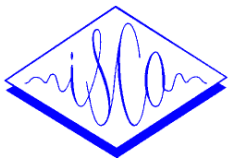
ISCA Archive
http://www.isca-speech.org/archive



4th European Conference on
Speech Communication and Technology
EUROSPEECH '95
Madrid, Spain, September 18-21, 1995

Optimizing Baseforms for HMM-Based Speech Recognition

Torbjørn Svendsen [1] *Frank K. Soong* [2] *Heiko Purnhagen* [3]

[1]The Norwegian Institute of Technology, Trondheim, Norway
[2]AT&T Bell Laboratories, Murray Hill, NJ, USA
[3]University of Hannover, Hannover, Germany

ABSTRACT

In this paper we propose a new, automatic optimal baseform determination algorithm. Given a set of subword Hidden Markov Models (HMMs) and acoustic tokens of a specific word, we apply the tree-trellis N-best search algorithm to find the optimal baseforms (transcriptions) in the maximum likelihood sense. Different token preselection algorithms have been investigated to facilitate fast search for representative baseforms and to alleviate the problem of representing vastly different pronounciations with a single baseform. The DARPA Resource Management database was used for evaluating the new baseform optimization algorithm, improvements of recognition rates using different token selection algorithms and the tree-trellis search have been consistently obtained.

1. INTRODUCTION

Modern large vocabulary speech recognizers employ subwords as the basic modeling units. This implies that in order to obtain word (or sentence) recognition, a lexicon which defines the composition of the vocabulary words in terms of the basic units must be made available to the recognizer. In most cases, linguistically defined subwords are chosen as the basic recognition units, typically phonemes or phone-like units.

In contrast to almost all other components in an HMM based speech recognizer which are optimized with respect to some objective criterion, the lexicon is commonly created by human experts or by the use of standard pronounciation dictionaries. Particular problems with these approaches arise, e.g., in cases where the pronounciation variations of many speakers with different dialects need to be represented by one or a small number of lexical entries. Also, the traditional approaches cannot readily be applied to systems employing non-linguistic recognition units.

Some efforts to automatically optimize this process have been reported over the past decade (see e.g., [1, 2, 3]). The present work proposes a method for automatic generation of pronounciation baseforms with respect to some pre-defined basic units such as phonemes according to an objective optimization criterion. The method is equally applicable to either linguistically or acoustically defined units, and generates optimal baseforms of the words in the vocabulary.

2. OPTIMAL BASEFORMS

The problem to be addressed can be formulated as follows: Given a number of sample utterances of a word, $U = \{U_1, U_2, \ldots, U_L\}$, and a Hidden Markov model, $\lambda = \{\lambda^1, \lambda^2, \ldots, \lambda^P\}$, describing all the subword units $p = \{p_1, p_2, \ldots, p_P\}$, find the most likely string of subword units, $\hat{S}$, over the set of all possible strings, S, subject to some given decoding grammar

$$\hat{S} = \underset{S}{\text{argmax}}\{P(S|U, \lambda)\} \qquad (1)$$

Assuming that all strings are equally probable, this is equivalent to maximizing the joint likelihood of the utterances, i.e.,

$$\begin{aligned} \hat{S} &= \underset{S}{\text{argmax}}\{P(U|S, \lambda)\} \\ &= \underset{S}{\text{argmax}} \prod_{i=1}^{L} P(U_i|S, \lambda) \end{aligned} \qquad (2)$$

If (2) is to be optimized over only one utterance, the problem can be easily solved by Viterbi decoding. However, the number of sample utterances per word will need to be large enough to produce word models that are robust to both inter- and intra-speaker variations yielding a non-trivial optimization problem. Fortunately, it turns out that a modification of the tree-trellis based fast search proposed by Soong and Huang [4] for finding the N best string hypotheses in principle guarantees finding the optimal baseforms.

The tree-trellis algorithm is based on the A*-algorithm for finding the optimal path(s) through a tree [5]. In order to provide the A*-search with a cost estimate, a time synchronous Viterbi search is performed in the (time-)forward direction, storing accumulated likelihood scores for every time instance for all word terminal states. The A*-search is then performed time asynchronously in the backward direction to find the N best strings. In order to maximize (2), the tree-trellis algorithm need some modifications.

The modified tree-trellis algorithm commences by computing a likelihood map of each of the L observations. Then, a frame synchronous Viterbi trellis search is performed in the forward direction according to some given decoding grammar. A possible choice is the simple free subword decoding (e.g., free-phoneme decoding) grammar shown in fig. 1, where any number of subword units in any order can constitute a string. The Viterbi search is executed using level building to accommodate length variations in the string hypotheses. For every observation, the Viterbi search also generates a partial path map which for every frame and grammar node contains the accumulated likelihood scores of all partial paths leading to the grammar node,

$$\max_{q_{1\to t-1}} P(O^{(l)}_{1\to t}, q_{1\to t-1}, q_t = i(\lambda_{k_n})|\Lambda_{l,n}) \quad (3)$$

where $i(\lambda_{k_n})$ is the final state of the kth arc leading to the nth grammar node and the model $\Lambda_{l,n}$ is a string model constructed by concatenating n subword models. $q_{1\to t-1}$ denotes the state sequence from time 1 to $t-1$ and $O^{(l)}_{1\to t}$ denotes the t first observation vectors of sample utterance l. In addition, an ordered list of the rank of each partial path is produced.

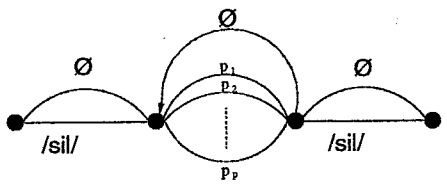


Figure 1. Free subword decoding grammar.

After completing the forward Viterbi search, a backward, frame asynchronous tree search is performed. The search finds the phoneme string that maximizes the combined likelihood scores for the L tokens of a word. If so desired multiple lexical entries can be found using this algorithm, producing a rank ordered list of transcriptions based on the corresponding likelihood scores.

The tree search is implemented with the A* algorithm. Here, use is made of an evaluation function

$$\hat{f}(n) = \hat{g}(n) + \hat{h}(n) \quad (4)$$

where $\hat{f}(n)$ is an estimate of the likelihood score of the best path passing through grammar node n, $\hat{h}(n)$ is an estimate of the likelihood score of the best path from grammar node n to the terminal node and $\hat{g}(n)$ is an estimate of the (yet incomplete) path from grammar node n to a goal node (which in our case is the first or the leftmost node in the grammar). As we proceed backward from the final node $\hat{h}(n)$ can be calculated exactly by working the Viterbi algorithm time reversed. If only one token per word were used, an exact value of $\hat{g}(n)$ could be directly obtained from the forward Viterbi search. Since the likelihood scores for different tokens may be the result of different paths through the string grammar network, we can only obtain an upper bound for the value of $\hat{g}(n)$. This is sufficient to guarantee that an optimal path will always be found [5], although it will not guarantee that the optimal path is found in a minimum number of steps.

3. TOKEN PRESELECTION

When generating a lexicon for speaker independent recognition applications, it is essential that the algorithm is presented with training tokens which represent typical variations in word pronounciation. However, if the number of training tokens is high, there may exist a large number of partial baseforms with similar likelihood scores during the search. This mainly occurs for longer words, and can lead to fan-out explosions in the backward tree search resulting in computer memory shortage. One possibility of reducing this problem is to randomly select a sub-set of the training tokens from the available speech corpus for baseform generation.
When multiple tokens of the same word exhibit large idiosyncratic or accentual differences among different speakers, or large intraspeaker variations, the algorithm will not be able to produce a single *representative* baseform. This is analog to the problem of finding a single, representative centroid from data that form multi-

clusters or distribute sparsely over a large region. Moreover, since no representative, optimal baseform exists, the search for the optimal baseform can be an agonizingly slow process. In order to minimize this problem, a string clustering algorithm has been developed in order to ensure that the modified tree-trellis algorithm is presented with similar acoustic realizations of the word in question. This clustering of the training tokens can also be used for the construction of multiple baseforms.

3.1. String clustering

If two training utterances of the same word result in similar subword strings being output from a subword recognizer, it is reasonable to believe that the two acoustic realizations of the word are similar. The string clustering algorithm has been developed in order to group training utterances with close matching subword transcriptions and is a modification of the algorithm presented in [6].

The source data for the string clustering algorithm is provided by an N-best subword recognizer. For each training token of a word, the tree-trellis algorithm is used to provide the N best string hypotheses. This yields NL strings, where L is the number of training tokens for a word. The distance between two strings is evaluated by using dynamic programming alignment between two strings and the Levenshtein distance measure [6] which sets the cost of insertions, deletions and substitutions uniformly to 1, otherwise the distance is 0.

The clustering is initialized by assigning the N string hypotheses of the first token to N clusters. For each new training token, the respective string hypotheses are assigned to the cluster which has the cluster center with the lowest distance to the present hypothesis, provided that this distance is below some predetermined threshold. Otherwise, a new cluster is created. After cluster assignment, the cluster member yielding the lowest intracluster distortion is chosen a new cluster center. After all training tokens have been assigned to their nearest neighbor cluster, the procedure is iterated until convergence.

The methods for preselection considered in this study were the following:

max_100: A selection from a randomized list of the training tokens. A maximum of 100 tokens were used per word. If the modified tree-trellis algorithm failed due to memory shortage, a back-off strategy of using the original transcription was employed.

rand_10: A reasonable number ($\sim$ 10) of training tokens is necessary in order to obtain a representative transcription. Also, long words tend to increase the risk of search failure due to the physical constraints like memory saturation, etc.. New transcriptions were only generated for words with no more than 10 phonemes in the original transcription and where at least 10 training tokens were available. 10 tokens per word chosen from a randomized list of the training tokens were used.

clust_100: The tree-trellis algorithm is used to provide a list of the N most likely transcriptions for each token. String clustering is then employed to group similar transcriptions of each word. Up to 100 tokens from the largest cluster for each word are preselected as training tokens.

max_clust: max_100 is used with a new back-off strategy. clust_100 is used for the words which exhibit problems due to a large number of fan-outs in the tree search.

4. EXPERIMENTS

The proposed method has been tested on the DARPA Resource Management speech corpus. Using the standard word transcriptions simple context independent phoneme models with a single Gaussian has been generated from the speaker independent training set. 47 phone models and two silence models were used. Feature vectors containing 39 parameters (12 MFCCs and normalized energy, velocity and acceleration parameters) were computed every 10 ms using a 25 ms window. Using the phone models, the continuous utterances were segmented on the word level. After preselection of training tokens, new lexical entries were then generated by the modified tree-trellis algorithm.

Of the 991 words in the RM vocabulary, 988 words occur at least once in the training set. 592 of these words qualified for the preselection method **rand_10** (i.e., at least 10 occurrences and a maximum of 10 phonemes in the standard transcription). Also, not all vocabulary words are in the Feb89 test set which was used for evaluation. In order to have a fair comparison between the various methods for preselec-

tion, new lexicon entries were generated for 301 words which were both in the training and test sets of all methods. For the remaining vocabulary words, the original phonemic transcriptions were used.

It was observed that the new lexical entries consistently improved the average per frame likelihood score on both the training and the test set. Almost all new lexical entries represented reasonable pronounciations. For instance, the word INDIAN, which had the original transcription /ih n d iy ax n/ was transcribed as /ih n y n/ by all methods. Typically, 20-25% of the lexical entries generated automatically were identical to the original transcriptions.

The methods were compared in terms of word recognition rates on the task of continuous recognition of the test sentences. The word-pair grammar was used for the recognition experiments. The results are given in Table 1.

Lexicon	Words	
	Correct	Accuracy
"original"	78.3%	76.1%
max_100	80.6%	78.3%
rand_10	79.2%	76.5%
clust_100	79.8%	77.4%

Table 1. Performance using 301 new entries.

As shown in Table 1 all methods for automatic baseform generation yielded an improvement over the original lexicon. It was however observed that the **max_100** preselection resulted in too large number of fan-outs in the backward tree search which forced the computer to abandon the search for optimal baseforms. This happened in as many as 187 of the 988 words present in the training corpus. The **clust_100** preselection on the other hand had no such problems and produced baseforms for all words. In the next experiment, the **max_clust** method described in the previous section was compared to the performance of the **clust_100** preselection and to the original transcription. In this case, no fan-out problems were encountered, and the results presented in Table 2 are obtained using the 988 automatically generated baseforms.

The hybrid preselection method is clearly superior, both to the **clust_100** preselection and to the original transcriptions.

Lexicon	Words	
	Correct	Accuracy
"original"	78.3%	76.1%
clust_100	78.6%	75.8%
max_clust	81.7%	79.8%

Table 2. Performance using 988 new entries.

5. CONCLUSIONS

A new method for automatic generation of acoustic baseforms has been presented. The method is based on selecting the baseforms which maximize the likelihood of the training data, which is consistent with the recognition criterion. The recognition results indicate that the automatically generated baseforms yield superior performance to manually created phonemic baseforms. The method is applicable also to non-linguistic subword units, and may be exploited to solve a major obstacle in employing non-linguistic subwords for speech recognition, namely the question of how to create an optimal lexicon.

REFERENCES

[1] J.M.Lucassen, R.L.Mercer: "An Information Theoretic Approach to the Automatic Determination of Phonemic Baseforms", Proc. ICASSP'84, pp. 42.5.1-42.5.4, San Diego, 1984

[2] A.Asadi, R. Schwartz, J.Makhoul: "Automatic Modeling for Adding New Words to a Large Vocabulary Speech Recognition System", Proc. ICASSP'91, pp. 305-308, Toronto, 1991

[3] L.R.Bahl et al.: "A Method for the Construction of Acoustic Markov Models for Words", IEEE Trans. on Speech and Audio Processing, pp. 443-452, Oct. 1993

[4] F.K.Soong, E.F.Huang: "A Tree-Trellis Based Fast Search for Finding the N Best Sentence Hypotheses in Continuous Speech Recognition", Proc. ICASSP'91, pp.701-704, Toronto, 1991

[5] N.Nilsson: *Problem-Solving Methods in Artificial Intelligence,* McGraw Hill, New York, 1971

[6] S.Y.Lu, K.S.Fu: "A Sentence-to-Sentence Clustering Procedure for Pattern Analysis", IEEE Trans. on Systems, Man and Cybernetics, pp. 381-389, May 1978