EXTENDED VITERBI ALGORITHM FOR OPTIMIZED WORD HMMS

Michael Gerber, Tobias Kaufmann, Beat Pfister

Speech Processing Group, Computer Engineering and Networks Lab., ETH Zurich, Switzerland
{gerber,kaufmann,pfister}@tik.ee.ethz.ch

ABSTRACT

This paper deals with the problem of finding the optimal sequence of sub-word unit HMMs for a number of given utterances of a word. For this problem we present a new solution based on an extension of the Viterbi algorithm which maximizes the joint probability of the utterances and all possible sequences of sub-word units and hence guarantees to find the optimal solution. The new algorithm was applied in an isolated word recognition experiment and compared to simpler approaches to determining the sequence of sub-word units. We report a significant reduction of the recognition error rate with the new algorithm.

Index Terms— Viterbi, hidden Markov model, optimizing word models, isolated word recognition

1. INTRODUCTION

Most modern HMM-based speech recognizers use sub-word units as the basic acoustic modeling elements. Sub-word models can be linguistically motivated such as e.g. phone models or they can be abstract in the sense that there is no direct correspondence to the linguistic units of a language.

If phone models are used, the word models of the recognition vocabulary can be created by concatenating phone models according to a pronunciation lexicon to generate an acoustical model for each vocabulary word. In the case of abstract sub-word models or if no suitable pronunciation lexicon is available for the given language or dialect, the optimal sequence of sub-word models that constitute a particular word model can be derived from several utterances of that word.

For a single utterance, the problem of finding the optimal (in terms of maximum likelihood) sequence of sub-word models can easily be solved by means of the Viterbi algorithm: The sub-word HMMs are connected in parallel to form a sub-word loop and the optimal state sequence $\hat{Q}$ through the resulting HMM λ is evaluated. Then we can derive the optimal sequence of sub-word units $\hat{Z}$ from $\hat{Q}$, which we denote as: $\hat{Z} = \text{SWU}(\hat{Q})$.

In contrast to this simple case, determining the sequence of sub-word models, which maximizes the joint likelihood of several utterances, leads to a non-trivial optimization problem. This problem can be stated more formally as follows: Given a set of M sub-word HMMs $\lambda_1, \ldots, \lambda_M$ and K utterances of a word, designated as $\mathbf{X}^{(1)}, \ldots, \mathbf{X}^{(K)}$, find the optimal sequence of sub-word units $\hat{Z}$, i.e. the sequence of sub-word units with the highest probability to produce the utterances $\mathbf{X}^{(1)}, \ldots, \mathbf{X}^{(K)}$.

Since the utterances $\mathbf{X}^{(1)}, \ldots, \mathbf{X}^{(K)}$ generally are not of equal length, it is not possible to find a common state sequence.[1] However, our aim is not to find the optimal common state sequence for $\mathbf{X}^{(1)}, \ldots, \mathbf{X}^{(K)}$, but the optimal common sequence of sub-word units $\hat{Z}$. We can formulate this optimization task more specifically as follows: we look for the K state sequences $Q^{(1)}, \ldots, Q^{(K)}$ that maximize the product of the joint probabilities $P(\mathbf{X}^{(k)}, Q^{(k)}|\lambda)$ under the condition that all state sequences correspond to the same sequence of sub-word units. Note that λ still designates the sub-word loop mentioned above.

Thus, the problem of finding the optimal sequence of sub-word units $\hat{Z}$ can be written as:

$$\hat{Z} = \underset{Z}{\text{argmax}} \prod_{k=1}^{K} \max_{Q \in \mathcal{Q}_Z^{(k)}} P(\mathbf{X}^{(k)}, Q|\lambda) \tag{1}$$

$$\text{where} \quad \mathcal{Q}_Z^{(k)} = \{Q \mid \text{SWU}(Q){=}Z \text{ and } |Q|{=}T_k{+}2\}$$

$\mathcal{Q}_Z^{(k)}$ is the set of all state sequences that are consistent with Z and include T_k emitting states (together with the start and end states the sequences are of length $T_k{+}2$).

The remainder of this paper is structured as follows: First an overview of related solutions from the literature is given in Section 2. In Section 3 we describe our own solution which is basically an extension of the Viterbi algorithm. In Section 4 our algorithm is applied in the context of an isolated word recognizer and the results are compared to other approaches to isolated word recognition.

[1] This depends on the type of HMM used. An HMM as used in this work is defined to have N states, to start in the non-emitting state S_1, to repeatedly transit with probability a_{ij} from a state S_i to a S_j and emit with probability $b_j(x)$ the observation x and finally to transit with probability a_{iN} from a state S_i to the final non-emitting state S_N. Therefore, the HMM generates a sequence of T observations while visiting T emitting states.

2. RELATED WORK

There are several approaches to solve the optimization problem for several utterances. Most of these approaches are based on heuristics and can be roughly divided into two categories. A first category of approaches is based on generating sequences of sub-word units for each utterance and choosing the one which best matches the complete set of utterances. The simplest variant of this category is to find the best sequence of sub-word units for every utterance in isolation and choose the one which best describes the whole set of utterances. The problem with this approach is that the globally optimal sequence of sub-word units is often not among the candidates. A solution to this problem was suggested in [1]. This solution employs the method described in [2] to generate the n best sequences of sub-word units for every utterance and chooses the best candidate from this extended set of utterances. This increases the probability to find the optimal sequence of sub-word units but cannot guarantee to find it.

The other category of approaches is based on A* tree search. In [3] a method was described which chooses the best node to be evaluated next in the tree search using a heuristically determined estimate of the likelihood for the remainder of the optimal path through that node. This approach finds the optimal solution only if this likelihood is overestimated. Then, however, the tree-search algorithm is likely to be intractable. An improvement to this algorithm was presented in [4]. Here the normal forward pass of Viterbi search is executed individually for each signal and the likelihoods are stored for all utterance-state-frame triples. The tree search is then performed in a backward pass, while a better estimate of the continuation likelihood of the backward path can be computed based on the stored likelihoods from the forward pass. Since this estimate is based on the forward scores of the individual paths it is still an over-estimate as is argued in [5]. Finding the optimal path is therefore still based on heuristics. An approach which uses breadth-first tree-search was presented in [6]. This approach does not guarantee optimality either since it requires strong pruning.

3. EXTENSION OF THE VITERBI ALGORITHM

The standard Viterbi algorithm is used to solve the decoding problem, i.e. to determine for an observation sequence $\mathbf{X} = x_1 x_2 \ldots x_T$ and a given HMM λ with states $S_1, S_2, \ldots, S_N$ (S_1 and S_N being the non-emitting start and end states, see the footnote on the previous page) the optimal sequence of states $\hat{Q} = S_1 \hat{q}_1 \hat{q}_2 \ldots \hat{q}_T S_N$. With the principle of dynamic programming, the joint probability of the partial observation sequence $\mathbf{X}_t = x_1 x_2 \ldots x_t$ and the optimal partial state sequence $\hat{Q}_t = S_1 \hat{q}_1 \hat{q}_2 \ldots \hat{q}_t$ that ends in state S_j at time t

$$\delta_t(j) = \max_{\text{all } Q_t \text{ with } q_t = S_j} P(\mathbf{X}_t, Q_t | \lambda) \tag{2}$$

can be computed recursively with

$$\delta_t(j) = \max_{1<i<N} \delta_{t-1}(i)\, a_{ij}\, b_j(x_t). \tag{3}$$

$\delta_T(N)$ is the probability $P(\mathbf{X}, \hat{Q}|\lambda)$. In order to find $\hat{Q}$ the optimal precursor state has to be stored for each time and state tuple:

$$\Psi_t(j) = \operatorname*{argmax}_{1<i<N} \delta_{t-1}(i)\, a_{ij}. \tag{4}$$

The optimal state sequence $\hat{Q}$ can then be determined recursively, starting at the end. $\hat{Q}$ can be visualized as a path through a two-dimensional trellis spanned by the observations and the states.

Now we extend the Viterbi algorithm such that it finds the optimal sequence of sub-word units $\hat{Z}$ for K observation sequences, as defined by equation (1). The corresponding trellis has $K+1$ dimensions and the path through this trellis has to meet the two following conditions:

1. The path has to proceed in every step by zero or one in every observation sequence (i.e. in all of the K temporal dimensions in the trellis) but has to proceed in at least one observation sequence.
2. A path Q through the trellis is valid if this path and its projections $Q^{(k)}$ meet the condition: $\text{SWU}(Q) = \text{SWU}(Q^{(k)}) = Z$. This condition is obviously met if each transition between states that do not belong to the same sub-word model, advances a time step in all observation sequences.

The mathematical formulation of our extension to the Viterbi algorithm is as follows: The joint probability of the partial observation sequences $\mathbf{X}^{(1)}_{t_1}, \mathbf{X}^{(2)}_{t_2}, \ldots, \mathbf{X}^{(K)}_{t_K}$ and the optimal partial state sequences $\hat{Q}^{(1)}_{t_1}, \hat{Q}^{(2)}_{t_2}, \ldots, \hat{Q}^{(K)}_{t_K}$ with $q^{(1)}_{t_1} = q^{(2)}_{t_2} = \cdots = q^{(K)}_{t_K} = S_j$ is defined in equation (5). Of course, the optimal partial state sequences have to meet the condition that the corresponding sequences of sub-word units are identical.

Probability $\delta_{t_1,\ldots,t_K}(j)$ can be computed recursively for all points in the trellis with equation (6). Note that all partial paths that violate the two above conditions are excluded. For backtracking the best path we have to save the optimal precursor state i like in the one-dimensional Viterbi algorithm. Additionally we need the time vector $(c_1, c_2, \ldots, c_K)$ which points from the current time point to the precursor time point. All these values are represented as a (K+1)-dimensional vector $\Psi_{t_1,t_2,\ldots,t_K}(j) = (c_1, c_2, \ldots, c_K, i)$. This vector is determined with equation (7).

Let's give some additional explanations to equations (6) and (7): With $\sum c_k > 0$ the first condition is met. In order to satisfy the second condition, transitions from one sub-word unit to a another one are allowed only if $\sum c_k = K$. Otherwise the sub-word unit must not change, i.e. $SWU(S_i)$ has to

$$\delta_{t_1,\ldots,t_K}(j) = \max_{\substack{\text{all } Q^{(1)}_{t_1},\ldots,Q^{(K)}_{t_K} \\ \text{with} \begin{cases} \text{SWU}(Q^{(1)}_{t_1})=\cdots=\text{SWU}(Q^{(K)}_{t_K}) \\ q^{(1)}_{t_1}=\cdots=q^{(K)}_{t_K}=S_j \end{cases}}} P(\mathbf{X}^{(1)}_{t_1},\ldots,\mathbf{X}^{(K)}_{t_K},Q^{(1)}_{t_1},\ldots,Q^{(K)}_{t_K}|\lambda) \quad (5)$$

$$\delta_{t_1,\ldots,t_K}(j) = \max_{\substack{(c_1,\ldots,c_K,i) \\ \text{with} \begin{cases} 1<i<N \\ c_k\in\{0,1\} \\ \sum c_k>0 \end{cases}}} \begin{cases} \delta_{t_1-c_1,\ldots,t_K-c_K}(i)\, a_{ij}^{\sum c_k}\, b_j(x^{(1)}_{t_1})^{c_1}\ldots b_j(x^{(K)}_{t_K})^{c_K} & \text{if } (\sum c_k = K) \text{ or } g(i,j) \\ 0 & \text{otherwise} \end{cases} \quad (6)$$

$$\Psi_{t_1,\ldots,t_K}(j) = \operatorname{argmax}_{\substack{(c_1,\ldots,c_K,i) \\ \text{with} \begin{cases} 1<i<N \\ c_k\in\{0,1\} \\ \sum c_k>0 \end{cases}}} \begin{cases} \delta_{t_1-c_1,\ldots,t_K-c_K}(i)\, a_{ij}^{\sum c_k}\, b_j(x^{(1)}_{t_1})^{c_1}\ldots b_j(x^{(K)}_{t_K})^{c_K} & \text{if } (\sum c_k = K) \text{ or } g(i,j) \\ 0 & \text{otherwise} \end{cases} \quad (7)$$

$$\text{where } g(i,j) = \begin{cases} true & \text{if } \text{SWU}(S_i)=\text{SWU}(S_j) \\ false & \text{otherwise} \end{cases}$$

be equal to $SWU(S_j)$. This condition allows multi-state sub-word units which may also contain skip arcs. By using c_k as an exponent in $b_j(x^{(k)}_{t_k})^{c_k}$, the probability of an observation $x^{(k)}_{t_k}$ is multiplied to the total path probability if and only if the path proceeds by one observation in dimension k. With $a_{ij}^{\sum c_k}$ also the transition probability a_{ij} is multiplied to the total path probability for each observation sequence in which the path proceeds by one.

The recursion of the algorithm is initialized with

$$\delta_{1,1,\ldots,1}(j) = a^K_{1j} \prod_{k=1}^{K} b_j(x^{(k)}_1) \quad (8)$$

and terminated with

$$\delta_{T_1,\ldots,T_K}(N) = \max_{1<i<N} [\delta_{T_1,\ldots,T_K}(i) a^K_{iN}] \quad (9)$$

$$\Psi_{T_1,\ldots,T_K}(N) = \operatorname{argmax}_{1<i<N} [\delta_{T_1,\ldots,T_K}(i) a^K_{iN}] \quad (10)$$

Note that the $c_1,\ldots,c_K$ in $\Psi_{T_1,\ldots,T_K}(N)$ are not defined by equation (10). They are implicitly defined as zero, since the end of all observation sequences has been reached. The optimal path $\hat{Q}$ through the trellis and thus the optimal sequence of sub-word units $\hat{Z}$ can now be found by backtracking.

The resulting sequence of sub-word HMMs is guaranteed to be the optimal one, because all allowed paths through the corresponding multidimensional trellis are considered.

The presented algorithm has a computational complexity of roughly $\mathcal{O}(T^K)$ for K utterances where T is the geometric mean of the utterance lengths.

4. EXPERIMENTS

We performed experiments with isolated word recognition.

4.1. Tested Recognizer

In the used recognizer all words were modeled as word HMMs. The word models of the recognition vocabulary were composed from two different types of sub-word units. In the first three experiments we used a set of 128 abstract sub-word units as described in [7]. Each of these abstract sub-word units was represented by a one-state HMM with 16 Gaussian mixtures. The first experiments differed in the method that was applied to compose the word models. For the fourth experiment, the word models were composed from phone models using a pronunciation lexicon.

4.1.1. Optimal template from several utterances (kViterbi)

In this experiment the extension of the Viterbi algorithm presented in this paper was applied to find for every word in the vocabulary the optimal sequence of abstract sub-word units.

4.1.2. Choose best-matching sequence (chooseBest)

In this experiment we used an approximate algorithm as it is used in [1] to find the optimal sequence of abstract sub-word units. We first determined for each utterance the 20 best sequences of abstract sub-word with the token-passing-based Viterbi algorithm with ten tokens per node. From these candidate word models we used the one with the highest mean likelihood over all utterances.

4.1.3. Best sequence for DTW-aligned signals (DTWalign)

In this experiment we used another approximate approach to determine the sequence of abstract sub-word units. All utterances of a reference word were aligned with DTW in order to get a time-aligned version of the sequences. All sequences were mapped on the sequence which had the smallest DTW-distance to all others. For these time-aligned sequences it was straightforward to perform a Viterbi search by using the joint observation probability of the aligned utterances to find the best sequence of sub-word units.

4.1.4. Lexicon-based word models (dict)

Here the word models were concatenated from phonemes according to a pronunciation lexicon. The phonemes were modeled by three-state monophones. Each state was modeled with 32 Gaussian mixture components.

4.2. Test Setup

For the experiments we used the Swiss German polyphone database. The signals were all recorded over telephone line using various types of telephones.

For training the abstract sub-word units and the phonemes, we used utterances from 3500 speakers of the database. The test speakers were not among those speakers.

Speaker-dependent word recognition experiments were performed for 53 speakers. For every speaker about 16 vocabularies were formed by randomly selecting ten words from a pool of mostly short German words. The used utterances of each word were also randomly selected. Every vocabulary was tested with about eight randomly chosen utterances. Thus a total of 854 vocabularies were created and 7237 tests were performed.

We used Mel-frequency cepstral coefficients (MFCCs) as features. The raw signals were filtered with a preemphasis filter with a coefficient of 0.9. Then the MFCCs 0-12 plus their temporal derivatives were computed every 10 ms for Hamming-windowed frames of length 37.5 ms.

4.3. Results

The recognition rates which we obtained for the experiments are given in Table 1. These results showed that the extension of the Viterbi algorithm for several utterances was better to determine the optimal sequence of sub-word units for word-models than alternative methods. According to the McNemar test (see for example [8]), the improvement relative to *chooseBest* and *DTWalign* were statistically significant on a level of < 0.1 %. Furthermore, the recognition rate of the experiment *kViterbi* in which the developed algorithm was used, was statistically significantly better (also < 0.1 % significance level) than the one of experiment *dict* where the word models were built according to a pronunciation lexicon.

Table 1. *Recognition rates for the experiments.*

experiment (see Section 4.1)	# of utterances per reference		
	1	2	3
kViterbi	92.9 %	95.6 %	96.6 %
chooseBest		95.1 %	95.8 %
DTWalign		94.1 %	94.1 %
dict	95.8		

5. CONCLUSION

The extended Viterbi algorithm is a new solution to the problem of finding the optimal sequence of sub-word HMMs for a word that has been uttered several times. The resulting sequence of sub-word HMMs is guaranteed to be optimal because the whole search space is considered.

There are plenty of applications which can profit from this algorithm. We have demonstrated its usability in an HMM-based isolated word recognizer where it has shown to perform better than a standard recognizer with phoneme HMMs and lexicon-based word models.

Since our algorithm guarantees to find the optimal sequence, it can be used as a reference for lower complexity algorithms which use approximations to determine the optimal sequence of sub-word units for several utterances.

6. REFERENCES

[1] H. Mokbel and D. Jouvet, "Derivation of the optimal set of phonetic transcriptions for a word from its acoustic realizations," *Speech Communication*, vol. 29, pp. 49–64, September 1999.

[2] F. K. Soong and E.-F. Huang, "A tree-trellis based fast search for finding the n best sentence hypotheses in continuous speech recognition," in *Proc. of the Workshop on Speech and Natural Language*, 1990.

[3] L. R. Bahl and P. F. Brown, et al., "A method for the construction of acoustic Markov models for words," *IEEE Transactions on Speech and Audio Processing*, vol. 1, no. 4, pp. 443–452, October 1993.

[4] T. Svendsen, F. K. Soong, and H. Purnhagen, "Optimizing baseforms for HMM-based speech recognition," in *Proc. of Eurospeech*, 1995.

[5] J. Wu and V. Gupta, "Application of simultaneous decoding algorithms to automatic transcription of known and unknown words," in *Proc. of ICASSP*, 1999.

[6] M. Bisani and H. Ney, "Breadth-first search for finding the optimal phonetic transcription from multiple utterances," in *Proc. of Interspeech*, 2001.

[7] M. Gerber and B. Pfister, "Fast search for common segments in speech signals for speaker verification," in *Proc. of Interspeech*, Brisbane (Australia), September 2008, pp. 375–378.

[8] L. Gillick and S. Cox, "Some statistical issues in the comparison of speech recognition algorithms," in *Proc. of ICASSP*, 1989, pp. 532–535.