Acta Cybernetica **17** (2005) 213–224.

A Hierarchical Evaluation Methodology in Speech Recognition

Gábor Gosztolya* and András Kocsor*

Abstract

In speech recognition vast hypothesis spaces are generated, so the search methods used and their speedup techniques are both of great importance. One way of getting a speedup gain is to search in multiple steps. In this multi-pass search technique the first steps use only a rough estimate, while the latter steps apply the results of the previous ones. To construct these raw tests we use simplified phoneme groups which are based on some distance function defined over phonemes. The tests we performed show that this technique could significantly speed up the recognition process.

Keywords: speech recognition, search methods, multi-stack decoding, multi-pass search, phoneme grouping.

1 Introduction

Automatic speech recognition (ASR) is a pattern classification problem [1] in which a continuously varying signal has to be mapped to a string of symbols (the phonetic transcription). Besides the identification of speech segments with grammatical phonemes [2], efficient searching in the induced hypothesis space [3] is of great importance as well. This work is related to both areas: first we give a hierarchical scheme of the Hungarian phonemes, then we try to exploit this structure in the search process.

In this paper we want to construct a multi-pass search method where the different steps are based on the selection of the different phoneme groups used. However this construction of the phoneme groups is not trivial, so the choice of the algorithm we use heavily affects the speed and recognition accuracy of the speech recognition system.

The structure of this paper is as follows. First we define the speech recognition problem and the search task. Then we construct a phoneme grouping method based on a distance function between phonemes. Lastly, after presenting and analyzing the test results, we mention some suggestions for future study.

*Research Group on Artificial Intelligence of the Hungarian Academy of Sciences and University of Szeged, H-6720 Szeged, Aradi vértanúk tere 1., Hungary, e-mail: {ggabor,kocsor}@inf.u-szeged.hu

2 Search Spaces in Speech Recognition

In speech recognition problems we have a speech signal represented by a series of observations $A = a_1 a_2 \dots a_t$, and a set of possible phoneme sequences (words or word sequences) which will be denoted by W. Our task is to find the word $\hat{w} \in W$ defined by

$$\hat{w} = arg \max_{w \in W} P(w|A), \tag{1}$$

which, using Bayes' theorem, is equivalent to the following maximization problem:

$$\hat{w} = arg \max_{w \in W} \frac{P(A|w) \cdot P(w)}{P(A)}. \tag{2}$$

Further, noting the fact that $P(A)$ is the same for all $w \in W$, we have that

$$\hat{w} = arg \max_{w \in W} P(A|w)P(w). \tag{3}$$

Speech recognition models can be divided into two types (the discriminative and generative ones), depending on whether they use Eq. (1) or Eq. (3). Throughout this paper we will apply the customary, generative approach [4].

Unified view

Both the generative and discriminative models exploit *frame-based* and/or *segment-based* [5] features, and this fact allows us to have a unified framework of the frame- and segment-based recognition techniques. To make this clearer, we will provide a brief outline of this framework along with the hypothesis structure that will be generated.

Now let us commence with some definitions. Let us define w as $o_1 o_2 \dots o_n$, where o_j is the jth phoneme of word w. Furthermore, let $A_1, A_2, \dots, A_n$ be non-overlapping segments of the observation series $A = a_1 a_2 \dots a_t$, where $A_j = a_{t_{j-1}} \dots a_{t_j}$, $j \in \{1, \dots, n\}$. An A_j segment is defined by its start and end times and will be denoted by $[t_{j-1}, t_j]$. For a segmentation $A = A_1, A_2, \dots, A_n$ we put the values of the time indices corresponding to each segment into a vector $T_n = [t_0, t_1, \dots, t_n]$ $(1 = t_0 < t_1 < \dots < t_n = t)$. We make the conventional assumption that the phonemes in a word are independent so that $P(A|w)$ can be obtained from $P(A_1|o_1), P(A_2|o_2), \dots, P(A_n|o_n)$ in some way. To calculate $P(A|w)$, various aggregation operators can be used at two distinct levels. In the first one the $P(A_j|o_j)$ probability values are supplied by a g_1 operator, i.e.

$$P(A_j|o_j) = g_1([t_{j-1}, t_j], o_j),$$

which provides an overall value for measuring how well the A_j segment represents the o_j phoneme based on local information sources. In the second one, another operator (g_2) is used to construct $P(A|w)$ using the probability values $P(A_1|o_1), \dots, P(A_n|o_n)$.

Frame-based approach

The well-known *Hidden Markov Model* (*HMM*) [6] is basically a frame-based approach, i.e. it handles a speech signal frame by frame. Usually a *Gaussian Mixture Model (GMM)* is applied to compute the $P(a_l|o_j)$ values (for delta and delta-delta features neighboring observations are also required) and for the A_j segment the $g_1([t_{j-1}, t_j], o_j)$ value is defined by

$$\prod_{l=t_{j-1}}^{t_j} c_{o_j} \cdot P(a_{l-k} \dots a_{l+k}|o_j), \tag{4}$$

where $0 \leq c_{o_j} \leq 1$. Practically speaking, g_1 includes all the information we have when we are in a particular state of a HMM model. We note here that, instead of GMM, Artificial Neural Networks (*ANN*s) and other machine learning algorithms that can be used for density estimation are also viable. This alternative provides a way for creating model hybrids. As for the $P(A|w)$ value, the g_2 operator is defined by

$$P(A_n|o_n) \prod_{j=1}^{n-1} (1 - c_{o_j}) P(A_j|o_j). \tag{5}$$

Segment-based approach

In the segment-based speech recognition approach – like the SUMMIT system of MIT [7] or our OASIS [8] – g_1 will usually be the direct output of some machine learning algorithm using features that describe the whole $[t_{j-1}, t_j]$ segment. Among the many possibilities the most conventional choice of g_2 is simply to multiply the probabilities, but in earlier works we showed that using other operators can be beneficial for both the speed and performance [9]. In the following we will stick to multiplication, but the improvements discussed here could also be implemented using other aggregation operators.

The hypothesis space

The task of speech recognition is essentially a selection problem over a Cartesian product space where the first dimension is a set of word hypotheses, while the second is a set of segmentations. 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

$$T^k = \{[t_0, t_1, \dots, t_k] : 1 = t_0 < t_1 < \dots < t_k \leq t\} \tag{6}$$

be the set of sub-segmentations made of k segments over the observation series $a_1 a_2 \dots a_t$. The hypotheses will be object pairs, i.e. they are elements of

$$H = \bigcup_{k=0}^{\infty} (Pref_k(W) \times T^k). \tag{7}$$

We will denote the root of the tree – the initial hypothesis – by $h_0 = (\emptyset, [t_0])$ ($h_0 \in H$). Here $Pref_1(W) \times T^1$ will contain the first-level nodes. For a $(o_1 o_2 \ldots o_j, [t_0, \ldots, t_j])$ leaf we link all $(o_1 o_2 \ldots o_j o_{j+1}, [t_0, \ldots, t_j, t_{j+1}]) \in Pref_{j+1}(W) \times T^{j+1}$ nodes.

Now we need to evaluate the nodes of the search tree. To this end let the g_1 and g_2 functions be defined by some aggregation operators. Then, for a node $(o_1 o_2 \ldots o_j, [t_0, \ldots, t_j])$, the value is defined by

$$g_2(g_1([t_0, t_1], o_1), \ldots, g_1([t_{j-1}, t_j], o_j)). \tag{8}$$

Note that, in practice, it is worth calculating this expression recursively. After defining the evaluation methodology we will look for a leaf with the highest probability.

This definition in typical circumstances leads to a huge hypothesis space, where a full search will be impractical because of the big run time and memory requirements. This leads us to employ heuristics like the well-known Viterbi beam search [10] or our choice, the multi-stack decoding algorithm [11].

3 Clustering the Phoneme Set

In this section we discuss the technique we used to create smaller, more compact phoneme groups. First we define two novel, similar functions between phonemes, prove that they have the right sort of properties to be distance functions, then utilize them in the phoneme-clustering problem.

There is no simple answer to the problem of how we should construct the phoneme groups mentioned above. We might base it on previous grammatical knowledge or use the confusion matrix of the phoneme classifier. The justification for the latter option is that the recognition process is already heavily based on the phoneme classifier.

A classifier gets some set of observations, and its task is to classify this set into one of the $\Omega = \{\omega_1, \omega_2, \ldots, \omega_K\}$ classes. A *confusion matrix* A is constructed in such a way that $a_{i,j}$ is the number of phonemes belonging to ω_j from a selected test set which we classified as ω_is by the classifier [12]. In our case the classifier is used to categorize the parts of speech into one of the phoneme classes. The confusion matrix of a good classifier is close to a diagonal matrix, which is why we will concentrate on the number of misclassified items (i.e. the number of examples that were incorrectly classified).

Grouping phonemes is a standard *clustering problem* [13]: some points (here, the phoneme classes) are to be assigned to a certain number of *clusters* (in our case, phoneme groups). There are some quite general algorithms for this task. The one we are going to use needs a distance function for two clusters, which will be defined below, but first we will explain how this algorithm works.

At the start each phoneme will be considered as different clusters. Then, in each step, we find those C_i and C_j clusters where $\mathcal{D}(C_i, C_j)$ is minimal, and combine

Table 1: An example of a confusion matrix

	1	2	3	4	5	6	7	8	9	10	11
1	2502	3	96	35	4	0	0	4	8	0	12
2	18	965	3	24	3	0	0	0	5	1	49
3	87	8	875	19	2	0	0	5	11	0	18
4	43	11	16	271	1	1	0	1	2	0	12
5	12	2	3	2	2250	257	80	101	53	5	48
6	0	1	0	0	51	299	17	22	8	24	17
7	1	0	0	0	46	31	208	6	1	15	5
8	3	4	3	1	70	39	8	5235	111	19	116
9	7	1	6	3	19	10	2	97	461	2	77
10	1	0	0	0	12	88	25	62	11	830	8
11	39	71	21	31	38	23	19	102	367	18	2316

them. We repeat this until $\mathcal{D}(C_i, C_j) \geq L$, where L is a parameter. (See Appendix A for the pseudocode of this algorithm.)

To define our novel distance functions first let A' be a normalized matrix for the confusion matrix A of the applied phoneme classifier. It takes the form

$$a'_{i,j} = a_{i,j} / \sum_k a_{k,j} \quad i, j \in \{1, \ldots, K\}.$$

We can assume that $\sum_k a_{k,j} \neq 0$, otherwise it would mean that the jth phoneme has no examples in the test database. Next we define a distance function based on this A' matrix. Let

$$d^1_{i,j} = \begin{cases} 0 & \text{if } i = j \\ \infty & \text{if } a'_{i,j} = a'_{j,i} = 0 \text{ and } i \neq j \\ -log(a'_{i,j}) & \text{if } a'_{j,i} = 0 \text{ and } a'_{i,j} \neq 0 \\ -log(a'_{j,i}) & \text{if } a'_{i,j} = 0 \text{ and } a'_{j,i} \neq 0 \\ \min(-log(a'_{i,j}), -log(a'_{j,i})) & \text{otherwise,} \end{cases} \tag{9}$$

and let

$$d^2_{i,j} = \begin{cases} 0 & \text{if } i = j \\ \infty & \text{if } a'_{i,j} = a'_{j,i} = 0 \text{ and } i \neq j \\ -log((a'_{i,j} + a'_{j,i})/2) & \text{otherwise.} \end{cases} \tag{10}$$

Now let D' be the output of some shortest path-finding algorithm with the input of the D^1 or D^2 matrix. (We can choose either of them, but of course if we use both, this choice leads to twice as many test cases. The figures we obtained can be seen in the results section.) D' is a distance function, moreover it satisfies the criteria of being a metric because

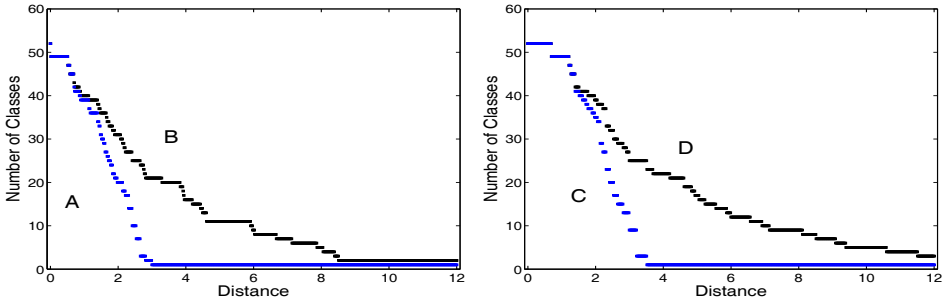

Figure 1: Number of phoneme groups (classes) – L limit diagram for the four distance-variations; d^1 and d^2, respectively. The A and C curves belong to $\mathcal{D}_{min}$, while the B and D curves belong to the $\mathcal{D}_{max}$ group distance function.

- $d'_{i,i} = 0$
- $d'_{i,j} = d'_{j,i}$
- $d'_{i,j} \leq d'_{i,k} + d'_{k,j}$

Now we have to define the distance $\mathcal{D}(C_i, C_j)$ of the clusters C_i and C_j, when we have only the $d'(x_i, y_i)$ values (the distance between different phonemes). To do this we have two straightforward options [13]:

$$\mathcal{D}_{min}(C_i, C_j) = min_{x,y}\{d'(x, y)|x \in C_i, y \in C_j\}, \tag{11}$$

and

$$\mathcal{D}_{max}(C_i, C_j) = max_{x,y}\{d'(x, y)|x \in C_i, y \in C_j\}. \tag{12}$$

The former tends to create longer, larger clusters, while the latter usually creates more compact ones. In our experiments we tested both versions.

We should mention here that the use of $\mathcal{D}_{max}$ in this algorithm could lead to a nondeterministic case if, at any given point, there exist some clusters C_i, C_j and C_k such that $\mathcal{D}_{max}(C_i, C_j) = \mathcal{D}_{max}(C_i, C_k)$. Note here that $\mathcal{D}_{min}$ is not a metric because in some cases the triangle inequality does not hold: there exist C_i, C_j and C_k clusters such that $\mathcal{D}_{min}(C_i, C_j) \nleq \mathcal{D}_{min}(C_i, C_k) + \mathcal{D}_{min}(C_k, C_j)$.

3.1 Tests

Applying the clustering algorithm (using one of the above $\mathcal{D}$ functions) will lead to a series of unions and a series of distance values. Based on them we can choose the possible values of the limit L, which will result in phoneme groups that will be used in the recognition process. Obviously, good L values are those where there is a nice gap between successive distance values in the output.

After examining Figure 1 we identified those bigger flat regions in each curve. For each of them we selected three Ls, resulting in the same number of phoneme

groups, which were later used in the multi-pass recognition method. The corresponding recognition steps were called Pass 1 (p_1), Pass 2 (p_2) and Pass 3 (p_3), with the number of phoneme groups varying from 27 to 34, from 17 to 21 and from 10 to 13, respectively. The default phoneme set was labelled p_0 and had 52 phonemes.

4 The Search Process

Given the phoneme groups – and hence the hypothesis space – we still have to search for the best hypothesis. There are standard search heuristics for this task, from which we chose the multi-stack decoding algorithm. Moreover, there is the possibility of constructing multi-pass methods where there are multiple steps in the search process. Here we decided to apply this idea using the already constructed phoneme groups.

Multi-pass Search Strategies

In general, **multi-pass methods** work in two or more steps: in the first pass the less likely hypotheses are discarded because of some condition requiring low computational time. Then, in the later passes, only the remaining hypotheses are examined by more complex, reliable evaluations, which will approximate the probabilities of the hypotheses more closely. (In the common search methods only the last pass remains, so more hypotheses are scanned there, making the process more time-consuming.)

To speed up the earlier steps, we need to construct faster phoneme classifiers, and the usual way of doing this is to reduce the number of features. (In our system, where ANNs are used, it also leads to a lower number of hidden neurons.) Here the number of phoneme groups was decreased. In the first pass a search with a very restricted phoneme set was performed. Then, in the later passes, more and more detailed phoneme groupings were used, where the dictionary consisted of the 'winning' words of the previous level. Obviously, during the last pass we had to use the original phoneme set to get only one word as a result, not a set of words. At each level we employed the multi-stack decoding algorithm in the search process.

The Multi-stack Search Method

The multi-stack decoding algorithm [11] is one of the heuristic search methods we mentioned earlier, and we chose this one as our basic search technique. To discuss the method first we have to give a definition. A *stack* is a structure for keeping hypotheses in. Moreover, we use limited-sized stacks: if there are too many hypotheses in a stack, we prune the ones with the highest cost.

In this algorithm we assign a separate stack to 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 (the initial hypothesis) 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 associated with their new end times. Algorithm 2 in Appendix B shows the pseudocode for multi-stack decoding.

The multi-stack decoding algorithm has one parameter, the *stack size*. Decreasing it usually decreases the accuracy of the method (i.e. the recognition performance), while a greater stack size leads to increased run times and therefore a slower speech recognition system. Thus it is very important to find the best parameter value, which might mean a trade-off between accuracy and speed. Above a certain parameter setting there is no change in accuracy and this is often what we call the optimal value.

5 Experiments and Results

For testing purposes we used a corpus of 500 children uttering 60 words each, making a total of 30,000 utterances of 2000 different Hungarian words with a variance related to everyday-use occurrence. 24,000 utterances were used for training, while 6,000 words remained for testing purposes – including being the basis for phoneme-group generation. Many of the young speakers had just learned to read and some of them had difficulties with pronunciation, which led to a diverse database. Moreover, many of the words in the database (and thus, in the dictionary) were similar to each other with a phoneme difference of just one or two, which made the recognition quite difficult.

The phoneme recognition rate was 79.47% on the original phoneme set, and it remained around 80% when we applied the restricted phoneme sets. The diversity of the database led to a basic word recognition percentage of 84.14% with the OASIS system, whereas the *HTK* system we used for reference achieved a score of 84.34%. In our tests we expected a word accuracy of at least 80% for a multi-pass configuration.

5.1 Results

The speed of a multi-pass configuration was measured in the number of phoneme classifications averaged for a word. (We found that this was analogous to the actual running speed.) As the ANNs used were of unequal size at each step, the results were normalized to the speed of the phoneme classifier on the last pass. Moreover, for a multi-pass configuration, because at each level there is a multi-stack decoding algorithm used with a different parameter (stack size), there is room for adjusting both the speed and performance. In this case all levels were tested with different parameters, and the best configuration was the one which satisfied the accuracy criterion, and proved to be the fastest among these.

After considering the phoneme-clustering methods, we found we had four possibilities for selecting the phoneme groups and hence selecting the first passes of the multi-pass search method employed. In the table we show all four, providing a way of comparing them exhaustively. "•" means that we applied the given recognition

Table 2: Recognition results

Used passes				d^1 (minimum)		d^2 (average)	
p_0	p_1	p_2	p_3	$\mathcal{D}_{min}$	$\mathcal{D}_{max}$	$\mathcal{D}_{min}$	$\mathcal{D}_{max}$
●	○	○	○	12,578.01	12,578.01	12,578.01	12.578.01
●	●	○	○	6,697.17	6,656.59	4,641.99	5,390.84
●	○	●	○	5,784.14	6,682.20	–	7,831.17
●	○	○	●	–	–	–	**3,713.60**
●	●	●	○	7,727.77	5,477.20	–	7,286.60
●	●	○	●	–	–	–	6,647.32
●	○	●	●	–	–	–	5,062.17
●	●	●	●	–	–	–	6,463.53

pass in the configuration, while "○" denotes that this pass was omitted; a value "–" means that the given configuration could not attain the required recognition accuracy; d^1 and d^2 denote the chosen distance function between phonemes, while $\mathcal{D}_{min}$ and $\mathcal{D}_{max}$ denote the chosen distance function between phoneme groups, respectively.

Examining the results led us to the following observations. First, it is clear that it is possible to speed up a speech recognition system with a multi-pass search method by creating phoneme groups. Because the last pass is always executed using the original phoneme set and thus on the original phoneme classifier, a faster multi-pass search algorithm means that in the earlier passes the list of possible words was drastically reduced; thus the last pass was able to attain a good recognition accuracy even with small-sized stacks. Second, it seems that using $\mathcal{D}_{min}$ in the clustering algorithm leads to a worse result than $\mathcal{D}_{max}$. Third, we noticed that two-pass searches performed better than three- or four-pass configurations. The one-pass configuration was the slowest of the ones we tested when we wanted to achieve an accuracy of at least 80%.

6 Conclusion

In this paper we defined a novel speech recognition method which applied a hierarchical scheme of phoneme-group clusterings. We based this clustering on novel distance functions between phonemes. These functions, which characterized the phoneme set, employed the confusion matrix of the phoneme-classifying neural networks. Then the application of a well-known shortest path-finding algorithm supplied the final distance values, which formed the input for a general clustering algorithm. With this approach using increasingly detailed phoneme structures we were able to create a hierarchical speech recognition method. According to the test results the proposed hierarchical recognition method was able to significantly speed up the speech recognition process by a factor of 3 or 4. Also, our method is insensitive to the type of the phoneme-classifier, so various techniques can be used like C4.5 and GMM, which will be the subject of future work.

References

[1] C.M. Bishop, *Neural Networks for Pattern Recognition,* Clarendon Press, Oxford, 1995.

[2] A. Kocsor, L. Tóth, A. Kuba Jr., K. Kovács, M. Jelasity, T. Gyimóthy and J. Csirik, *A Comparative Study of Several Feature Space Transformation and Learning Methods for Phoneme Classification,* International Journal of Speech Technology, Vol. 3, Number 3/4, pp. 263-276, 2000.

[3] G. Gosztolya and A. Kocsor, *Improving the Multi-Stack Decoding Algorithm in a Segment-based Speech Recognizer,* Proceedings of the 16th International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems, IEA/AIE 2003, LNAI 2718, pp. 744-749, Springer Verlag, 2003.

[4] F. Jelinek, *Statistical Methods for Speech Recognition,* The MIT Press, 1997.

[5] M. Ostendorf, V. Digalakis and O. A. Kimball, *From HMMs to Segment Models: A Unified View of Stochastic Modeling for Speech Recognition,* IEEE Transactions on Acoustics, Speech and Signal Processing, volume 4, pp. 360–378, 1996.

[6] L.Rabiner and B.-H. Juang, *Fundamentals of Speech Recognition* Prentice Hall, 1993.

[7] 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.

[8] A. Kocsor, L. Tóth and A. Kuba Jr., *An Overview of the Oasis Speech Recognition Project,* Proceedings of ICAI '99, Eger-Noszvaj, Hungary, 1999.

[9] G. Gosztolya and A. Kocsor, *Aggregation Operators and Hypothesis Space Reductions in Speech Recognition,* Text, Speech and Dialogue '04, Brno, Czech Republic, 2004.

[10] 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.

[11] L.R. Bahl, P.S. Gopalakrishnan, R.L. Mercer, *Search issues in large vocabulary speech recognition,* Proceedings of the 1993 IEEE Workshop on Automatic Speech Recognition, Snowbird, UT, 1993.

[12] R. O. Duda, P. E. Hart And D. G. Stork, *Pattern Classification,* John Wiley & Sons Inc., 2001.

[13] D. Hand, H. Mannila and P. Smyth, *Principles of Data Mining,* MIT Press, 2001.

Appendix A

The pseudocode of a general clustering algorithm. "$\leftarrow$" means that a variable is assigned a value. The parameters are $x_1, x_2, \ldots, x_n$, the initial points to be clustered (grouped), a $\mathcal{D}(C_i, C_j)$ distance function, and an L value for the stopping criterion.

Algorithm 1 General clustering algorithm

```
for i = 1, ..., n do
  C_i ← {x_i}
end for
while there is more than one cluster left do
  (i, j) ← arg max D(C_i, C_j) is minimal
  if D(C_i, C_j) > L then
    break
  end if
  C_i ← C_i ∪ C_j
  Remove C_j
end while
```

Appendix B

The multi-stack decoding pseudocode described by Algorithm 2. "$\Leftarrow$" means pushing a hypothesis into a stack. $Stack[t_i]$ means a stack belonging to the t_i time instance. A $H(w, T)$ hypothesis denotes a phoneme sequence and time-instance se-

Algorithm 2 Multi-stack decoding algorithm

```
Stack[t_0] ⇐ h_0(∅, [t_0])
for i = 0 ... n do
  while not empty(Stack[t_i]) do
    H(w, T) ← top(Stack[t_i])
    if t_i = t_max then
      return H
    end if
    for t_l = t_{i+1} ··· t_{i+maxlength} do
      for all {v | wv ∈ Pref_{1+length of w}} do
        H'(w', T ∪ t_l) ← extend H with v
        Stack[t_l] ⇐ H'
      end for
    end for
  end while
end for
```

quence pair. *Extending* a hypothesis $H(w, T) = H(w, [t_0, \ldots, t_k])$ with a phoneme v and a time t_i results in a hypothesis $H'(wv, T \cup t_i) = H'(wv, [t_0, \ldots, t_k, t_i])$, where the cost of the new hypothesis is calculated via the g_2 operator, after applying the g_1 function. Here we denote the maximal length of a phoneme by $maxlength$.