A NOVEL MACHINE-LEARNING ALGORITHM FOR IMPROVING RECOGNITION PERFORMANCE IN A FEATURE-BASED, DELAYED-COMMITMENT CONTINUOUS SPEECH RECOGNITION SYSTEM

M. O'Kane, P. Kenne, D. Landy and S. Atkins

University of Canberra, Canberra, Australia

ABSTRACT

This paper describes an algorithm for machine learning in a feature-based continuous speech recogniser. This algorithm provides the recogniser with an automatic means of achieving high recognition scores when operating in speaker-independent mode. The learning algorithm operates at a lexical level because the fact the recogniser adheres to the principle of 'delayed commitment' in using speech signal information in making lexical decisions.

1. INTRODUCTION

The introduction by Kai-Fu Lee [1] of successful automatic learning and adaptation algorithms in Hidden Markov model speaker-independent continuous speech recognisers led to demonstrations of very significant improvements in the performance of recognisers using this architecture. To date no such similar learning algorithms have been devised for speaker-independent continuous speech recognisers using rule-based architectures. This paper presents a procedure which addresses the issue of learning techniques for rule-based recognisers.

2. THE RULE-BASED RECOGNISER

The rule-based recogniser used in this work was the FOPHO recogniser [2] which is a completely rule-based recogniser in which new lexical items and quasi-phonetic units alike are added to the system in the form of nested production rules. All rules are written in the special-purpose programming language, WAL (Wave Analysis Language) [3]. A simple rule in WAL involves associating a label with any time-segment of a speech waveform for which a specified signal processing function is true. For example the rule

```
feature
  name : stry,
  wave : speech,
  (association : zcross(20000,50)
      is
   long_high_amplitude(20,100))
end.
```

will give the label "stry" to any portion of a waveform for which the number of zero crossings averaged over a 50 msec time-window remains above 100 for 20 msec or more. Complex rules in WAL cause labels to be associated with time-logical combinations of simple signal processing functions or of other rules, simple or complex, specified either directly or by label reference. Available time-logical operators in WAL are : and, or, not, then, before, after, includes, strict_then, strict_before, strict_after. Rules can also have 'characteristics' (time_long, time_short, extendr, extendl) which can be thought of as extra conditions which rules have to meet in order to 'fire'. Thus the (complex) rule:

```
feature
  name : four,
  wave : speech,
  ((association : speech is f)
  strict_then
  (association : speech is or with
  characteristic time_long (60))
 end.
```

means that in order for the lexical label "four" to be associated with a speech segment, the rule "f" must fire and then the rule "or" must fire for a time interval of at least 60 msec.

One of the fundamental principles of FOPHO is that all rules leading to quasi-phonetic labels should embody the Klatt principle of delayed commitment [3] in that they should fire on all exemplars of a given phonetic unit at the cost of overfiring on a certain percentage of near-confusion sounds to the target sound. Word hypotheses activated by incorrect fires are hopefully eliminated by the requirement that for a given lexical item to be recognised, the phonetic features have to occur in a certain sequence.

The expressive ease with which it is possible to write and test FOPHO rules using WAL, combined with the overfiring consequences of using the delayed commitment approach poses a problem for immediate introduction of formal automatic learning techniques in a system such as FOPHO. In order to proceed with the learning algorithm described here it was first necessary to automate the checking of the correctness or otherwise of rule fires against a labelled speech database and to re-structure the rule file on which automatic learning was to take place.

3. AUTOMATIC CHECKING

The Chk Program takes rule results from the WAL interpreter working on a selected speech database and compares the results with the hand-marked labels in the database. Correct fires, misses, misfires and overfires are all reported and analysed on an utterance-by-utterance basis (the verbose version) or as a summary for the whole database. If required the Chk Program also reports firing results in terms of rule components. For example, for the rule "four" given above, the program will, if the appropriate switch is set, indicate how many times and where both components fire and how many times only one or none of the components fires. This Chk Program forms the core of the learning algorithm.

4. RULE FILE RE-STRUCTURING

In order to make automatic learning easier to implement it has been necessary to re-structure the rule file before learning commences. Under the re-structuring, the "or" time-logical operator is replaced by separate rule statements. Also rule components at the simple rule level are tagged as 'capture' or 'eliminator' rules. The process of phonetic rule development is a process of 'capturing', if possible, all examples of a target phoneme while 'eliminating' as many phonemes as possible that are not in the target set. The tagging referred to above provides an explicit record of this process.

5. PRELIMINARY PROCESSING

To provide a baseline for learning, ordinary rule development for a particular vocabulary is carried out for a single speaker. Rule development for this speaker is typically done over at least twenty examples of each lexical item in spontaneous continuous speech. When the rules for the speaker are operating near-perfectly, they are used as the base set of rules for further work. This provides a suitable baseline for a speaker-independent recogniser. For example, FOPHO rules were developed for one female speaker for the 'hard' vocabulary of the digits and the words "phone" and "double". When these rules were tested on 160 other speakers speaking a total of 250 utterances in spontaneous continuous speech with each utterance typically consisting of strings of fourteen to sixteen lexical items, the average percent correct over all lexical items was

$$\% \text{ correct} = \frac{\text{No. correct}}{(\text{No. correct} + \text{No. misses})} = 63.3$$

while the average misfire to correct ratio was

$$\frac{\text{No. misfires}}{\text{No. correct}} = 0.51$$

Of course the scores for particular lexical items in this vocabulary varied somewhat with the best case being for the word "six" for which the percent correct was 86% and the misfire to correct ratio was 0.60. The worst case was the word "double" where the percent correct was only 44.3% and the misfire to correct ratio was 1.0. By tuning the rules on extensive data for a second speaker, an improvement of 4% was achieved with no change in the misfire to correct ratio. It is on this single-speaker-developed, second-speaker-tuned recogniser that the learning algorithm was tested.

6. THE LEARNING ALGORITHM

The aim of the learning algorithm is to improve recognition scores at the lexical item level. Generally this will lead to improved scores at the quasi-phonetic level although, because of the principle of delayed commitment, this improvement will not necessarily follow.

As indicated in Section 4, the process of recognition rule development is a trade-off process of capturing as many examples as possible of a particular lexical item while simultaneously keeping misfires as low as possible. This trade-off notion is carried through into the learning algorithm.

In order to illustrate the operation of the algorithm we will trace its performance for the rule for the word "four" in the FOPHO digit-vocabulary system for which baseline data is given above. The baseline data for "four" in that system after second-speaker tuning is 67% correct and a misfire to correct ratio of 0.26.

The learning algorithm first considers the percent correct results for the large speaker set for the major (i.e. immediate sub-lexical) components of a lexical rule. In the case of the rule for "four" these components are the rule for "f" and the rule for "or" which have baseline percent correct scores of 91% and 71% respectively for occurrences within examples of "four". The algorithm uses this data to find the weakest component in a rule (in this case "or") and selects this for further processing.

In the next step the algorithm concentrates on eliminator rules for the selected component. If the Chk Program results shows that these eliminators are falsely eliminating any known "four" the learning algorithm proceeds to relax the rule parameters by a fixed percentage (currently 10%) in all the eliminator rules and keeps doing this until no examples are falsely eliminated. Results of this for "four" are given in Table 1 as 'after step 1' results. There is an 8% improvement in the percent correct results although the misfire to correct ratio does not change much because, while the number correct is improving, not as many cases as should be are being eliminated any longer.

In the next step, the algorithm concentrates on the capture rule components and proceeds to relax

both rule and time parameters again by a fixed percentage (10% in this case) for all capture rules for this component. The results of doing this are the 'after step 2' results in Table 1. There has been a 12% improvement in the percent correct score at the cost of almost doubling the misfire to correct ratio. Accordingly, the next step in the algorithm involves an attempt to lower this ratio. This is done by adding new eliminator rules designed to eliminate the major misfire lexical items (in this case "five", "phone" and "one"). These eliminator rules are found by table-lookup, thus this step is only as good as the table contents. The 'after step 3' results show that the table contents probably should be strengthened.

It is not surprising that the major misfire groups for this example are the words "five", "phone" and "one", as in Australian English the truncated start of the words "five" and "phone" do in many cases sound like continuous speech "four", as does the start of "one" when heard following a word ending in /f/. Issues such as these pose a problem in selecting a suitable stopping condition for the learning algorithm. Presently the learning algorithm arbitrarily stops after three steps. An obvious extension is to have it improve performance on the next weakest component. At present the issue of the not-too-surprising misfires is handled by lexical eliminator rules which for this example leads to the FOPHO rule "finalfour":

```
feature
  name : finalfour,
  wave : speech,
  ((((association : speech is four)
  and
  not (association : speech is five))
  and
  not (association : speech is
       phone))
  and
  not (association : speech is one))
end.
```

7. ACKNOWLEDGEMENTS

This work was supported by grants from the Industry, Research and Development Board, the Australian Research Council and Wang Australia.

8. REFERENCES

[1] LEE, K-F., *"Automatic Speech Recognition - The Development of the SPHINX System"*, Kluwer Academic Publishers, Boston, 1989.

[2] ATKINS, S., KENNE, P., LANDY, D., NULSEN, S and O'KANE, M., "WAL - A Speech Recognition Programming Language", *Proceedings of the ICSLP 90*, Kobe, Vol.1, pp233-236, November 1990.

[3] KLATT, D., "Speech perception : a model of acoustic-phonetic analysis and lexical access", *J Phonet, 7*, pp279-312, 1979

[4] O'KANE, M., "The FOPHO Speech Recognition Project", *Proceedings of the Eighth International Joint Conference on Artificial Intelligence*, Karlsruhe, pp630-632, 1983.

	% correct	misfire/correct ratio
Baseline data	67	.26
After step 1	75	.27
After step 2	87	.50
After step 3	87	.43

Table 1 : Learning algorithm results for "four".