A template matching approach for detecting pronunciation mismatch

Lavanya Prahallad, Radhika Mamidi, Kishore Prahallad
International Institute of Information Technology, Hyderabad, India.
{lavanya@research, radhika.mamidi, kishore}@iiit.ac.in

ABSTRACT
In this paper, we study the usefulness of the best path and the complete trellis in dynamic programming based template matching approach for detecting pronunciation mismatch. We show that there exists cues in trellis (a matrix representing all paths), which could be exploited for detecting pronunciation mismatch. Such an approach could be used to build a template based approach for detecting pronunciation mismatch independent of the language.

KEYWORDS: Pronunciation mismatch, dynamic programming, template matching.

KEYWORDS IN L_2: .

Proceedings of the Workshop on Speech and Language Processing Tools in Education, pages 79–84,
COLING 2012, Mumbai, December 2012.

1 Template based approach

Detection of pronunciation mismatch plays a crucial role in the computer-assisted pronunciation training (CAPT). A CAPT system is built using an automatic speech recognition system (Delmonte, 2000) (Ehsani and Knodt, 1998) (Eskenazi, 2009) (Stenson et al., 1992). A template matching algorithm provides greater flexibility than a model-based (such as hidden Markov models) approach in building an automated tutor for second language learning and acquisition. Model-based approaches are resource intensive. It needs a lot more speech data than single templates or exemplars to build acoustic models of the native and non-native speakers. In the case of template based approaches, a single reference template is deemed sufficient. Template based approaches could also be scaled to a new language easily.

Amongst template matching algorithms, dynamic programming is mostly commonly used as it performs nonlinear alignment of test and reference patterns. The best path aligning the test and reference templates is chosen using the Viterbi algorithm. The cost of this alignment is typically subjected to a threshold to detect pronunciation mismatch in the test and reference templates. The cost of the alignment depends on several factors including the recording environment, speaker/learner, and the amount of pronunciation mismatch in the test template. Thus, the alignment cost is subjected to some form of normalization (what type of normalization etc., give references). Inspite of normalization of scores, it is not easy to detect subtle pronunciation variations in the template based approaches. For example, in English, it is hard to detect /sh/ when it is replaced with /s/ using a template based approach.

In this paper, we show that it is important to look at the cues present in the scores of the entire trellis than computing the best path. A trellis is a matrix representing all the possible alignment paths between the test and reference templates. The best path computation using the Viterbi algorithm is designed to optimise the score between the test and reference template. Thus, we argue that it is better to delay the process of computing the best path, and focus on the scores along the diagonal of the trellis.

In order to demonstrate the usefulness of the cues in the trellis we make use of synthetic speech, i.e., speech generated by a text-to-speech system. A good text-to-speech (TTS) system faithfully generates a waveform corresponding to the text input or to a sequence of phonemes. A TTS system is usually built using a single speaker's voice. Thus a test or reference example generated by a TTS will have the same speaker's voice. Thus the variability in the speaker characteristics is suppressed. This acts as an advantage to our study to show that the cues in the trellis highlight the subtle pronunciation mismatches in the test and reference templates.

This paper is organized as follows. Section 2 discusses the synthetic data used in this work. Section 3 explains the significance of trellis in highlighting the mismatches. Section 4 discusses the results.

2 Synthesized speech data set

To generate synthesized speech data, we have used US KAL diphone voice in festival speech synthesis system. A diphone voice is unit selection voice, where the required set of diphones are collected from the speaker, these diphones are modified in terms of duration and intonation based on the context. A smooth concatenation of these prosodically modified diphones is done to generate a synthetic speech. In this work, we have generated 10 synthesized words with correct and incorrect pronunciation. The correct pronunciation of a word is automatically generated by pronunciation lexicon in the festival speech synthesis system, to generate incorrect

Word	Correct Pron.	Incorrect Pron.
sugar	sh uh g er	s uh g er
gym	jh ih m	jh ay m
honey	hh ah n iy	hh aa n ey
though	dh ow	dh ao
daughter	d ao t er	d aa t er
zero	z ih r ow	jh ih r ao
snack	s n ae k	s n aa k
honor	aa n er	aa n er
caught	k aa t	k aa t
eight	ey t	ay t ey

Table 1: List of words with correct and incorrect pronunciation

pronunciation we have manually identified a sequence of phonemes for each word, where there is at least one incorrect phoneme. Table 1 shows the set of 10 words, their correct and incorrect pronunciation.

3 Alignment using dynamic programming

Given that we have generated synthetic speech data with correct and incorrect pronunciation, we now describe the features extracted and the dynamic programming based alignment in the below sections:

3.1 Feature extraction

The speech signal is processed to generate linear prediction cepstral coefficients (LPCCs). These features are generated for every block of 10 milli seconds using a frame shift of 5 milli seconds. For each frame a 12^{th} order linear prediction analysis is applied to extract 17 dimension LPCCs.

3.2 Forward algorithm

Let $Y = \{\boldsymbol{y}(1), \boldsymbol{y}(2), \ldots, \boldsymbol{y}(T)\}$ be a sequence of observed feature vectors for the correct pronunciation. Let $X = \{\boldsymbol{x}(1), \boldsymbol{x}(2), \ldots, \boldsymbol{x}(M)\}$ be a sequence of observed feature vectors for the incorrect pronunciation. The dynamic programming aligns the feature vectors Y with X. The result is stretched or shrunk signal $X' = \{\boldsymbol{x}(1), \boldsymbol{x}(2), \ldots, \boldsymbol{x}(T)\}$. The dynamic programming algorithm to compute X' is as explained below. This is explained in the probability-like domain, as apposed to tradition Euclidean distance domain.

Let $1 \leq j \leq M$, $1 \leq i \leq M$, and $1 \leq t \leq T$. Let us define $\alpha_t(j)$ as a cost or score incurred to align j^{th} feature of X with t^{th} feature vector of Y.

The $\alpha_t(j)$ could be computed frame-by-frame using the recursive Viterbi equation

$$\alpha_t(j) = \max_i \{\alpha_{t-1}(i) a_{i,j}\} P(\boldsymbol{y}(t), \boldsymbol{x}(j)), \tag{1}$$

where $P(\boldsymbol{y}(t), \boldsymbol{x}(j)) = exp(\|\boldsymbol{y}(t) - \boldsymbol{x}(j)\|^2)$, and $\|.\|^2$ represents the Euclidean distance between two feature vectors. Here $i = \{j, j-1, j-2\}$. The value of $a_{i,j} = 1$, thus making all paths (including non-diagonal) leading from $(i, t-1)$ to (j, t) are given uniform weightage.

The value $P(\boldsymbol{y}(t), \boldsymbol{x}(j))$ is typically less than 1. For large values of t, $\alpha_t(.)$ tends exponentially to zero, and its computation exceeds the precision range of any machine (Rabiner and Juang, 1993). Hence $\alpha_t(.)$ is scaled with term $\frac{1}{\max_i\{\alpha_t(i)\}}$, at every time instant t. This normalization ensures that values of $\alpha_t(.)$ are between 0 and 1 at time t.

Given $\alpha_t(.)$, a backtracking algorithm is used to find the best alignment path. In order to backtrack, an addition variable ϕ is used to store the path as follows.

$$\phi_t(j) = \underset{i}{\text{argmax}}\{\alpha_{t-1}(i)a_{i,j}\}, \tag{2}$$

where $\phi_t(j)$ denotes the frame number (index of the feature vector) at time $(t-1)$ which provides an optimal path to reach state j at time t.

3.3 Best path

Given values of $\phi_t(.)$, a typical backtracking done to obtain the best path is as follows:

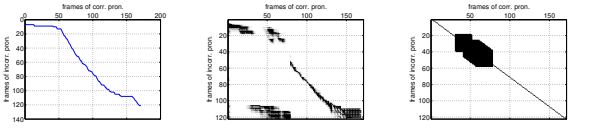
$$y(T) = N \tag{3}$$

$$y(t) = \phi_{t+1}(y(t+1)),\ t = T-1, T-2, \ldots, 1. \tag{4}$$

4 Results and discussion

4.1 Best path Vs. Full trellis

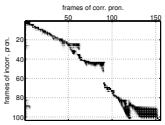
Any typical template matching algorithm works with the best path. It has to be noted that the best path is one of the paths in the entire trellis. Our argument is that there exists cues in the trellis about the weak alignment spots, which are not clearly indicated in the best path. In order to demonstrate it, we align the utterances corresponding to /s uh g er/ (incorrect pronunciation) and /sh uh g er/ (correct pronunciation) using dynamic programming. Fig. 1(a) shows the best path of this alignment, i.e., $y(t)$ for $t = 1 \ldots T$, as defined in Eq. (4). Fig. 1(b) shows the complete trellis, i.e, $\alpha_t j$ for all values of t and j. It could be observed that the best path in Fig. 1(a) does not highlight or show explicitly any type of mismatch between the aligned utterances. Fig. 1(b) highlights that there is a weak spot (a set of white pixels denoting a white patch) along the diagonal. The time stamps of this weak spot correspond to alignment of /sh/ with /s/. Thus the full trellis could be useful is detecting a pronunciation mismatch. Using a simple algorithm, we have automatically detected this weak spot, and is as shown in Fig. 1(c).



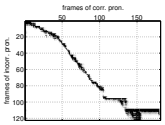
(a) Best path (b) Complete trellis (c) Automatically detected weak spots in the trellis

Figure 1: Alignment of /s uh g er/ with /sh uh g er/.

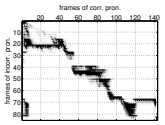
Fig. 2 provides the similar results for the rest of the utterances.



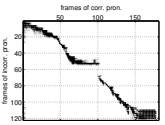
(a) /jh ay m/ Vs /jh ih m/



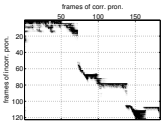
(b) /hh aa n ey/ Vs /hh ah n iy/



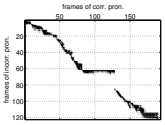
(c) /dh ao/ Vs /dh ow/



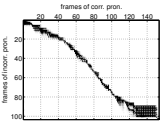
(d) /d aa t er/ Vs /d ao t er/



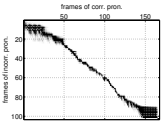
(e) /jh ih r ao/ Vs /z ih r ow/



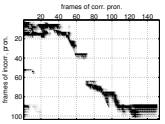
(f) /s n aa k/ Vs /s n ae k/



(g) /aa n er/ Vs /aa n er/



(h) /k aa t/ Vs /k aa t/



(i) /ay t ey/ Vs /ey t/

Figure 2: Trellis of the dynamic programming alignments

5 Conclusions

In this paper, we have demonstrated the usefulness of complete trellis in a template matching approach for detecting pronunciation mismatch. We have shown that the complete trellis provide more cues than using the best path, as done traditionally. The experiments in this paper are done using synthetic speech samples. We plan to investigate the usefulness of the trellis on real data sets, and build a pronunciation learning system using template based approach.

A major question is whether such cues are present in real datasets involving two speakers. Our informal experiments show that such cues exists. Moreover, our approach of building a pronunciation learning system is to have the learner imitate a teacher or a reference template. One could use thresholds suiting to a requirement of a strict or a lenient system during automatic detection of weak spots in the trellis.

References

Delmonte, R. (2000). Slim prosodic automatic tools for self-learning instruction. *Speech Communication*, 30(2–3):145 – 166.

Ehsani, F. and Knodt, E. (1998). Speech technology in computer-aided language learning: Strengths and limitations of a new call paradigm. *Language Learning and Technology*, 2(1):45–60.

Eskenazi, M. (2009). An overview of spoken language technology for education. *Speech Communication*, 51(1):832–844.

Rabiner, L. R. and Juang, B. H. (1993). *Fundamentals of Speech Recognition*. Prentice-Hall.

Stenson, N. et al. (1992). The effectiveness of computer-assisted pronunciation training. *Calico Journal*, 9(4):5–19.