DEVELOPMENT OF THE 1998 OGI-FONIX
BROADCAST NEWS TRANSCRIPTION SYSTEM

Xintian Wu and Yonghong Yan
Oregon Graduate Institute of Science and Technology, 20000 N.W. Walker Road, P.O. Box 91000, Portland, OR 97291-1000, USA, (xintian@cse.ogi.edu)

ABSTRACT
In speech recognition systems, it is generally required that the training environment be identical to the decoding environment. Any mismatch between them may result in performance degradation. This paper tries to improve the performance of a speech recognition system by compensating for the training and decoding mismatches. The baseline system [1][2] is a multiple pass decoding system capable of transcribing broadcast news, which achieved 30.5% word error rate on the 1997 DARPA HUB4E test set. Three approaches were investigated: (1) Delete long silence in both training and decoding utterances; (2) Enlarge the second-pass decoding dictionary; (3) Merge utterance fragments into a complete sentence. These approaches resulted in 2.8%, 0.3%, and 23% absolute error reductions on the 1997 test set, respectively. The combined approach achieved more than 4% absolute error reduction. On the official 1998 DARPA HUB4E evaluation, the resulting system achieved 27.9% word error rate for the 97 part evaluation data and 23.6% word error rate for the 98 part evaluation data.

1. INTRODUCTION
In speech recognition systems, the parameters of acoustic models and language models are estimated from a training set (corpus) and used in a decoding environment. It is important to make the training environment as similar as possible to the test environment. Any mismatches between them may cause performance degradation, especially when a decoding task is large and complex, such as the DARPA Broadcast News tasks. It often makes significant performance difference between a matched and a mismatched training and decoding environments.

In 1998, with support from FONIX, OGI developed a large vocabulary continuous speech recognition system to participate in the 1998 DARPA Broadcast News evaluation. The baseline system achieved 30% word error rate on the hub4e97 test set. In this paper, we present the experiments we did during the development period to improve the baseline performance. The focus is to compensate for the mismatches between the training and decoding environments. We examined each step in system building stages and present three approaches concerning with aspects of feature generation, acoustic model training and language model application. We also present a method of generating MLLR regression class trees using the decision-tree clustering technique. Overall, more than 4% absolute word error rate was achieved by applying these compensations.

In the following sections, we briefly introduce the baseline system in Section 2, and then present the comparison experiments of each approach in Section 3.

2. THE BASELINE SYSTEM
The OGI-FONIX large vocabulary speech recognition system is a continuous HMM-based system. It is developed for transcribing Broadcast News tasks challenged by the DARPA evaluation. The decoding environment contains complicated conditions such as background music, commercials, and telephone conversations.

To make the system run on the three-hour test data, recognition was performed in multiple stages: (1) The Bayesian Information Criterion segmentation [3] and mono-phone recognition were used to segment and cluster the test data; (2) A first decoding pass generated a word graph for each segment of speech; (3) The second decoding pass generated the final transcriptions.

The system has 56k active words in vocabulary. The language models were trained with WSJ and BNL data released in 1997 and 1998. The trigram language model has 7M bigrams and 14M trigrams.

The system uses the traditional 39 parameters MFCC feature (12 mel parameters plus energy coefficients and their first and second derivatives).

The standard Hub4 acoustic training data for 1996 and 1997 released by LDC were used. There are two sets of acoustic models: a within-word triphone model (6800 states with 36 Gaussians per state) for the first decoding pass, and a crossword triphone model (7500 distinct states with 21 Gaussians per state) for the second decoding pass.

During the multiple-pass decoding, unsupervised MLLR was used to adapt the model parameters to each speaker.

3. APPROACHES AND EXPERIMENTS
During the system development period, many experiments were done to identify and compensate for the mis-
matches between the training and decoding environments so that system performance may be improved. To speed up the experiment turn-around time, a 1000 seconds development subset was carefully chosen from the 1996 PE test set. The subset has the same condition (F0-FX) in terms of distribution, the same language model complexity, and the same OOV (Out Of Vocabulary) rate as the Hubele97 test set. Most of the training and decoding experiments were based on this subset. The Hubele97 evaluation set was also used as a development set.

Silence Deletion and Cluster Based CMS

In feature generation, Cepstrum Mean Subtraction (CMS) is often used to normalize the speaker variations. The following two issues may affect the performance of CMS:

(1) On very short segments, the cepstrum mean value does not represent the speaker or channel attributions well due to insufficient data. Therefore, the traditional way of performing CMS on each individual segment may result in an inaccurate CMS mean value for the corresponding segment. We use the cluster-based CMS approach, where the cepstrum mean value is calculated from the speaker cluster, all the segments from the same speaker. The speaker cluster ensures that the CMS mean value is calculated from enough speech observations.

(2) CMS is a mixed effect of speaker and channel normalization. We observed that the balance of speaker to channel information was different from segment to segment. To make CMS represent more of the speaker variations, it is desirable to minimize the influence of channel information. Long silence in a segment is the easiest channel information that can be identified in a reasonable accuracy. The silence deletion technique was introduced by Yan in his work on language identification [4]. We find it quite effective in improving recognition performance. The idea is to delete most long silence (only leave several frames) in segments from both training and decoding. The following steps shows how to delete silence for a segment:

(a) Run a mono-phone decoder on the segment and detect silence.

(b) If silence is at the beginning of a segment, the segment start-frame is adjusted to preserve at least 7 frame silence at the beginning.

(c) If silence is at the end of a segment, the segment end-frame is adjusted to preserve at least 7 frame silence at the end.

(d) If silence is in the middle of a segment, the segment is shortened to preserve at least 8 frames silence in the middle.

(e) re-calculate CMS for the segment.

In the above procedure, the frame numbers in step (b), (c) and (d) are empirical numbers we found from experiments.

The experiments were performed on the 1000 seconds development set. Word error rate results are shown in Table 1. Individually, the silence deletion yielded 2.8% improvement and the cluster based CMS yielded 0.3% improvement. The mixed effect of silence deletion and cluster based CMS yielded 3% performance improvement on the subset.

<table>
<thead>
<tr>
<th>1000s subset</th>
<th>WER</th>
</tr>
</thead>
<tbody>
<tr>
<td>baseline</td>
<td>38%</td>
</tr>
<tr>
<td>Silence Deletion</td>
<td>35.2%</td>
</tr>
<tr>
<td>Clustered CMS</td>
<td>37.7%</td>
</tr>
<tr>
<td>mix both</td>
<td>35%</td>
</tr>
</tbody>
</table>

Table 1. Silence Deletion and Cluster Based CMS

Using Two Dictionaries

In acoustic modeling, the training process usually uses a very big dictionary. The best pronunciations are selected by the force-alignment program in iterative processes. The decoding dictionary is much smaller due to the computational complexity and the accuracy of language models. To help the acoustic calculation in decoding, two dictionaries were used in the system. The two dictionaries have the same vocabulary size but different numbers of alternative pronunciations. Additional reasons of using two dictionaries include:

(1) For the within-word decoding pass, the computational complexity limits the use of a very big dictionary.

(2) For the crossword decoding pass, each word graph has a limited vocabulary size. The computational complexity caused by enlarging the dictionary is manageable. The additional alternative pronunciations helps the calculation of acoustic probability.

The procedure to change dictionaries is illustrated in Figure 1:

for each word graph,

for each word in the word graph:

(1) Remove the alternative pronunciation marker.

(2) Find all the alternative pronunciations in the big dictionary and insert them by duplicating the corresponding arcs.

Experiments were done on the HUBele97 evaluation set. With a 56K vocabulary size, the dictionary used for the within-word decoding pass has 63K pronunciation entries, and the dictionary for the cross-word decoding pass has 66K pronunciation entries. In these early experiments, we explore the case of having 3K pronunciation difference mainly from common words in the vocabulary. The experimental results are shown in Table 2. The use of two dictionaries yields 29.1% WER, which is a 0.2% absolute WER improvement over the baseline of 29.3% WER.
Chop and Merge Segment

In the segmentation stage, long segments are chopped first at the sentence boundary (silence detected by the mono-phone decoder) and then at the word boundary (detected by the fast word recognizer). The resulting segments are short enough (about 4 to 8 seconds) to ensure that they can be handled by the decoder. These segments can be any part of a sentence. We observed two impacts on using short segments:

(1) The language model is trained on sentence base. However, during decoding, we always assume that each segment is a sentence: beginning with a sentence-begin token and ending with a sentence-end token. These sentence boundary tokens cause the training and decoding mismatch. CMU [5] first brought up this question and tried to address the mismatch by "introducing two words of context and retrain the language model." The result is unsatisfied. "The standard technique of modeling the begin-of-sentence token and assuming the end-of-sentence token provided the lowest word error rate."

(2) For short segments with only few words, it is difficult to apply high-order language models. Therefore, more high-order language models can be applied to a complete sentence than to several short segments chopped from the same sentence. In the later case, the power of high-order language models is restricted.

To address these problems, we perform the usual decoding (on short segments) in the within-word decoding pass. The language model remains unchanged. In the crossword decoding pass, the decoding segments are re-arranged so that segments chopped from the same long utterance are merged back to form a complete sentence. Thus the boundary issue caused by segmentation is alleviated.

<table>
<thead>
<tr>
<th>hub4e97 WER</th>
<th>1 dict only</th>
<th>using two dicts</th>
</tr>
</thead>
<tbody>
<tr>
<td>Long segments (&lt;30s)</td>
<td>30.0%</td>
<td>-</td>
</tr>
<tr>
<td>chop segment (&lt;10s)</td>
<td>29.3%</td>
<td>29.1%</td>
</tr>
<tr>
<td>chop and merge segment</td>
<td>28.0%</td>
<td>27.7%</td>
</tr>
</tbody>
</table>

Table 3. Using two dictionaries in recognition

The procedure to merge two segments (word graphs) is illustrated in figure 2. The words at the end of the first word graph and the words at the beginning of the second word graph are fully connected. Time sequences should be kept for the merging segments to ensure a correct merge.

Experiments were done on the HUB4e97 test set. The word error rate results were shown in table 3. In the first column of the table, chopping the long segments into short ones yielded 0.7% improvement (due to relaxing the beam). Merging the small segments back yielded another 1.3% improvement. In the second column of the table, the jointed effect with using two dictionaries was studied. Totally 2.3% improvement was observed.

**Figure 1. Procedure to Change Dictionary**

w1, w2, and w3 are recognized words in the small dictionary. w1', w2', and w3' are corresponding words in the big dictionary. "(0)" and "(1)" are alternative pronunciation markers. In the figure it is assumed that word w2 has totally two alternative pronunciations in the big dictionary, noted as w2'(0) and w2'(1).

**Figure 2. Procedure to merge two segments**

Segments from the same sentence are merged back to form a complete sentence. The resulting word graph is recognized by a crossword decoding pass.

**Table 2. Using two dictionaries in recognition**

<table>
<thead>
<tr>
<th>hub4e97 test set</th>
<th>word error rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Using two dictionaries</td>
<td>29.1%</td>
</tr>
</tbody>
</table>
MLLR Regression Tree

We present a way of using the decision tree method to build the MLLR regression class tree. The purpose of this method is to incorporate phonetic knowledge into building the regression tree clustering. Since acoustic models are clustered using the decision-tree method during the training, the resulting regression class tree may be more identical to the training configuration than those traditional regression tree using only data-driven methods.

The following issues are addressed to apply the decision tree method:

1. Assume Gaussian mixtures within a HMM state is similar enough so that they are by default clustered together (adapted by the same transform matrix).

2. The phonetic questions should be extended to consider not only left and right context of a triphone but also the central mono-phone of triphones.

3. The same as in decision-tree clustering, we perform the regression-class tree clustering in the single Gaussian case, the mean and variance parameters used for a HMM model may have two choices: (a) Use the parameters corresponding to the central state to represent the HMM model. (b) Include questions of the state number in the decision-tree question set. In practice, we find little difference between these two choices. Either of them works fine for the experiments we designed.

![Decision-tree based regression-class tree](image)

Figure 3. Decision-tree based regression-class tree

"Vowel", "unfortenesis", "nasal" and "low" are different phonetic questions. The prefix "C." indicates that the question is asked on the central mono-phone. "State_02" and "State_12" are questions on HMM states.

Experiments were done on the huble_07 test set. The regression-class tree was built to have 8 leaves and totally 15 regression classes. Figure 3 shows part of the MLLR regression-class tree. An occupation threshold was set to control the tailoring of the regression-class tree during adaptation. Table 4 lists the results. The resulting regression-class tree achieved a 0.9WER improvement over the global adaptation case.

<table>
<thead>
<tr>
<th></th>
<th>word error rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Huble_07, test set</td>
<td>21.7%</td>
</tr>
<tr>
<td>Global Adaptation</td>
<td>26.7%</td>
</tr>
<tr>
<td>Regression Tree</td>
<td>25.8%</td>
</tr>
</tbody>
</table>

Table 4. Using decision tree based regression tree clustering

4. CONCLUSION

Keeping the training environment as similar as possible to the decoding environment is important in speech recognition systems. By identifying the mismatches and compensating for these mismatches, we improve the system performance. The approaches we present in the paper are still in their early research stage. More experiments are being done to further test the ideas.

REFERENCES


