Segment Boundaries in Low Latency Phonetic Recognition

Giampiero Salvi

KTH, Royal Institute of Technology,
TMH dept. for Speech, Music and Hearing,
giampi@speech.kth.se

Abstract. This study analyses how the reduction of the look-ahead length of a two pass phonetic decoder influences the alignment of the segment boundaries. It is shown how the optimization of some tuning parameters, such as the insertion penalty, is dependent on the look-ahead length. It is also suggested that the insertion penalty be dynamically adjusted to some measure of similarity of the phonetic segments following each other.

1 Introduction

Two pass decoders such as different variants of the Viterbi decoder [1] are extensively used in speech recognition. In applications where the input signal is captured in a continuous stream, and where the latency of the system response is a constraint, the optimal Viterbi solution needs to be approximated. This is done by limiting the length of the look ahead, i.e. the number of future frames that are observed before making a decision at time t.

The effects of this approximation have been studied in the area of broadcast news recognition/transcription [2,3]. The characteristics of the decoder (large vocabulary word recognition), and the constraints considered in these studies, are far from the ones investigated in this paper.

The application we have in mind [4] aims at mapping the speech signal into visemic classes that can be used to control the lip movements of an avatar [5]. As the system should be used as a lip reading support in telephone conversation, the time alignment of the classified segments is as important as the classification accuracy, and the latency constraints are especially severe.

This study analyses the effect of latency constrains on the segment alignment using the Synface phonetic recognizer [6].

2 The Recognizer

The Synface recognizer is a hybrid of recurrent neural networks (RNNs), and hidden Markov models (HMMs). The input layer of the RNN contains thirteen units that represent the Mel frequency cepstral coefficients $C_0, .., C_{12}$. The activity of the output layer represents the posterior probability of each of the N_p

phonetic classes given the input vector, plus N_n acoustic classes including silence and different kinds of noise. The total number N of output units depends on the language, see Section 3.

The only hidden layer contains 400 units and is fully connected with the input and output layers with time delayed connections. Time delayed self-connections are also drawn between the hidden layer an itself.

The output activities of the neural network are used as probability estimators in a Viterbi-like decoder where the look-ahead length can be varied.

The recognition network specified by a Markov chain defines a free loop of phonemes, where every phoneme is represented by a three state left-to-right HMM.

3 Data

The recognizer was trained on the SpeechDat database independently on three languages (Swedish, English and Flemish). The experiments in this paper refer to Swedish. The Swedish database has been divided into a training, a validation and a test set, respectively of 33062, 500 and 4150 utterances.

The phonetic transcriptions shown in the examples use SAMPA symbols [7] with few exceptions [8] presented in Table 1. The total number of acoustic classes is 50 for Swedish, with 46 phonemes and 4 kinds of noise/silence.

original	}	2	{	9	@
modified	uh	ox	ae	oe	eh

Table 1. Modification to the SAMPA phonetic symbols

4 The Problem

Hidden Markov model decoding is based on the integration of two concurrent production models. A transition model π, A describes the probability of jumping between two of a finite number of predefined states. An emission model B estimates the probability of emitting a particular observation given one of the states.

All transitions in the model are treated homogeneously in decoding phase, but they can be divided into classes, depending on the way they are obtained during training. If the recognizer is based on a set of N linguistically significant units $\{\lambda_k\}$, each made up of a number m_i of states $\{s_i\}$, with $M = \sum_i m_i$ total number of states, and we call $\Lambda(.)$ a relation of correspondence from the set of states to the set of units ($\Lambda(s_i) = k \iff s_i \in \lambda_k$), we can define the *intra*-unit transitions, i.e. transitions between states of the same unit as $T = \{t_{ij} : \Lambda(s_i) = \Lambda(s_j)\}$. The

probability of these transitions commonly obey to a standardized left-to-right topology and are usually estimated from the acoustical training data. We will refer to them as *acoustic* transitions.

The *inter*-unit transitions defined as $G = \{g_{ij} : \Lambda(s_i) \neq \Lambda(s_j)\}$, on the contrary, have a topology that strongly depends on the task and on the resources available. They could be specified according to *a priori* information, or estimated from large text-based corpora in form of N-grams. We will refer to them as *grammar* transitions[1].

In any case, these probability models need some tuning, for which an optimization is run on an independent data set. The parameters that are subject to optimization are the *insertion penalty* (ip) and the *grammar factor* (gf). The first simply introduces a penalty whenever a grammar transition is traversed and is used to find a balance between the number of symbol insertions and deletions in the decoding phase. The second defines how much weight should be given to the grammar model compared to the acoustic models.

The resulting modified grammar transition probabilities in negative logarithmic domain are specified as:

$$G' = \{g'_{ij} = \mathrm{gf}\ g_{ij} + \mathrm{ip}, \quad g_{ij} \in G\}$$

that corresponds in linear domain to:

$$G' = \{g'_{ij} = e^{-\mathrm{ip}}\ g_{ij}{}^{\mathrm{gf}}, \quad g_{ij} \in G\}$$

Noticing that T and G are disjoint sets, the total transition model is given by:

$$A = A(\mathrm{ip}, \mathrm{gf}) = T \cup G'$$

In the following experiments, the grammar factor is an uninteresting parameter as the grammar specifies a free loop of phonemes with equal transition probabilities. The role of the grammar factor would become essential if a non-uniform distribution of probabilities, e.g. a phone bigram, was to be used. Note that ip and gf are global parameters.

We call A^* the grammar obtained by optimizing the performance of the recognizer as a function of the insertion penalty on the development set. The more the grammar A contains short paths (e.g. phonetic recognition as opposed to word recognition) the higher we expect the optimal value of the insertion penalty to be in order to reduce insertions.

When the look-ahead length is decreased, the solution A^* is no longer optimal. One reason for this is that at time t, the acoustic evidence over a certain number f of frames is needed in order to exceed the transition penalty p. The higher p the higher f. This is especially noticeable for transitions between similar acoustic segments. In this case, the contribution of each future frame is less effective if compared to the penalty p.

[1] in the following the notation t_{ij} and g_{ij} will be used indifferently to refer to a transition or to the probability associated to it

5 Insertion Penalty Optimization

As noted in Section 4 the optimal value A^* depends on the look-ahead length. Figure 1 shows the optimization run on the value of the insertion penalty ip for different values of the look-ahead length. For low insertion penalties, the number of insertions is high, reducing the accuracy, but increasing the % of correct symbols.

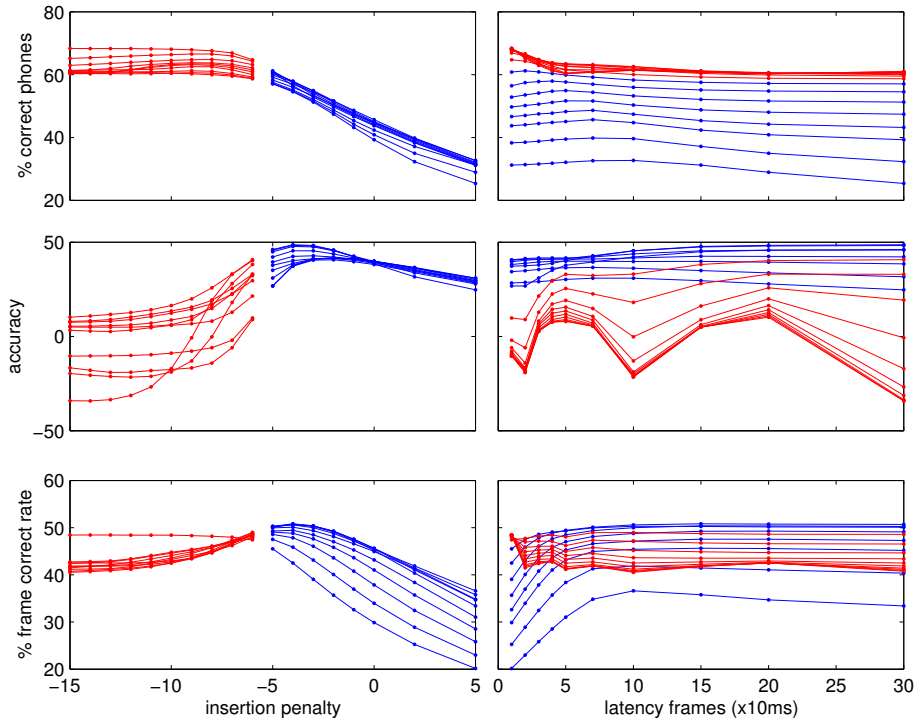

Fig. 1. % correct symbols, accuracy and frame-by-frame correct rate for different values of look-ahead length and insertion penalty. The colors in the figure are used only to simplify the visualization.

6 Experimental Observation

The three criteria indicated in the previous paragraph (% correct symbols, accuracy and frame-by-frame correct rate) do not take into account the segment boundaries explicitly.

Figure 2 shows one of the phenomena announced in Section 4 and that characterizes the recognition results for very low look-ahead lengths.

The segment of speech contains the beginning of the word “känsla” (känsloyttringarna). Silence (`sil`) is followed by the fricative `C` and by the vowel `E`. The first transcription pane from the top is the reference transcription, the second the maximum a posteriori (map) solution obtained selecting the phonetic class with the highest neural network activity. The following six panes contain the approximated Viterbi solution with look-ahead length from 1 to 6. Finally the last pane is the standard Viterbi solution.

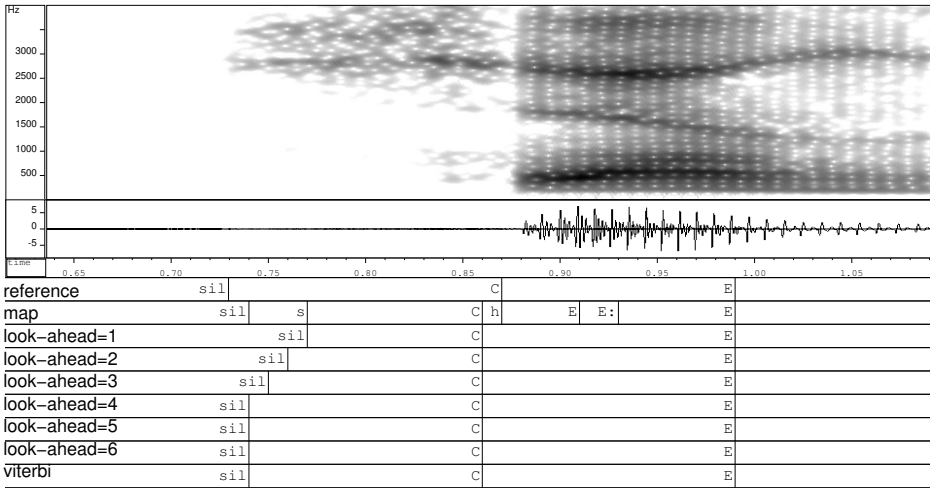

Fig. 2. Spectrogram, waveform and transcriptions of a segment of the phrase “Men känsloyttringarna”. The transcriptions contain from the top: reference (forced alignment), maximum a posteriori solution, approximated Viterbi solution with look-ahead = 1,2,3,4,5,6 and standard Viterbi solution.

In the first transition (`sil` to `C`), even though the map solution shows that acoustically `s` and `C` are more probable than `sil` from $t = 0.74$ sec to $t = 0.86$ sec, for short look-ahead, the transition is delayed. This delay is three frames for look-ahead = 1 frame, which suggests that the Viterbi accumulators need to incorporate four frames of acoustic evidence to overcome the insertion penalty. Consistently the hypothesis changes after two frames for look-ahead = 2, after one frame for look-ahead = 3 and synchronously with the map solution when look-ahead > 3.

This does not happen in the transition between `C` and `E`. This is probably due to the fact that, in spite of `E` being acoustically similar to `E:` (and even `h` in the beginning), it is well separated from `C`. That is the acoustic probability of stretching the segment `C` over `E` is very low, compensating for the insertion penalty even at short look-ahead lengths.

7 Preliminary Conclusions

The preliminary analysis in this paper shows one phenomenon that arise when the look-ahead in a Viterbi-like decoder is reduced to a length that is comparable with the length of the shortest path in the recognition grammar. The main observable effect is the misalignment of phonetic boundaries (usually a delay), that sometimes results in the deletion of an entire segment. Optimizing the insertion penalty to the length of the look-ahead is necessary but not sufficient to avoid the problem. A suggestion is to use a measure of similarity of the previous and next segment (possibly based on mutual information) to adjust the insertion penalty dynamically to each case.

8 Acknowledgments

This research was carried out at the Centre for Speech Technology supported by Vinnova (The Swedish Agency for Innovation Systems), KTH and participating Swedish companies and organisations.

References

1. Viterbi, A.J.: Error bounds for convolutional codes and an asymtotically optimum decoding algorithm. IEEE Trans. Inform. Theory **IT-13** (1967) 260–269
2. Imai, T., Kobayashi, A., Sato, S., Tanaka, H., Ando, A.: Progressive 2-pass decoder for real-time broadcast news captioning. In: ICASSP. (2000) 1937–1940
3. Ljolje, A., Hindle, D.M., Riley, M.D., Sproat, R.W.: The AT&T LVCSR-2000 system. In: Speech Transcription Workshop, University of Maryland (2000)
4. Karlsson, I., Faulkner, A., Salvi, G.: SYNFACE - a talking face telephone. In: Proc. Eurospeech. (2003) 1297–1300
5. Beskow, J.: Trainable articulatory control models for visual speech synthesis. Journal of Speech Technology (in press)
6. Salvi, G.: Truncation error and dynamics in very low latency phonetic recognition. In: ISCA Tutorial and Research Workshop on Non-linear Speech Processing (NOLISP), Le Croisic, France. (2003)
7. Gibbon, D., Moore, R., Winski, R., eds.: SAMPA computer readable phonetic alphabet, Part IV, section B. In: Handbook of Standards and Resources for Spoken Language Systems. Mouton de Gruyter, Berlin and New York (1997)
8. Lindberg, B., Johansen, F.T., Warakagoda, N., Lehtinen, G., Kačič, Z., Žgank, A., Elenius, K., Salvi, G.: A noise robust multilingual reference recogniser based on SpeechDat(II). In: 6th Intern. Conf. on Spoken Language Processing. Volume III. (2000) 370–373