Abstract

Building high accuracy speech recognition systems with limited language resources is a highly challenging task. Although the use of multi-language data for acoustic models yields improvements, performance is often unsatisfactory with highly limited acoustic training data. In these situations, it is possible to consider using multiple well trained acoustic models and combine the system outputs together. Unfortunately, the computational cost associated with these approaches is high as multiple decoding runs are required. To address this problem, this paper examines schemes based on log-linear score combination. This has a number of advantages over standard combination schemes. Even with limited acoustic training data, it is possible to train, for example, phone-specific combination weights, allowing detailed relationships between the available well trained models to be obtained. To ensure robust parameter estimation, this paper casts log-linear score combination into a structured support vector machine (SSVM) learning task. This yields a method to train model parameters with good generalisation properties. Here the SSVM feature space is a set of scores from well-trained individual systems. The SSVM approach is compared to lattice rescoring and confusion network combination using language packs released within the IARPA Babel program.

Index Terms: system combination, structured support vector machines, speech recognition, keyword spotting

1. Introduction

Automatic speech recognition (ASR) systems generally require training on large amounts of data to achieve high accuracy. However, sufficient data cannot normally be guaranteed for many low resource languages. Automated approaches can be used to increase the amount of training data, such as synthesising speech with the known transcriptions [1] and using untranscribed audio in semi-supervised training [2]. An alternative solution to this problem is to use data from other languages to train a multilingual deep neural network (DNN) [3, 4] to produce more robust bottleneck features. Although these data augmentation schemes yield improvements [5, 6], performance is often unsatisfactory. Thus, a combination of multiple well trained systems might be preferred.

In general system combination approaches can be divided into two distinct groups, i.e. hypothesis combination and log-likelihood score combination. Recogniser output voting error reduction (ROVER) [7] and confusion network combination (CNC) [8] are typical examples of combining the hypotheses generated by different recognisers. In these approaches, different hypotheses to be combined are generated from separate decoding runs on various systems. It is computationally expensive to run multiple passes of decoding, and the cost is even higher when speech recognition is an intermediate process, such as in keyword spotting (KWS) [9]. Thus, more efficient approaches based on log-likelihood score combination might be preferred. Here, the log-likelihoods from different systems are combined, and only a single decoding run is required.

Joint decoding [10] is a typical example of log-likelihood score combination, where the state (frame level) log-likelihoods of tandem [11] and hybrid [4] systems are linearly combined. The systems to be combined share the same hidden Markov model (HMM) topology and the combination weights are typically manually set. This makes joint decoding highly constrained. An alternative to combining frame level log-likelihoods is to combine segment level HMM log-likelihoods. This relaxes the frame level Markov assumption to the segment level, and synchronises systems at the phone (or word) level rather than frame. This relaxation also allows long-span dependency within the segments to be captured.

In [12], structured discriminative models are trained using the feature space based on phone log-likelihoods with the same context but different central phone generated by tandem and hybrid systems. Small gains were observed from using additional log-likelihoods extracted from the same models. [13] examines combination of hybrid and tandem systems with log-linear models, and applies learnt phone-specific combination weights to frame level joint decoding, achieving a small performance gain. [14] discusses model combination at sentence level, using system-specific combination weights. A more general framework was introduced by [15], where systems are combined at word level, and the word-specific combination weights are trained with the minimum Bayes risk (MBR) criterion. Another approach was investigated in [16]. However, these approaches are still limited as some words in decoding may not appear in training, especially in low resource language tasks.

The log-likelihood score combination approach is investigated in this paper. This approach is cast into a structured support vector machine (SSVM) [17, 18, 19] learning task to robustly estimate phone-specific combination weights. In this work, a more meaningful feature space is used, which is based on phone log-likelihoods from multiple systems, rather than using extra phone log-likelihoods with the same context extracted from a single system. This paper also discusses assumptions applied in typical combination approaches, such as frame level and segment level lattice rescoring, and investigate what impact these assumptions have on system combination gains. To assess the impact of different approaches on a downstream task, KWS performance is also examined. All experiments are car-
ried out on the highly challenging IARPA Babel evaluation task. The paper is organised as follows. In section 2 different commonly used combination approaches and the generation of complementary systems are discussed. The SSVM is introduced in section 3. Finally, experimental results and conclusions are presented in sections 4 and 5.

2. System Combination

In standard decoding, the Viterbi algorithm is employed to find the best hypothesis \( \hat{W} \). Given an utterance \( O \), the decoding process can be described as:

\[
\hat{W} = \arg \max_W \left\{ p(O|W)P(W)^\alpha \right\} \tag{1}
\]

where \( p(O|W) \) and \( P(W) \) are the likelihood and probability given by the acoustic (AM) and language (LM) models. \( \alpha \) is the LM scale factor. In hypothesis combination approaches, such as ROVER [7], multiple decoding runs are required which is computationally expensive. Alternatively, a single set of lattices generated by one system can be rescoring by using other systems. Then the hypothesis combination approaches can be applied to the lattices rescoring by different systems. This type of approach can significantly reduce the computational overhead as the search space (or hypotheses) is given by the lattice.

The rescoring process can be described as:

\[
\hat{W} = \arg \max_W \left\{ p(O|W)P(W)^\alpha \right\} \tag{2}
\]

where \( \hat{W} \in L \) denotes a possible hypothesis given by a determined lattice. \( p(O|W) \) and \( P(W) \) are the likelihood and probability given by AM and LM, respectively, which might differ from the ones used to generate the lattice. This approach is referred to as frame level lattice rescoring in this paper.

A far more efficient way to rescore the lattice is to fix both the search space and phone segmentations, referred to as segment level lattice rescoring. It can be expressed as:

\[
(W, \hat{\rho}) = \arg \max_{W, \rho \in L} \left\{ \prod_{i=1}^{\rho} p(O_{(i)}|w_{i}) P(W)^{\alpha} \right\} \tag{3}
\]

where \( W, \rho \in L \) denotes a possible hypothesis \( \alpha \) and the corresponding segmentation given by a lattice, and \( p(O_{(i)}|w_{i}) \) is the likelihood (corresponding to segment \( O_{(i)} \)) with triphone label \( w_{i} \) given by an AM, which might differ from the one generating the lattice. \( \rho \) is the number of segments. By using logarithms, equation (3) can be rewritten as:

\[
(W, \hat{\rho}) = \arg \max_{W, \rho \in L} \left\{ \frac{1}{\alpha} \left[ \sum_{i=1}^{\rho} \log p(O_{(i)}|w_{i}) \right] \right\} \tag{4}
\]

This is a linear combination of log-likelihoods from a single system. Before introducing the SSVM approach, the possible ways of generating complementary systems (that make different errors) will be discussed in the following subsection.

2.1. Complementary System Generation

In system combination, it is assumed that the systems to be combined complement each other. The most commonly used approach to generating complementary systems is simply to train a number of independent systems with different acoustic models. These individual systems might use different front-ends, segmentations, dictionaries or decision trees [20, 21]. Figure 1 illustrates the framework of the tandem and stacked hybrid systems used in this work. In this framework, complementary systems might be generated, for example, by using different (unilingual or multilingual) data to train the bottleneck DNN, employing different input feature types, using different supervisions to train the transforms, semi-supervised AM training, or using different structures or activation functions for the bottleneck or hybrid DNNs. All these approaches lead to potentially complementary systems, but there is no theoretical guarantee, hence a number of experiments must be performed to select the optimal combination [20]. However, standard approaches to assess how complementary individual systems are would involve multiple decoding runs which are time consuming. Alternatively, complementarity can be efficiently evaluated by the combination approaches based on lattice rescoring with different AMs. This is examined in the experimental section.

3. Structured Support Vector Machines

In its basic form, a support vector machine (SVM) is a linear classifier [22]. To classify observation sequences \( O \) into one of many possible sentences \( W \) with a SVM, the simplest option is to map them jointly into a fixed dimensional representation \( \Phi(O, W) \). Unfortunately extracting fixed dimensional features from variable-length observation sequences and modelling the vast, unstructured, mostly unseen space of possible sentences is non-trivial [23]. Instead, a structured assumption is usually imposed on the label space making sentences to be variable-length sequences of finite vocabulary units such as words or phones. The variable-length issue is then delegated to those units by aligning them with observations to yield alignment \( \rho \) dependent feature vectors \( \Phi(O, W, \rho) \). This serves the basis of structured discriminative models including SSVMs. Classification is performed by solving a semi-Markov inference problem [24]:

\[
(W, \hat{\rho}) = \arg \max_{W, \rho} \left\{ \alpha \Phi(O, W, \rho) \right\} \tag{5}
\]

Efficient inference can be performed if the search space in equation (5) is constrained to fewer hypotheses encoded compactly in a lattice [25, 26]. Such lattices can be efficiently generated using standard HMM-based approaches [27]. It is simple to notice that the inference problem in equation (5) or its lattice based approximation includes equation (4) as a subproblem. This relationship will be explored in the rest of this section.

The key concept of SVMs is a margin defined as the distance between the closest correct and incorrect class example [22]. For SSVMs this can be defined by:

\[
M(W_n, \rho_n, W, \rho; \alpha, O_n) = \alpha \Phi(O_n, W_n, \rho_n) - \alpha \Phi(O_n, W, \rho) \tag{6}
\]
where \((W_n, p_n)\) and \((W, p)\) are correct and incorrect alignments for observation sequence \(O_n\). For a sequence classification task it is important to take into account loss, \(\mathcal{L}(W_n, W)\), as different alignments make different number of transcription errors. Training then consists of minimising the largest violation of the loss-augmented margin [17]:

\[
F_{lm}(\boldsymbol{\alpha}) = -\log p(\boldsymbol{\alpha}) + \sum_{n=1}^{N} \left[ \max_{W \neq W_n} \left\{ \mathcal{L}(W_n, W) - \mathcal{M}(W_n, p_n, W; \boldsymbol{\alpha}, O_n) \right\} \right]
\]

(7)

where \(N\) is the number of training examples, and \([\cdot]_+\) is the hinge loss. \(W \neq W_n\) denotes any possible hypothesis that differs from the reference \(W_n\). \(p(\boldsymbol{\alpha}) = \mathcal{N}(\mu_0, C\boldsymbol{I}) \propto \exp\{-\frac{1}{2}||\boldsymbol{\alpha} - \mu_0||^2\}\) is the Gaussian prior with mean \(\mu_0\) and diagonal covariance \(C\boldsymbol{I}\). The objective function in (7) can be efficiently solved by using the cutting plane algorithm [28]. In this work, the segmentations corresponding to the references are the most likely segmentations given by the HMMs. An alternative approach to obtain optimal segmentations by using discriminative models is discussed in [17].

There are a number of possible forms for alignment dependent feature vectors \(\Phi(O, W, \rho)\). Typically they consist of observation and transition features [26]. Observation features are extracted from variable-length observation sequences associated with units whereas transition features are extracted on transitions from one unit to another. One simple form is [29]:

\[
\Phi(O, W, \rho) = \left[ \sum_{i=1}^{n} \delta(w_i, w_1) \phi(O_i, w_i) \right] \vdots \left[ \sum_{i=1}^{n} \delta(w_i, v_j) \phi(O_i, w_i) \right] \log P(W)
\]

(8)

where \(\{w\}_{i=1}^{n}\) denotes all possible phone units in the dictionary. \(\delta(\cdot)\) is the Kronecker delta, \(\phi(O_i, w_i)\) is the observation feature vector for segment \(O_i\). In general, this feature vector can be produced by using any approach capable of mapping variable length observation sequences to fixed length. It can be a first and higher order observation statistics traditionally used with frame level models [30, 31]. Other examples include score spaces [32, 33, 34] and event detectors [26, 16]. A simplest example is given in (4), where the mapping for a segment \(O_i\) with label \(w_i\) is a log-likelihood, namely \(\phi(O_i, w_i) = \log p(O_i|w_i)\). This work employs a more general form, which consists of log-likelihoods from multiple AMs, rather than a single one. Let \(K\) be the number of AMs, the observation feature vector \(\phi(O_i, w_i)\) and the corresponding phone-specific weights can be described as:

\[
\phi(O_i, w_i) = \left[ \log p_1(O_i, w_i) \right] \vdots \left[ \log pk(O_i, w_i) \right], \alpha_{w_i} = \left[ \alpha_{w_i}^{1} \right] \vdots \left[ \alpha_{w_i}^{K} \right]
\]

(9)

By setting the weights corresponding to an individual system to 1 and others to 0, the SSVM will retrieve the performance of segment level lattice rescoring described in (4). Moreover, by using these manually set weights in the prior, optimal weights can be learnt by using the large margin training criterion (7).

This approach is adopted in the experiments, where the SSVMs are trained based on these priors with strong baselines.

### 4. Experiments

Different system combination approaches discussed in the previous sections are examined in this section, with ASR and KWS performance presented. Experiments are performed on the Swahili and Javanese full language packs \(^4\) released within the IARPA Babel program, which contain around 40 hours of transcribed conversational telephone speech data for training.

**Table 1: Resource requirements of different approaches.**

<table>
<thead>
<tr>
<th>Approach</th>
<th>Decoding #</th>
<th>KWS #</th>
</tr>
</thead>
<tbody>
<tr>
<td>Standard</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>Rescoring (frame)</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>Rescoring (segment)</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>SSVM</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

4Swahili IARPA-babel202b-v1.0d. Javanese IARPA-babel402b-v1.0a.

\(^2\)These DNNs are generated by different sites (CUED and RWTH).

**Table 2 gives the word error rates (WER) on the Swahili dev set.**

<table>
<thead>
<tr>
<th>System</th>
<th>WER (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>S1</td>
<td>39.4</td>
</tr>
<tr>
<td>S2</td>
<td>36.2</td>
</tr>
<tr>
<td>S3</td>
<td>36.2</td>
</tr>
<tr>
<td>S4</td>
<td>34.5</td>
</tr>
</tbody>
</table>

S1, S2, S3 and S4 are joint decoding systems, which combine the tandem and hybrid log-likelihoods at frame level. S1, S2 and S3 differ in the structure of the bottleneck DNN: they use 26, 39 and 62 dimensional bottlenecks respectively. S4 uses 39-d bottleneck and semi-supervised training. Lattices generated by the S1 system are used for rescoring experiments. For example, the SSVM combination weights are trained based on lattices generated with a bigram language model (LM). In decoding, bigram lattices are rescored with a trigram LM and the scores from the systems to be combined. In the first block (from row 1 to 4) of Table 2, all the numbers are confusion network (CN) results. The second block lists the CNC results.

Since S1, S2, S3 and S4 use different model structures and are trained with different training technologies, they may complement each other. Moreover, these 4 joint decoding systems have comparable performance (see “Standard” column), this makes a good combined performance possible, and experiments show that the CNC result (43.5%) is much better than the result of any individual system. For the system combination results,
Table 2: WER performance on Swahili (202) Eval15 dev set.

<table>
<thead>
<tr>
<th>System</th>
<th>Standard</th>
<th>Rescoring</th>
<th>SSVM</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Frame</td>
<td>Segment</td>
</tr>
<tr>
<td>S1</td>
<td>44.7</td>
<td>44.8</td>
<td>45.0</td>
</tr>
<tr>
<td>S2</td>
<td>45.6</td>
<td>45.3</td>
<td>45.7</td>
</tr>
<tr>
<td>S3</td>
<td>44.6</td>
<td>44.6</td>
<td>45.1</td>
</tr>
<tr>
<td>S4</td>
<td>44.7</td>
<td>44.7</td>
<td>45.5</td>
</tr>
<tr>
<td>CNC</td>
<td>43.5</td>
<td>43.5</td>
<td>43.7</td>
</tr>
</tbody>
</table>

Time (hrs): 19 7 10 9 9

Table 3: MTWV performance on Swahili (202) Eval15 dev set.

<table>
<thead>
<tr>
<th>System</th>
<th>Standard</th>
<th>Rescoring</th>
<th>SSVM</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Frame</td>
<td>Segment</td>
</tr>
<tr>
<td>S1</td>
<td>0.543</td>
<td>0.543</td>
<td>0.541</td>
</tr>
<tr>
<td>S2</td>
<td>0.537</td>
<td>0.535</td>
<td>0.539</td>
</tr>
<tr>
<td>S3</td>
<td>0.540</td>
<td>0.543</td>
<td>0.545</td>
</tr>
<tr>
<td>S4</td>
<td>0.544</td>
<td>0.543</td>
<td>0.535</td>
</tr>
<tr>
<td>PM</td>
<td>0.570</td>
<td>0.566</td>
<td>0.561</td>
</tr>
</tbody>
</table>

Time (hrs): 25 14 15 3 3

Table 4: WER performance on Javanese (402) dev set.

<table>
<thead>
<tr>
<th>System</th>
<th>Standard</th>
<th>Rescoring</th>
<th>SSVM</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Frame</td>
<td>Segment</td>
</tr>
<tr>
<td>J1</td>
<td>5.0</td>
<td>5.0</td>
<td>5.0</td>
</tr>
<tr>
<td>J2</td>
<td>5.1</td>
<td>5.1</td>
<td>5.1</td>
</tr>
<tr>
<td>J3</td>
<td>5.1</td>
<td>5.1</td>
<td>5.1</td>
</tr>
<tr>
<td>J4</td>
<td>5.1</td>
<td>5.1</td>
<td>5.1</td>
</tr>
<tr>
<td>CNC</td>
<td>52.3</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Time (hrs): 49 13 13

Table 5: MTWV performance on Javanese (402) dev set.

<table>
<thead>
<tr>
<th>System</th>
<th>Standard</th>
<th>Rescoring</th>
<th>SSVM</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Frame</td>
<td>Segment</td>
</tr>
<tr>
<td>J1</td>
<td>0.451</td>
<td>0.451</td>
<td>0.451</td>
</tr>
<tr>
<td>J2</td>
<td>0.446</td>
<td>0.446</td>
<td>0.446</td>
</tr>
<tr>
<td>J3</td>
<td>0.420</td>
<td>0.420</td>
<td>0.420</td>
</tr>
<tr>
<td>J4</td>
<td>0.362</td>
<td>0.362</td>
<td>0.362</td>
</tr>
<tr>
<td>PM</td>
<td>0.463</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Time (hrs): 40 8 8

4.2. Experiments on Javanese

It is time consuming to generate multiple complementary systems using different model structures or modelling techniques. A simple way to create possible complementary systems is to use different bottleneck features, and this approach is examined on the Babel Javanese data. The systems J1, J2, J3 and J4 presented in Table 4 only differ in the bottleneck DNNs, which are trained on different data sets: J4 – unilingual; J3 – 11 languages; J2 – 24 languages; J1 – 24 languages with fine-tuning to Javanese. These individual systems can be relatively easily generated, but the results in Table 4 show that these systems are not good examples of complementary systems. The CNC result (52.4%) is not significantly better than the J1 result (53.0%). These systems have high deletion rate. The SSVM can only match the CNC result. This indicates that it is hard for SSVMs to take advantage of extra features from other systems, when individual systems are not complementary. However, the SSVM with trained weights is the most efficient and effective approach.

For the KWS results shown in Table 5, the SSVM with manually set weights, has a comparable result (0.461) to the PM (0.463), but only 1 KWS run is required by the SSVM, and only needs 1/5 of the time used by the standard approach.

5. Conclusions

This paper has examined the impact of different combination approaches on both the ASR and KWS performance. The proposed SSVM combination approach, that only needs 1 decoding and 1 KWS run, can achieve good ASR and KWS performance very efficiently. However, training the SSVM weights gives worse KWS performance which needs to be investigated further.

---

7Cantonesian IARPA-babel101b-v0.4c, Assamese IARPA-babel102b-v0.5a, Bengali IARPA-babel103b-v0.4b, Pashto IARPA-babel104b-v0.4a, Turkish IARPA-babel105b-v0.4, Tagalog IARPA-babel106-v0.2f, Vietnamese IARPA-babel107b-v0.7, Haitian Creole IARPA-babel201b-v0.2b, Lao IARPA-babel202b-v3.1a, Tamil IARPA-babel204b-v1.1b, Zhu IARPA-babel206b-v0.1d.

8Plus Kurmanji Kurdish IARPA-babel205b-v1.0a, Tok Pisin IARPA-babel207b-v1.0b, Cebuano IARPA-babel301b-v2.0b, Kazakh IARPA-babel302b-v1.0a, Telugu IARPA-babel303b-v1.0a, Lithuanian IARPA-babel304b-v1.0b, Swahili IARPA-babel202b-v1.0d, Guarani IARPA-babel305b-v1.0a, Igbo IARPA-babel306b-v2.0c, Amharic IARPA-babel307b-v1.0b, Mongolian IARPA-babel401b-v2.0b, Javanese IARPA-babel402b-v1.0b, Dholuo IARPA-babel403b-v1.0b.

9Since the individual systems do not have comparable performance, the system-dependent weights are set to (0.6α, 0.3α, 0, 0).
6. References


