Improving the Multi-stack Decoding Algorithm in a Segment-Based Speech Recognizer

Gábor Gosztolya and András Kocsor

Research Group on Artificial Intelligence of the Hungarian Academy of Sciences and University of Szeged, H-6720 Szeged, Aradi vértanúk tere 1., Hungary,
ghosty@rgai.inf.u-szeged.hu, kocsor@inf.u-szeged.hu

Abstract. During automatic speech recognition selecting the best hypothesis over a combinatorially huge hypothesis space is a very hard task, so selecting fast and efficient heuristics is a reasonable strategy. In this paper a general purpose heuristic, the multi-stack decoding method, was refined in several ways. For comparison, these improved methods were tested along with the well-known Viterbi beam search algorithm on a Hungarian number recognition task where the aim was to minimize the scanned hypothesis elements during the search process. The test showed that our method runs 6 times faster than the basic multi-stack decoding method, and 9 times faster than the Viterbi beam search method.

Keywords. search methods, segmental speech model, speech recognition, multi-stack decoding, Viterbi beam search.

In the last two decades the performance of speech recognition has greatly improved. With the growth of computational power, the same algorithms are computed faster, and this also makes further refinements possible, which may result in a more accurate recognition system. However the more complex the approach, the bigger the hypothesis space might be. The search over the hypothesis space is a vital issue because there are usually millions of hypotheses, and most of them have very low probabilities. This is why the search methods remain an important issue in speech recognition even today.

The multi-stack decoding method [3] is a well-known search algorithm with a generally good performance. However, we sought to improve it by constructing a faster method which recognized the same amount of test words. Although we tested the improvements on Hungarian numbers, all of these strategies can be used for other languages and utterances as well. The tests were performed within the framework of our segment-based speech recognition system called the OASIS Speech Laboratory [5], [7].

The structure of the paper is as follows. First we briefly discuss the fundamentals of speech recognition, then compare the frame-based and segment-based approaches. Next we explain the standard multi-stack decoding method and the Viterbi algorithm, and our improvements on the former. The final part of the paper discusses the experiments, the effectiveness of each improvement, and the results we obtained in practice.

1 Frame-Based vs. Segment-Based Speech Recognition

In the following the speech signal A will be treated as a chronologically increasing series of the form $a_1 a_2 \dots a_t$, while the set of possible phoneme-sequences (words) will be denoted by W. The task is to find the word $\hat{w} \in W$ defined by

$$\hat{w} = arg \max_{w \in W} P(w|A) = arg \max_{w \in W} \frac{P(A|w) \cdot P(w)}{P(A)} = arg \max_{w \in W} P(A|w) \cdot P(w),$$

where $P(w)$ is known as the *language model.* If we optimize $P(w|A)$ directly, we are using a discriminative method, while if we use Bayes' theorem and omit $P(A)$ as we did in the formula above the approach is generative. The speech recognition process can be frame-based or segment-based, depending on whether the model incorporates frame-based or segment-based features [4].

Frame-based speech recognition. In speech theory a widely-used model is Hidden Markov Modelling (HMM), which is a frame-based generative method. HMM models speech as a collection of states which are connected by transitions. Each state associates an output observation with an output probability, and each transition has an associated transition probability. Here, the feature vectors are the output observations. The output probability models the acoustic constraints, while the transition probability between HMM states models duration constraints. For further details on HMM see [6].

Segment-based speech recognition. In this less commonly used approach we assume for a word $w = o_1 \dots o_l$ that a phoneme o_i is based on $A_i = a_j a_{j+1} \dots a_{j+r-1}$ (an r-long segment of A, where $A = A_1 \dots A_n$). With this A_i segment first long-term features are extracted, then a *phoneme classifier* is used to identify the most probable phoneme covered by the underlying segment. In our framework Artificial Neural Networks (*ANN)* are employed, but the way the classifier actually works is of no concern to us here. We further assume that $P(w|A) = \prod_i P(o_i|A) = \prod_i P(o_i|A_i)$, i.e. that the phonemes are independent. In order to determine $P(o_i|A_i)$, we need to know the exact values of the A_is. This is a hard task, and because automated segmentation cannot be done reliably, the method will make many segment bound hypotheses. So we must include this segmentation S (which determines $A_1 \dots A_n$) in our formulae:

$$P(w|A) = \sum_S P(w, S|A) = \sum_S P(w|S, A) \cdot P(S|A) \approx \max_S P(w|S, A) \cdot P(S|A)$$

For a given S, both $P(w|S, A)$ and $P(S|A)$ can be readily calculated using a classifier. The former is computed via a phoneme classifier, while the latter leads to a two-class classification problem with classes called "phoneme" and "anti-phoneme" [1], [7].

2 The Search through the Hypothesis Space

In this section we will build the hypothesis space by using the segment-based approach, but in a frame-based system an equivalent hypothesis space would appear. In the following instead of a probability p we will use the cost $c = \log p$. The hypothesis space is a subset of a Cartesian product space of two spaces. The first space consists of phoneme sequences, while the second consists of segmentations. An array $T_n = [t_0, t_1, \ldots, t_n]$ is called a *segmentation* if $0 = t_0 < t_1 < \cdots < t_n = t$ holds, which defines n neighboring $[t_i, t_{i+1}]$ intervals. We also require that every phoneme fit into some overlapping interval $[t_i, t_q]$ $(i, q \in \{0, \ldots, n\}, 0 \leq i < q \leq n)$, i.e. the former A_i speech segment here is referred by its start and end times. Given a set of words W, we use $Pref_k(W)$ to denote the k-long prefixes of all the words in W having at least k phonemes. Let $T_n^k = \{[t_{i_0}, t_{i_1}, \ldots, t_{i_k}] : 0 = i_0 < i_1 < \cdots < i_k \leq n\}$ be the set of sub-segmentations over T_n with k connected intervals. Now we will define the hypothesis space recursively by constructing a search tree containing all the hypothesis elements as its nodes. We will denote the root of the tree (the initial hypothesis) by $h_0 = (\emptyset, [t_0])$, and $Pref_1(W) \times T_n^1$ will contain the first-level nodes. For a $(o_1 o_2 \ldots o_j, [t_{i_0}, \ldots, t_{i_j}])$ leaf we link all $(o_1 o_2 \ldots o_j, [t_{i_0}, \ldots, t_{i_j}, t_{i_{j+1}}]) \in Pref_{j+1}(W) \times T_n^{j+1}$ nodes. We are looking for a leaf with the lowest cost.

If a hypothesis is discarded because of its high cost (i.e. we do not scan its descendants), we say that it was *pruned.* A *stack* is a structure for keeping hypotheses in. Moreover, we use limited size stacks in the multi-stack decoding algorithm: if there are too many hypotheses in a stack, we prune the ones with the highest cost. Extending a hypothesis $(o_1 o_2 \ldots o_j, [t_{i_0}, \ldots, t_{i_j}])$ having a cost c with a phoneme v and a time instance $t_{i_{j+1}}$ will result in the hypothesis $(o_1 o_2 \ldots o_j v, [t_{i_0}, \ldots, t_{i_j}, t_{i_{j+1}}])$, which has a cost of c together with the cost of v in the interval $[t_{i_j}, t_{i_{j+1}}]$.

Multi-stack decoding method. In this algorithm we assign a separate stack for each time instance t_i and store the hypotheses in the stack according to their end times. In the first step we place h_0 into the stack associated with the first time instance then, advancing in time, we pop each hypothesis in turn from the given stack, extend them in every possible way, and put the new hypotheses into the stack belonging to their new end time [3]. Algorithm 1 in Appendix shows the pseudocode for multi-stack decoding.

Viterbi beam search. This algorithm differs only in one feature from the multi-stack decoding approach: instead of keeping the n best hypotheses, a variable T called the *beam width* is employed. For each time instance t we calculate D_{min}, i.e. the lowest cost of the hypotheses with the end time t, and prune all hypotheses whose cost D falls outside $D_{min} + T$ [2].

3 How Might Multi-stack Decoding Be Bettered?

When calculating the optimal stack size for multi-stack decoding, it is readily seen that this optimum will be the one with the smallest value where no best-scoring hypothesis is discarded. But this approach obviously has one major drawback: most of the time bad scoring hypotheses will be evaluated owing to the constant stack size. If we could find a way of estimating the required stack size associated with each time instance, the performance of the method would be significantly improved.

i) One possibility is to combine multi-stack decoding with a Viterbi beam search. At each time instance we keep only the n best-scoring hypotheses, and also discard those which are not close to the peak (thus the cost will be higher than $D_{min} + T$). Here the beam width can also be determined empirically.

ii) Another approach is based on the observation that, the later the time instance, the smaller the required stack. We attempted a simple solution for this: the stack size at time t_i will be $s \cdot m^i$, where $0 < m < 1$ and s is the size of the first stack. Of course m should be close to 1, otherwise the stacks would soon be far too small for safe use.

iii) Another technique is a well-known modification of stacks. It can easily happen that there are two or more hypotheses which have the same phoneme-sequence and the same end times (it may be that some earlier phoneme bound is at a different time instance). In this case it is sufficient to retain only the most probable ones. Surprisingly, a simpler version of it worked better: we allowed the stacks to store more of these “same” hypotheses, but when extending a hypothesis in the stack, we popped it and then compared its phoneme sequence with the previously popped one (when there was one). The current hypothesis is extended only when the phoneme sequences differ.

iv) Yet another approach for improving the method comes from the observation that we need big stacks only at those segment bounds where they exactly correspond to phoneme bounds. So if we could estimate at a given time instance what the probability is of this being a bound, we could then reduce the size of the hypothesis space we need to scan. We trained an ANN for this task (on derivative-like features) where its output was treated as a probability p. Then a statistical investigation was carried out to find a function that approximates the necessary stack size based on this p. First, we recognized a set of test words using a standard multi-stack decoding algorithm with a large stack. Then we examined the path which led to the winning hypothesis, and noted the required stack size and the segment bound probability p for each phoneme. The result represented as a stacksize–probability diagram was used to obtain a proper fitting curve estimating the required stack size. It can be readily shown that most of the higher stack sizes are associated with a high value of p, so the stack size can indeed be estimated by this probability. This observation was confirmed by the test results.

v) The last suggested improvement is based on our observation that the minimum size of a stack depends as well on the number of hypotheses the earlier stacks hold. The size of a new stack was estimated by the previous ones using preliminary statistics. But this type of prediction is only of value if other improvements were initially applied and stack-size reduction was achieved beforehand.

Table 1. Summary of the best performances of all the method combinations used.

Method combinations	total ANN calls	avg. ANN calls
Viterbi beam search	*5,469,132*	*17,529.26*
multi-stack decoding	*3,754,456*	*12,033.51*
multi-stack + *i*	2,263,738	7,255.57
multi-stack + *ii*	3,701,035	11,862.29
multi-stack + *iii*	**1,555,747**	**4,986.36**
multi-stack + *iv*	3,283,114	10,522.80
multi-stack + *v*	—	—
multi-stack + *iii* + *i*	**888,445**	**2,847.58**
multi-stack + *iii* + *ii*	1,409,434	4,517.41
multi-stack + *iii* + *iv*	1,280,783	4,105.07
multi-stack + *iii* + *v*	—	—
multi-stack + *iii* + *i* + *ii*	861,253	2,760.42
multi-stack + *iii* + *i* + *iv*	**728,702**	**2,335.58**
multi-stack + *iii* + *i* + *v*	808,635	2,591.77
multi-stack + *iii* + *i* + *iv* + *ii*	722,902	2,316.99
multi-stack + *iii* + *i* + *iv* + *v*	**678,709**	**2,175.34**
multi-stack + *iii* + *i* + *iv* + *v* + *ii*	**677,309**	**2,170.86**

4 Results and Conclusion

We had six speakers who uttered 26 numbers, each one twice, giving a total of 312 occurrences. We expected the methods to score the same results, which they did, although with different parameters even for each grouping. We monitored the methods by the number of hypothesis elements scanned: the lower the number, the better the method. We employed sequential forward selection: first we checked all improvements separately with their best parameters, then we chose the best and checked all the others combined with it, and so on. For comparison we also performed a standard Viterbi beam search test.

It is quite surprising that improvement *iii* produced the best results. After that, improvement *i* (the Viterbi beam search) reduced the search space the most, followed by improvement *iv* (with the segment bound probability). It can be seen that all improvements enhanced the performance of the recognition in every case; the best result was obtained by using all of them together. But we also suggest partial combinations such as combination *iii* + *i* + *iv* + *v* or *iii* + *i* + *iv*, having found that further extensions did not greatly better the performance.

Overall we conclude that, on examining the test results, it is apparent that we can indeed marry the multi-stack decoding and Viterbi beam search methods without any loss of accuracy, and with a marked improvement in performance. Further significant search space reductions were also possible, so our new combination ran some 6 times faster than the multi-stack decoding method, and some 9 times faster than the Viterbi beam search method.

5 Appendix

The multi-stack decoding pseudocode described by Algorithm 1. "$\leftarrow$" means that a variable is assigned a value; "$\Leftarrow$" means pushing a hypothesis into a stack. $Stack[t_i]$ means a stack belonging to the t_i time instance. A $H(t,p,w)$ hypothesis is a triplet of time, cost and a phoneme-sequence. *Extending* a hypothesis $H(t,c,w)$ with a phoneme v and a time t_i results in a hypothesis $H'(t+t_i, c', wv)$, where $c' = c + c_i$, c_i being the cost of v in the interval $[t, t_i]$. We denote the maximal length of a phoneme by *maxlength*.

Algorithm 1. Multi-stack decoding algorithm

```
Stack[$t_0$] $\Leftarrow h_0(t_0, 0, \text{``''})$
for $i = 0 \ldots n$ do
  while not empty(Stack[$t_i$]) do
    $H(t,c,w) \leftarrow$ top(Stack[$t_i$])
    if $t_i = t_max$ then
      return $H$
    end if
    for $l = i+1 \cdots i + maxlength$ do
      for all $\{v \mid wv \in Pref_{1+length\ of\ w}\}$ do
        $H'(t_l, c', w') \leftarrow$ extend $H$ with $v$
        Stack[$t_l$] $\Leftarrow H'$
      end for
    end for
  end while
end for
```

References

1. J. Glass, J. Chang, M. McCandless, *A Probabilistic Framework for Features-Based Speech Recognition,* Proceedings of International Conference on Spoken Language Processing, Philadelphia, PA, pp. 2277-2280, 1996.
2. P.E. Hart, N.J. Nilsson and B. Raphael, *Correction to "A Formal Basis for the Heuristic Determination of Minimum Cost Paths",* SIGART Newsletter, No. 37, pp. 28-29, 1972.
3. X. Huang, A. Acero, H.-W. Hon, *Spoken Language Processing,* Prentice Hall PTR, 2001.
4. F. Jelinek, *Statistical Methods for Speech Recognition,* The MIT Press, 1997.
5. A. Kocsor, L. Tóth and A. Kuba Jr., *An Overview of the Oasis Speech Recognition Project,* Proceedings of ICAI '99, Eger-Noszvaj, Hungary, 1999.
6. L.Rabiner and B.-H. Juang *Fundamentals of Speech Recognition* Prentice Hall, 1993.
7. L. Tóth, A. Kocsor and K. Kovács, A Discriminative Segmental Speech Model and its Application to Hungarian Number Recognition, *Text, Speech and Dialogue, 2000.*