguish them is limited to context and linguistic information. The question selection stage in decision tree generation is locally optimal, which means that altering it can lead to very different decision trees, and thus different HMM sets. For these reasons, the decision tree algorithm is a good stage to focus on for complementary system generation. A further advantage of altering the decision tree algorithm is that no changes need to be made to the training algorithm. However, the decision tree generation stage typically occurs early in the process of building a system, and so it is time-consuming to build many systems with different decision trees. The remainder of this paper considers one existing and one new approach to generating complementary systems by altering the decision tree algorithm. These differ from previous work on decision tree generation in their aim to generate complementary systems rather than one single best system.

3. RANDOM DECISION TREES

Introducing randomness is one way to create varying classifiers [5]. HMM parameter estimation algorithms are fairly robust to randomness, so an alternative is to introduce randomness into the decision tree algorithm by altering the question selection stage [10]. Instead of selecting the single best question, the tree is grown by randomly choosing a question from the best N questions. An example of this question selection compared to the standard approach is given in figure 1. The decision tree algorithm becomes:

1. **Statistics**
   - Obtain statistics for seen triphone contexts using the forward-backward algorithm

2. **Question Selection**
   - Recursively build the tree by *randomly selecting from the top N questions* which give the highest data likelihood

3. **Stopping criterion**
   - Stop building the tree when the data likelihood falls below a threshold

	Question	Likelihood
BEST ►	**Left front fricative?**	**[70.1]**
	Right back fricative?	[69.6]
	Right nasal?	[69.5]
	Left vowel central?	[68.8]
	Right liquid?	[67.5]
	Left nasal?	[65.4]
	Left unvoiced fricative?	[64.3]

RANDOM (N=5)

Fig. 1. Random tree question selection

A randomised decision tree does not guarantee that the systems are complementary, but using multiple systems built on random trees makes it more likely that confusable states will be separated in at least *some* of the trees. Hence, it is expected that a combination of outputs from systems built on random trees will give improvements.

The problem that systems should have comparable error rates is no longer an issue as altering the value of N provides some control over individual system performances. A larger value of N will tend to lead to trees being very different from the best tree, and it is likely that system performance will degrade. However, as there is no guarantee of obtaining complementary systems, random decision tree systems suffer from the problem that the optimal order of combination cannot be predicted and hence all system combinations must be performed in order to find the best.

4. DIRECTED DECISION TREES

In order to address the weaknesses of standard and random decision trees, it is desirable to build decision trees in such a way that explicitly separates confusable states. Directed decision trees do this by using a second set of statistics when building the tree. This second set of statistics is obtained by using a weighted forward-backward algorithm, where the weighting function assigns a greater weight to portions of the training data that are poorly modelled by a baseline model. Hence, a higher weight will be assigned to states that are easily confused. These weighted statistics are used to perform the question selection, while the original statistics are used for the stopping criterion. An example of this question selection is shown in figure 2. The directed decision tree thus aims to separate confusable states in the decision tree, hence resolving confusions made by the baseline recogniser, and is expected to be complementary to the baseline. The directed decision tree algorithm is:

1. **Statistics**
   - Obtain original statistics for seen triphone contexts
   - Obtain *weighted* statistics for seen triphone contexts

2. **Question Selection**
   - Recursively build the tree by selecting the question which gives the highest likelihood with respect to the *weighted* statistics

3. **Stopping criterion**
   - Stop building the tree when the data likelihood falls below a threshold, with respect to the *original* statistics

Question	Original	Weighted
Right liquid?	**[67.5]**	**[37.4]** ◄ DIRECTED
Right nasal?	[69.5]	[37.2]
Left nasal?	[65.4]	[36.7]
Left vowel central?	[68.8]	[36.3]
Left front fricative?	*[70.1]*	*[35.8]* ◄ *ORIGINAL*
Left unvoiced fricative?	[64.3]	[35.3]
Right back fricative?	[69.6]	[34.2]

Fig. 2. Directed tree question selection

A suitable method for weighting the training data is needed. There are many existing applications which make use of a weighted training data set, typically as part of the HMM training algorithm. For example, boosting [6, 7], active training [12] and MBRL [8]. These apply weightings at various levels; at the utterance level [7], the word level [8] or the frame level [6]. They also differ in the form of weighting function used, and how that weighting is applied during training.
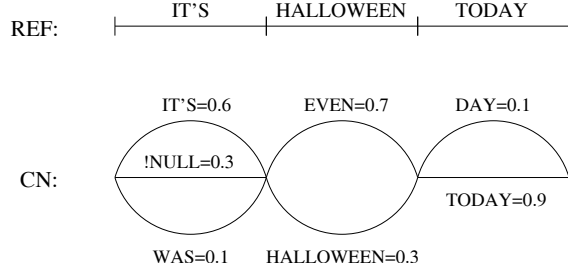
Fig. 3. Calculating the weighting function

Reference word	Weight
IT'S	0.4
HALLOWEEN	0.7
TODAY	0.1

Table 1. Weights for words in figure 3 using equation 1

The weighting function itself should reflect how well classified the training data is. One possibility is to use a confidence measure, for example the likelihood-ratio [12] to identify correctly recognised portions of the training data. The drawback of this approach is that confidence measures are not always reliable. Alternatives include using the reference transcription to identify errors and basing the weighting function around, for example, word posteriors [8] or a discriminative criterion such as the pseudo-loss used in boosting.

The final consideration is how to apply the weighting function in training. Applying a weighting at the utterance level is straightforward as the utterance boundaries are already defined, either manually or by automatic segmentation. Applying the weight at a finer granularity, e.g. word level, requires some further processing. One option is to force-align the training set utterances to obtain word boundaries, and hence weight the observations for a particular word [6]. This may lead to errors at word boundaries. In contrast, word-level weights can be applied directly to states and hence avoid the need for force-alignment [8].

In this paper, the weighting is done as follows. Weights are calculated for each reference word by obtaining confusion networks for each training set utterance and aligning these with the reference transcription, as in figure 3. This makes it straightforward to see whether a word is correctly modelled or not. A weighting function based on average word posteriors is used:

$$l(\tilde{\mathcal{W}}) = \left(1 - \frac{1}{S}\sum_{s=1}^{S} P(\tilde{\mathcal{W}}|\mathcal{O}; \mathcal{M}^{(s)})\right)^{\alpha} \qquad (1)$$

The second term in the weighting function is the average posterior of the reference word given a number of previous systems $\mathcal{M}^{(1)} \cdots \mathcal{M}^{(S)}$ and so is directly related to how well that particular word is modelled. The weighting is applied by multiplying each state occupancy count by the weight of the word it relates to. Using this weighting function with $\alpha = 1$, the losses of the words in figure 3 are given in table 1. The effect is to assign a higher weighting to reference words with low posterior. The effect of increasing α in this equation is to reduce the proportional influence of well modelled words and increase the influence of poorly modelled words. This in turn leads the algorithm to focus more and more on the very poorly

Decision Tree		CN-decoding			
		dev04f	eval03m	eval04	y1q1
BASELINE		15.8	8.7	26.4	31.3
RANDOM	BEST	15.7	8.9	26.5	31.0
	WORST	16.4	9.1	27.0	31.3
DIRECTED		15.3	8.8	26.2	31.0

Table 2. Random and Directed Tree baseline results (CER %)

Decision Tree		CNC with BASELINE			
		dev04f	eval03m	eval04	y1q1
RANDOM	BEST	15.0	7.7	25.8	29.5
	WORST	15.4	7.9	26.0	30.0
DIRECTED		15.1	7.8	25.9	29.5

Table 3. Random and Directed Tree combination results (CER %)

modelled training data. If a particular model has a large number of reference words with posterior close to 1, it may be useful to increase α and so decrease the influence of these words.

5. EXPERIMENTAL RESULTS

Experiments were performed on a Broadcast News Mandarin task. The baseline systems were trained using 148 hours of data; 28 hours of Hub-4 data released by the Linguistic Data Consortium (LDC) with accurate transcriptions, and 120 hours of TDT4 data with only closed-caption references provided. Light supervision techniques were used on the latter portion. A trigram language model with a 50k wordlist was used for decoding. Trigram lattices were converted to confusion networks in order to perform confusion network decoding for the individual system results [13]. This allows the gains achieved by combination to be seen easily. Results are given on four test sets: dev04f consists of 0.5 hours of CCTV data from shows broadcast in November 2003, eval04 includes 1 hour of data from CCTV, RFA and NTDTV broadcast in April 2004, eval03m is 0.6 hours of mainland shows from February 2001 and y1q1 is 3 hours of data from October 2005. This system is fully described in [3]. In contrast to [3], these experiments use no speaker adaptation, and the first set of experiments uses an ML trained baseline.

5.1. Random and Directed trees

The first set of experiments compares directed to random decision trees. The baseline system was built from a standard decision tree, with 16 component GMMs for each state and a 39 dimensional feature vector. This vector consisted of 13 PLP features with 1st and 2nd derivatives added, and there were 6070 unique states. Five random decision trees were also built using a value of $N = 5$, and the number of unique states varied from 5585 to 5618. A directed decision tree was built using the baseline ML system to gather the weighted statistics and the weighting described in the previous section with $\alpha = 1$; this tree contained 5830 unique states. The number of states in a system did not appear to have any correlation with the final error rates, implying that any gains obtained were from the different methods of state tying rather than a change in the number of model parameters.

Table 2 shows the individual system results for the baseline, random and directed decision tree systems. For the random systems, the best and worst performances are given. The best and worst models differed between testsets, and numbers are *individually* chosen for each testset from the five random tree systems. It can be seen that the

directed decision tree system performs as well as or better than the best random decision tree system on all testsets, and outperforms the baseline on three of the four sets. In contrast, the individual random systems tend to perform slightly worse than the baseline system.

Table 3 shows the performance of the random and directed tree systems when combined with the baseline. Again, the best and worst performances for the random systems were picked separately for each test set. The combination of random systems with the baseline gave improvements over the individual system performances, as has been seen previously [10]. The directed tree system performs marginally worse than the best random tree system, yet significantly better than the worst. Interestingly, the best performing individual random system did not necessarily correspond to the best result in combination, highlighting the fact that the optimal order of combination is not predictable.

It is expected that if many random trees are built, the best of these would outperform both the baseline and the directed tree systems. However, these experiments show that the directed tree performance is close to that of the best random tree, without the uncertainty associated with randomness. In addition, the directed tree led to a time saving when building the systems as there was no need to train multiple systems based on multiple trees.

5.2. Discriminative training

The previous section showed how directed decision trees perform when combined with an ML baseline, but it is unclear whether these gains will still be seen with a more complex model. Hence, a second set of experiments evaluated whether a discriminatively trained baseline could benefit from combination with a directed decision tree system.

An MPE baseline was built from the previous ML system by first expanding the feature vector to 52 dimensions by adding the 3rd derivatives, then mapping down to 39 dimensions using an HLDA transform. Next, pitch was added, and a re-ordering of components was performed based on state occupancies. Thus the feature vector was 42 dimensions, and there was an average of 16 components per state. Finally, MPE training was performed with an MMI prior for I-smoothing. A new directed decision tree was built using weighted statistics from this MPE baseline, and an MPE trained directed-tree system was built in the same way.

Table 4 shows the results obtained using these two systems. Firstly, the individual results for each system are presented. In contrast to the ML system, the directed decision tree system no longer outperforms the baseline. However, when the two systems are combined, improvements in error rate are still seen. For example, on the y1q1 set, the individual system error rates are 30.0 and 30.1% CER, yet the combination gives an error rate of 28.9%; a gain of 1.1% absolute on top of the gain achieved from discriminative training. Similar relative gains are seen on the three other testsets.

$\mathcal{M}$	Decision Tree	dev04f	eval03m	eval04	y1q1
S0	BASELINE	12.5	7.2	21.2	30.0
S1	DIRECTED	12.1	7.4	21.2	30.1
S0 + S1	CNC	11.7	6.3	20.5	28.9

Table 4. MPE system combination results (CER %)

6. CONCLUSIONS

This paper has presented a new algorithm for building complementary systems based on separating confusable states in decision trees.

This algorithm differs from standard decision tree generation by using weighted statistics alongside the original statistics when growing the decision tree. By weighting the training data to reflect confusability, states which are easily confused in the training set are separated in the decision tree. This implicitly builds complementary systems, compared to previous explicit methods which alter the training algorithm.

This algorithm was evaluated using a weighting function based on word posteriors, and was compared to a previous approach based on random decision trees. Using this particular weighting function, the directed tree system was found to outperform both the baseline and random tree systems individually. When combined with the baseline system, it was found to perform almost as well as the best random tree system, yet significantly better than the worst random system. The advantage of the directed tree algorithm is that only one system needs to be built, rather than multiple systems. Further experiments showed that the gains from using directed trees are additive when discriminative training is also performed.

7. REFERENCES

[1] J.G. Fiscus, "A post-processing system to yield reduced word error rates: Recogniser output voting error reduction (ROVER)," in *Proc. IEEE ASRU Workshop, pages 347-352*, 1997.

[2] G. Evermann and P. C. Woodland, "Posterior probability decoding, confidence estimation and system combination," in *Proceedings Speech Transcription Workshop*, College Park, MD, 2000.

[3] R. Sinha, M.J.F. Gales, D.Y. Kim, X.A. Liu, K.C. Sim, and P.C. Woodland, "The CU-HTK Mandarin broadcast news transcription system," in *Proceedings ICASSP*, 2006.

[4] M.J.F. Gales, D.Y. Kim, P.C. Woodland, H.Y. Chan, D. Mrva, R. Sinha, and S.E. Tranter, "Progress in the CU-HTK broadcast news transcription system," *IEEE Trans. Speech and Audio Processing*, vol. 14, pp. 1513–1525, 2006.

[5] T.G. Dietterich, "Machine learning research: Four current directions," *The AI Magazine*, 1997.

[6] R. Zhang and A. I. Rudnicky, "A frame level boosting training scheme for acoustic modelling," in *Proceedings ICSLP*, 2004.

[7] C. Meyer, "Utterance-level boosting of HMM speech recognisers," in *Proceedings ICASSP*, 2002.

[8] C. Breslin and M.J.F. Gales, "Generating complementary systems for speech recognition," in *Proceedings ICSLP*, 2006.

[9] L. Breiman, "Bagging predictors," *Machine Learning*, vol. 24(2), pp. 123–140, 1996.

[10] O. Siohan, B. Ramabhadran, and B. Kingsbury, "Constructing ensembles of ASR systems using randomized decision trees," in *Proceedings ICASSP*, 2005.

[11] J. Odell, *The use of context in large vocabulary speech recognition*, Ph.D. thesis, 1995.

[12] L.M. Arslan and J.H.L. Hansen, "Selective Training for Hidden Markov Models with Applications to Speech Classification," *IEEE Trans. Speech and Audio Processing*, vol. 7, pp. 46–54, 1999.

[13] L. Mangu, E. Brill, and A. Stolke, "Finding consensus among words: Lattice-based word error minimization," in *Proceedings Eurospeech*, 1999.