Directed Decision Trees for Generating Complementary Systems

C. Breslin, M.J.F. Gales

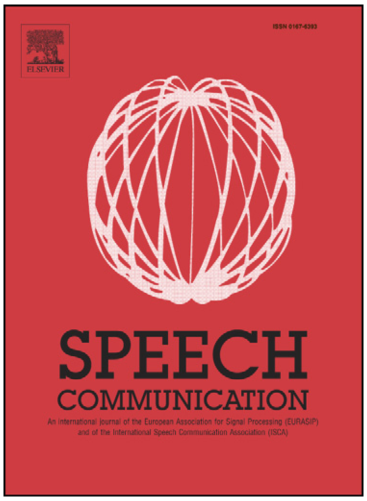
Directed Decision Trees for Generating Complementary Systems

C. Breslin, M.J.F. Gales

PII: S0167-6393(08)00138-6
DOI: 10.1016/j.specom.2008.09.004
Reference: SPECOM 1754

To appear in: *Speech Communication*

Received Date: 17 December 2007
Revised Date: 10 September 2008
Accepted Date: 14 September 2008



Directed Decision Trees for Generating Complementary Systems

C. Breslin [*,1], M.J.F. Gales [*]

Engineering Department, Cambridge University, Trumpington Street, Cambridge CB2 1PZ, United Kingdom

Abstract

Many large vocabulary continuous speech recognition systems use a combination of multiple systems to obtain the final hypothesis. These complementary systems are typically found in an ad-hoc manner, by testing combinations of diverse systems and selecting the best. This paper presents a new algorithm for generating complementary systems by altering the decision tree generation, and a divergence measure for comparing decision trees. In this paper, the decision tree is biased against clustering states which have previously led to confusions. This leads to a system which concentrates states in contexts that were previously confusable. Thus these systems tend to make different errors. Results are presented on two Broadcast News tasks - Mandarin and Arabic. The results show that combining multiple systems built from directed decision trees give gains in performance when confusion network combination is used as the method of combination. The results also show that the gains achieved using the directed tree algorithm are additive to the gains achieved using other techniques that have been empirically shown as complementary.

Key words: Speech recognition, Complementary Systems, System Combination

1. Introduction

State of the art large vocabulary continuous speech recognition (LVCSR) systems often use a multipass decoding strategy (e.g. Nguyen et al. (2002); Gauvain et al. (2002); Gales et al. (2006)). Such a framework may have many different passes, depending on the task at hand, but typically will include an initial transcription stage to obtain an approximate hypothesis, speaker adaptation and normalisation using this approximate hypothesis, then lattice generation and rescoring using more complex systems. Often, the final pass makes use of multiple systems, and schemes such as ROVER (Fiscus (1997)) and Confusion Network Combination (CNC) (Evermann and Woodland (2000)) are used to combine the outputs and obtain a final hypothesis.

It is only useful to use a combination of multiple systems if they are complementary. That is, the systems must make different errors before their combination can improve the final hypothesis. An ad-hoc approach to generating complementary systems is to independently train a number of different systems and select the ones which combine to give the best results. It is not possible to predict which systems are complementary based on individual performance alone, so all possible combinations must be performed to find the best. Also, it is not guaranteed that any independently trained systems will in fact be complementary and, as the number of individual systems increases, it becomes increasingly time-consuming to perform all the different combi-

[*] Corresponding author.
Email addresses: `catherine.breslin@crl.toshiba.co.uk` (C. Breslin), `mjfg@eng.cam.ac.uk` (M.J.F. Gales).
URL: `http://mi.eng.cam.ac.uk/~cb404` (C. Breslin).
[1] Supported by Toshiba Research Europe Ltd and the EPSRC

nations.

In previous work, examples of this ad-hoc approach have used individual systems with different segmentations (Gales et al. (2006)), frontends (Hain et al. (2007); Hwang et al. (2007)), phoneme sets (Stuker et al. (2006)) or dictionaries (Gales et al. (2006)). It has often been seen that the combination of independent systems with very different error rates yields no gain, and so it is desirable to combine independent systems with comparable error rates (Gales et al. (2006); Stuker et al. (2006)). It is difficult to build independent systems which simultaneously have comparable performance and yet make different errors, which limits the types of independently trained system that can be combined in practice.

Ensembles of complementary classifiers have successfully been used for machine learning problems as, for both theoretical and practical reasons, they often outperform the individual constituent classifiers. This is particularly true in situations where simply building a more complex model would lead to overtraining, yet ensembles of models can yield improvements in performance (Dietterich (2000)).

For these reasons, recent work has begun to look at algorithms for explicitly training complementary systems for ASR. Generic algorithms, such as boosting (Freund and Schapire (1996)) exist to build complementary classifiers for static data with a finite number of classes. However, they require a number of approximations before they can be used with dynamic speech data. Algorithms for explicitly generating complementary ASR systems have either operated in a boosting-like manner where later systems focus on portions of the training data which are harder to recognise (Zhang and Rudnicky (2004, 2003); Meyer (2002)), or have altered the decision tree generation scheme (Siohan et al. (2005); Breslin and Gales (2007b,a)). These approaches no longer attempt to train individual systems with optimal performance, but rather to train an ensemble of systems which have optimal performance when combined together.

This paper describes an approach to generating complementary systems for automatic speech recognition by altering the decision tree generation, previously presented in (Breslin and Gales (2007b,a)). The decision tree is directed so as to bias against tying states which cause confusions. A system built with a directed tree will hopefully resolve confusions, but may introduce new errors, and so will be complementary to the original system. In contrast to other work on decision trees, for example (Nock et al. (1997); Hu and Zhao (2007); Xue and Zhao (2007)), this approach is not concerned with improved decision tree generation, but with generating a sequence of decision trees that can be used to build systems that are complementary.

This paper is organised as follows. First, sections 2 and 3 review system combination methods and previous algorithms for generating complementary systems. Next, sections 4, 5 and 6 describe decision trees, random decision trees and directed decision trees. Section 7 then presents a divergence measure for the comparison of decision trees. Section 8 presents experimental results and discussion on two large vocabulary tasks, Broadcast News transcription in Mandarin and Arabic, and finally section 9 draws conclusions.

2. System Combination

If multiple systems are to be used for decoding, a method for combining them is needed. The most popular methods for this purpose are hypothesis combination schemes where each system independently decodes the data and the resulting hypotheses are combined. Two examples of this type of scheme are ROVER (Fiscus (1997)) and Confusion Network Combination (CNC) (Evermann and Woodland (2000)). Both take the outputs from multiple systems, align them, and then perform a voting at the word level to give the final hypothesis. The two schemes differ in both the form of system output used and the voting scheme.

ROVER begins with 1-best outputs from each of the systems, and aligns these to give a word transition network (WTN). Voting for each slot, or set of competing words, in the WTN is based on the frequency of occurrence $N(\mathcal{W})$ and, if available, word confidence $C(\mathcal{W})$, typically the word posterior probability.

Rather than use the 1-best hypotheses, CNC begins with lattice outputs and first converts these to confusion networks. A confusion network is a linear graph such that each arc represents a word, and aligned arcs represent competing words from the lattice. They are obtained from lattices by clustering arcs that overlap in time (Mangu et al. (1999)). Confusion network arcs are annotated with word posterior probabilities of the word, $\mathcal{W}$, given the model, $\mathcal{M}^{(s)}$, i.e. $P(\mathcal{W}|\mathcal{M}^{(s)})$. The confusion networks for all systems are aligned, and the best word for each segment is chosen using a weighted voting scheme and the posteriors from each confusion network.

The order in which systems are combined is important for both ROVER and CNC, and previous experiments have suggested that systems should be combined in increasing order of word error rate (Hoffmeister et al. (2006)).

ROVER combination for two systems reduces to picking the word with the highest confidence where there is a confusion. Hence, if the recogniser confidence scores are not reliable, then ROVER between two systems often does not perform well. As more systems are combined, ROVER becomes more robust to the confidence scores. This issue is not as important with CNC as many more words are used, so it is more robust to the exact values of confidence score. For this reason, and as many of the experimental results in section 8 combine just two systems, CNC is used as the method of combination in this paper, for both the training algorithm and in decoding.

Another, more indirect, approach to system combination is cross-adaptation (Gales et al. (2006); Stuker et al. (2006)). This is a scheme which naturally arises in a multipass adaptive framework, where the output transcriptions from one system are used in the subsequent pass as the input hypothesis to perform speaker adaptation of a second system. In practice, this form of system combination has led to improvements in performance. The experimental results in section 8 make use of both CNC and cross-adaptation.

3. Complementary System Generation

As described in section 1, an ad-hoc approach to generating complementary systems for ASR is to train a variety of diverse systems, evaluate their performance in combination, and select those which perform well together. An alternative way to build multiple diverse systems is to introduce randomness into the training algorithm (Dietterich (2000)). This can be done, for example, by adding random noise onto the training data, initialising the parameters randomly, or by bagging (Breiman (1996)) - selecting random subsets of the training data. These methods can be used regardless of the classifier, the task, and the training algorithm. If the specific classifier in question is a decision tree, then it is possible to grow the trees in a random manner and hence build a random forest (Breiman (2001)). This approach is considered in more detail for ASR in section 5.

Building diverse systems, either through different training algorithms or by injecting randomness, does not guarantee that the multiple systems are complementary. Hence, approaches have been proposed that aim to explicitly generate complementary systems. These typically operate in an iterative framework to build multiple systems, where each new system focuses on errors made by previous systems.

Boosting is one example of such an approach, and AdaBoost (Freund and Schapire (1996)) is the most commonly used boosting algorithm. It is an algorithm which operates on a binary classification task, using any form of classifier. A multi-class variant of AdaBoost, called AdaBoost.M2, was developed for a classification task with a finite, ideally small, number of output classes. Both algorithms build a series of classifiers which can then be combined. Typically the classifier is a weak learner, i.e. it performs slightly better than random, but by combining many weak learners it is possible to achieve the performance of a strong classifier.

AdaBoost maintains a distribution over all the training data, where harder examples are allocated greater weight. A new classifier is trained by taking this distribution into account, and so the algorithm begins to focus on harder training data. After each classifier has been built, the distribution over the training data is updated. As part of the training algorithm, a classifier importance is calculated, and this is used as a weight to combine all the classifiers using a voting scheme and obtain the final hypothesis.

Where there is low classification noise on the training set, i.e. the training labels are accurate, boosting outperforms randomisation as a method for generating complementary systems. In high classification noise however, the reverse is true as the later classifiers built by the boosting algorithm begin to focus on errors rather than on data which is truly hard to classify. In contrast, bagging and randomisation work much better in this situation because the randomness overcomes the classification noise (Dietterich (1999)).

There are a number of issues with applying boosting, or boosting-like, algorithms to speech recognition. Speech is not a binary classification task, there are a large number of output classes, the forms of classifier used are highly complex, the training data labels may not be accurate, the data is dynamic and, finally, the simple weighted voting scheme is not directly applicable for system combination. Hence some previous work on boosting for speech recognition has recast the problem as a phone classification one, and avoided some of these problems. For example, (Zweig and Padmanabhan (2000)) builds GMMs for each phone and performs boosting for phone classification at the frame level, (Schwenk (1999)) applies boosting to the Neural Network part of a hybrid HMM/NN system, and (Dimitrakakis and Bengio (2004)) applies boosting to whole-phone HMMs.

Boosting-like algorithms for continuous speech recognition have been implemented at the utterance level (Zhang and Rudnicky (2003); Meyer (2002)), at the word level (Breslin and Gales (2006)), and at the frame level (Zhang and Rudnicky (2004)). As these schemes depend on reweighting the training data, they have some similarities to discriminative (Povey (2005)) and active training (Kamm and Meyer (2003)), while aiming to build an ensemble of classifiers rather than a single best.

Other approaches to building complementary systems for ASR have focused on the decision tree, and these are described in the following sections.

4. Decision Trees for ASR

A typical ASR system uses parameter tying to share parameters over a number of HMM components. In this work, parameter tying is done at the state output distribution level, using a binary decision tree so multiple states share an output distribution (Odell (1995)). Decision trees contain questions at their nodes, and states are clustered at their leaves.

Decision tree clustering is a top-down clustering algorithm which maximises the likelihood of the data. As the decision tree is grown, the states clustered at each node are split according to the question which gives the best local increase in data likelihood. This is continued recursively until the total data likelihood falls below a threshold. A forward-backward pass over the data is performed to obtain an initial alignment of states to frames and hence obtain the statistics required for decision tree clustering. There are three stages in the algorithm:

(i) **Statistics**
  - Obtain statistics for all *seen triphone* contexts (i.e. those that appear in the training data) using the forward-backward algorithm

(ii) **Question Selection**
  - Recursively build the tree by selecting the question which gives the highest change in data likelihood at each node

(iii) **Stopping criterion**
  - Stop construction of the tree when the data likelihood falls below a threshold

The effect of the algorithm is to cluster states which are close in acoustic space, as these are likely to be well modelled by a shared distribution. A disadvantage of parameter sharing is that only contextual and linguistic information is available to distinguish clustered states, as they share an output distribution. This is not ideal if clustered states are confusable and lead to errors.

As the decision tree algorithm is only locally optimal, slightly altering any stage of the process can lead to very different decision trees being built. For this reason, 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 speech recognition system, and so it is time-consuming to build and evaluate many systems with different decision trees.

5. Random Decision Trees

Randomness can be introduced to build diverse ASR systems via the decision tree generation. A random decision tree is built by altering the question selection stage of the tree generation. Instead of selecting the best question when splitting the data at each node, a random choice from the top N is made (Breiman (2001); Siohan et al. (2005)). Thus the decision tree algorithm becomes:

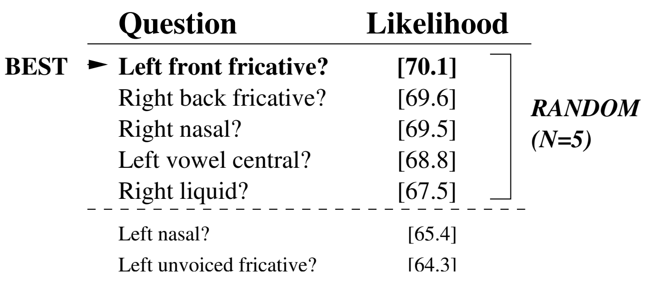
(i) **Statistics**
  - Obtain statistics for all seen triphone contexts using the forward-backward algorithm

(ii) **Question Selection**
  - Recursively build the tree by *randomly selecting from the top N questions* which give the highest change in data likelihood

(iii) **Stopping criterion**
  - Stop construction of the tree when the data likelihood falls below a threshold

An example of the random tree generation is shown in figure 1. Rather than select the question *'Left front fricative?'* which is the optimal in standard decision tree generation, a random question from the top five is selected instead.

Introducing randomness does not guarantee to build complementary systems, but using multiple random systems can lead to an average performance



	Question	Likelihood	
BEST ►	**Left front fricative?**	**[70.1]**	*RANDOM (N=5)*
	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]	

Fig. 1. *Random decision tree question selection*

that is better than the individual systems (Dietterich (1999)). In the case of random decision trees for ASR, 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 to combine well 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 original tree, and it is likely that system performance will degrade.

Injecting randomness has the advantage that any number of diverse systems can be built, although the process isn't deterministic and hence not repeatable. Also, as there is no guarantee of obtaining complementary systems, random decision tree systems suffer from the problem that the optimal number of systems and order of combination cannot be predicted, hence all system combinations must be performed in order to find the best. This is typically addressed by building many systems based on random trees and combining them all together (Huang et al. (2007); Ramabhadran et al. (2006)).

6. Directed Decision Trees

It is desirable to build decision trees in such a way that confusable states are explicitly separated. In this way, previous errors may be resolved, though new errors may be introduced by clustering states which were previously separate. However, if the two systems make different errors they will be complementary, and hence should lead to improved performance when combined.

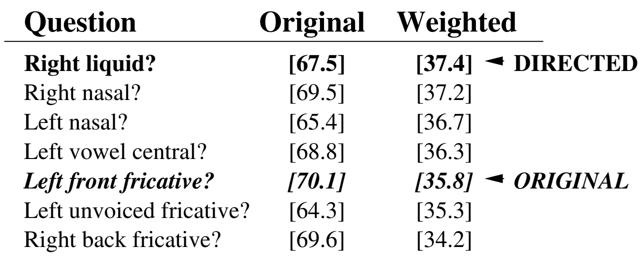
Directed decision trees (Breslin and Gales (2007a,b)) aim to separate confusable states by using a second set of statistics when selecting the best question in decision tree generation. This second set of statistics is weighted so as to reflect confusions in the training set, so states which often lead to confusions are allocated a higher weight than those which don't. These weighted statistics are used in the question selection stage of the decision tree generation, so that states with high weights are not clustered together. The original statistics are used for the stopping criterion, to ensure that the trees are of a similar size and to avoid having to tune a new stopping criterion threshold. The directed decision tree algorithm is:

(i) **Statistics**
  – Obtain original statistics for seen triphone contexts
  – Obtain *weighted* statistics for seen triphone contexts statistics

(ii) **Question Selection**
  – Recursively build the tree by selecting the question which gives the highest change in likelihood with respect to the *weighted* statistics

(iii) **Stopping criterion**
  – Stop construction of the tree when the data likelihood falls below a threshold, with respect to the *original* statistics

An example of this question selection is shown in figure 2. The question *'Right liquid?'* is chosen as this gives the largest data likelihood with respect to the weighted statistics, rather than the question *'Left front fricative?'* which is optimal 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 decision tree question selection*

This algorithm does not explicitly generate complementary systems. However, it is hoped that by giving a higher weight to states associated with confusions during the decision tree clustering, these states will be separated and the system built using the directed tree will make different errors to the baseline system.

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. Active training weights the available data for use in maximum likelihood training. For example, by emphasising certain portions of the training set, task dependent models can be built (Cincarek et al. (2006)), or else it is possible to reduce the amount of training data needed while still retaining good performance (Kamm and Meyer (2003)). Discriminative training also weights the training data to improve performance, such as in MPE training (Povey (2005)) where a higher weight is allocated to areas with high phone error. Boosting (Zhang and Rudnicky (2004); Meyer (2002)) and MBRL (Breslin and Gales (2006)) both make use of a weighting to focus on hard portions of the training data for building complementary models. These methods apply a weight at different levels, including the utterance level (Meyer (2002)), the word level (Breslin and Gales (2006)), the phone level (Povey (2005)) or the frame level (Zhang and Rudnicky (2004)).

For the purpose of building a directed decision tree, the weighting function should reflect how well the training data is modelled. One possibility is to use a confidence measure as a weighting, for example the hypothesised word posterior probability (Jiang (2005)) or likelihood-ratio (Arslan and Hansen (1999)) to identify correctly recognised portions of the training data. The drawback of this approach is that confidence measures are not always reliable. An alternative is to directly use the reference transcription to identify errors and to base the weighting function around, for example, reference word posteriors, as in (Breslin and Gales (2006)).

The final consideration is how to apply the weighting during the forward-backward pass for obtaining the statistics. Applying a weighting at the utterance level is straightforward as utterances are segmented either manually or automatically and thus are clearly defined. Applying the weight at a finer granularity, such as the word or phone level, requires some further processing. One option is to force-align the training data to obtain boundaries for words, phones or states, and hence directly weight the observations, e.g. (Zhang and Rudnicky (2004)). However, this provides a hard assignment of frames to lexical units, and may lead to errors at the boundaries. In contrast, it is possible to directly weight the states, as in (Breslin and Gales (2006)), by applying the weight directly to the state occupation statistics.

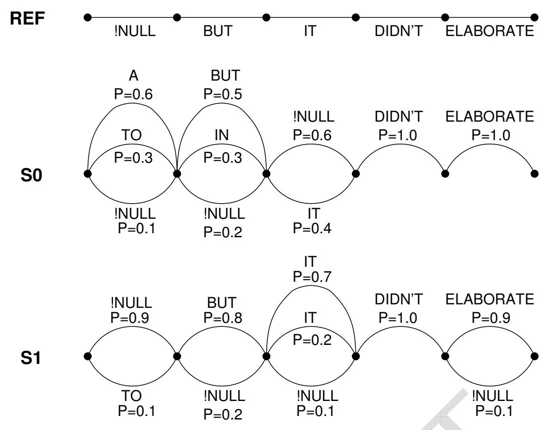


Fig. 3. *Calculating the loss function with multiple previous systems*

Reference word	Weight
BUT	0.35
IT	0.45
DIDN'T	0.00
ELABORATE	0.05

Table 1
Weights for words in figure 3 using equation 1, $\alpha = 1$

This latter approach has the advantage that it links more closely with CNC as words are weighted rather than frames. It also allows for more flexibility as the effective weight for each frame may change as training progresses and the alignment of states to frames is altered. Furthermore, weighting at the state level easily allows the application of a weight obtained with one system to multiple different systems without the need to recompute the state alignment.

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, so each reference word $\mathcal{W}_{ref}$ is assigned a weight $l(\mathcal{W}_{ref})$:

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

The second term in the weighting function, $\frac{1}{S}\sum P(\mathcal{W}_{ref}|\mathcal{O}, \mathcal{M}^{(s)})$, is the average posterior of the reference word given S previous systems $\mathcal{M}^{(1)} \cdots \mathcal{M}^{(S)}$ and the training data $\mathcal{O}$, and is taken directly from the reference word posterior probabilities of arcs in the aligned confusion networks. The weighting is then applied in the forward-backward pass by multiplying each state occupancy count, $\gamma_j(t)$, by the weight of the word it belongs to. When $\boldsymbol{o}_t$ is the observation at time t, the accumulated statistics change from $\sum_t \gamma_j(t)\boldsymbol{o}_t$ to $\sum_t l_j(t)\gamma_j(t)\boldsymbol{o}_t$. The state level loss $l_j(t)$ comes from the word level loss $l(\mathcal{W}_{ref})$, where state j forms part of word $\mathcal{W}_{ref}$. There are many options for the loss function, the form in equation 1 is just one variant.

Using this weighting function with $\alpha = 1$, the weights of the words in figure 3 are given in table 1. A higher weighting is assigned to reference words with low posterior ('BUT' and 'IT'), and vice versa for words with high posterior ('DIDN'T' and 'ELABORATE'). The effect of increasing α 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 modelled training data. If a particular model leads to 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.

Now, a boosting-like framework for generating multiple complementary systems can be used, as shown in figure 4 for building three complementary systems. A baseline system, S0, is first trained in the usual way using unweighted statistics for the decision tree generation. Then, S0 is used to obtain confusion networks for the training data, these are aligned with the reference and used to generate the weighted statistics for building a second decision tree. Next, system D1 is built from this decision tree, and so will be complementary to S0. Using confusion networks from both S0 and D1 allows a second set of weighted statistics to be obtained and a third decision tree to be generated. This decision tree is then used to build D2, which is complementary to both S0 and D1. This iterative method for building complementary systems has the advantage that the final order of combination is simply determined by the order in which systems were trained. The method of training and the system design for S0, D1 and D2 does not need to be the same, and further diversity can be obtained by varying, for example, the training approach, frontend or topology for each of the three systems.

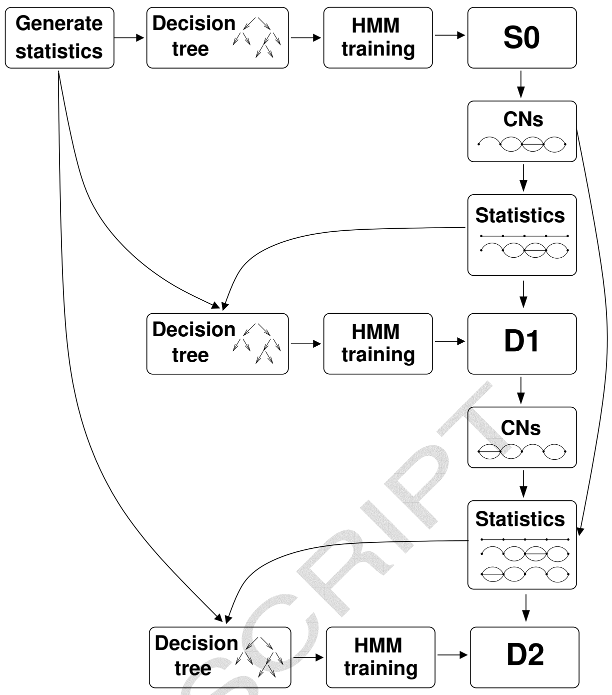


Fig. 4. *Framework for building multiple directed decision trees*

7. Decision Tree Divergence Measure

When combining systems that use different decision trees, it is important to generate trees that are different. In this work, decision trees cluster states of triphones, and so they are built as a first step in training a triphone system. It is computationally expensive then to evaluate multiple decision trees by training the multiple triphone systems and evaluating their performance. Hence, it is useful to have a method for comparing decision trees without having to fully train the corresponding system. This section describes a divergence measure for this purpose, which evaluates the similarity between two decision trees.

Figure 5 shows an example of two decision trees with different clusterings. It is possible to compare these clusterings directly using, for example, a cluster similarity measure like that in (Rand (1971)). However, these use many pairwise comparisons between clustered elements and prove expensive in practice, so it is therefore useful to have a scheme that makes use of the properties of decision trees for ASR.

The decision tree can be viewed as a mapping from unclustered to clustered states. That is, for each unclustered state θ, the decision tree mapping for the s^{th} tree, $f_s(.)$, yields the clustered state θ_s:

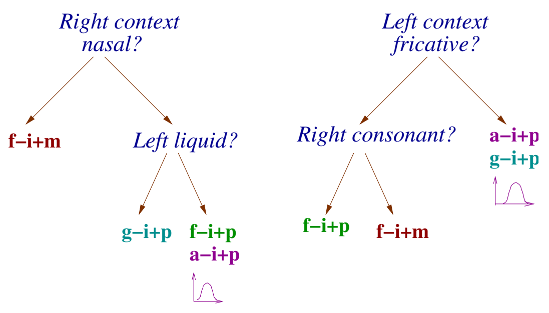


Fig. 5. *Comparing Decision Trees*

$$\theta_s = f_s(\theta) \tag{2}$$

Each clustered state θ_s has an associated Gaussian distribution, which depends on the clustering. Thus, the output distribution associated with state θ_1 will differ from θ_2 due to the different tree clustering in trees 1 and 2. For example, see the triphone *a-i+p* in figure 5.

Rather than directly measure cluster similarity, the divergence measure proposed here makes use of the fact that if two trees have similar clusterings then, for each state, the corresponding clustered state output distributions from the two trees will also be similar. Similarly, if the clustering is very different, then it is expected that the state output distributions from the two trees will differ too. The tree divergence $\mathcal{D}$ is calculated as an average, over all observed unclustered states, of divergences between Gaussian distributions from the two trees, weighted by the state occupation count γ_θ

$$\mathcal{D} = \sum_{\theta} \gamma_\theta \mathrm{KL}_{sy}(\theta_1, \theta_2) \tag{3}$$

where

$$\mathrm{KL}_{sy}(\theta_1, \theta_2) = \frac{1}{2}\left(\mathrm{KL}(\theta_1, \theta_2) + \mathrm{KL}(\theta_1, \theta_2)\right) \tag{4}$$

and θ_1 and θ_2 are the clustered states from trees 1 and 2, corresponding to the unclustered state θ. The Kullback-Leibler divergence $\mathrm{KL}(\theta, \theta)$ (Kullback and Leibler (1951)) is used as a measure of divergence between the state output distributions from the two trees, but any suitable distance metric could be employed.

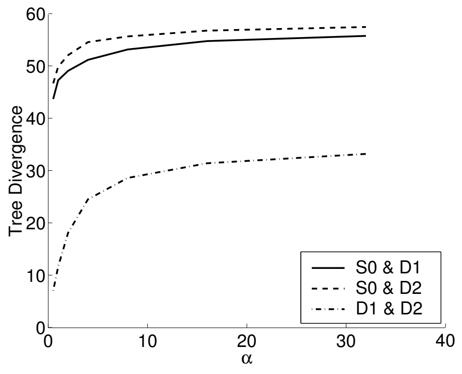
In equation 3, the sum is over all states of the unclustered triphones that were seen in the training data. The symmetric KL divergence is then between the distributions associated with that state in the two decision trees. Thus, if there are 10,000 seen triphones in the training set, each with three states, the symmetric KL divergence is calculated 30,000 times and a weighted summation performed.

This divergence measure is not expected to correlate directly with performance, nor to be an indicator of whether two trees are complementary or not. For example, it is possible to build an arbitrary decision tree with very different clustering from the optimal tree. These two trees would have a high divergence, but the arbitrary tree would not be expected to perform well either by itself, or in combination with the optimum tree. Rather, the divergence measure is useful in this work to determine whether two trees are close together. If two trees are very similar then it is unlikely that the resulting systems will differ enough for gains to be seen when combining them, and so it is not worthwhile to build the system.

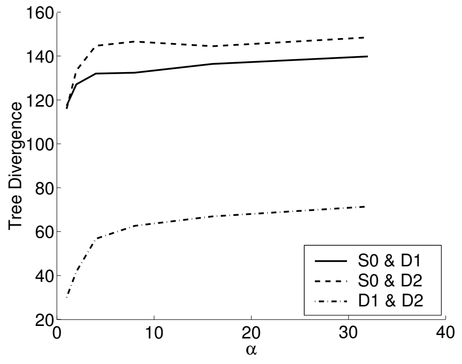
As an example, this divergence measure was evaluated on the two Broadcast News tasks described below in section 8. A baseline system S0 was first built. Next, D1 was built to be complementary to S0 using a directed decision tree, and D2 was then built to be complementary to both D1 and S0. D1 and D2 were built in exactly the same way as S0, the only difference being in the decision tree generation. The weighting for reference words in the statistics generation was previously given in equation 1. The effect of α on the decision tree generation is interesting, so to avoid the computational cost of building and decoding with many systems, the effect of α on decision tree divergence measure is examined rather than its effect on final word error rate.

The tree divergence was measured for both Mandarin and Arabic broadcast news tasks, and figure 6 shows how the tree divergence varies with α in the loss function calculation. D1 was compared to just the baseline, S0, while D2 was compared both to S0 and to D1. The divergence tends to increase with α, as more emphasis is placed on harder to recognise training data. Both D1 and D2 are a similar distance from S0, but D1 and D2 are much closer together. For the Mandarin system the absolute values of the divergence are smaller. This could be because the Mandarin system includes tonal questions, hence there are many more similar questions for the Mandarin task than there are for Arabic.

The tree divergence was measured for ten random decision trees on the Mandarin task as well as for the directed trees. The divergences between S0 and the random trees are $\{182.1, 170.5, 188.7, 168.0, 167.7\}$ for $N = 10$ and $\{102.2, 96.8, 105.5, 103.3, 109.5\}$ for $N = 5$. This is greater than the divergences be-



(a) Mandarin



(b) Arabic

Fig. 6. *Decision Tree Divergence with α when comparing S0+D1 (dotted line), S0+D2 (solid), and D1+D2 (dashed)*

tween the baseline and the directed tree in figure 6(a), which is to be expected as the random tree algorithm is far less restrained.

For the directed tree results in the following section, D1 was built with $\alpha = 1$ and D2 was built with $\alpha = 16$, to avoid D1 and D2 being too similar to each other and not giving gains when the systems are combined.

8. Experimental Results

8.1. *Broadcast News Mandarin Experimental Setup*

Broadcast news Mandarin transcription was chosen as the initial task for development. For preliminary results, a simple baseline ML model was built using a 39 dimensional feature vector including 12 PLP coefficients and energy, plus the first and second derivatives. This system used a standard decision tree for state clustering, where the state output distributions were 16 component GMMs and there were 6070 distinct states. To take account of the tonal nature of Mandarin, tonal questions were used in the decision tree.

A more complex MPE trained system was built using the same decision tree but with a 42 dimensional feature vector. 12 PLP coefficients and energy were used, plus first, second and third derivatives. An HLDA transform projected back to 39 dimensions, before pitch and its first and second derivatives were added. The number of states per component was reordered to be proportional to the state occupancy count, keeping an average of 16 components per state. MPE training was performed with a dynamic MMI prior for I-smoothing. This system is more fully described in (Sinha et al. (2006)).

The baseline acoustic models for the Mandarin task 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 (Lamel (2002)).

A trigram language model with a 50k wordlist was used for decoding in a singlepass unadapted framework. Trigram lattices were converted to confusion networks in order to perform confusion network decoding for the individual system results (Mangu et al. (1999)). This allows the gains achieved by combination to be seen easily. Results are given on the `bnm-dev06` test set, which is a combination of `dev04f` (0.5 hours of CCTV data from shows broadcast in November 2003), `eval04` (1 hour of data from CCTV, RFA and NTDTV broadcast in April 2004), `eval03m` (0.6 hours of mainland shows from February 2001) and `y1q1` (3 hours of data from October 2005). Statistical significance tests were performed using the matched pairs test (Gillick and Cox (1989)).

8.2. *Broadcast News Mandarin Results*

First, the performance of random decision trees was considered. Ten random decision trees were built: five with N=10 and five with N=5. These systems were built in the same way as the baseline, except for the decision tree. The individual system results, and their performance in combination, are given in table 2.

The performance of the random tree systems is similar to that seen previously in (Siohan et al. (2005)). The results show that the random trees individually perform worse than the baseline, and as N is increased the individual system performance gets worse. The average result for N=10 is 23.8%

CER, compared to 23.6% for when N=5, and 23.4% for the baseline. Yet when these random systems were combined with the baseline the performance improved, dropping from an error rate of 23.4% baseline to an average of 22.9% for combination with the baseline when N=5, and 23.0% when N=10. Thus, both values of N yield gains in combination, with N=5 performing slightly better than N=10.

Statistical significance tests were performed using a significance level of 95%. For the individual system results, when N=5 the random tree systems do not differ from the baseline system, while for N=10 the differences are statistically significant. For the combination of the baseline and the random tree systems, all results are significant with the exception of S0+R5 for the case when N=5.

It is also worth noting that the best individual system does not necessarily give the best results in combination, and there is some variation between best and worst performance, both for the individual systems and for their combination with the baseline.

Decision Tree		bnm-dev06 (CER %)	
		Individual	CNC with S0
BASELINE	S0	23.4	-
	R1	23.5	22.9
	R2	23.5	22.9
RANDOM:	R3	23.6	22.8
N=5	R4	23.6	22.9
	R5	23.7	23.1
	AVG	*23.6*	*22.9*
	R1	23.6	22.9
	R2	23.9	22.9
RANDOM:	R3	23.9	23.1
N=10	R4	23.9	23.0
	R5	23.8	22.8
	AVG	*23.8*	*23.0*

Table 2
Random Tree Mandarin performance for ML trained systems (CER %)

Table 3 shows a comparison of the average random and the directed tree performance. For the random systems, the average performance for the case N=5 is given, as this allows for easy comparison. Thus, R refers to the average of the random system results, S0+R refers to the average performance when combining one random system with the baseline, and S0+R+R refers to the average performance when combining two random systems with the baseline.

It can be seen that the individual directed decision tree systems perform slightly better than the baseline while the random trees perform worse. In combination with the baseline system however, the directed trees perform as well as the average random tree. For example, S0+D1+D2 gave an error rate of 22.7%, which a 0.7% absolute improvement over the baseline performance, while the average combination of S0 and two random trees gave a comparable error rate of 22.8%. Significance tests showed that the individual systems D1 and D2 did not differ from the baseline, but the combinations S0+D1 and S0+D1+D2 are statistically significant when compared with the baseline system, S0.

The majority of the gain with both the random and directed tree systems is seen when combining just two systems, with a much smaller gain achieved by the third. For the random trees, this contrasts with (Siohan et al. (2005)) where two random tree systems gave good improvements before slowing as more systems were added. This difference could be due to the different languages, or the different task used in this paper. For the directed tree systems, this pattern is similar to that seen in the divergence measure in the previous section, and suggests that further gains could be obtained from a second complementary system if the diversity of the tree could somehow be increased. Additionally, it might be expected from the divergence measurements of the previous sections that the random tree systems perform better as they diverge more from the baseline tree. However, the results show this not to be the case, reinforcing the intuition that high divergence does not necessarily correlate with an improved performance.

The directed decision tree algorithm makes use of a data weighting, and thus has some implicit discriminative effect in training. This could account for the improvements seen in the individual directed tree results over the baseline ML trained system.

It is expected that if many random trees are built, then the best of these would outperform the sub-optimal directed tree systems. Thus, it is not expected that the directed tree systems outperform the best random tree system. However, these results show that, for a small number of random trees, the directed tree performance is close to that of the best random tree, without the uncertainty associated with randomness. Furthermore, the computational cost of building multiple systems based on random trees is reduced with the directed tree ap-

	Complementary	System	CER (%) bnm-dev06
BASELINE	-	S0	23.4
DIRECTED	S0	D1	23.3
	S0+D1	D2	23.2
RANDOM	-	R	23.6
CNC		S0+D1	22.9
		S0+D1+D2	22.7
		S0+R	22.9
		S0+R+R	22.8

Table 3
Comparison of Directed and Random Tree Mandarin results for ML trained systems (CER %)

proach.

Next, the effect of directed decision trees on an MPE trained system was considered. It is computationally expensive to generate lattices for discriminative training, so MPE training was only performed for the baseline and for one directed tree system. Table 4 shows the results obtained. First, the individual results for the baseline and the directed tree are shown. In contrast to the ML system, the individual directed decision tree system no longer outperforms the baseline. This suggests that any implicit discriminative effect from the data weighting is subsumed by the explicit discriminative training. When the two systems are combined, improvements in error rate are still seen. The improvement from the baseline to the combined performance was 0.4% absolute, a drop in error from 18.4% to 18.0%. Again, statistical significance tests showed that the combination S0+D1 is statistically different from the baseline system S0 at a significance level of 95%.

System	Decision Tree	Complementary	bnmdev06
S0	BASELINE	-	18.4
D1	DIRECTED	S0	18.5
S0 + D1	CNC		18.0

Table 4
Directed tree performance for Mandarin with MPE trained systems (CER %)

8.3. *Broadcast News Arabic Experimental Setup*

For decoding in a multipass adaptive framework, the task of broadcast news transcription in Arabic was chosen. This task allows for faster development than with the Mandarin task due to the lack of tonal information, and also avoids the mismatch between a word level weighting in training and the character level error metric used to evaluate the Mandarin system performance. Additionally, Arabic has the interesting property that short vowels are not normally transcribed, so word pronunciations are generated using a set of rules (Buckwalter (2004)). This leads to a large number of pronunciations per word - on average 4.3 compared to 1.1 for English. Techniques for addressing the large number of pronunciations have yielded systems with similar performance yet which perform well in combination.

One example of such a technique is to consider how the large number of pronunciations affects the MPE training criterion, which makes use of the phonetic transcription in training (Gales et al. (2007)). The MPE training criterion with training data $\mathcal{O}$ and model parameters $\mathcal{M}$ is

$$\mathcal{F}_{MPE}(\mathcal{M}) = \sum_{\mathcal{H}} P(\mathcal{H}|\mathcal{O},\mathcal{M})\mathcal{L}(\mathcal{H},\mathcal{H}_{ref}) \qquad (5)$$

The loss function $\mathcal{L}(\mathcal{H},\mathcal{H}_{ref})$ between a hypothesis $\mathcal{H}$ and the reference $\mathcal{H}_{ref}$ can be calculated in several ways when there are many reference pronunciations. *Multiple pronunciation* MPE training involves taking the minimum phone loss between the hypothesis and all possible reference pronunciations. An alternative is to take the best single pronunciation of the reference given the current model parameters, and use this one pronunciation as the reference. This is the *single pronunciation* training used in this work. These two forms of training lead to systems which perform similarly, but give improvements upon combination, and thus can be used in addition to the directed tree approach to incorporate extra diversity.

The baseline phonetic system for the Arabic task was trained using 102 hours of data. A PLP frontend was used; 12 PLP coefficients and energy, plus first, second and third derivatives, with an HLDA transform projecting down to 39 dimensions. The number of components per state is proportional to the state occupancy, and an average of 16 components per state is maintained. MPE training was performed using an MMI prior, and gender dependent models were built. Two directed tree systems, D1 and D2, were built. The first is complementary to the baseline S0, and the second complementary to S0+D1. Three random trees were built with N=5, and were trained in the same way as the directed tree systems. The random tree results below give the average performance over the three random systems for easy comparison.

Results are given on the average of four test sets, each around 3 hours long: `bnat06` and `bnad06` are broadcast news data while `bcat06` and `bcad06` consist of broadcast conversation shows, and hence are less closely matched to the training data.

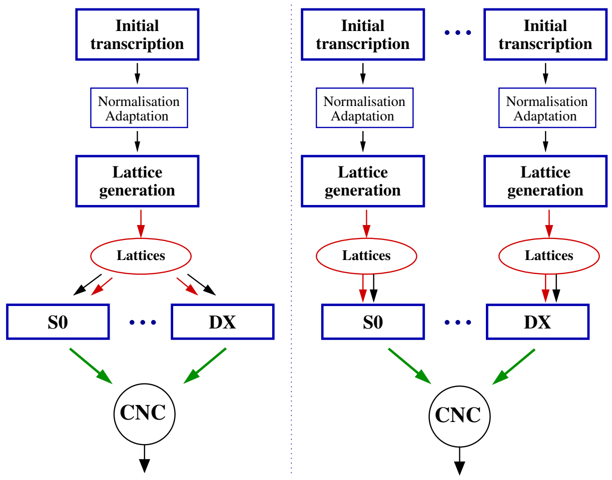


Fig. 7. *Multipass framework with (a) common lattice generation, (b) separate lattice generation passes*

Decoding was performed in two multipass frameworks both shown in figure 7. In these frameworks, an initial transcription was generated using gender independent models so that normalisation and adaptation could be performed, before lattices were generated. These lattices were then rescored using the various gender dependent models, and their outputs combined using confusion network combination. Both MLLR mean and variance adaptation were used, along with lattice-based adaptation, as in (Gales et al. (2006)). The first framework in figure 7(a) uses a common adaptation and lattice generation stage, using the S0 model, while the second in 7(b) uses separate passes for each of the systems. In decoding, the large number of pronunciations are handled using pronunciation probabilities.

8.4. *Broadcast News Arabic Results*

The initial baseline system, S0, was trained using the single pronunciation MPE criterion described above. The results obtained using a multipass decoding framework with a common lattice generation pass are shown in table 5. These results show a similar pattern to the previous unadapted singlepass results on the Mandarin task (table 4). The average baseline error rate over the four testsets is

	System	Complementary to	WER (%)
BASELINE	S0	-	36.6
RANDOM	R	-	36.7
DIRECTED	D1	S0	36.6
CNC	S0+D1		36.3
	S0+R		36.3

Table 5
Arabic results using a common adaptation and lattice generation pass with single pronunciation MPE trained systems

36.6%, the directed tree system has an average error rate of 36.6% and the random tree systems have a performance of 36.7%. The combination of the directed tree systems and the baseline leads to a improved error rate of 36.3% WER, which is statistically significant when compared to the baseline system alone. The average performance of the random tree systems in combination with the baseline is the same as for the directed tree systems. These results show that the behaviour seen previously when using an unadapted singlepass decoding strategy carry through to the multi-pass decoding framework.

M0 is a second baseline system trained using the multiple pronunciation MPE criterion described above. The only difference between M0 and S0 is the loss function used in the MPE training stage. DM1 and DM2 are both built using directed decision trees in the same way as M0, yet are complementary to S0. Hence they should retain the gains seen previously, but also include extra diversity from the difference in training. The average performance of the three random tree systems (RM), also trained using the multiple pronunciation MPE training, are given.

Table 6 shows the results obtained using these systems in the two multipass frameworks previously described in figure 7. Gains can be seen when combining the single and multiple pronunciation MPE systems. For example, the baseline performances are 36.6% and 36.1% for S0 and M0 respectively with a common lattice generation pass, and combining them gives a performance of 35.7%. In the shared lattice pass framework, M0 has an error rate of 36.3% and S0+M0 yields 35.9%. This difference between the two frameworks is the result of cross-adaptation.

The directed tree systems DM1 and DM2 perform similarly to the baseline system M0, while the random tree systems perform slightly worse. Again, the cross-adaptation effect is noticeable as the individual system results obtained using the common lattice generation pass are better than those obtained using the separate lattice generation passes. However, the performance of S0+DM1 improves over that of S0+M0. For example, when using a separate lattice generation pass, S0+DM1 gives an error rate of 35.5%, compared to 35.9% obtained from combining S0 and M0. Further small gains can be obtained from introducing a second directed tree system, DM2. This system was built to be complementary to S0+DM1. Comparing table 6 to table 5 where both D1 and D2 were single pronunciation systems, the effect of introducing the additional form of complementary training is to further improve performance. The error rate of S0+D1 is 36.3% in table 5, while in table 6 the error rate of S0+DM1 is 35.6%, an absolute gain of 0.7%, implying that the directed tree gains are additional to those obtained from the multi-pronunciation training. Again, the directed tree systems perform slightly better in combination than the random tree systems.

The results in table 6 show clearly the cross adaptation effect that can be achieved when using one system to perform adaptation and generate lattices, and another to rescore them. The effect is demonstrated by the improved individual system performances when using a shared adaptation and lattice generation pass over using independent passes. It is interesting then that these gains don't follow through when system combination is performed. It is the framework with no cross-adaptation effect that gives the best results when combining the systems based on different decision trees. Again this shows that individual system error rate is not necessarily a good indicator of whether two systems are complementary or not, and, as such, simply aiming to optimise individual system performances may not lead to optimal performance in combination. The first framework is more efficient for decoding, and it may be the case that relaxing the pruning on the first passes to give more diverse lattices for rescoring will combine the efficiency benefits of the first framework with the performance gains of the second.

A second observation concerns the relative gains obtained when combining the baseline with either one or two complementary systems. The largest gains over the baseline are obtained by combining two complementary systems and the baseline, though the contribution from adding the second complementary system is small compared to that of just the first. Although the directed decision tree algorithm is not a boosting approach, it does try and focus the statistics generation on harder parts of the training set. Also, the broadcast news task makes use of lightly supervised training so there are likely to be some transcription errors in the training data. However, unlike in (Dietterich (1999)), the observation that random trees perform better in classification noise is unlikely to apply here. Firstly, only a small number of decision trees are built, too few for the effect to be noticed, and secondly the weighting is only done at a state level so the influence of poor transcriptions is expected to be minimal. It is likely that the two directed decision trees are not diverse enough, as seen with the tree divergence measure in figure 6(b). The fact that D1 and D2 are close together in both divergence and error rate, suggest a need to somehow increase diversity between these two trees in order to see gains from a second complementary system.

While the gains seen from the directed decision trees aren't as large as those from combining S0 and M0, they consistently add to the the gains from combining the single and multiple pronunciation MPE systems, while performing slightly better than the average random tree systems.

9. Conclusions

This paper has presented an algorithm for generating complementary decision trees, along with a divergence measure for quantifying the similarity between decision trees without having to build and evaluate entire systems.

Experimental results were presented on two Broadcast News tasks - Mandarin and Arabic. The divergence measure was first evaluated on these two tasks. There appears not to be a strong correlation between divergence and word error rate, as seen by the high divergences between the random trees and the baseline on the Mandarin task, which in fact have similar performances to the directed trees. The divergence was instead used to evaluate the similarity of the trees and hence determine suitable parameters for building the directed decision trees.

Preliminary results on the Mandarin task show that systems built using directed trees perform as well, or slightly better, than the average of those built using random trees, without the individual fluctuations in performance seen when using multiple random trees. Compared to the random tree systems, the directed tree is a deterministic algorithm allowing more control over the tree generation process, and also has the advantage that the order of

	System	Complementary to	Lattice generation (a) common	(b) separate
BASELINE	S0	-	36.6	36.6
	M0	-	36.1	36.3
RANDOM	RM	-	36.3	36.6
DIRECTED	DM1	S0	36.0	36.4
	DM2	S0+DM1	36.1	36.4
CNC	S0+M0		35.7	35.9
	S0+DM1		35.6	35.5
	S0+RM		35.8	35.6
	S0+DM1+DM2		35.5	35.4
	S0+RM+RM		35.7	35.5

Table 6
Arabic results using (a) a common, and (b) a separate adaptation and lattice generation pass and both single/multiple pronunciation MPE training, averaged over all four testsets (WER %)

combination in decoding is the same as that in training.

Next, results presented on the two tasks show that the combination of complementary directed tree systems and the baseline give consistent performance gains in a range of conditions - using both ML and MPE trained models, and performing unadapted singlepass decoding or using a more complex multipass decoding scheme. Typically, the largest gains are obtained from the combination of one directed tree system with the baseline, but further small gains are seen from the addition of a second directed tree system.

Directed decision trees were then used in addition to a different form of MPE training for the Arabic task. This multiple pronunciation training criterion has previously been shown to be complementary to the single pronunciation training, and the combination of a directed tree with this form of training has shown gains additional to those obtained from the training criterion and the tree separately. Again, the largest gains are seen from the addition of one complementary system, with further small gains from a second, and the directed trees typically perform as well as or better than the average random tree.

Finally, the results show that individual system performance is not a good indicator of whether two systems are complementary. While cross-adaptation improves individual system results, the gains do not add to those from CNC. Hence, care must be taken when combining multiple systems to ensure that performance is optimal.

References

Arslan, L., Hansen, J., 1999. Selective Training for Hidden Markov Models with Applications to Speech Classification. IEEE Trans. Speech and Audio Processing 7, 46–54.

Breiman, L., 1996. Bagging Predictors. Machine Learning 24(2), 123–140.

Breiman, L., 2001. Random Forests. Machine Learning 45, 5–32.

Breslin, C., Gales, M., 2006. Generating Complementary Systems for Speech Recognition. In: Proceedings ICSLP.

Breslin, C., Gales, M., 2007a. Building Multiple Complementary Systems using Directed Decision Trees. In: Proceedings Interspeech.

Breslin, C., Gales, M., 2007b. Generating Complementary Systems using Directed Decision Trees. In: Proceedings ICASSP.

Buckwalter, T., 2004. Buckwalter Arabic morphological analyzer version 2.0. In LDC2004L02, Linguistic Data Consortium.

Cincarek, T., Toda, T., Saruwatari, H., Shikano, K., 2006. Utterance Based Selective Training for the Automatic Creation of Task Dependent Acoustic Models. IEICE Transactions on Info and Systems 89 (3), 962–969.

Dietterich, T., 1999. An Experimental Comparison of Three Methods for Constructing Ensembles of Decision Trees: Bagging, Boosting and Randomization. Machine Learning, 1–22.

Dietterich, T., 2000. Ensemble Methods in Machine

Learning. Lecture Notes in Computer Science 1857, 1–15.

Dimitrakakis, C., Bengio, S., 2004. Boosting HMMs with an Application to Speech Recognition. In: Proceedings ICASSP.

Evermann, G., Woodland, P. C., 2000. Posterior Probability Decoding, Confidence Estimation and System Combination. In: Proceedings Speech Transcription Workshop.

Fiscus, J., 1997. A Post-processing System to Yield Reduced Word Error Rates: Recogniser Output Voting Error Reduction (ROVER). In: Proceedings IEEE ASRU Workshop.

Freund, Y., Schapire, R., 1996. Experiments with a New Boosting Algorithm. Proceedings of the thirteenth International Conference on Machine Learning.

Gales, M., Diehl, F., Raut, C., Tomalin, M., Woodland, P., Yu, K., 2007. Development of a Phonetic System for Large Vocabulary Arabic Speech Recognition. In: Proceedings ASRU.

Gales, M., Kim, D., Woodland, P., Chan, H., Mrva, D., Sinha, R., Tranter, S., 2006. Progress in the CU-HTK Broadcast News Transcription System. IEEE Trans. Speech and Audio Processing 14, 1513–1525.

Gauvain, J., Lamel, L., Adda, G., 2002. The LIMSI Broadcast News Transcription System. Speech Communication, 37(1–2):89–108.

Gillick, L., Cox, S., 1989. Some statistical issues in the comparison of speech recognition algori thms.

Hain, T., Burget, L., Dines, J., Garau, G., Wan, V., Karafiat, M., Vepa, J., Lincoln, M., 2007. The AMI System for the Transcription of Speech in Meetings.

Hoffmeister, B., Klein, T., Schluter, R., Ney, H., 2006. Frame Based System Combination and a Comparison with Weighted ROVER and CNC. In: Proceedings ICSLP.

Hu, R., Zhao, Y., 2007. A Bayesian Approach for Phonetic Decision Tree State Tying in Conversational Speech Recognition. In: Proceedings ICASSP.

Huang, J., Marcheret, E., Visweswariah, K., Libal, V., Potamianos, G., 2007. Detection, Diarization and Transcription of Far-Field Lecture Speech. In: Proceedings Interspeech.

Hwang, M., Wang, W., Lei, X., Zheng, J., Cetin, O., Peng, G., 2007. Advances in Mandarin Broadcast Speech Recognition.

Jiang, H., 2005. Confidence measures for speech recognition: A survey. Speech Communication, 45:455–470.

Kamm, T., Meyer, G., 2003. Word-selective Training for Speech Recognition. In: Proceedings IEEE Workshop on Automatic Speech Recognition and Understanding.

Kullback, S., Leibler, R., 1951. On Information and Sufficiency. Annals of Mathematical Statistics, 22:79–86.

Lamel, L., 2002. Lightly Supervised and Unsupervised Acoustic Model Training. Computer, Speech and Language.

Mangu, L., Brill, E., Stolke, A., 1999. Finding Consensus Among Words: Lattice-Based Word Error Minimization. In: Proceedings Eurospeech.

Meyer, C., 2002. Utterance-level Boosting of HMM Speech Recognisers. In: Proceedings ICASSP.

Nguyen, L., Matsoukas, S., Davenport, J., Kibala, F., Schwartz, R., Makhoul, J., 2002. Progress in transcription of Broadcast News using Byblos. Computer, Speech and Language.

Nock, H., Gales, M., Young, S., 1997. A Comparative Study of Methods for Phonetic Decision-Tree Clustering. In: Proceedings Eurospeech.

Odell, J., 1995. The Use of Context in Large Vocabulary Speech Recognition. Ph.D. thesis, University of Cambridge.

Povey, D., 2005. Discriminative Training for Large Vocabulary Speech Recognition. Ph.D. thesis, University of Cambridge.

Ramabhadran, B., Siohan, O., Mangu, L., Zweig, G., Westphal, M., Schulz, H., Soneiro, A., 2006. The IBM 2006 Speech Transcription System for European Parliamentary Speeches. In: TC-STAR Workshop on Speech-to-Speech Translation. Barcelona, Spain, pp. 111–116.

Rand, W. M., 1971. Objective Criteria for the Evaluation of Clustering Methods. Journal of the American Statistical Association 66, 846–850.

Schwenk, H., 1999. Using Boosting to Improve a Hybrid HMM/Neural-Network Speech Recogniser. In: Proceedings ICASSP.

Sinha, R., Gales, M., Kim, D., Liu, X., Sim, K., Woodland, P., 2006. The CU-HTK Mandarin Broadcast News Transcription System. In: Proceedings ICASSP.

Siohan, O., Ramabhadran, B., Kingsbury, B., 2005. Constructing Ensembles of ASR Systems using Randomized Decision Trees. In: Proceedings ICASSP.

Stuker, S., Fugen, C., Burger, S., Wolfel, M., 2006. Cross-System Adaptation and Combination for Continuous Speech Recognition: The Influence of

Phoneme Set and Acoustic Front-end. In: Proceedings ICSLP.

Xue, J., Zhao, Y., 2007. Novel Lookahead Decision Tree State Tying for Acoustic Modelling. In: Proceedings ICASSP.

Zhang, R., Rudnicky, A., 2004. A Frame Level Boosting Training Scheme for Acoustic Modelling. In: Proceedings ICSLP.

Zhang, R., Rudnicky, A. I., 2003. Improving the Performance of an LVCSR System through Ensembles of Acoustic Models. In: Proceedings ICASSP.

Zweig, G., Padmanabhan, M., 2000. Boosting Gaussian Mixtures in an LVCSR System. In: Proceedings ICASSP.