SUPRASEGMENTAL DURATION CONTROL WITH MATRIX PARSING IN CONTINUOUS SPEECH RECOGNITION

Harald Singer and Shigeki Sagayama

ATR Interpreting Telephony Research Laboratories

ABSTRACT - This paper describes a unified framework for continuous speech recognition (CSR) under grammatical constraints, where trellis calculations and parsing are performed by the same simple fundamental operations, namely multiplication and addition of likelihood matrices. The matrix parser is shown to be a generalization of the CYK parser, which because of its simplicity lends itself to efficient hardware implementation. It also facilitates explicit supra-segmental duration control for all grammatical categories. Preliminary results showed, that improved duration control on the mora level raised the recognition accuracy from 86.6 % to 88.2 %.

INTRODUCTION

An important research issue in CSR is the question, of how to combine the acoustic model with the language model or, in other words, how to efficiently incorporate the grammar in the search algorithm without losing optimality (in a Viterbi sense).

Previous HMM or NN based search algorithms like the HMM-LR (Kita et al., 1989) use duration control on the state or phoneme level. Well-known durational compensation effects on other levels (c.f. mora-timing) suggest duration control also for non-terminal symbols. Incorporation of these supra-segmental durations in a LR or a FSN (finite state network) based recognition system is however quite difficult. Furthermore these CSR systems perform the same likelihood calculations over and over again as the same non-terminal symbols appear in different hypotheses or branches of the network. Thus, pruning of hypotheses becomes necessary and may introduce additional errors.

This paper is organized as follows: First, we will derive the matrix parser from the CYK parser, then introduce matrix parsing for CSR, next formulate a unified framework for linguistic parsing and acoustic likelihood calculation, and finally report experimental results using mora duration control.

DERIVATION OF THE MATRIX PARSE ALGORITHM

From the Cocke-Younger-Kasami (CYK) Algorithm to the Matrix Parser

A reformulation of the classical CYK algorithm (Fu, 1982) is given in Fig. 1. The left side of Fig. 3 shows an example grammar and the resulting parse table L for input string `I read a book`. First, the terminals are rewritten to non-terminals on the first minor diagonal, e.g. `I` to `N`, `read` to `V` etc. . Next, all cells above the diagonal are visited in the order shown by the dotted arrow line. For each cell, all possible rules are tried, e.g. for cell 0,2 the elements of cells 0,1 and 1,2 are checked if they comply to any grammar rule. In this case only the rule `S` $\longrightarrow$ `NP VP` is applicable. The set of symbols for cell 0,2 thus consists of the non-terminal `S`. The same operation is performed for all cells above the diagonal. If, at the end of the parse, the cell 0,N (here 0,4) contains the sentence symbol `S` the parse has been successful, i.e. the input string is a valid sentence in this grammar.

The loops over position indices i, j and the combination of $l_{i,k}, l_{k,j}$ show some similarity to matrix multiplication. To explore this idea further, we changed the order of operations, i.e. we loop first over all grammar rules and then over position indices i, j. Looping over i, j can be implemented as a matrix operation. This is a key difference to a previously reported generalization of the CYK parser (Ney, 1987).

Fig. 2 shows the matrix parse algorithm. Formally, each symbol A is represented by a $N+1, N+1$ matrix L_A, with N the number of input symbols in the string to be parsed. First, the terminal symbol matrices L_A are initialized. The grammar is then traversed recursively, top-down and depth-first, starting at the

```
Given a Chomsky normal form context-free
grammar $G = (V_N, V_T, P, S)$, $A, B, C \in V_N$
and an input string $w = a_1 ... a_N$

for $i = 1$ to $N$
  $l_{i,i-1} = \{A | A \to a_i\}$
for $i = 1$ to $N$
  for $j = i - 1$ down to 0
    $l_{i,j} = \{A | (\exists_{\substack{k \\ i<k<j}} \exists_{B \in l_{i,k}} \exists_{C \in l_{k,j}} A \to BC) \vee$
        $(\exists_{B \in l_{i,j}} A \to B)\}$
```

Figure 1: CYK algorithm

```
for $i = 1$ to $N$
  $l_{A_{i,i-1}} = \{1 | A \in V_T, A = a_i\}$
GETL($S$)

subroutine GETL($A$)
  if $A \in V_T$ return $L_A$
  $L_A = 0$
  forall $R$ LEFT($R$) $= A$
    $L = I$
    forall $W \in$ RIGHT($R$)
      $L = L \cdot$ GETL ($W$)
    $L_A = L_A + L$
  return $L_A$
```

Figure 2: Matrix parse algorithm

"sentence" symbol s by applying subroutine GETL(A). Rules of the form $A \longrightarrow BC$ are equivalent to matrix multiplication $L_A = L_B L_C$, $A \longrightarrow B|C$ are equivalent to matrix addition $L_A = L_B + L_C$.

L_A stands for matrix of symbol A, $l_{A_{i,j}}$ is the value of the ith column, jth row in the matrix for symbol A, 0 and I are null-matrix and identity matrix, respectively, R stands for rules, LEFT(R) is the symbol on the left-hand side of rule R, $\cdot$ and $+$ are matrix multiplication and addition. To recover the parse tree, an additional traceback matrix for each non-terminal grammar symbol specifying rule number and position index is required.

Fig. 3 suggests a simple interpretation of the matrix parse algorithm: Imagine the parse table of the standard CYK parser being split up into different parse tables for each grammar symbol. Each cell now contains only one value (for unambiguous grammars either 0 or 1) instead of a set of grammar symbols. Parsing then consists of simple matrix multiplications. The parse has been successful, as $l_{S_{0,N}} = 1$.

Matrix Parser for Continuous Speech

The algorithm can easily be enhanced to "noisy" input sequences, where the input is given as a vector sequence of symbol observation probabilities and not as a sequence of character strings. A typical application is frame-by-frame phoneme spotting. Basic elements are now square matrices of likelihoods for each grammar symbol A, i.e. each terminal (here phoneme) and each non-terminal (e.g. noun, adverb etc.).

$$L_A = \begin{pmatrix} 0 & l_{A_{0,1}} & l_{A_{0,2}} & \cdots & l_{A_{0,N}} \\ & 0 & l_{A_{1,2}} & \cdots & l_{A_{1,N}} \\ & & 0 & \cdots & l_{A_{2,N}} \\ & & & \ddots & \vdots \\ \text{\Large 0} & & & & l_{A_{N-1,N}} \\ & & & & 0 \end{pmatrix}$$

The matrix element $l_{A_{t,s}}$ stands for the likelihood that symbol A was uttered between time points t and s, consuming observation frames $t+1$ to s. N stands for the total number of observation frames. The likelihood values can be obtained, for example, by a phoneme spotting neural network (Singer and Sagayama, 1992).

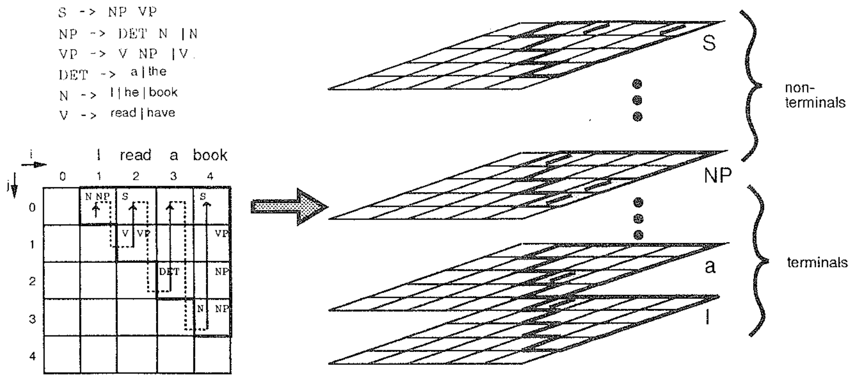

Figure 3: Example grammar, CYK parse table and symbol matrices (only non-zero entries are shown) for input string "I read a book"

Speech recognition is often formulated as an optimal path search problem, where likelihood values for different paths are required instead of their sum. Viterbi-type optimal path search, instead of a trellis-type algorithm, can also be defined by a matrix representation. **Viterbi matrix multiplication** and **Viterbi matrix addition** are defined as

$$\begin{array}{ll} \textbf{Multiplication}: & l_{AB_{t,s}} = \max\limits_{t<\tau<s} l_{A_{t,\tau}} l_{B_{\tau,s}} \\ \textbf{Addition}: & l_{AB_{t,s}} = \max(l_{A_{t,s}}, l_{B_{t,s}}) \end{array}$$

Matrix Representation of HMM

Generalization of the matrix parsing can be carried one step further reformulating trellis (or Viterbi) likelihood calculations inside the HMM's as matrix operations. Let us consider a state in an HMM. The likelihood matrix of an HMM state for a single clock has non-zero components only in the upper diagonal components:

HMM state transition. The likelihood matrix of a transition from state i to j of an HMM to observe x_t is

$$Q_{ij}(\mathbf{x}) = a_{ij} \begin{pmatrix} 0 & b_{ij}(x_1) & & & \mathbf{0} \\ & 0 & b_{ij}(x_2) & & \\ & & \ddots & \ddots & \\ & & & 0 & b_{ij}(x_N) \\ \mathbf{0} & & & & 0 \end{pmatrix}$$

where a_{ij} is the transition probability and $b_{ij}(x_t)$ is the output probability.
The likelihood matrix of a null transition from state i to j is independent of the observation x_t

$$Q_{ij0}(\mathbf{x}) = a_{ij0} I$$

where a_{ij0} is the null transition probability, and I stands for the identity matrix.

In the case of self loops ($i = j$), the likelihood matrix of staying at state i for one clock time is given by Q_{ii} letting $i = j$ in the above matrix. The likelihood matrix of staying at i for two clocks is Q_{ii}^2, which has non-zero values only at the second upper off-diagonal components. Generally, the likelihood matrix of staying at state i for exactly k clocks is given by Q_{ii}^k. Therefore, if the maximum duration of the observed signal is N clocks, the likelihood of staying at i for arbitrary time is given by the triangular matrix:

$$P_{ii} = \sum_{k=1}^{N} Q_{ii}^k = \begin{pmatrix} 0 & a_{ii}b_{ii}(x_1) & a_{ii}^2 b_{ii}(x_1)b_{ii}(x_2) & \cdots & a_{ii}^N b_{ii}(x_1)\cdots b_{ii}(x_N) \\ & 0 & a_{ii}b_{ii}(x_2) & \cdots & a_{ii}^{N-1} b_{ii}(x_2)\cdots b_{ii}(x_N) \\ & & 0 & \cdots & a_{ii}^{N-2} b_{ii}(x_3)\cdots b_{ii}(x_N) \\ & & & \ddots & \vdots \\ & & & & a_{ii}b_{ii}(x_N) \\ 0 & & & & 0 \end{pmatrix}$$

Fig. 4 shows an HMM (4-state 3-loop with null transitions), which is defined by the likelihood matrices P_{11}, P_{22}, P_{33} of self loops at states 1, 2, and 3 and the likelihood matrices Q_{12}, Q_{23}, Q_{34} of inter-state transitions. The likelihood matrix P for this left-right HMM is computed by

$$P \;=\; (P_{11} + I)(Q_{12} + Q_{120})(P_{22} + I)(Q_{23} + Q_{230})(P_{33} + I)(Q_{34} + Q_{340})$$

where I stands for the identity matrix.

Min-max type duration control can be explicitly implemented by ignoring matrix components outside the legal duration range, i.e. setting to zero of matrix elements with less than a minimum duration and more than a maximum duration. This is applicable to all levels such as states, phonemes, and words and is also helpful to reduce the computational cost.

A more sophisticated way of duration control can also be implemented: as phoneme durations are not evenly distributed between minimum and maximum thresholds but are closer to a Gauss distribution, likelihood values can be weighted with a Gaussian window using mean and standard deviation duration values (see Fig. 5). Obviously, this is not limited to Gaussian distributions, but any other distribution, e.g. a gamma distribution, could also be used.

EXPERIMENTS

Recognition experiments were performed with a task-dependent context free grammar of 582 rules. The phoneme test set perplexity was about 3.8, word test set perplexity was about 150. Due to implementational considerations recursions were removed from the grammar.

As we want to compare the HMM matrix parser with the HMM-LR parser we tried to stay as close to the experimental conditions for the HMM-LR as possible. The following details must therefore be considered:

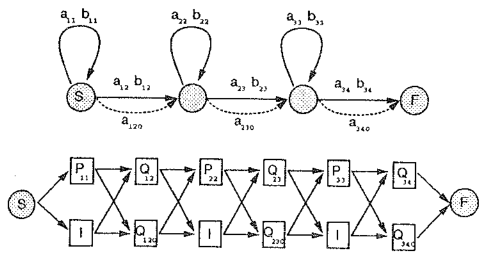

Figure 4: HMM topology and corresponding signal source network for HMM with null transitions

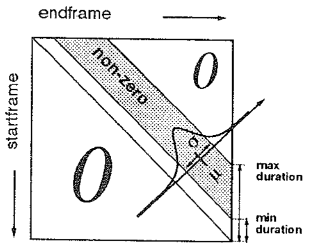

Figure 5: A likelihood matrix with minimum and maximum duration constraints (only shaded components are non-zero)

```
<utterance>[1,48]:          -32.51
  Q[1,5]:                   -31.43
  <p>l[5,43]:               -31.65
    <np>l[5,43]:            -31.65
      <n>b[5,22]:           -26.28
        k1[5,9]:            -29.59
        o[9,12]:            -17.55
        r[12,15]:           -25.73
        e[15,22]:           -28.36
      <suffix>e[22,43]:-36.00
        <wa$>e[22,43]:-36.00
          w[22,30]:  -35.29
          a3[30,43]: -36.44
  Q[43,48]:                 -39.87
```

Figure 6: Parse tree for utterance /korewa/ with log likehood values

```
<utterance>[1,48]:                -32.51
  Q[1,5]:                         -31.43
  <p>l[5,43] :                    -31.65
    <np>l[5,43]:                  -31.65
      <n>b[5,22]:                 -26.28
        <ko@>b[5,12]:             -24.43
          k1[5,9]:                -29.59
          o[9,12]:                -17.55
        <re@>[12,22]:             -27.57
          r[12,15]:               -25.73
          e[15,22]:               -28.36
      <suffix>e[22,43]:           -36.00
        <wa$>e[22,43]:            -36.00
          <wa@>e[22,43]:-36.00
            w[22,30]:   -35.29
            a3[30,43]: -36.44
    Q[43,48]:                     -39.87
```

Figure 7: Parse tree for utterance /korewa/ with log likelihood values (mora level added)

- For the vowels, the LR parser distinguishes between phrase initial/medial and phrase final position. The decision which HMM model to use for the verification is made according to the grammar, that is if the phrase is allowed to end after the current vowel.
- For the consonants (ch, ts, p, t, k, b, d, r), the LR parser distinguishes between phrase initial and medial position. For these consonants phrase final position is not possible in Japanese.

In the matrix parser, we choose to implement these rules not inside the parser but to change the grammar accordingly. These changes have also to be made for non-terminals. The number of grammar symbols thus increased about 25 % to 150 symbols (61 terminals, 89 non-terminals).

53 phoneme HMM's were trained on a labelled Japanese database of 5240 common words and 216 phonetically balanced words for one speaker. The models were 3-state, 3-mixture Gaussian continuous output probability density function HMM's. All models were left-to-right without any null transitions, except for unvoiced /u/ and /i/, which were allowed to have null transitions (see Fig. 4). A rudimentary duration control limited the minimum duration of a phoneme to 3 frames, i.e. 30 ms., except for unvoiced /u/ and /i/ with a minimum duration of 0 frames, i.e. unvoiced /u/ and /i/ can be skipped.

The HMM matrix parser achieved a recognition rate of 86.6 % with this rudimentary duration control. As expected this result was completely identical to the result achieved with an LR parser where the beam width had been chosen large enough so that no pruning occurred (in this experiment beam width was set to 1000).

Fig. 6 shows a parse tree for the phrase /korewa/. Non-terminal symbols are marked by <>. The letters b, e, l at the end of non-terminals stand for "begin", "end" and "lone". `<np>l` is a "lone" noun phrase, i.e. it can only be preceded and followed by a pause (`Q`).

In a slightly more sophisticated duration control implementation, mean and standard deviation duration values for each phoneme are calculated from the even-numbered words of 5240 isolated spoken words and then adapted to the faster phrase utterance speed. Minimum and maximum duration are set to half and twice the mean duration values, respectively. The effect of this type of duration control had however a negative effect and reduced the recognition rate to 83.2 %.

We then defined a **mora** level inside the grammar, i.e. rewriting rules like `<n> -> s o k o` were changed to `<n> -> <so@> <ko@>, <so@> -> s o, <ko@> -> k o`. The resulting grammar had an additional 205 mora type non-terminal symbols, raising the number of non-terminal symbols to 294. Fig. 7 shows the parse tree for the same phrase /korewa/ with the morae `<ko@>`, `<re@>` and `<wa@>`.

Duration control on the mora level with minimum and maximum duration values estimated from a large continous speech data base raised the recognition rate to 88.2 % and showed the effectiveness of supra-segmental duration control, a feature that can not easily be implemented in the LR parser.

CONCLUSION

In this paper we showed that the matrix parser can be interpreted as a generalized CYK parser. A unified framework for continuous speech recognition has been developed, where HMM trellis and Viterbi calculations, and parsing are performed by the same simple matrix operations. As these operations are very regular they can efficiently be implemented on hardware. Another advantage of the matrix parser compared to other parsers like the LR parser is that duration control can easily be implemented not only on the state or phoneme level, but on any desired grammar level. Using this feature for implementation of **mora timing** raised the recognition accuracy from 86.6% to 88.2 %.

A major problem, however, is the reliable estimation of μ and σ for the duration of supra-segmental non-terminals as rare morae like /pya/ don't exist in sufficient number in our database. In future research, we will address this problem. Another drawback is the huge computational cost incurred: processing time is on the order of N^3, required space is on the order of N^2, where N is the number of input frames. We expect nevertheless, that real-time performance for a hardware implementation can be achieved if a proper pruning strategy is used.

REFERENCES

Fu, K.S. (1982) *Syntactic Pattern Recognition and Applications*, Prentice-Hall.

Kita, K., Kawabata, T., and Saito, H. (1989) *HMM Continuous Speech Recognition Using Predictive LR Parsing*, Proc. ICASSP-89, pp.53–56.

Ney, H. (1987) *Dynamic Programming Speech Recognition Using a Context-Free Grammar*, Proc. ICASSP-87, pp.69–72.

Sagayama, S. *A Matrix Representation of HMM-based Speech Recognition Algorithms*, Proc. Euro-Speech'91, pp. 1225, Genoa.

Singer, H. and Sagayama, S (1992) *Matrix Parsing Applied to TDNN-Based Speech Recognition*, Proc. ASJ Spring Meeting, 3-1-10, pp.89-90.