IMPROVING VOCABULARY INDEPENDENT HMM DECODING RESULTS BY USING THE DYNAMICALLY EXPANDING CONTEXT

Mikko Kurimo

Neural Networks Research Centre, Helsinki University of Technology
P.O.Box 2200, FIN-02015 HUT, Finland
mikko.kurimo@hut.fi

ABSTRACT

A method is presented to correct phoneme strings produced by a vocabulary independent speech recognizer. The method first extracts the N best matching result strings using mixture density hidden Markov models (HMMs) trained by neural networks. Then the strings are corrected by the rules generated automatically by the Dynamically Expanding Context (DEC). Finally, the corrected string candidates and the extra alternatives proposed by the DEC are ranked according to the likelihood score of the best HMM path to generate the obtained string. The experiments show that N need not be very large and the method is able to decrease recognition errors from a test data that even has no common words with the training data of the speech recognizer.

1. INTRODUCTION

The use of hidden Markov models (HMMs) [7] is nowadays clearly the dominant trend in the automatic speech recognition (ASR). The HMMs connect two contributing stochastic processes, the production of the signal state sequence and the generation of the acoustic features in each state, into a single probabilistic framework. Using models corresponding to phonemes, the probabilities of different phoneme sequences can be estimated and compared after the extraction of the short-time feature vectors from the signal. The HMMs are normally applied to generate the most probable phoneme string or to select it from a given set of alternatives.

If the recognition task is to transcribe the speech into phoneme sequences that are not limited to any vocabulary, there is normally not much to be done to correct the occurring recognition errors. Some rules for certain common differences between the spoken and written text can be checked, but there is no general way to correct misspelled text, because there would be several alternatives and actually, no list of acceptable words is available. In [2] the Dynamically Expanding Context algorithm is suggested to correct erroneous phoneme strings in the postprocessing stage of a speech recognizer for unlimited vocabulary of Finnish using context-dependent production rules. The method learns the rules automatically from lists of source and target string pairs.

In many ASR applications well-defined vocabularies or grammatical constraints limit the variety of possible words and the structure in each utterance. To optimally select the best matching word the information extracted from HMM decoding can include N best HMM strings ($N > 1$) and their current likelihoods. This HMM information is then combined with the available additional information in the postprocessing stage of the recognition process.

In this work the recognition task is the vocabulary independent ASR. The objective is to improve the HMM result strings so that the final result would be the best string that follows the DEC rules. Since it would be difficult to bring the DEC rules directly into the HMM decoding, the problem is approached by transforming all the best HMM string candidates by the DEC and selecting the best by computing the HMM scores for the obtained set of promising valid strings (see Figure 1).

2. DYNAMICALLY EXPANDING CONTEXT

The Dynamically Expanding Context (DEC) [2] is a method to generate context-dependent grammatical mappings. It has been used, for example, as the postprocessing stage of a speech recognizer for unlimited vocabulary of Finnish [2, 3]. The rules are generated automatically by comparing phoneme strings produced by the speech recognizer with the corresponding lists of correct words.

A production rule is a mapping from a string segment A of the source string S into the corresponding segment B of the transformed string T. The rule is applied, if the context segments around A in S matches with the specifications of the rule. For example, the recognizer might hear the Finnish diphthong /au/ as /aou/ and so the transformation /aou/ $\rightarrow$ /au/ should be included into the rule set. The rule is then formed to always replace the string segment $a(o)u$ by $a()u$, where a and u denote the context and $A = (o)$ and $B = ()$. The transformed result T of the whole string S is generated simply by concatenating all the transformed segments B.

The DEC implements the idea of storing minimum amount of context for each rule while keeping the productions unique. For each segment the length of the context is the smallest possible one without conflicts in the training sample set. A rule causes a conflict, if there are more than one possible transformation results for the same segment and context.

2.1. The generation of the DEC rules

For the postprocessing of speech recognition results the possible transformations between string segments are extracted from lists of string pairs (S, T) where the source string S is the result string of the recognizer and the target string T is the corresponding correct word from the dictated word list. The method applied to generate the rules from the (S, T) pairs is described here only briefly (for more details c.f. [3]).

The processing of a sample pair (S, T) begins by aligning the corresponding segments in S and T. The alignment of the cor-

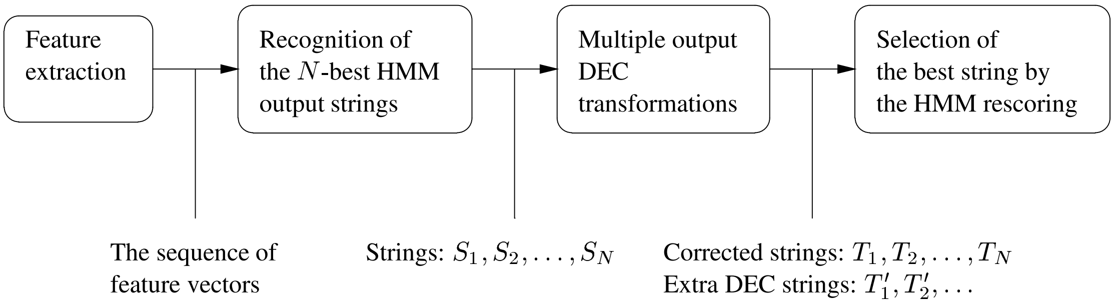

Figure 1: The stages of the N-best HMM-DEC decoding and the information that is transmitted between the stages.

rect and the erroneous string can often be done in several alternative ways. However, the correct alignment is most often the one which introduces the smallest amount simple segment transitions. This alignment is defined by minimizing the Levenshtein distance (the sum of inserted, deleted and substituted segments) between the strings. For example, for the string transformation /kolomä/ → /kolme/ (the number three in Finnish), the correct alignment should give the segment transitions /o/ → // and /ä/ → /e/.

The production rules are first stored using a small context, e.g. l(o)m → () for the given example. However, if a conflict is encountered, the rule is marked ambiguous and new rules are introduced with a little longer context, e.g. ol(o)m → (). Thus the generation of a new rule is always started by checking the conflicts against the old rules.

It is often reasonable to set an application dependent upper limit for the length of the context. To be sure that the DEC is correct, the training data can be scanned a few times. The sufficient number of iterations depends on the chosen maximal context length. The theoretical upper limit is half of the number of possible context expansions steps plus one [3].

2.2. Some problems in the training

In practice, one important problem in training the DEC is how to obtain the enough long list of the string pairs (S, T) to cover all the main phoneme contexts where transformations may exist. Those contexts should also appear often enough that all the expectable transformations would emerge and could be adopted by the DEC. In ASR this can be helped by selecting the dictated words so that the number of phoneme combinations is somehow balanced. A different data set not used in the training of the current HMMs might reveal more variation to the DEC, but unfortunately, all available training data are normally needed to train the HMMs.

Another problem is how to check that each obtained DEC rule is defined by a context specific enough so that no correct string outside the training material would get transformed to an incorrect one. Due to the storage and collecting difficulties the training material is normally so limited that such strings can always exist. In this paper the DEC was additionally trained to form (T, T) identity string pairs for some large enough written text corpus to ensure that at least all segments in correct words will remain unchanged.

2.3. Finding the candidates of corrected strings

First the string S is divided into a sequence of successive segments. The DEC production rules are then sought for each segment. If no matching rules are found, the segment is accepted to be correct as such. If the matching rule is ambiguous, the context of the segment is expanded and the search is continued to the next context level. When a unique rule is encountered its outcome is concatenated to the target string.

It is possible that the sequence of production rule expansions found in the search end up to an ambiguous rule. To select the best one from the available target segments, the majority voting procedure described in [3] can be applied. In the current paper, however, all the candidates are collected by duplicating the target string whenever there is more than one possible output segments. After the checking all the segments there may thus exist several alternative candidates for the corrected string (see Figure 1).

3. N-BEST HMM

The conventional way to decode the sequence of feature vectors by HMMs is to extract the most probable sequence of the model states by using the Viterbi search. In ASR it is nowadays common (for example, c.f. [1, 6]) that the calculation of the short-time probabilities connecting each observed feature vector to each model state will be the most time consuming part of the process. and the Viterbi search by itself can be quite fast.

When the recognition result given by the HMMs are passed to some further postprocessing transformations, it is possible that the transformed result corresponds no longer the best most probable result in the sense that the HMM decoding result was selected. For example, some rival HMM result that was maybe defeated by a narrow margin might actually be closer to the correct result and give a more probable result string in the postprocessing transformation. Another way to view this problem is to note that if the HMM decoding is not the final stage of the recognition process, it could be more robust to feed the next stage with several alternative strings instead of compressing all the classification information into a single phoneme string.

Since it does not require any new computation of short-time observation probabilities, the decoding of N-best HMM strings ($N > 1$) does not excessively increase the computational effort of the recognition procedure. In fact, there are fast N-best search algorithms, for example [8], which need only a relatively small amount of extra computation compared to the best candidate Viterbi search. The N-best HMM decoding has long been used to provide candidate words for various postprocessing systems. In this paper it is applied is to generate candidate strings to be converted by the DEC into correct words which are ranked by their best HMM alignment scores (see Figure 1). The DEC only corrects the improper segments in the decoded strings so the task of the HMM decoding is to extract the most likely coarticulation and other errors in order to inspire the most likely correct words. Analogous situation is in the automatic correction of spelling errors, where, for example, a rule-based system generates a whole set of close-by correction candidates and then applies some matching measure to select the most probable correct word [4].

4. THE FINAL RECOGNITION RESULT

The previous sections already described, how to generate a list of possible output strings by N-best HMMs and a multiple output DEC. At least one fundamental problem still exists: How to select the correct output from that list?

If the task would include processing of further constraints based on, for example, some grammatical or other knowledge of the possible words and their order, all the corrected strings could be fed forward as such. In this paper, however, the task is to convert the spoken utterances to the best matching phoneme strings not limited by any particular vocabulary or structure. The output string should thus be the one that fits best to the feature vector sequence using some applicable measure.

Since the short-time observation probabilities for HMMs have already been computed, it is a rather quick and straight-forward procedure to find the best time alignment of the HMMs for each corrected string. Then the likelihood score of this best time alignment can act as the comparison measure between the candidate strings, just as if we were selecting the most probable decoding in a normal HMM.

5. EXPERIMENTS

The recognition experiments were made using the speech recognition system based on mixture density HMMs [5] trained by Self-Organized Maps [3] and Segmental LVQ3 as described in [6]. The basic feature vector is a 15-dimensional mel-cepstrum concatenated by the energy component. 140 mixture Gaussians are trained for each HMM using 80-dimensional context vectors formed by concatenating 5 basic feature vectors for time windows of different lengths around the current frame [6].

The data includes four sets of 350 words per each speaker. Three sets are used for training the models and remaining one for testing. To get the average recognition rates used in method comparisons the speaker-dependent tests are repeated for seven different speakers. In Tables 1 and 2 three different rates are given to describe the correctness of the results. The phoneme error rate counts the insertion, deletion and substitution errors, the correct phoneme rate counts only the correct phonemes and the correct word rate counts only the fully correct words with no phoneme errors. To estimate a limit for the best possible string ranking method with the given HMM and DEC results the top-N rate is computed as well. It counts how often the correct word is among the top N result strings.

As can be seen from Table 1, the DEC results were very good in the original test set. Actually even by using the DEC in the conventional way [3] to correct only the best HMM result string the rates are about as good as by the 5-best HMM-DEC. The DEC seems to be working very well as noted also in [9].

One problem with the DEC is that if it encounters recognition errors in untrained phoneme contexts, it may try to use invalid rules for corrections and end up to an even worse string. To prevent this the DEC should be trained using large variety of typical erroneous

Rate %	HMM	DEC-1	DEC-5
Original test set			
Phoneme errors	5.2	2.3	2.2
Correct phonemes	96.0	98.2	98.3
Fully correct words	73.8	90.3	90.1
Correct in top 5	79.4	-	97.3
Reorganized test set			
Phoneme errors	7.2	7.8	6.6
Correct phonemes	94.8	94.2	95.4
Fully correct words	66.8	69.1	70.9
Correct in top 5	72.6	-	80.9

Table 1: The average test set recognition rates using only the HMM decoding “HMM”, using the conventional single output DEC for the best HMM string “DEC-1” and using the proposed 5-best HMM-DEC “DEC-5”.

strings, but this is normally difficult to execute, for example for a new speaker. Anyhow, the phoneme contexts in the current test set that was used in [6] as well, are quite well covered in the training data although it is all recorded on a separate session in a different day.

To put the DEC (and the HMMs as well) into a more challenging test, the division of the data into training and test sets is reorganized. For each speaker the data is divided now so that in each speech recording, including the previous test set recording as well, every fourth word is selected for the test set and the other three for the training set. So the size of the test and training sets are about the same as in the previous test, but the word lists used in the training and test sets are now completely distinct. An additional difficulty is that the training material will not any more be balanced for the phoneme combinations, which may be fatal, because the remaining set of 263 different training words is not very large. Some phoneme pairs or even some rare phonemes may now be completely missing from the training data and at the same time over-emphasized in the test data. At least for the HMMs this will surely introduce some additional recognition errors and some DEC rules will probably be missing as well (the total size of the DEC can be over 30 % smaller).

The rates for the reorganized test set in Table 1 show that the HMM recognition rates were clearly better in the original test set. Although the number of fully correct words can still be increased by the conventional DEC, the total increase of phoneme errors reveals that some unseen test strings introduce new problems. The 5-best DEC-HMM system selects the result string from the generated set of several alternative corrected strings and the improved rates show that on the average it also succeeds to reduce errors.

In theory, the DEC should be correct for the training data only after four iterations, but the experiments indicated that after the third and subsequent iterations with the current training data the average test data result is still the same as after the second iteration. The reason is that after two iterations the DEC can already handle the strings of the whole training data almost 100.0 % correctly so that very few rule modifications can still take place and results from those modifications will not be seen as a significant change in the test data result. After the first DEC iteration the column of average performance rates corresponding to the last column of Table 1 were: 2.3 %, 98.3 %, 89.3 % and 96.6 % and for the reorganized test set: 6.7 %, 95.4 %, 70.4 and 79.8 %.

The checking of the rules using a large vocabulary will, at least, prevent that any unseen phoneme combinations in correct strings would get transformed into incorrect results. This checking was already included in the DEC results of Table 1, but without this check the three recognition rates corresponding to the last column of Table 1 were 2.4 %, 98.1 % and 89.2 % for the original test set and 11.5 %, 91.6 % and 51.2 % for the reorganized test set. So, the latter extreme test shows how vital this check is if untrained phoneme contexts are to be expected.

Rate %	1	2	3	4	5	6	10
Orig.	Single output DEC						
Errors	2.3	2.1	2.1	2.2	2.2	2.3	2.3
Phon.	98.2	98.4	98.4	98.3	98.3	98.3	98.3
Words	90.3	90.6	90.4	90.1	90.0	89.8	89.6
Top-N	90.3	93.8	95.3	95.6	95.6	95.6	95.6
Orig.	Multiple output DEC						
Errors	1.9	1.9	2.1	2.1	2.2	2.2	2.3
Phon.	98.6	98.6	98.5	98.4	98.3	98.3	98.2
Words	92.1	91.6	90.8	90.5	90.1	89.9	89.3
Top-N	93.2	96.1	97.2	97.3	97.3	97.3	97.3
Reorg.	Single output DEC						
Errors	7.8	7.4	7.1	7.1	7.0	7.0	7.0
Phon.	94.2	94.8	95.1	95.2	95.3	95.3	95.2
Words	69.1	70.2	70.6	70.7	70.8	70.7	70.4
Top-N	69.1	74.4	76.4	77.3	77.8	77.9	78.1
Reorg.	Multiple output DEC						
Errors	7.0	6.7	6.6	6.6	6.6	6.6	6.6
Phon.	94.9	95.3	95.4	95.4	95.4	95.4	95.4
Words	69.8	70.6	70.7	70.8	70.9	70.8	70.5
Top-N	71.9	77.4	79.3	80.1	80.9	81.1	81.4

Table 2: The average rates of phoneme errors "Errors", correct phonemes "Phon.", fully correct words "Words" and fully correct words to belong in the top-N string set. $N = 1, 2, \ldots$ is the amount of strings extracted from the HMM decoding and corrected by the DEC. Results on both the original test set "Orig." and the difficult reorganized test set "Reorg." are given.

At first one might think that the result of N-best DEC-HMM could be improved with small computational effort by expanding to more and more result candidates (N), giving them to DEC and taking the winner string according to the HMM score. Anyhow, according to the experiments (Table 2) the increased recognition time, for example in expanding N from 5 to 10 is only about 3 % as expected, but, more importantly, the recognition performance does not improve. The experimental results seem to indicate that there is actually no notable difference between the rates using values $N = 2, \ldots, 10$. One reason might be that while more candidates are given to DEC the new strings are further and further away from the correct word and thus they may, in fact, introduce more new errors to the ranking of candidate strings than correct words. For the correct words that might occasionally be found by increasing N, the chances to win the ranking of all the new result strings will probably get lower as N increases. The natural explanation is that errors inevitably occur when the string that fits to the DEC rules and matches best to the HMMs simply is incorrect. For the difficult reorganized test set the multiple output DEC and the increase of N give some slight improvements for the recognition rates. Using only this test it is impossible to select any generally superior value for N, but anyhow, there seems to be no reason to use $N > 5$.

6. CONCLUSIONS

This paper presents a method to connect a N-best HMM system and a multiple output DEC to find out the best result string obeying the DEC rules. The method is approximative in the sense that only the N best matching HMM strings are transformed by the DEC, but the experiments show that N need not be very large. According to the experiments the DEC is a very successful tool to correct erroneous strings, but naturally, the rules are valid only for the phoneme combinations seen on the training data. When tested on an extreme data in which the dictated word set is completely distinct from the set used for training, some of the errors can still be corrected by using the suggested N-best HMM-DEC system.

7. REFERENCES

[1] Enrico Bocchieri. Vector quantization for the efficient computation of continuous density likelihoods. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, volume 2, pages 692–695, 1993.

[2] Teuvo Kohonen. Dynamically expanding context, with application to the correction of symbol strings in recognition of continuous speech. In *Proceedings of the 8th International Conference on Pattern Recognition*, pages 1148–1151, Paris, France, 1986.

[3] Teuvo Kohonen. *Self-Organizing Maps*. Springer, Berlin, 1995.

[4] Karen Kukich. Techniques for automatically correcting words in text. *ACM Computing Surveys*, 24(4):377–439, December 1992.

[5] Mikko Kurimo. Hybrid training method for tied mixture density hidden Markov models using learning vector quantization and Viterbi estimation. In *Proceedings of the IEEE Workshop on Neural Networks for Signal Processing*, pages 362–371, Ermioni, Greece, September 1994.

[6] Mikko Kurimo. *Using Self-Organizing Maps and Learning Vector Quantization for Mixture Density Hidden Markov Models*. PhD thesis, Helsinki University of Technology, Espoo, Finland, 1997.

[7] Lawrence R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

[8] Frank K. Soong and Eng-Fong Huang. A tree-trellis based fast search for finding the n best sentence hypotheses in continuous speech recognition. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, volume 1, pages 705–708, 1991.

[9] Kari Torkkola, Jari Kangas, Pekka Utela, Sami Kaski, Mikko Kokkonen, Mikko Kurimo, and Teuvo Kohonen. Status report of the Finnish phonetic typewriter project. In T. Kohonen, K. Mäkisara, O. Simula, and J. Kangas, editors, *Artificial Neural Networks*, volume I, pages 771–776, Amsterdam, Netherlands, 1991. North-Holland.