4:53

R13. Decisions about features. Scott M. Brill, Michael S. Phillips, Moshe J. Lasry, and Richard M. Stern (Departments of Computer Science and Electrical Engineering, Carnegie–Mellon University, Pittsburgh, PA 15213)

This paper describes the methods of statistical analysis used to classify letters in a feature-based, speaker-independent isolated letter recognition system. A hierarchical decision structure was implemented so that decisions at each node of the decision tree could be made on the basis of a small number of relevant features. For example, the 26 letters were first classified into vowel categories on the basis of first and second formant frequencies. The specific decisions, and the features used to make them, were selected by a clustering analysis of training data. At each decision node of the recognition system the test utterance was first analyzed using Fisher linear discriminant functions, with threshold weights individually set for each pairwise decision in order to minimize misclassifications. When a decision could not be made with certainty, classification was performed using a maximum likelihood procedure assuming multivariate Gaussian statistics. The sequential use of nonparametric and parametric discriminant functions produced superior classification to that obtained with either of the separate analyses. The overall system structure will be discussed in terms of practical tradeoffs between the number of features used at each decision node and the system's overall probability of error. [Work supported by NSF.]

5:07

R14. A powerful post-processing algorithm for time domain pitch trackers. Philippe Specker (Department of Computer Science, Carnegie–Mellon University, Schenley Park, Pittsburgh, PA 15213)

Existing pitch tracking algorithms are not precise nor reliable enough to be useful in feature-based recognition systems. It is possible, however, to analyze the errors produced by a particular algorithm and then reduce these errors using post-processing techniques. Error patterns were analyzed for a time-domain pitch tracker [W. H. Tucker and R. H. T. Bates, IEEE Trans. **ASSP-26**, 597–604 (1978)] and a post-processing algorithm, based on artificial intelligence techniques, was written in order to eliminate errors. Performance was compared for the original and modified versions of the pitch tracker for a number of speakers using both isolated words and sentences. All types of errors were reduced by the post-processing algorithm. Voiced–voiceless decisions were performed with less than 1% error for 2080 letters produced by males and females. The fundamental frequency microstructure was tracked sufficiently well to be used in extracting phonetic features in a feature-based recognition system.

5:21

R15. Very large vocabulary recognition (VLVR): using prosodic and spectral filters. Alex Waibel (Computer Science Department, Carnegie–Mellon Uiversity, Pittsburgh, PA 15213)

The use of Very Large Vocabularies ($\sim$20 000 words) imposes two major constraints on the design of isolated or connected word recognition systems: the efficient reduction of the large search space to a subvocabulary of manageable size [V. Zue and D. Shipman, J. Acoust. Soc. Am. Suppl. 1 **71**, C7 (1982)] and the robustness of the search space reduction heuristics involved. Psychological evidence suggests that prosodic and robust segmental features may be used as preliminary decision criteria in human speech perception. In this paper we present attempts to apply such heuristics to the design of VLVR systems. As a data base a 20 000 word vocabulary has been compiled providing phonemic, prosodic, and pragmatic information. Based on this corpus, the tradeoffs between the robustness of certain features and their power to reduce the search space have been studied. Our results indicate that combining prosodic information (syllable counts, stress patterns) with a set of robustly detectable features (frication, stops, vowel nuclei of stressed syllable) can reduce the vocabulary size to groups of less than 400 words. Additional potentially useful prosodic features, e.g., rhythmic patterns, are currently being investigated.

5:35

R16. Alignment classification method to facilitate automatic acoustic–phonetic statistics collection. Janet M. Baker (DRAGON Systems, Inc., 173 Highland Street, West Newton, MA 02165)

Automatic labeling methods have been developed which allow speech recognition systems to be trained and tested on very large data bases (up to 140 000 word tokens). But are these automatic labeling methods accurate enough to collect statistics of direct value to speech science? Despite the success of automatic methods in effectively training speech recognition systems, the alignment routines still make a small percentage of gross phone alignment errors—in sharp contrast to human spectrogram or waveform experts. Automatic training procedures seem, nonetheless, to be tolerant of this small percentage of misaligned sounds provided they are given a sufficiently large number of correctly aligned instances of each sound. In this paper, a new method is proposed to directly address the following classification problem: is a particular putative phone alignment correct or is it an alignment error? Applying this classification method to a large data base previously labeled by a conventional automatic routine, acoustic–phonetic statistics for each sound may be obtained for all instances of that sound which are classified as correctly labeled.

5:49

R17. Unifying dynamic programming methods. James K. Baker (DRAGON Systems, Inc., 173 Highland Street, West Newton, MA 02165)

Dynamic programming has come to be widely used and accepted in automatic speech recognition. However, two different but similar applications have often been described more in terms of their differences than their similarities. On the one hand, dynamic programming is used to find the best nonlinear dynamic time warping to align two instances of a word. On the other hand, dynamic programming may be used to find the best state sequence for a hidden Markov process. Not only are these procedures essentially equivalent, but significant generalization comes from an explicit unification. Dynamic programming may be used not only to align two instances of a word, but also to align an instance of a word with an arbitrary finite state model for the word, or even to align two arbitray models. Multiple instances of a word may contribute to a single model, and multiple passes on a finite set of training data can be used to further refine word models.