LATTICE COMBINATION FOR IMPROVED SPEECH RECOGNITON

Xiang Li, Rita Singh, Richard M. Stern

Department of Electrical and Computer Engineering and School of Computer Science
Carnegie Mellon University
Pittsburgh, Pennsylvania, 15213, USA
{xiangl, rsingh, rms} @cs.cmu.edu

ABSTRACT

Combining recognition outputs from different features or different systems will generally improve the recognition accuracy compared to that obtained with any single feature/system alone. Several attempts have been made to combine different systems together, but they all restricted to the use of only single best hypothesis from different feature/systems during combination. We present a new multiple hypothesis based combination method named lattice combination. The idea of lattice combination is to construct a mixed lattice by combining and modifying lattices from individual feature sets or systems together, and output the best path from the combined lattice as the final hypothesis. Experiments in five different database indicate the consistent improvements in recognition accuracy of lattice combination over conventional methods.

1. INTRODUCTION

The performance of an automatic speech recognition (ASR) system depends critically on the feature set that it uses. Even though there currently exist several different kinds of features (e.g. MFCC, PLP) which may generate good results under certain conditions, none of them works perfectly for all conditions. Instead, they represent different subsets of information embedded in the speech signal. It stands to reason that combining the information from these features properly would result in better recognition accuracy than the result obtained with any single feature set alone.

Generally speaking, there are two ways of combining information from different features together: we can either concatenate different feature vectors to form a bigger feature and perform recognition based on this combined feature, or perform recognition directly based on single features and combine their outputs together. Compared with the former method, we are currently more interested in the later category because of its advantages like parallel processing in training and recognition, flexibility in adding new features and ability of combining different systems without additional effect.

To our knowledge, there currently exist two methods of combining speech recognition outputs together, *ROVER* [1] and *Hypothesis Combination* [2]. Even though they both effectively improve the recognition performance, they have the same constraints that they perform the combination only using the single best hypothesis from the recognizer. We know that in many situations, words that present in the utterance do not appear in the single-best hypothesis. As a result, combination scheme that is based on single best hypothesis would miss such words. On the other hand, researches do show that appropriate processing of multiple recognition outputs (e.g. lattice) can give us better performance than just picking up the single-best hypothesis[3].

Motivated by above facts, we propose a new multiple outputs based combination scheme named lattice combination. To combine different ASR systems together, it first mix lattices from these systems together, then modify this mixed lattice, and finally search the best path in this mixed lattice as the combination output. The reason of choosing lattice is that compared with another multiple outputs format as N-best lists, lattice preserves more information of the hypothesis space, hence would benefit us more during the combination, if we can process them properly.

In the following sections, we first describe the details of lattice combination in Section 2, then we proceeds to the Section 3 with experimental results and analyses, followed by the discussion and future work in Section 4, and summary in Section 5

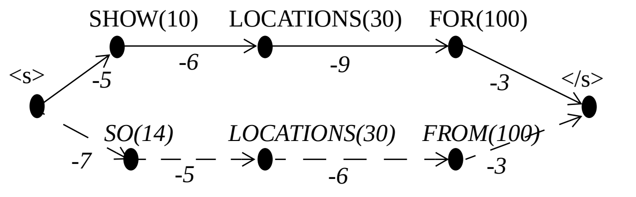
2. LATTICE COMBINATION

As convenient output format for speech recognizer, lattice is a directed acyclic graph, in which nodes are associated with words and their starting/ending frames, and the edges represent the possible transition of words in the hypothesis. Because the acoustic model (AM) is context dependent, the acoustic scores are stored in the edges instead of the nodes in the lattice. To avoid the overflow during computation, the acoustic scores are stored as log value.

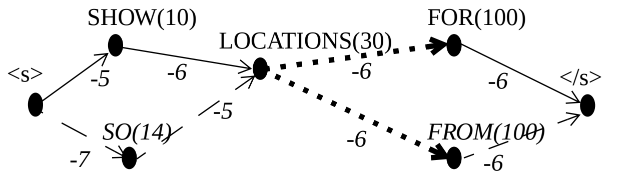
In lattice combination, we first merge all the corresponding starting and ending nodes of lattices generated from different systems together to form a bigger mixed lattice, then we edit the nodes and edges in this mixed lattice by *merging edges* from different lattices, *building new edges with/without AM recomputation* among nodes from different lattices and *renormalizing acoustic scores* from different lattices. Once all these are done, we use a search algorithm (e.g. viterbi or A*) to find the path with maximal cumulative score in the mixed lattice as combination output.

2.1 Merging edges

We first merge different edges originated from distinctive lattices together so long as their outgoing nodes have the same word label, same beginning and ending frame, and the ending nodes of these edges have the same first phone. Since the acoustical score of an edge only depends on the above constraints, edges originating from different lattice should have the same acoustic score in the mixed lattice if they satisfy the above constraints. To merge these edges, We first merge their outgoing nodes together to a new node, then change their acoustic score by taking either the



(a) Original mixed lattice



(b) Mixed lattice after merging edge

igure 1: Example of merging edges. The <S> and </S> represent the starting and ending nodes of lattice respectively.

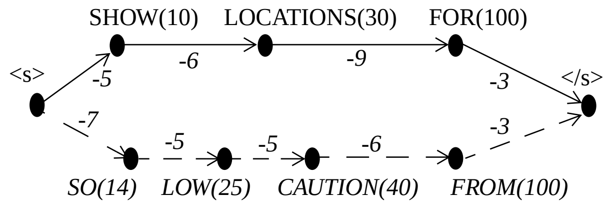
sum or maximal value among their previous acoustic scores. We choose the later method since this approach works better experimentally. Here is an example: In the figure 1(a), we have a mixed lattice generated from two single lattices, the plain words represent the words from lattice1 while italic words came from lattice2; solid line represent the edges from lattice 1 and dash line represents edges from lattice 2. The number in the ( ) behind each node is the starting frame of that node. As shown in figure 1(b), we merge edge from LOCATIONS to FOR with edge from *LOCATIONS* to *FROM* by first merging their outgoing nodes LOCATIONS and *LOCATIONS* together, then updating acoustic scores of edges from LOCATIONS to FOR and from LOCATIONS to *FROM* as represented by the dotted lines to be the maximum among their previous scores.

2.2 Creating new edges without AM recomputation

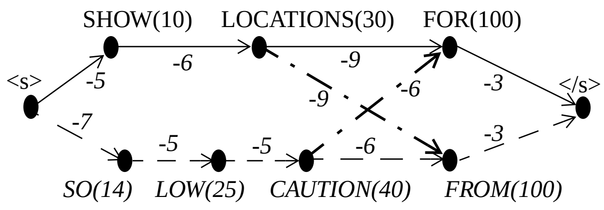
In addition to merging edges, we also build new edges between nodes from different lattices provided that their word label and time information satisfy certain kind of constraints. Specifically, we can build a new edge from node *A* to node *B* so long as there exists an edge from node *A* to node *C* whose word label has the same first phone as the word label in node *B*, and the difference in the beginning frame of node *B* and node *C* lies within our tolerance (e.g. 30 or 40 ms). Since the edge from *A* to *C* tells us that node *A* can end just before node *C, A* can also end just before node *B* so long as node *B* and *C* have similar beginning time. The acoustical score associated the new edge is assigned according to the following linear formulation:

$$W_{A \to B} = \frac{D_{A \to B}}{D_{A \to C}} \cdot W_{A \to C} \qquad (1)$$

where $W_{A \to B}$ and $W_{A \to C}$ are the acoustic scores (log value) of the edges *A* to *B* and *A* to *C*, $D_{A \to B}$ and $D_{A \to C}$ are the duration of corresponding edges. The motivation in Equation (1) is that since the acoustic score of an edge is the product of the acoustic likelihood of each frame in the duration of that edge, the longer the duration of an edge, the less its acoustical score. Figure 2 is an example of creating new edges without AM recomputation: we built an new edge from nodes LOCATIONS to *FROM* as represented by the dash - dotted line, given the existing edge LOCATIONS to FOR . Because of the same starting frame of node FOR and *FROM* , the score of the new edge is the same as the existing one. Similarly, we create another edge from *CAUTION* to FOR .



(a) Original mixed lattice



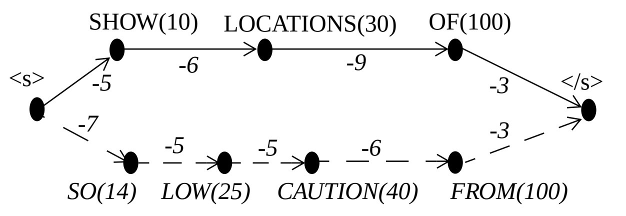
(b) Mixed lattice after creating new edges

Figure 2: Example of creating new edges without AM recomputation

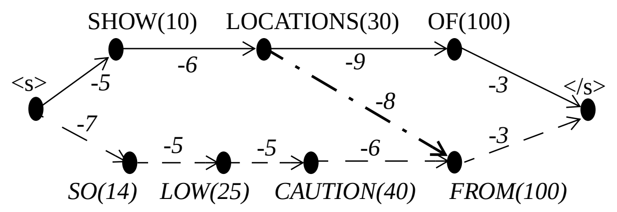
2.3 Creating new edges with AM recomputation

Theoretically speaking, when we create new edges between different nodes from different lattices, the constraint of same first phone for the incoming node of the existing edge (e.g. FOR in the figure2) and the node into which we want to build new edge (e.g. *FROM* in figure 2) is too strict. In fact, we should be able to create new edges so long as the starting time of ending nodes of existing edges is the same as the starting time of the node into which we want to build new edge. For the example in figure 3, we should still be able to build an edge from LOCATIONS to *FROM* if the only existing edge from LOCATIONS is to OF , so long as the starting time of OF is the same as starting time of *FROM* .

The reason we impose same phone constraints is that if the first phones of node B and C are different, we have no ways to assign the score to the new edge A->B, since the acoustic models for A-B and A-C are different. Fortunately, it s quite often that in addition to the lattice, we also have the acoustic model and feature for the different systems that we want to combine, in that situation, we can recompute the acoustic score of any edge that we want to create directly instead of using linear equation (1) to predict the new score, if we care more about the recognition accuracy than computation effect. But there is another thing we should care about: if we create too much new edges in the mixed lattice, at the same time we create some correct yet missing edges (the one presented in the utterance but be deleted during decoding), we will also create some new incorrect edges which may dominate other



(a) Original mixed lattice



(b) Mixed lattice after creating new edges

Figure 3: Example of creating new edges with AM recomputation

correct edges and degrade the performance.

Based on the above consideration, Once we want to create a new edge from node A originated from lattice 1 to node B originated from lattice 2, the way we create new edge with AM recomputation is performed as the following 4 steps:

- First check whether there exist another edge from Node A to Node C and the difference in the starting time of Node C and Node B lies within our tolerance (e.g. 30 ms or 40 ms).
- If there do exist such edge from A to C, then we use the acoustic model and feature of system 1 to compute the AM score of node A with right context to be the first phone of node B and ending time is one frame before the starting time of node B
- If the first phone of Node C is in the same confusable phone set as the first phone of node B, then we assign the AM score computed in the step 2 to the new edge A->B.
- If the first phones of node B and node C are in different confusable phone sets, we multiply the AM score generated from step 2 by a weight less than 1 and assign the weighted AM score to the new edge A->B.

2.4 Score normalization

Generally speaking, the distribution of AM score generated by different features or different systems are different. Since the final result in the lattice combination is the path with maximal cumulative score and this path may consists of different parts originated from different features or systems, we need to normalize the AM score of lattices from different systems before we combine them together. The way we did for normalization is that we keep the maximal acoustic modeling score in each frame during decoding, for each edge in the lattice, we sum up all the maximal acoustic scores in the duration of that edge as the corresponding normalization factor.

3. EXPERIMENTAL RESULTS AND ANALYSIS

We tested the performance of the lattice combination in the following corpus: the DARPA Resource Management (RM) corpus, the Telefornica (TI+D) Cellular Telephone corpus with artificially corrupted traffic noise at SNRs of 5 and 10 dB, the Speech In Noisy Environments 1 (SPINE1) corpus and the Speech In Noisy Environments 2 (SPINE2) corpus. Among these corpus, RM is the clean speech database without additional noise, TI+D is a telephone bandwidth spanish database.

The speech recognizer we used was CMU SPHINX III system. To the HMM structure, we use continuous model for the RM, SPINE1 and SPINE2 corpus, while semi-continuous in the TI+D 5dB and 10 dB corpus. For each testing corpus, two different features(feature1 and feature2) were used to perform the combination. For the RM, TI+D 5dB and TI+D 10 dB corpus, the feature1 is the MFCC while the feature2 is the PLP; for the SPINE1 corpus, feature1 is still the MFCC but the feature2 is a modified version of MFCC generated by performing IDCT instead of DCT from log-spectral; for the SPINE2 corpus, we first perform the KLT to generate a 20 dimensional feature vector from the 40 dimension log-spectral vector, then perform 2 different kinds of LDA to further transfer this 20 dimensional feature vector to 2 different 13 dimensional feature vectors as feature1 and feature2. The difference between these 2 LDA is that they were designed to discriminate amongst two different sets of subword-unit classes in each case.

To perform the combination, we first generated the single lattice from each feature, then we use the viterbi search script (since all the Language Modeling (LM) we used were Bi-gram) to generate the single best hypothesis for each feature, combine these single best hypothesis using *ROVER* and *Hypothesis combination* as comparison. Since we don t have the confidence score for each word generated, we only use the frequency of occurrence voting in the *ROVER*. We then combine the lattices generated from different features together, perform viterbi search on the combined lattice to generate the combination result.

WER (%)	Feature 1	Feature 2	ROVER	Hyp-Comb	Lat-Comb
RM	11.5	12.0	11.1	8.4	8.0
TI+D 5dB	25.5	26.6	26.1	25.6	24.7
TI+D10dB	12.5	13.0	13.7	11.9	11.3
SPINE 1	35.1	36.2	35.4	34.2	33.2
SPINE 2	17.5	16.6	17.8	15.9	15.0

Table 1: Recognition accuracy of three combination schemes on ve corpus. The lattice combination scheme uses merge edge nd build new edge without AM recomputation.

RM	TI+D 5 dB	TI+D 10 dB	SPINE 1	SPINE 2
0.3	0.05	0.18	0.12	0.04

Table 2: Statistical significance level between hypothesis com-ination and lattice combination

Table 1 shows the Word Error Rate (WER) obtained from each single feature and various combination schemes. For the lattice combination, we used the *merge edge, build new edge without AM recomputation*.

In addition to the WER, we also test the statistical significance between lattice combination and hypothesis combination using the matched pairs method[4]. The two-tailed significance levels are shown in the table 2.

In RM, TI+D 5dB and TI+D 10 dB corpus, we also test the performance of *building new edges with AM recomputation*. Table 3 shows the recognition accuracy of lattice combination and statistical significance level (P) between the result of lattice combination and hypothesis combination. The lattice combination is the same as the one used in table 1 except that we used the *build new edge with AM recomputation* instead of *without AM recomputation.*

From all these tables, we can see the following:

Lattice combination improve the recognition accuracy consistently in all the various testing corpus. The relative improvement of lattice combination over hypothesis combination range from 3 to 6 percent without the AM recomputation and from 6 to 8 percent with the AM recomputation. For some testing corpus like TI+D 5dB, lattice combination is the only combination scheme that reduce the WER.

The intrinsic similarity of the feature sets plays an important role in the combination. From the results in table 1 and table 2, it turns out that the highest improvement in WER and the most significant difference between hypothesis combination and lattice combination has been achieved in the SPINE 2 corpus for which the feature sets had been developed specifically to maximize the difference between different sub-word classes in the speech sentence. Actually, it s in accordance to our expectation, we know that the advantages of multiple output based combination come from finding a composite path which may consists of different parts from different system or different features. Because of the constraints during decoding (e.g. language modeling), the single best hypothesis outputted from single system is only the best in the overall sense. In many situations, words that are presented in the utterance do not appear in this single best path even though they may appear in some other paths during decoding. And this situation become more apparent when the feature sets we use are specified to certain sub-word classes.

4. DISCUSSION AND FUTURE WORK

As to the possible future work, here are some points:

First, the more appropriate way to update the score of the edges during the *merge edge* procedure. If we can find some other more appropriate ways to update the score instead of simply taking the maximal value among existing edges as the new score, we should get some additional benefits.

	RM	TI+D 5dB	TI+D 10dB
WER (%)	7.8	24.3	11.1
P	0.16	0.03	0.08

Table 3: The recognition accuracy and the statistical significance level (P) between hypothesis combination and lattice combination with merge edge and build new edge with AM recomputation

The second thing we can do is to find an adaptive way of setting threshold instead of using a fixed value for the whole testing corpus when we *merge edge* and *create new edges with/without AM recomputation*. Since different words may vary in number of phones, duration etc., using different threshold for different words or phones will also helps us.

Another thing from which we can get the benefit is to find more appropriate ways of normalization. Using maximal decoding score as the normalization factor is just kind of local normalization and our goal is to find an appropriate global normalization method.

The last but not the least point is to find the feature sets that are as different as possible. As can be seen from our experimental results, even though the implementation of lattice combination is independent of the feature sets used in the combination, the combination effect significantly depends on these feature sets. Developing maximally different features will help us take the full advantages of lattice combination.

5. SUMMARY

We have developed a new approach to combine the results of different feature or different systems together in speech recognition. Instead of using only the single best hypothesis during the combination, our lattice combination method exploits the benefit of multiple hypothesis output by preform the combination using the lattices generated from different systems. By merging edges, building new edges with/without AM recomputation and score normalization, our lattice combination method achieved consistent improvement in recognition accuracy over five different speech corpora.

6. REFERENCE

[1] Fiscus, J.G., A post-processing system to yield reduced word error rates: Recognizer Output Voting Error Reduction (ROVER), *Proc. IEEE Workshop on Automatic Speech Recognition and Understanding*, pp. 347-354, 1997.

[2] Singh, R., Seltzer, M., Raj, B., and Stern, R.M, Speech in noisy environments: robust automatic segmentation, feature extraction, and hypothesis combination , *Proc. ICASSP 2001*, Salt lake city, Vol 1, pp.273-276, 2001

[3] Mangu, L., Brill, E., Stolcke, A., Finding Consensus Among Words: Lattice-Based Word Error Minimization , *Proc. EUROSPEECH 1997*, Vol 1, pp. 495-498, 1997

[4] Gillick, L., Cox, S.J., Some statistical issues in the comparison of speech recognition algorithms , *Proc. ICASSP 1989*, Vol 1, pp. 532-535, 1989.