Iterative Posterior-Based Keyword Spotting Without Filler Models: Iterative Viterbi Decoding and One-Pass Approach

Marius-Călin Silaghi [1] Hervé Bourlard [1,2]

IDIAP–RR 99-27

January 2000

1 Swiss Federal Institute of Technology at Lausanne (EPFL)
2 Dalle Molle Institute for Perceptual Artificial Intelligence (IDIAP), Martigny, Switzerland

IDIAP Research Report 99-27

ITERATIVE POSTERIOR-BASED KEYWORD SPOTTING WITHOUT FILLER MODELS: ITERATIVE VITERBI DECODING AND ONE-PASS APPROACH

Marius-Călin Silaghi Hervé Bourlard

JANUARY 2000

Abstract. This paper addresses the problem of detecting keywords in unconstrained speech without explicit modeling of non-keyword segments. The proposed algorithm is based on recent developments in confidence measures based on local a posterior probabilities, together with the (known) approach of relaxing the begin/endpoints in a *Dynamic Programming (DP)* matching of partial hypothesized sub-sequences on a specific (keyword) hidden Markov model (HMM), followed by a time normalization of the resulting scores[1]. In this case, straightforward DP is no longer optimal and various algorithms were devised, usually requiring more computation and more memory. In this paper, we present an alternative, quite simple and efficient, approach to this problem, which can also be easily generalized to more complex matching scores. The proposed algorithm is based on (1) the average observation posterior along the most likely state sequence, and (2) an iterative Viterbi decoding algorithm, which does not require scoring for all begin/endpoints and which can be proved to converge (quickly) to the "optimal" solution without requiring any specific filler models. Results obtained with this method on 100 keywords chosen at random from the BREF database [5] are reported. Finally, we also develop a new pruning strategy of a one-pass approach that can optimize the same criterion, and yields the same performance than the iterative Viterbi approach. This report is an extension of [8], where a detailed convergence proof of the Iterative Viterbi Decoder is provided, together with an extension to the one-pass decoder.

[1]Dividing the resulting scores by the length of the matched sub-sequences. This approach, sometimes referred to as template-matching word spotters or sliding model method, was initially proposed in DTW-based approaches [4], and then generalized to HMM-based systems [11].

1 Introduction

This paper addresses the problem of *keyword spotting (KWS)* in unconstrained speech without explicit modeling of non-keyword segments (typically done by using filler HMM models or an ergodic HMM composed of context dependent or independent phone models without lexical constraints). Although several algorithms [1] tackling this type of problem have already been proposed in the past, e.g., by using Dynamic Time Warping (DTW) [4] or Viterbi matching [11] allowing relaxation of the (begin and endpoint) constraints, these are known to require the use of an “appropriate” normalization of the matching scores since segments of different lengths have then to be compared. However, given this normalization and the relaxation of begin/endpoints, straightforward DP is no longer optimal (or, in other words, the DP optimality principle is no longer valid) and has to be adapted, involving more memory and CPU. Indeed, at any possible ending time e, the match score of the best warp and start time b of the reference has to be computed [4] (for all possible start times b associated with unpruned paths). Moreover, in [11], and in the same spirit than what is presented here, for all possible ending times e, the average observation likelihood along the most likely state sequence is used as scoring criterion. Finally, this adapted DP quickly becomes even more complex (or intractable) for more advanced scoring criteria (such as the confidence measures mentioned below).

More recently, work in the field of confidence level, and in the framework of hybrid HMM/ANN systems, it was shown [1] that the use of accumulated local posterior probabilities (as obtained at the output of a multilayer perceptron) normalized by the length of the word segment (or, better, involving a double normalization over the number of phones and the number of acoustic frames in each phone) was yielding good confidence measures and good scores for the re-estimation of N-best hypotheses. Similar work, where this kind of confidence measure was compared to several alternative approaches, was reported in [10] and confirmed this conclusion. However, so far, the evaluation of such confidence measures involved the estimation and rescoring of N-best hypotheses. Similar work and conclusions (also using N-best rescoring) were also reported in using likelihood ratio rescoring and non-keyword rejection [9].

In this paper, we will use a similar scoring technique for keyword spotting without explicit filler model. Compared to previously devised “sliding model” methods (such as [4, 11]), the algorithm proposed here is based on:

1. A matching score defined as the average observation posterior along the most likely state sequence. It is indeed believed that local posteriors (or likelihood ratios, as in [9]) are more appropriate to the task.
2. The iteration of a Viterbi decoding algorithm, which does not require scoring for all begin/endpoints or N-best rescoring, and which can be proved to (quickly) converge to the “optimal” [2] solution without requiring any specific filler models, using straightforward Viterbi alignments (similar to regular filler-based KWS, but at the cost of a few iterations).

2 KWS without filler models

Let $X = \{x_1, x_2, \ldots, x_n, \ldots, x_N\}$ denote the sequence of acoustic vectors in which we want to detect a keyword, and let M be the HMM model of a keyword M and consisting of L states $\mathcal{Q} = \{q_1, q_2, \ldots, q_\ell, \ldots, q_L\}$. Assuming that M is matched to a subsequence $X_b^e = \{x_b, \ldots, x_e\}$ ($1 \leq b \leq e \leq N$) of X, and that we have an implicit (not modeled) *garbage/filler state* q_G preceding and following M[3], we define (approximate) the log posterior of a model M given a subsequence X_b^e as the average posterior probability along the optimal path, i.e.:

[1] Sometimes referred to as “sliding model methods”.

[2] From the point of view of the chosen scoring functions.

[3] Thus implicitly introducing the grammatical constraint that we have only one keyword, preceded and followed by a non-keyword segment.

$$\begin{aligned}
-\log P(M|X_b^e) &\simeq \frac{1}{e-b+1}\min_{\forall Q\in M} -\log P(Q|X_b^e) \\
&\simeq \frac{1}{e-b+1}\min_{\forall Q\in M} \{-\log P(q^b|q_G) \\
&\quad -\sum_{n=b}^{e-1}[\log P(q^n|x_n)+\log P(q^{n+1}|q^n)] - \log P(q^e|x_e) - \log P(q_G|q^e)\} \quad (1)
\end{aligned}$$

where $Q = \{q^b, q^{b+1}, ..., q^e\}$ represents one of the possible paths of length $(e-b+1)$ in M, and q^n the HMM state visited at time n along Q, with $q^n \in \mathcal{Q}$. In this expression, q_G represents the "garbage" (filler) state which is simply used here as the non-emitting initial and final state of M. Transition probabilities $P(q^b|q_G)$ and $P(q_G|q^e)$ can be interpreted as the keyword entrance and exit penalties, as optimized in [3], but these have not been optimized here. In our case, local posteriors $P(q_\ell|x_n)$ were estimated as output values of a multilayer perceptron (MLP) used in a hybrid HMM/ANN system [2].

For a specific sub-sequence X_b^e, expression (1) can easily be estimated by dynamic programming since the sub-sequence and the associated normalizing factor $(e-b+1)$ are given. However, in the case of keyword spotting, this expression should be estimated for all possible begin/endpoint pairs $\{b,e\}$ (as well as for all possible word models), and we define the matching score of X on M as:

$$S(M|X) = -\log P(M|X_{b^*}^{e^*}) \quad (2)$$

where the optimal begin/endpoints $\{b^*, e^*\}$, and the associated optimal path Q^*, are the ones yielding the lowest average local posterior:

$$\langle Q^*, b^*, e^*\rangle = \underset{\{Q,b,e\}}{\text{argmin}} \; \frac{-1}{e-b+1}\log P(Q|X_b^e) \quad (3)$$

Of course, in the case of several keywords, all possible models will have to be evaluated.

As shown in [1, 10], a double averaging involving the number of frames per phone and the number of phones will usually yield slightly better performance:

$$\langle Q^*, b^*, e^*\rangle = \underset{\{Q,b,e\}}{\text{argmin}} \frac{-1}{J}\sum_{j=1}^{J}\left(\frac{1}{e_j-b_j+1}\sum_{n=b_j}^{e_j}\log P(q_j^n|x_n)\right) \quad (4)$$

where J represents the number of phones in the hypothesized keyword model and q_j^n the hypothesized phone q_j for input frame x_n.

However, given the time normalization and the relaxation of begin/endpoints, straightforward DP is no longer optimal and has to be adapted, usually involving more memory and CPU. A new (and simple) solution to this problem will be proposed in Section 4.

3 Filler-based KWS

Although various solutions have been proposed towards the direct optimization of (2) as, e.g., in [4, 11], most of the keyword spotting approaches today prefer to preserve the optimality and simplicity of Viterbi DP by modeling the complete input [6] and explicitly [7] or implicitly [3] modeling non-keyword segments by using so called filler or garbage models as additional reference models. In this case, we assume that non-keyword segments are modeled by extraneous garbage models/states q_G (and grammatical constraints ruling the possible keyword/non-keyword sequences).

In this paper, we will consider only the case of detecting one keyword per utterance at a time. In this case, the keyword spotting problem amounts at matching the whole sequence X of length N onto an extended HMM model $\overline{M}$ consisting of the states $\{q_G, q_1, \ldots, q_L, q_G\}$, in which a path (of length N) is denoted

$\overline{Q} = \{\overbrace{q_G, ...q_G}^{b-1}, q^b, q^{b+1}, ..., q^e, \overbrace{q_G, ...q_G}^{N-e}\}$ with $(b-1)$ garbage states q_G preceding q^b and $(N-e)$ states q_G following q^e, and respectively emitting the vector sequences X_1^{b-1} and X_{e+1}^N associated with the non-keyword segments.

Given some estimation of $P(q_G|x_n)$ (e.g., using probability density functions trained on non keyword utterances), the optimal path $\overline{Q}^*$ (and, consequently b^* and e^*) is then given by:

$$\begin{aligned}\overline{Q^*} &= \underset{\forall \overline{Q} \in \overline{M}}{\text{argmin}} - \log P(\overline{Q}|X) \\ &= \underset{\forall \overline{Q} \in \overline{M}}{\text{argmin}} \{- \log P(Q|X_b^e) - \sum_{n=1}^{b-1} \log P(q_G|x_n) - \sum_{n=e+1}^{N} \log P(q_G|x_n)\} \qquad (5)\end{aligned}$$

which can be solved by straightforward DP (since all paths have the same length). The main problem of filler-based keyword spotting approaches is then to find ways to best estimate $P(q_G|x_n)$ in order to minimize the error introduced by the approximations. In [3], this value was defined as the average of the N best local scores while, in other approaches, this value is generated from explicit filler HMMs. However, these approaches will usually not lead to the "optimal" solution given by (2).

4 Iterating Viterbi Decoding (IVD)[6]

In the following, we show that it is possible to define an iterative process, referred to as *Iterating Viterbi Decoding (IVD)*[4] with good/fast convergence properties, estimating the value of $P(q_G|x_n)$ such that straightforward DP (5) yields exactly the same segmentation (and recognition results) than (3). While the same result could be achieved through a modified DP in which all possible combinations (all possible begin/endpoints) would be taken into account, it is possible to show that the algorithm proposed below is more efficient (in terms of both CPU and memory requirements).

4.1 IVD: General description

The IVD algorithm is based on the same criterion than the filler based approaches (5), but rather than looking for explicit (and empirical) estimates of $P(q_G|x_n)$ we aim at mathematically estimating its value (which will be different and adapted to each utterance) such that solving (5) is equivalent to solving (3). Thus, we perform an iterative estimation of $P(q_G|x_n)$, such that the segmentation resulting of (5) is the same than what would be obtained from (3).

Defining $\varepsilon = -\log P(q_G|x_n)$, the proposed algorithm can be summarized as follows:

1. Start from an initial value $\varepsilon_0 = \varepsilon$[5], (e.g., with ε equal with a cheap estimation of the score of a "match"). In the experiments reported below, ε was initialized to $-\log$ of the maximum of the local probabilities $P(q_k|x_n)$ for each frame x_n.

   An alternative choice could be to initialize ε_0 to a pre-defined score that expression (1) should reach to declare a keyword "matching" (see point 4 below). In this last case, if ε increases at the first iteration, then we can (as proven) directly infer that the match will be rejected, otherwise it will be accepted.

2. Given the current estimate ε_t of $P(q_G|x_n)$ at iteration t, find the optimal path $\langle \overline{Q}_t, b_t, e_t \rangle$ according to (5) and matching the complete input.

3. Update ($t = t + 1$) the estimated value of ε_t, defined as the average of the local posteriors along the optimal path Q_t (matching the $X_{b_t}^{e_t}$ resulting of (5) on the keyword model) i.e.:

$$\varepsilon_{t+1} = -\frac{1}{(e_t - b_t + 1)} \log P(Q_t|X_{b_t}^{e_t}) \qquad (6)$$

[4]Patent pending.

[5]It is actually proven that the iterative process presented here will always converge to the same solution (in more or less cycles, with the worst case upper bound of N iterations) independently of this initialization.

4. Return to (2) and iterate until convergence. If we are not interested in the optimal segmentation, this process could also be stopped as soon as ε reaches a (pre-defined) minimum threshold below which we can declare that a keyword has been detected.

Convergence proof of this process and generalization to other criteria, are given in section 4.2: each IVD iteration (from the second iteration) will decrease the value of ε_t, and the final path yields the same solution than (3).

4.2 IVD: Convergence proof

The proof of all lemmas and theorems below are given in the Appendix.

Proposition 1 *Obviously, a path $\overline{Q}$ returned by the Viterbi algorithm (5) for X and $\overline{M}$ is completely determined by b and e, independently of the value of $-log(P(q_G|x_n))$ (i.e. ε).*

Lemma 1 *If $\varepsilon = S(M|X)$ given by (2), then solving (5) yields $\langle Q^*, b^*, e^* \rangle$ given by (3).*

We recall that $\forall\varepsilon$, if the resulting Viterbi path over X and $\overline{M}$ is $\overline{Q_b^e}$, then

$$-\log P(M|X_b^e) \geq -\log P(M|X_{b^*}^{e^*}). \tag{7}$$

Equality appears when the path Q^* is not unique. In that case, it is the implementation that decides between equivalent solutions.

Lemma 2 *If $\varepsilon > S(M|X)$ given by (2), then the path and corresponding begin and endpoints $\langle \overline{Q}, b, e \rangle$ resulting of (5) are such that:*

$$e^* - b^* \leq e - b \tag{8}$$

Lemma 3 *If $\varepsilon > S(M|X)$, then the path $\langle \overline{Q}, b, e \rangle$ resulting of (5) corresponds to: $\overline{Q_b^e}$, then:*

$$-\log P(M|X_b^e) < \varepsilon.$$

Theorem 1 *The IVD algorithm described in Section 4.1 will converge to $\langle Q^*, b^*, e^* \rangle$ defined in (3).*

In the sequel of this section we give an upper bound of the number of steps needed for convergence. We denote the path computed by (5) for a certain value of ε by $\overline{Q_\varepsilon}$. Similarly to Lemma 2 we then have:

Lemma 4 *If $\varepsilon_1 > \varepsilon_2$, then either the sequence Q_{ε_1} is longer than Q_{ε_2}, or else we have reached convergence, i.e.:*

$$e_2 - b_2 \leq e_1 - b_1 \tag{9}$$

From the previous lemma and from Lemma 3, we note that at each step before convergence, the length of Q_ε decreases. The number of steps is therefore roughly upper bounded by N which is the same with the number of DP runs needed in order to compute the matches for all the subsequences of X. The cost of running $DP(\varepsilon)$ is similar with the one of running DP over the whole X. Running N times $DP(\varepsilon)$ it therefore about two times more expensive than solving the problem with DP over all subsequences of X. However, the number of steps needed for convergence can be much smaller than N.

4.3 Generalization of IVD

IVD thus provides us with a way to transform the computation of a matching score over all possible subintervals of X into the computation of that score several times but only for the whole sequence. Therefore the technique will be useful whenever the computation on a fixed sequence of X is sensibly cheaper than the computation for all pairs of begin and end points.

IVD consists in iteratively substituting the emission probability of the garbage model with the estimated score obtained for the nongarbage section at the previous step where the global cost was optimized, up to convergence. A sufficient set of conditions that have lead to the validity of IVD were:

- A fixed point for the iterative process appears when the emission probability of the garbage equals the optimal matching score that is looked for.
- When the $-log$ emission probability of the garbage, ε, is bigger than the optimal matching score, then the obtained estimation of ε is lower than the initial one.
- The obtained estimation of ε cannot be lower than the optimal score.

The first condition is typical of the averaging operation and is expected for many score functions based on it. The next requirements are equivalent to any other convergence processes. The second one is fulfilled if the score is a linear combination of ε and of the score on the nongarbage section. In that case, the optimal path will offer an estimation to be chosen against all the matches with a nongarbage score above or equal to ε.

Let us now analyze, as example, the score function presented in 4. This function is a combination of averaging operations. When $\overline{M}$ is analyzed, the garbage is treated as two begin and end phonemes. The emission probability in our model is constant along them and their contribution in the global average is equal with ε. As described above, when ε equals the optimal average on the nongarbage phonemes, that optimal alternative will now minimize the global cost and is chosen leading to a fix point. The second condition is similarly fulfilled. The last condition is again insured by the definition. We mention however that for this function, the optimal path is chosen at the first step for whatever value of ε since the contribution of the garbage is fixed to ε for any path in the HMM $\overline{M}$. The problem with this particular scoring function lies in the fact that cost of the associated estimation is not so cheap as with IVD while a beam search is needed. In the next sections we will present some ways of improving the related beam search.

5 One-pass keyword spotting

5.1 General Description

The above algorithm has a very good experimental convergence speed. However, we cannot guarantee the convergence speed of the process. For this reason, a one step computation is potentially interesting. In the next subsection we show that the standard DP cannot be used for solving the equation (3).

5.2 The Principle of Optimality

Let us define $T(\overline{M}, X)$ as the DP table of emission probabilities for an utterance X and the states of the hypothesized word W. When solving by standard DP, we would compute for each entry of the table $T(\overline{M}, X)$ at frame k of X and state s of $\overline{M}$ three values: S_{ks}, L_{ks} and C_{ks}, where S_{ks} corresponds to the sum of the posteriors on the optimal path that leads to the entry, L_{ks} holds the length of the optimal path computed so far, and C_{ks} is the estimation of the cost on the optimal expanded path.

By a path leading to an entry $T(k, s)$ we mean a sequence of entries in the table T, such that there is exactly an entry for each time frame $t{\le}k$. At each entry $T(k, s)$, DP selects a locally optimal path noted P_{ks}.

At each step k, we consider all pairs of entries of table $T(\overline{M}, X)$ of type $T(k, s)$, $T(k-1, t)$. We update for each such pair, the current cost C_{ks} (initially $\inf$) as:

$$
\begin{aligned}
S_{ks} &= S_{(k-1)t} - \log p(s|x_k)p(s|t) \\
L_{ks} &= L_{(k-1)t} + 1, \forall t > 0, t{\le}L \\
C_{ks} &= \frac{S_k}{L_k}
\end{aligned}
\qquad (10)
$$

wanting to have at step k the path P_{ks} from the paths $P_{(k-1)t}$ that minimizes C_{NL}. With DP, one will choose the P_{ks} with minimal C_{ks}.

In order for the previous computation to be correct, the optimality principle needs to be respected. The optimality principle of Dynamic Programming requires that the path to the frame $k-1$ that minimizes C_{NL}, also minimizes C_{ks} for an entry at frame k of table $T(\overline{M}, X)$.

We note that:

Lemma 5 *The expression 10 does not respect the optimality principle of Dynamic Programming*

5.3 Punning with beam search[6]

The Dynamic Programming can be viewed as a set of safe prunings that are applied at each entry of the DP table and has the property that only one alternative is maintained. We have thus shown that Dynamic Programming cannot be used, since the principle of optimality is not respected. We try therefore to detect the type of safe pruning that can be done.

Lemma 6 *If at a frame* a *we have two paths* P'_a *and* P''_a *with* $S''_a < S'_a$ *and* $L'_a < L''_a$*, then at no frame* $c{\geq}a$ *will a path* P''_c *be forsaken for a path* P'_c *if* $P'_a{\subset}P'_c$*,* $P''_a{\subset}P''_c$ *and* $P'_c{\setminus}P'_a{\equiv}P''_c{\setminus}P''_a$*. We will note the order relation as* $P''_a{\prec}P'_a$*.*

Lemma 7 *A path P' may be discarded only for a lower cost one, P".*

$$P'{\prec}P'' \Rightarrow C'_k < C''_k$$

Theorem 2 *Algorithm 1 computes* $S(M,X)$ *and* Q^* *from equation (3).*

```
    procedure OneStep(W,X)
        SetOfPaths(1..N, 1..K)←∅
        for all frame=1; $frame <= N$; frame++ do
            for all state=1; $state <= K$; state++ do
                for all candidate $p_i$∈SetOfPaths(frame-1, 1..K) do
                    Add($p_i$, SetOfPaths[frame, state])
                end
            end
        end
        SetOfPaths[frame, K] ← best of the candidates
    end.
    procedure Add(path, set-of-paths)
        for all $p_i$∈set-of-paths do
1.1         if $path{\prec}p_i$ then
                delete $p_i$
            end
1.2         if $p_i{\prec}path$ then
                return
            end
        end
        Insert $p_i$ in set-of-paths
    end.
```

Algorithm 1: *One Step Algorithm*

By ordering the set of paths, according to Lemma 7, we only need to check the line 1.2 of algorithm 1 up to the eventually insertion place. The last paths are candidates for pruning in line 1.1. In order for the pruning to be acceptable, we will prune only paths that were too long on the last state. An additional counter is needed for storing the state length. This counter is reset when the state is changed and is incremented at each advance with a frame.

6 One pass confidence-based keyword spotting

6.1 The Method of Double Normalization[6]

The corresponding confidence measure is defined as:

$$\frac{1}{NVP}\sum_{p_i\in VP}\frac{\sum_{pst\in p_i}-\log(pst)}{length(p_i)} \tag{11}$$

where NVP stands for the *number of visited phonemes* and VP stands for the *set of visited phonemes*. An average is computed over all posteriors *pst* of the emission probabilities for the time frames matched to the visited phonem p_i. The function $length(p_i)$ gives the number of time frames matched against p_i.

This method consists into a breath first Beam Search algorithm. It refers to a set of reduction rules and certain normalizations:

For the state q_G, in this method, the logarithm of the emission posterior is equal with zero. For each frame e and for each state s, the set of paths/probabilities of having the frame e in the state s is computed as the first N maxima of the confidence measure for all paths in HMM $\overline{M}$ of length e and ending in the state s. The paths that according to the reduction rules will loose the final race when compared with another already known path, will be deleted as well.

We note a_1, p_1, l_1, a_2, p_2 and l_2 the confidence measure for the previous phonemes, the posterior in the current phoneme and the length in the current phoneme for the path Q_1, respectively the path Q_2. The rules that may be used for the reduction of the search space by discarding a path Q_1 for a path Q_2 are in this case any of the next ones:

1. $l_2 \geq l_1$, $A > 0, B \leq 0$ and $L_c^2 A + L_c B + C \geq 0$
2. $l_2 \geq l_1$, $A \geq 0, B \geq 0$ and $C \geq 0$
3. $l_2 \geq l_1$, $A \leq 0, C \geq 0$ and $L^2 A + LB + C \geq 0$
4. $l_2 \geq l_1$, $A = 0, B < 0$ and $LB + C \geq 0$

where $A = a_1 - a_2$, $B = (a_1 - a_2)(l_1 + l_2) + p_1 - p_2$, C=$(a_1 - a_2)l_1 l_2 + p_1 l_2 - p_2 l_1$, $L = L_{max} - \max\{l_1, l_2\}$, $L_c = -B/2A \geq 0$ while L_{max} is the maximum acceptable length for a phoneme.

By discarding paths only if one of the above rules is satisfied, the optimum defined by the confidence measure with double normalization can be guaranteed, if no phone may be avoided by the HMM M. Any HMM may be decomposed in HMMs with this quality. The 4-th rule is included in the 3-rd and its test is useless if the last one was already checked.

First test, $l_2 \geq l_1$ tells us if Q_2 has chances to eliminate Q_1, otherwise we will check if Q_1 eliminates Q_2. These tests were inferred from the conditions of maintaining the final maximal confidence measure while reduction takes place. In order to use the method of double normalization without decomposing HMMs that skip some phonemes, the previous rules are modified taking into account the number of visited phonemes for any path F_1 respectively F_2 and the number of phonemes that may follow the current state.

A simplified test may be:

- $l_2 \geq l_1$, $A \geq 0, p_1 \geq p_2$ respectively $F_2 \geq F_1$ for the HMMs that skips phonemes.

This test is weaker than the 2^{nd} reduction rule. For example a path is eliminated by a second path if the first one has an inferior confidence measure (higher in value) for the the previous phonemes, a shorter length and the minus of the logarithm of the cumulated posterior in the current phoneme also inferior (higher in value) to that of the second one.

An additional confidence measure based on the maximal length, L_{max}, and on the maximum of the minus of the logarithm of the cumulated and normalized posterior in phoneme, P_{max}, is used in order to limit the number of stored paths.

- $p > L_{max}P_{max}$ in any state
- $\frac{p}{l} > P_{max}$ at the output from a phoneme

where p and l are the values in the current phoneme for the minus of the logarithm of cumulated posterior and for the length of the path that is discarded. These tests allow for the elimination of the paths that are too long without being outstanding, respectively of the paths with phonemes having unacceptable scores, otherwise compensated by very good scores in other phonemes.

If N is chosen equal with one, the aforementioned rules are no longer needed, but always we propagate the path with the maximal current estimation of the confidence measure. The obtained results are very good, even if the defined optimum is guaranteed for this method only when N is bigger than the length of the sequence allowed by L_{max} or of the tested sequence.

The same approach is valid for the simple normalization, where the HMM for the searched word will be grouped into a single phoneme.

7 Experimental results

Preliminary tests of the IVD algorithm were performed on the BREF database [5], a continuous, read speech microphone database. As done in [1], $3,736$ utterances were used for training an artificial neural network (multilayer perceptron) to generate local (context-independent) phone posterior probabilities. 242 utterances (with a $2,300$ word lexicon), from which *100 keywords* were selected at random, were used for testing. These keywords were simply represented by simple hybrid HMM/ANN models [2] based on context-independent phones.

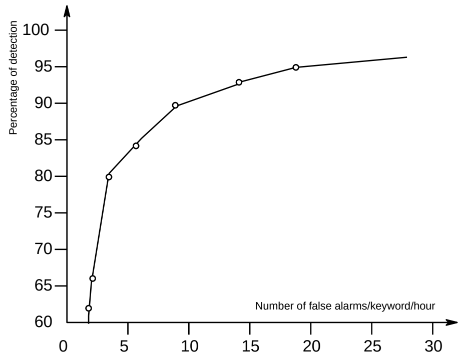

Figure 1: *ROC of the IVD-based keyword detection based on (2) as a function of number of false alarms/keyword/hour, as obtained on 242 BREF test sentences and 100 keywords selected at random.*

The resulting ROC (Receiver Operating Characteristics) curve, using IVD to estimate (2), is presented in Figure 1 and shows good performance compared to similar experiments [3], although no parameters (such as keyword entrance penalties) were tuned to optimize performance. For computing the segmentation, 3 to 5 iterations were needed. If the segmentation is not needed, the “matching” decision can be taken with only one iteration as described in the initialization step of the algorithm.

For comparison, the ROC curve obtained (for the same keywords and test sentences) with criterion (4), involving a double normalization, is reported in Figure 2. As also reported for confidence measure rescoring [1], this measures is yielding even better KWS performance.

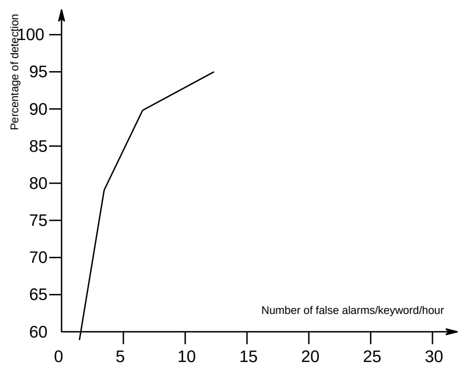

Figure 2: *ROC using criterion (4) (double normalization), on 242 BREF test sentences containing 100 keywords selected at random.*

8 Conclusions

In this paper, we have thus proposed a new method for keyword spotting, based on recent advances in confidence measures, using local posterior probabilities, but without requiring the explicit use of filler models.

A new algorithm, referred to as *Iterating Viterbi Decoding (IVD)*, to solve the above optimization problem with a simple DP process (not requiring to store pointers and scores for all possible ending and start times), at the cost of a few iterations.

While the proposed approach allows for an easy generalization to more complex criteria, preliminary results obtained on the basis of 100 keywords (and without any specific tuning) appear to be particularly competitive to other alternative approaches.

9 Acknowledgments

We thank Giulia Bernardis for helping us with the BREF database and providing us with a neural network trained for this task. We also acknowledge the useful discussions with Prof. Boi Faltings.

References

[1] Bernardis, G. and Bourlard, H., "Improving posterior-based confidence measures in hybrid HMM/ANN speech recognition systems," *Proceedings of Intl. Conf. on Spoken Language Processing* (Sydney, Australia), pp. 775-778, 1998.

[2] Bourlard, H. and Morgan, N., *Connectionist Speech Recognition - A Hybrid Approach*, Kluwer Academic Publishers, 1994.

[3] Bourlard, H., D'hoore, B., and Boite, J.-M., "Optimizing recognition and rejection performance in wordspotting systems," *Proc. of IEEE Intl. Conf. on Acoustics, Speech, and Signal Processing* (Adelaide, Australia), pp. I:373-376, 1994.

[4] Bridle, J.S., "An efficient elastic-template method for detecting given words in running speech," *Proc. of the Brit. Acoust. Soc. Meeting*, pp. 1-4, April 1973.

[5] Lamel, L.F., Gauvain, J-L., and Eskénazi, M., "BREF, a large vocabulary spoken corpus for French," *Proceedings of Eurospeech'91*, pp. 505-508, 1991.

[6] Rohlicek, J.R., “Word spotting,” in *Modern Methods of Speech Processing*, R.P. Ramachandran and R. Mammone (Eds.), Kluwer Academics Publishers, pp. 123-157, 1995.

[7] Rose, R.C. and Paul, D.B., “A hidden Markov model based keyword recognition system,” *Proc. of ICASSP’90*, pp. 129-132, 1990.

[8] Silaghi, M.-C., Bourlard, H., “Iterative Posterior-Based Keyword Spotting Without Filler Models,” *Proceedings of the IEEE Automatic Speech Recognition and Understanding (ASRU’99) Workshop*, Keystone, Colorado, Dec. 12-15, 1999.

[9] Sukkar, R.A. and Lee, C.-H., “Vocabulary independent discriminative utterance verification for nonkeyword rejection in subword based speech recognition,” *IEEE Trans. on Speech and Audio Processing*, vol. 4, no. 6, pp. 420-429, 1996.

[10] Williams, G. and Renals, S., “Confidence measures for hybrid HMM/ANN speech recognition,” *Proceedings of Eurospeech’97*, pp. 1955-1958, 1997.

[11] Wilpon, J.G., Rabiner, L.R., Lee C.-H., and Goldman, E.R., “Application of hidden Markov models of keywords in unconstrained speech,” *Proc. of ICASSP’89*, pp. 254-257, 1989.

Appendix

Lemma 1 *If $\varepsilon = S(M|X)$ given by (2), then solving (5) yields $\langle Q^*, b^*, e^* \rangle$ given by (3).*

Proof. Let we note $w = S(M|X)$. We see that choosing $\overline{Q^*}$, $-\log P(\overline{Q^*}|X) = N * w$. Another path $\overline{Q}$ would have yielded $-\log P(\overline{Q}|X) = N * w + (e - b + 1) * (w' - w) > N * w$ while $w' > w$ from the definition of Q^* (if unique). Here we have noted by b, e, w' the values of the first frame, last frame respectively average probability in whatever path $\overline{Q_b^e}$ where Q_b^e is an alternative to Q^*.

$$w' = -\log P(M|X_b^e) = \frac{1}{e-b+1} - \log P(\overline{Q}|X_b^e)$$

While Viterbi yields the optimum (minimum), it chooses $\overline{Q^*}$. □

Lemma 2 *If $\varepsilon > S(M|X)$ given by (2), then the path and corresponding begin and endpoints $\langle \overline{Q}, b, e \rangle$ resulting of (5) are such that:*

$$e^* - b^* \leq e - b \tag{12}$$

Proof. We use the notations from the previous proof:

$$-\log P(\overline{Q^*}|X) = (b^* + N - 1 - e^*) * \varepsilon + (e^* - b^* + 1) * w$$

We demonstrate that the path Q^* is better than any shorter one Q_b^e:

$$-\log P(\overline{Q_b^e}|X) = (b + N - 1 - e) * \varepsilon + (e - b + 1) * w'$$

If we assume that Q_b^e is chosen with

$$e^* - b^* > e - b.$$

$((e^* - b^*) - (e - b)) > 0$. $w' \geq w$ from the definition of Q^*. $\varepsilon > w$ from hypothesis. Therefore, $\forall\, e^* - b^* > e - b$:

$$(e^* - b^* + 1) * w < ((e^* - b^*) - (e - b)) * \varepsilon + (e - b + 1) * w'$$

We get: $(b^* + N - 1 - e^*) * \varepsilon + (e^* - b^* + 1) * w < (b + N - 1 - e) * \varepsilon + (e - b + 1) * w'$, showing that such a path $\overline{Q_b^e}$ would not have been preferred to $\overline{Q^*}$.

$\Rightarrow$ The assumption was wrong and the chosen $\overline{Q_b^e}$ will have $e^* - b^* \leq e - b$. □

Lemma 3 *If $\varepsilon > S(M|X)$, then the path $\langle \overline{Q}, b, e \rangle$ resulting of (5) corresponds to: $\overline{Q_b^e}$, then:*

$$-\log P(M|X_b^e) < \varepsilon.$$

Proof.

$$-\log P(\overline{Q_b^e}|X) = (N - 1 - e + b) * \varepsilon + (e - b + 1) * w'$$

If the $\overline{Q^*}$ sequence was not preferred then:

$$-\log P(\overline{Q_b^e}|X) \leq -\log P(\overline{Q^*}|X)$$

This can be written as: $(N - 1 - e + b) * \varepsilon + (e - b + 1) * w' \leq (N - 1 - e^* + b^*) * \varepsilon + (e^* - b^* + 1) * w$.

After reordering we obtain: $(e - b + 1) * w' \leq (N - 1 - e^* + b^*) * \varepsilon + (e^* - b^* + 1) * w - (N - 1 - e + b) * \varepsilon$, and by regrouping the right term:

$$(e - b + 1) * w' \leq (e^* - b^* + 1) * w + ((e - b) - (e^* - b^*)) * \varepsilon \tag{13}$$

Since $e^* - b^* + 1 > 0$ and from the hypothesis that $w < \varepsilon$:

$$(e^* - b^* + 1) * w + ((e - b) - (e^* - b^*)) * \varepsilon < (e - b + 1) * \varepsilon \tag{14}$$

From the inequalities (13) and (14):

$$(e - b + 1) * w' < (e - b + 1) * \varepsilon$$

$e - b + 1 > 0 \Rightarrow w' < \varepsilon$.

If Q^* was preferred, then again

$$w' = w < \varepsilon.$$

□

Theorem 1 *The IVD algorithm described in Section 4.1 will converge to $\langle Q^*, b^*, e^* \rangle$ defined in (3).*

Proof. From inequality (7), after the 2^{nd} step we have $\varepsilon \geq w$. From Lemma 3, each further cycle of the algorithm decreases ε.

From Lemma 1 and Lemma 3, results that the convergence appears at $w' = w$. While the number of possible paths is limited, the algorithm is sure to converge. □

In the rest of this section we give an upper bound of the number of steps needed for convergence. Similarly to Lemma 2 we show that:

Lemma 4 *If $\varepsilon_1 > \varepsilon_2$, then either the sequence Q_{ε_1} is longer than Q_{ε_2}, or else we have reached convergence, i.e..*

$$e_2 - b_2 \leq e_1 - b_1 \tag{15}$$

Proof. Since $\overline{Q_{\varepsilon_1}}$ was preferred to $\overline{Q_{\varepsilon_2}}$ for $\varepsilon = \varepsilon_1$, it means that $\varepsilon_1(b_1 - e_1 + N - 1) + w_1(e_1 - b_1 + 1) \leq \varepsilon_1(b_2 - e_2 + N - 1) + w_2(e_2 - b_2 + 1)$.

$$\varepsilon_1((e_2 - b_2) - (e_1 - b_1)) + (e_1 - b_1 + 1)w_1 - (e_2 - b_2 + 1)w_2 \leq 0 \tag{16}$$

If for ε_2, $\varepsilon_2 < \varepsilon_1$, we would obtain $\overline{Q_{\varepsilon_2}}$ with $e_2 - b_2 \geq e_1 - b_1$, $(e_2 - b_2) - (e_1 - b_1) \geq 0$, then by subtracting $(\varepsilon_1 - \varepsilon_2)((e_2 - b_2) - (e_1 - b_1)) \geq 0$ from the first part of the inequality 16 we obtain:

$$\varepsilon_2((e_2 - b_2) - (e_1 - b_1)) + (e_1 - b_1 + 1)w_1 - (e_2 - b_2 + 1)w_2 \leq 0$$

showing that $\overline{Q_{\varepsilon_1}}$ will still be preferred to $\overline{Q_{\varepsilon_2}}$. If $\overline{Q_{\varepsilon_1}} \neq \overline{Q_{\varepsilon_2}}$ then this is a contradiction and we can infer that $e_2 - b_2 < e_1 - b_1$. Otherwise we have $\overline{Q_{\varepsilon_1}} = \overline{Q_{\varepsilon_2}}$ and we reach convergence with $e_2 - b_2 = e_1 - b_1$. □

Lemma 5 *The expression 10 does not respect the optimality principle of Dynamic Programming*

Proof. The path chosen at an entry E of the table $T(\overline{M}, X)$ at the frame a may not be included in the optimal path P to an entry F at the frame $c > a$ even if $\{E, F\} \subset \overline{Q^*}$. We keep the same notation as in the previous part, so that b and e represent the first respectively the last frames of X in a path Q.

Let k be a frame of X, $b^* \leq k \leq e^*$. We demonstrate that if $(x_k, q_m) \in Q^*$ and M_i^j is the HMM that remains from M when we keep only the states $\{q_i, ..., q_j\}$, then we may have for the path Q that optimizes $S(\overline{M}_0^m, X_1^k)$:

$$Q \not\subset \overline{Q^*}.$$

We take into account two paths P', P'' at each of the frames a and c, for the entries E respectively F, $b^* \leq a \leq e^*$, $b^* \leq c \leq e^*$, noted P'_c, P'_a, P''_c, P''_a so that, if $P'_c \backslash P'_a$ and $P''_c \backslash P''_a$ represent the sequences of states in the paths P' respectively P'' from the frame a to the frame c, then

$$P'_c \backslash P'_a = P''_c \backslash P''_a.$$

We note the three stored values of the two paths S', L', C' respectively S'', L'', C'' at each of the frames a and c using the index of the frame and we have:

$$\Delta S = S''_c - S''_a = S'_c - S'_a$$

$$\Delta L = c - a$$

We can show that $\exists\ \Delta L, \Delta S$ so that

$$\frac{S''_a}{L''_a} < \frac{S'_a}{L'_a}$$

but

$$\frac{S''_a + \Delta S}{L''_a + \Delta L} > \frac{S'_a + \Delta S}{L'_a + \Delta L}$$

For example, this case appears is $S''_a = 2$, $S'_a = 5$, $L''_a = 7$, $L'_a = 14$, $\Delta S = 10$, $\Delta L = 1$.

Therefore P” is chosen at frame a and P’ at frame c. If P’=Q^*, then P” was chosen in E without belonging to Q^*. □

Lemma 6 *If at a frame a we have two paths P'_a and P''_a with $S''_a < S'_a$ and $L'_a < L''_a$, then at no frame $c \geq a$ will a path P''_c be forsaken for a path P'_c if $P'_a \subset P'_c$, $P''_a \subset P''_c$ and $P'_c \setminus P'_a \equiv P''_c \setminus P''_a$. We will note the order relation as $P''_a \prec P'_a$.*

Proof. We use the notation:

$$\Delta S = S''_c - S''_a = S_c - S_a$$

$$\Delta L = c - a$$

From hypothesis:

$$\frac{S''_a}{L''_a} < \frac{S'_a}{L'_a}$$

Since, $\forall \Delta S \geq 0, \Delta L \geq 0$ and from the hypothesis that $S''_a < S'_a$ and $L'_a < L''_a$ we have:

$$L'_a * \Delta S + S''_a * \Delta L < L''_a * \Delta S + S'_a * \Delta L, \forall \Delta S, \Delta L$$

therefore, by adding the two inequalities, we obtain after some simple transformations:

$$\frac{S''_a + \Delta S}{L''_a + \Delta L} < \frac{S'_a + \Delta S}{L'_a + \Delta L}$$

□

Lemma 7 *A path P’ may be forsaken only for a lower cost one, P”.*

$$P' \prec P'' \Rightarrow C'_k < C''_k$$

Proof. This is a direct consequence of the formula 10 and Lemma 6. □

Theorem 2 *Algorithm 1 computes $S(M, X)$ and Q^* from equation (3).*

Proof. The theorem is the direct result of the Lemma 6. The procedure *Add*(path, set-of-paths) insures that a certain amount of paths will be discarded. The long paths have however a continuous advantage over the shorter ones, therefore maintaining the sets of paths at acceptable sizes requires additional pruning. □