Proceedings of The Institute of Acoustics

A PROBABILISTIC CONTROL STRATEGY FOR PARSING MULTIPLE STRINGS ON A LATTICE

G. Holmes, A. J. Hewett and S. J. Young.

Cambridge University Engineering Department.

INTRODUCTION

A hierarchical speech understanding system (SUS) with the capability of producing 'n best strings' in the form of a lattice as output from the recognition component must contain a control algorithm to restrict the search space of the lattice when parsing. The control algorithm confronts the problems of queueing the alternatives and placing a reliability measure on the phrase structures formed by the parser.

In this paper we describe an experimental model of a SUS with a limited vocabulary (the digits 0 to 9), which uses a connected word template matching algorithm modified to generate multiple solutions. The output of the recognition component is thus a lattice of alternative word matches (see fig. 1). The quality scores obtained from the recognition component are converted into likelihood estimates. The control algorithm uses these likelihood estimates to queue the alternatives and to fit a reliability measure to the individual word matches and the phrase structures formed during parsing.

A brief description of the system is given followed by an example of the control algorithm and the parsing process.

SPEECH UNDERSTANDING SYSTEM

```
            ------------  Word       ------   Phrase       ---------
          | Connected   | Lattice  | Chart  | Structures  | Higher    |
Speech -->| Word        |--------->| Parser |------------>| Level     |
          | Recogniser  |          |        |             | Knowledge |
            ------------             ------                ----------
```

fig.1 - SUS Model

The recognition component of the system performs template matching using the Itakura distance measure and 15-pole linear prediction analysis. The dynamic programming algorithms for producing a recognition lattice of the form discussed in the next section are described in [1].

WORD LATTICE

We define a word lattice to be a set of n alternative word matches $W_1,\ldots,W_n$. Each word W_i is a 5-tuple $\langle t,s,e,d_i,l,w\rangle$ where t is a template number, s a start frame, e an end frame, d_i a quality score derived from the DP algorithms, l a likelihood estimate and w is a word name. The lattice is ordered on least cumulative quality score because this quantity is the best indicator of the performance of the DP algorithms. The likelihood estimate is calculated for use by the control algorithm because it forms a better framework for asking questions like; how good is score x ?, and how good is the combined score of x and y ?

LIKELIHOOD ESTIMATION

The likelihood estimation is obtained in the following way. For each word W_i in the vocabulary we construct a probability distribution function P_i by employing the method described in [2] (with the exception that the DP algorithm used in constructing the probability density functions is a version of the Bridle and Brown algorithm [3], this is used to make the analysis consistent with [1]). The quality score d_i is the minimum of the distances produced by the reference templates for W_i (five reference templates are used per word). The likelihood function $l(W_i)$ is obtained from the equation

$$l(W_i) = 1 - \int_0^{d_i} P_i(x)\,dx \qquad (1)$$

Since no constraints are placed on word order during recognition, we make the assumption that the string likelihood of a set of words is simply the product of the individual likelihoods of those words. Clearly, there is a need to incorporate syntax statistics and string wellformedness into this measure for a more representative string likelihood function. Furthermore the statistics used in calculating P_i are obtained from isolated word recognition, and so do not perfectly reflect the situation of connected word recognition (this has the effect of producing relatively low values for equation (1)). We note these problems and hope to resolve some of them at a later date, the result we wish to present here is the control algorithm and parsing process.

CHART PARSING

The parsing mechanism employed in this discussion is the chart parser. A full description, plus code for this type of parser is given in [4]. The parsing method we use is a bottom-up version which has the option of using lookahead to restrict the parallelism of the parser. Efficiency is important since parsing a lattice involves much more search than conventional natural language parsing.

CONTEXT FREE GRAMMAR

The grammar is written in EBNF. The sample grammar below is used in the parsing example given later.

```
digit = Xzero | Xone | Xtwo | Xthree | Xfour | Xfive |
      = Xsix | Xseven | Xeight | Xnine
Lcode = Xzero Xone
Ccode = Xzero Xnine
Ecode = Xzero Xtwo
    L = Lcode digit digit
    C = Ccode digit digit
    E = Ecode digit digit
```

The dictionary used by the parser simply indicates that the word 'one' say, has terminal category Xone.

CONTROL ALGORITHM

The control algorithm below uses a heuristic threshold to limit the selection of word matches. The threshold value is used because it is better to return partially complete reliable strings than spanning utterances containing poorly recognised words. The control algorithm is given by

Step 1 : Begin at the left most start frame, call it x.

Step 2 : Select the set S_j with start frame x and likelihoods above the threshold. Ask if the pre-terminal categories of S_j can start any rule. If yes then create active edges for these with the proviso that - if any of these rules have terminal symbols as the next symbol along from the start symbol then perform a simple lookahead to see if this can be satisfied by any of the inactive edges of the chart - if not then do not create an active edge for that rule (this condition restricts the parallel activity of the parser). Move sequentially to the end frames of the words in S_j. Ask if the end frame is the end of the speech for all of the S_j. If yes then stop else set x equal to the end frame and goto step 2.

A successful parse occurs if there exists one (or more) inactive edges which span the entire utterance. If this is not the case then the largest spanning utterance may be returned (or some other error recovery mechanism may be invoked).

This procedure is best illustrated through a simple example. Here we present the initial, mid-process and final lattice configurations.

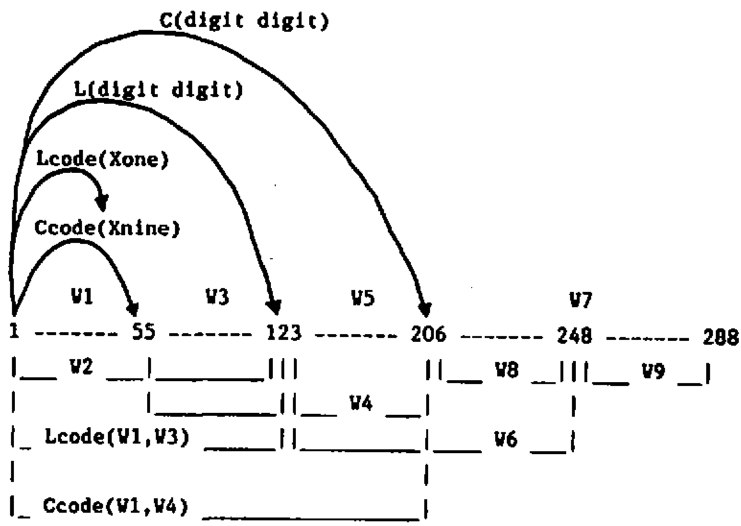
Initial Lattice

W_1 = <3,1,55,40247,0.490018,zero>
W_2 = <10,1,123,84567,0.232858,two>
W_3 = <4,55,123,24973,0.519384,one>
W_4 = <2,55,206,36750,0.844644,nine>
W_5 = <12,123,206,24975,0.752321,three>
W_6 = <23,123,248,42045,0.590458,four>
W_7 = <41,206,288,39906,0.338721,four>
W_8 = <19,206,248,44245,0.186734,one>
W_9 = <6,248,288,27656,0.659972,seven>

LATTICE PARSING STRATEGY

In the lattice representation below inactive edges are those drawn below the initial line and active edges those drawn above. An inactive edge is one which requires no further processing to become complete, i.e. if a right hand side of a rule is satisfied or if the edge is a member of the initial lattice, and their contents are given in parentheses. The active edges require further syntactic processing and these are represented below by category (left hand side) with their requirements for completion given in parentheses.

Lattice after algorithm has progressed to speech frame 206



For clarity the immediate creation of inactive edges with the digit category is omitted. Active edges for the rule Ecode is not generated since there are only two possibilities for continuation at speech frame 123, namely with V_3 and V_4. Note that V_8 (although syntactically valid) will not be persued since its likelihood estimate falls below the threshold (currently set at 0.2).

LATTICE PARSING STRATEGY

Final Lattice

```
     V1         V3         V5              V7
1 ------- 55 ------- 123 ------- 206 ------- 248 ------- 288
|___ V2 ___|_________|||          ||___ V8 ___|||___ V9 ___|||
|          |_________||___ V4 ___|          |           ||
|_ Lcode(V1,V3) _____||__________|____ V6 ___|           ||
|                                |                       ||
|_ Ccode(V1,V4) _________________|                       ||
|                                                        ||
|______________ L(V1,V3,V5,V7) __________________________||
|                                                         |
|______________ L(V1,V3,V6,V9) ___________________________|
```

The result edges which span the entire utterance are given by

```
(L - 0.099178
   (Lcode - 0.2545075
          (Xzero - 0.490018) (Xone - 0.519384))
   (digit - 0.590458
          (Xfour - 0.590458))
   (digit - 0.659972
          (Xseven - 0.659972)))

(L - 0.0648553
   (Lcode - 0.2545075
          (Xzero - 0.490018) (Xone - 0.519384))
   (digit - 0.752321
          (Xthree - 0.752321))
   (digit - 0.338721
          (Xseven - 0.338721)))
```

and the partially spanning result edges are given by

(Ccode - 0.4138907
(Xzero - 0.490018) (Xnine - 0.844644)) + V_7

(Ccode - 0.4138907
(Xzero - 0.490018) (Xnine - 0.844644)) + V_9

Our present parser would rank and return the complete edges (labelled L in the above) before the fragmented ones (labelled Ccode in the above). Here we see the need for including some measure of syntactic formation into the scoring function. It is still not clear, in the context of the application, which ordering to adopt when returning inactive edges following parsing.

The individual scores of the terminal symbols are converted to labels (good,bad,indifferent) as a measure of the reliability we are placing on them. This is achieved by thresholding. However, we have already mentioned the problems associated with attempting this labelling using the estimate represented by equation (1). We, therefore, defer any comment on our results so far until further experiments with different measures have been performed.

Note that the chart parser provides a good framework for error recovery within a SUS. Here, for example, we could return the Ccode edge and W_7 with their corresponding estimates. The result being that the higher level knowledge could use this information in its own strategy for disambiguating the requests of the user. For example, it could accept the Ccode edge and then ask a specific question to the user in order to obtain the final two digits.

CONCLUSION

We have presented a control algorithm and parsing process which can provide a framework for error recovery based on likelihood estimates within a SUS. The parallel activity of the parser and the combinatorics of parsing a lattice are reduced by lookahead and thresholding, respectively.

We have already mentioned the problems associated with using equation (1) as a likelihood measure for connected word recognition. Further, when this measure is used for speaker independent connected word recognition the scores it produces are relatively low. This does not affect the efficiency of the control algorithm and parsing processes but does make the reliability estimation more difficult.

Future work will concentrate on improving the statistics used in the likelihood measure to more accurately reflect connected word recognition, and also to include syntax statistics and syntactic vellformedness into string likelihood estimation.

ACKNOWLEDGEMENTS

This work vas supported by a joint SERC/British Telecom grant (BT/SERC Contract No. SR/D/4361.7). Acknovledgement is made to the Director of Research at British Telecom for permission to publish this paper.

REFERENCES

[1] Young, S.J. "Generating Multiple Solutions from Connected Word DP Recognition," Proc. Institute of Acoustics., 1984.

[2] Hewett, A.J., Holmes, G. and Young, S.J. "Dynamic Speaker Adaptation in Speaker-Independent Word Recognition," Proc. Institute of Acoustics., 1986 see these proceedings.

[3] Bridle, J.S., Brown, M.D. and Chamberlain, R.M. "An Algorithm for Connected Word Recognition", IEEE ICASSP 1982, pp. 899-902.

[4] O'Shea, T., Eisenstadt, M. "Artificial Intelligence Tools, Techniques and Applications", Harper and Row 1984.