Abstract

Large Vocabulary Continuous Speech Recognition (LVCSR) systems often use a multi-pass recognition framework where the final output is obtained from a combination of multiple models. Previous systems within this framework have normally built a number of independently trained models, before performing multiple experiments to determine the optimal combination. For two models to give improvements upon combination, it is clear that they must be complementary, i.e., they must make different errors. While independently trained models often do give improvements when they are combined, it is not guaranteed that they will be complementary. This paper presents a new algorithm, Minimum Bayes Risk Leveraging (MBRL), for explicitly generating systems that are complementary to each other. This algorithm is based on Minimum Bayes Risk training, but within a boosting-like iterative framework. Experimental results are reported on a Broadcast News Mandarin task. These experiments show small but consistent gains when combining complementary systems using confusion network combination.

Index Terms: Automatic Speech Recognition, Complementary System Generation, System Combination.

1. Introduction

Large Vocabulary Continuous Speech Recognition systems, such as those developed at Cambridge University for Broadcast News transcription [1, 2], typically use a multi-pass recognition framework, where a number of independently trained models are combined in the final stage. The system combination is performed using schemes such as ROVER [3] and CNC [4]. However, the models that are combined are normally not guaranteed to be complementary, and often the gains achieved from combination are small. Complementary system generation has been well documented in the context of machine learning [5] and there are many existing algorithms for generating complementary systems. Due to the increased complexity of the task, most need some modifications before they are applicable to ASR. Complementary system combination and generation for ASR has received growing attention in recent years, and is used in most state-of-the-art systems, such as those described in [2]. The most common approach for creating diverse speech recognition systems is simply to use a number of different acoustic modelling techniques to build several independent models. The models might use different frontends, segmentations, or phone sets. Independently trained systems often give gains when combined together, but there is no guarantee that this will be the case. The major drawback of this approach is that it's not possible to predict which systems have complementary errors without actually performing the combination. Hence, a number of experiments, such as those in [2, 6], must be performed to select the optimal combination. This is time-consuming, and becomes increasingly impractical as the training set size and number of alternative systems increases.

Boosting is a machine learning technique that is specifically designed for generating a series of complementary systems; AdaBoost [7] is the most popular boosting algorithm. It maintains a distribution over the training set, giving increased weight to poorly modelled training examples. Training is performed with respect to this distribution, and so that as it progresses, the distribution evolves so later classifiers focus on the 'difficult' examples. The resulting classifiers are then combined together with a weighted voting scheme, with weights predicted by the boosting algorithm. AdaBoost is only suitable for classification tasks involving a finite number of classes. For continuous speech recognition there can be an infinite number of classes, and so several approximations are needed before boosting is suitable for ASR. Boosting-like (or leveraging) algorithms for continuous speech recognition have previously been applied at the frame [8] or utterance level [9].

For training an ensemble of systems for ASR, it would be preferable to use a training algorithm that is explicitly tuned to the final combination scheme; this is the approach adopted in this paper. The combination scheme used is CNC [4], and the approach described, Minimum Bayes Risk Leveraging (MBRL), is based on modifying the Bayes loss function to reflect the errors in combination. This algorithm is described in detail in the next section, followed by preliminary results on a Broadcast News Mandarin system, before conclusions are drawn.

2. Minimum Bayes Risk Leveraging

Minimum Bayes Risk Leveraging (MBRL) is an approach to training complementary systems based on Minimum Bayes Risk training, but with a modified loss function to reflect the fact that multiple systems will be combined together. The standard expression for Minimum Bayes Risk (MBR) training [10] is

$$\mathcal{F}(\mathcal{M}) = \sum_{r=1}^{R} \sum_{m \in \mathcal{H}} P(H_m|O_r;\mathcal{M}) \mathcal{L}(H_m, \tilde{H})$$

(1)

where $\tilde{H}$ is the correct hypothesis for data $O_r$, $\mathcal{H}$ is the set of all possible hypotheses and $\mathcal{M}$ is the current model. This objective function is a generalisation of many existing discriminative criteria, such as Minimum Phone Error (MPE) and Maximum Mutual Information (MMI) [11]. In common with many discriminative criteria, there is no simple closed-form update approach to minimising this expression, so a range of approximations have been...
developed; see for example \cite{10, 11}.

MBRL uses the same general form of objective function, but also considers a number of previous classifiers during the training. There are two ways to introduce the dependency on previous models, $\mathcal{M}^{(0)}, \ldots, \mathcal{M}^{(S-1)}$, into the objective function. One option is to use the posterior probability of a word dependent on all previous systems. For the $S$th model, this yields

$$\mathcal{F}(\mathcal{M}) = \sum_{r=1}^{R} \sum_{h_m \in \mathcal{E}} P(H_m|\mathcal{O}_r; \mathcal{M}^{(0)}, \ldots, \mathcal{M}^{(S-1)}, \mathcal{M})(h_m, \mathcal{H})$$

To implement this form of combination is computationally expensive. Alternatively, the dependency can be introduced via a modified loss function, which gives

$$\mathcal{F}(\mathcal{M}) = \sum_{r=1}^{R} \sum_{h_m \in \mathcal{E}} P(H_m|\mathcal{O}_r; \mathcal{M})\mathcal{L}(H_m, \mathcal{H}|\mathcal{M}^{(0)}, \ldots, \mathcal{M}^{(S-1)})$$

The form and structure of $\mathcal{M}$ can be independent of all previous models, $\mathcal{M}^{(0)}, \ldots, \mathcal{M}^{(S-1)}$. Having determined the form of the objective function, it is necessary to evaluate precisely how the previous models, should alter the value of the modified loss function. This modified loss function reflects whether the training data is well modelled or not by the previous systems. Words which are correctly classified by earlier systems should be assigned minimal loss. This may cause later systems to badly model previously well modelled words, but shouldn’t affect the results after combination. As CNC \cite{4} is used for system combination, it should also be used to determine the effect of the previous systems on this loss. Thus, the loss function is calculated at a word level; this is in keeping with the original motivation for MBR training, which was to have training criterion which reflects word error rate as the assessment criterion. To calculate the loss function, confusion networks are generated from the training data using the existing models. These confusion networks are then combined together using CNC (the individual system word posteriors are simply averaged), and aligned with the reference transcription. This yields the posterior probability for each of the hypothesis words given all existing models, so that it is complementary to $\mathcal{M}^{(0)}$. With respect to $\mathcal{M}^{(0)}$, the word TODAY is well modelled, while the words HALLOWEEN, HELLO and EVEN are poorly modelled. By assigning a high value of loss to the incorrectly modelled words, and a low value to the correctly modelled words, $\mathcal{M}^{(S)}$ can focus on the mistakes made by $\mathcal{M}^{(0)}$.

There are two simple ways of using the word posteriors from CNC to alter the word-level loss function. First, the posterior for each word may be used directly in the loss function calculation. Thus, the modified word-level loss function for building the $S$th model, $\mathcal{L}(W_m, \mathcal{W}; S)$, becomes

$$\mathcal{L}(W_m, \mathcal{W}; S) = \begin{cases} P(W_m|\mathcal{O}_r S) & W_m \neq \mathcal{W} \\ 0 & \text{otherwise} \end{cases}$$

The problem with this approach is that the posteriors produced by ASR systems are not normally reliable. To reduce the effects of this, a simple threshold approach may be used. Thus

$$\mathcal{L}(W_m, \mathcal{W}; S) = \begin{cases} 1 & P(W_m|\mathcal{O}_r S) < \alpha, W_m \neq \mathcal{W} \\ 0 & \text{otherwise} \end{cases}$$

This threshold-based approach may be viewed as a form of training data pruning; words that are well modelled by earlier systems are not used to train latter systems. This form of thresholding has some similarities to active training \cite{12}, but with the view to building multiple systems, instead of one single best system. The threshold form of modified loss function is used in this paper.

**Initialise:**

From an initial model, $\mathcal{M}^{(0)}$ generate a set of training data Confusion Networks

**For:** \(s = 1:S\)

Combine confusion networks from $\mathcal{M}^{(0)}, \ldots, \mathcal{M}^{(S-1)}$ with the reference transcription. Train a model $\mathcal{M}^{(s)}$ to give $\mathcal{M}^{(s)}$ by minimising a cost function based on:

$$\mathcal{F}(\mathcal{M}) = \sum_{r=1}^{R} \sum_{w_m \in \mathcal{W}} P(W_m|\mathcal{O}_r; \mathcal{M})\mathcal{L}(W_m, \mathcal{W}; s)$$

Generate training data confusion networks for the new system $\mathcal{M}^{(s)}$

**Output:**

The final hypothesis is based on CNC using models $\mathcal{M}^{(0)}, \ldots, \mathcal{M}^{(S)}$

Figure 2: Minimum Bayes Risk Leveraging Algorithm

The Minimum Bayes Risk Leveraging algorithm is given in figure 2. In comparison to standard discriminative training which trains a single best system, the aim of this algorithm is to train a number of systems which may perform poorly individually, but which perform well in combination. similarities can also be drawn between this algorithm and boosting; both algorithms aim to train an ensemble of classifiers which perform well when combined. The loss function in MBRL has the same purpose as the distribution over the training data in boosting. In contrast to boosting however, the form of system combination is left open and no classifier weights are calculated as part of the algorithm. Also, there is
no need to alter the training algorithm or resample the training set to take account of the weighting on the training data; this is done implicitly by the MBRL objective function.

Previous work on weighting training data at a smaller granularity than the utterance level (e.g. [8]) has relied on force-aligning the data in order to determine which frames correspond to a particular word or phone. Force-aligning is not guaranteed to be accurate, and can lead to errors at the word or phone boundaries. This problem is avoided here by mapping word losses to states rather than to frames. Furthermore, [8] uses a confidence measure based on word posteriors to determine a weighting over the training data. Word posteriors are not always well correlated with correctness. The alternative proposed here, aligning confusion networks with the reference transcription, is fast and it provides an easy way to determine word correctness.

For the preliminary experiments presented in this paper, the models are trained using Maximum Likelihood (ML) rather than a discriminative criterion such as MPE. Though this will make the absolute baseline performance significantly worse, it simplifies the training and allows the ability of MBRL-style approaches to build complementary systems to be investigated. The use of the full MBRL algorithm described in figure 2 will be investigated in future work. The use of ML requires the effect of previous systems on the loss function to be modified. Here

\[ L(W_m, \bar{W}; S) = \begin{cases} 1 & P(W|O_S) < \alpha, W_m = \bar{W} \\ 0 & \text{otherwise} \end{cases} \]  

(6)

As the loss function is calculated at the word level, the corresponding state occupation counts are weighted, and hence the effect on the update formulae is minimal. For example, if state \( \theta \) belongs to reference word \( \bar{W} \), the modified mean update is given by

\[ \mu_{\theta}^{(S)} = \frac{\sum_{t=1}^{T} L(W_m, \bar{W}; S) \gamma_{\theta}(t) \alpha_{m}(t)}{\sum_{t=1}^{T} L(W_m, \bar{W}; S) \gamma_{\theta}(t)} \]  

(7)

where \( \gamma_{\theta}(t) \) is the occupation count for state \( \theta \) at time \( t \). The variance and prior updates are affected in a similar manner. ML-MBRL is the final algorithm used in this paper.

3. 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 140 hours of TDT4 data with only closed-caption references provided. Light supervision techniques were used on the latter portion. The feature vector consists of 13 PLP features with 1st, 2nd and 3rd derivatives appended. An HLDA transform is used to map this vector to 39 dimensions, and then pitch and its derivatives are added. Thus, the final feature vector has 42 dimensions. Results are given on two test sets: dev04f consists of 0.5 hours of CCTV data from shows broadcast in November 2003, and eval04 includes 1 hour of data from CCTV, RFA and NTDTV broadcast in April 2004. This system is fully described in [1]. In contrast to [1], these experiments use an ML trained baseline with no speaker adaptation.

Two baseline systems, \( G_0 \) and \( H_0 \), were built. \( H_0 \) was built using the standard HLDA frontend described above, while \( G_0 \) used a Gaussianised frontend [1]. Both systems have on average 16 components per state, and approximately 6000 unique states, after decision-tree based state clustering. From these starting systems, standard ML training using all of the training data was performed to give two further systems, \( G_1 \) and \( H_1 \). Also, two complementary systems, \( G_{1c} \) and \( H_{1c} \) were built using ML-MBRL. Both of these complementary systems were built to be combined with \( G_0 \) using a threshold of \( \alpha = \exp(-1) \) in the loss function (equation 6). This corresponded to 32% of the training data words. In addition, the state boundaries were fixed during training as initial experiments showed that these could drift significantly when using ML-MBRL training.

Figure 3 shows the ordered distribution of training set word posteriors obtained using \( G_0 \); the threshold of \( \exp(-1) \) corresponding to 32% of words is also marked. Approximately 20% of words have zero posterior probability with respect to \( G_0 \). Figure 4 shows how these word posteriors change after carrying out ML-MBRL training. This graph shows the distribution of word posteriors obtained using \( G_{1c} \), reordered for both portions of training data (i.e. the 32% used for training and the remaining 68%). It can be seen that the algorithm has increased the posterior probability of many previously badly recognised words, but has also had the effect of decreasing the posteriors for previously well recognised words.

Table 1 shows the confusion network decoding results for the individual baseline and complementary models. Performing further iterations of ML training has very little effect on the error rate; for example \( G_0 \) and \( G_1 \) both have an error rate of 14.3% on the dev04f set. However, performing ML-MBRL training degrades the individual system results; the error rate for \( G_{1c} \) is 14.7%, which is 0.3% worse than \( G_1 \). This effect is seen for both complementary models, on both test sets. This illustrates the fact that complementary system training schemes are not concerned with obtaining the best error rates for single systems.

The results from confusion network combination are also given in table 1. Combining two independent models does give improvements in error rate, as has been seen in previous work.
For example, G0 and H1 have individual error rates of 14.3% and 14.4% on dev04f, and their combination decreases the error rate to 13.8%. However, combining two complementary models can give greater improvements. For example, G0 and Hlc have individual error rates of 14.3% and 14.7%, but their combination gives an error rate of 14.3%. This is a gain of 0.4% absolute over the independent system combination of G0 and H1, despite the fact that the individual error rate for Hlc is 0.3% worse than for H1. As expected due to their similarity, the gains got from combining two GAUSS systems is small, while larger gains are seen from combining an HLDA and a GAUSS system.

For another degraded the error rate of the individual systems when compared to standard training. However, combining complementary systems led to improvements over combining independently trained systems. This is in contrast to previous work with CNC, which has found that optimal combination is obtained when the systems being combined have similar error rates. The results of an ideal combination scheme indicate that standard CNC is not an optimal method of combination for complementary systems, and that an alternative form of combination is needed to fully take advantage of the complementary nature of these models.

### Table 1: Individual System and CNC Results (CER %)

<table>
<thead>
<tr>
<th>Model</th>
<th>System</th>
<th>dev04f</th>
</tr>
</thead>
<tbody>
<tr>
<td>G0</td>
<td>GAUSS</td>
<td>14.3</td>
</tr>
<tr>
<td>H0</td>
<td>HLDA</td>
<td>14.4</td>
</tr>
<tr>
<td>G1</td>
<td>GAUSS</td>
<td>14.3</td>
</tr>
<tr>
<td>G1c</td>
<td></td>
<td>14.7</td>
</tr>
<tr>
<td>H1</td>
<td>HLDA</td>
<td>14.4</td>
</tr>
<tr>
<td>H1c</td>
<td></td>
<td>14.7</td>
</tr>
<tr>
<td>G0 + G1</td>
<td>CNC</td>
<td>14.1</td>
</tr>
<tr>
<td>G0 + G1c</td>
<td></td>
<td>14.0</td>
</tr>
<tr>
<td>G0 + H1</td>
<td>CNC</td>
<td>13.8</td>
</tr>
<tr>
<td>G0 + H1c</td>
<td></td>
<td>13.4</td>
</tr>
</tbody>
</table>

Table 2: IDEAL and CNC combination Results (CER %)

<table>
<thead>
<tr>
<th>Model</th>
<th>CNC</th>
<th>IDEAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>G0 + G1</td>
<td>14.1</td>
<td>14.0</td>
</tr>
<tr>
<td>G0 + G1c</td>
<td></td>
<td>14.0</td>
</tr>
<tr>
<td>G0 + H1</td>
<td>14.0</td>
<td>13.3</td>
</tr>
<tr>
<td>G0 + H1c</td>
<td></td>
<td>14.0</td>
</tr>
</tbody>
</table>

5. References


