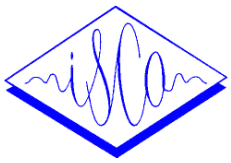

First International Conference on
Spoken Language Processing (ICSLP 90)
Kobe, Japan
November 18-22, 1990

A One–Pass Search Algorithm for Continuous Speech Recognition directed by Context–Free Phrase Structure Grammar

Michio OKADA
NTT Basic Research Laboratories.
Musashino-shi, Tokyo, 180 Japan

ABSTRACT

This paper presents an efficient parsing algorithm for integrating the search problems both in speech and language processing in general use for speech understanding systems.

The parsing algorithm we propose is regarded as an extension of the finite–state–network–directed, one–pass search algorithm to the one directed by a context–free grammar with retention of the time–synchronous procedure. Our attempt is to embed an active chart parser in the one–pass search algorithm for top–down prediction. Extended formalization of the chart parser makes it possible to incrementally generate both new vertices and multiple lexical edges based on a data–driven beam search technique.

Moreover, some extensions of the algorithm are discussed. This paper shows that the one–pass search algorithm proposed can be extended to a time–synchronous Viterbi–style beam search procedure that is guaranteed to find the N most likely sentences under the constraints of a unification-based phrase structure grammar.

1 Introduction

This paper presents an efficient parsing algorithm for integrating the search problems both in speech and language processing in general use for speech understanding systems.

The roles of the syntactic parser for continuous speech recognition systems are not only to derive syntactic and semantic structures from spoken natural language, but also to produce predictive information for perplexity reduction in order to reduce the search spaces using all the linguistic knowledge as a constraint.

Moreover, the efficient search algorithms at many levels in continuous speech recognition have an important role for handling the difficulties such as recognition uncertainty and ambiguity associated with the acoustic–phonetic processing, and syntactic/semantic ambiguities associated with linguistic processing. In addition to these difficulties, the syntactic search algorithms have to deal with a loose grammatical structure of spoken language, simultaneously.

2 Integration of speech and language processing

Several search algorithms and language models for incorporating linguistic processing into the acoustic–phonetic process itself, have been proposed. The most essential search algorithms in continuous speech recognition are several connected word recognition algorithms[1–4] based on dynamic programming algorithms. These algorithms are guaranteed to find an global optimal word sequence, and can be modified to deal with syntactic constraints in terms of a finite state syntax.

The most successful continuous speech recognition systems[5–7] use the statistical language models, such as Trigram, Bigram, as a weak grammatical constraint.

The Viterbi–style HMM decoding algorithm with between–word transition rules in terms of the trigram grammar, is basically identical to the finite–state–network–directed, one–pass dynamic programming search algorithm for connected word recognition.

On the other hand, in the domain of speech understanding, the use of the strict grammatical language models in addition to the weak constraints should be needed both for finding out the exact meaning or sentence structure, and for predicting succeeding words, phrases and utterances using all the possible knowledge sources[10, 11].

So far, there are several different approaches for the use of the syntactic constraint based on context–free grammar in search algorithms for continuous speech recognition.

Although the lattice parsing approach[8] has often been employed in continuous speech recognition systems previously proposed, one of its drawbacks is that, with its attempt to spot blindly all the words, it cannot take account of predictive information in parsing processes for creating a word lattice.

Meanwhile, Ney's C.K.Y.–based parsing algorithm[9] can find a global optimal solution under the context–free grammar constraint. The computational complexity of this algorithm, however, is proportional to the cube of the length of utterance and proportional to the number of lexical items, inevitably making it difficult to extend it for treatment of tasks with large vocabulary.

The parsing algorithm we propose is regarded as an extension of the finite–state–grammar–directed one–pass search algorithm[4] to the one directed by a context–free grammar with retention of the time–synchronous procedure.

To do this, we modified the active chart parser[12] to incrementally generate sub–networks which grammatically guide the one–pass search algorithm. Particular attention is given to the algorithm to bundle sub–sentence alternatives with the same grammatical functions in sub–networks in order to avoid tree structure formation in the search.

This integrating algorithm can approximately find out global optimal sentence hypotheses, and also have no overhead such as the hierarchical systems based on the lattice parsing approach[8]. The computational complexity of this algorithm is proportional to the length of input speech, but it is independent of the number of words and the length of words. Moreover, it is easy to extend to deal with the unification–based grammar[13], which can integrate syntactic information with semantic information and dynamically varying constraints in dialogue situation[11].

3 The context–free–grammar–directed, one–pass search algorithm

3.1 One–pass search algorithm

The one–pass dynamic programming search algorithm is the most efficient optimal search algorithm for connected word recognition, and is formulated to deal with syntactic constraints based on finite state networks in a frame–synchronous process.

This syntax–directed search algorithm requires pre–compiled finite state networks for finding out a global optimal word sequence. For the context–free grammar, it is impossible to compile the context–free rules into the finite state network. Consequently, the computation of the search algorithm for dealing with the context–free grammar can not be finished infinitely.

Active zones in the one–pass algorithm

The data–driven dynamic programming beam search technique can significantly reduce the amount of computation by pruning the DP scores (the cumulative likelihood) which are removed from further consideration. In this method, the range of the cumulative likelihood computation in current input frame, which is called "active zone", is decided in consideration of the cumulative likelihood at the previous input frame. In this approach, we use a data driven beam search technique in order to deal with context–free grammar in addition to reducing the amount of computation.

Incremental sub–network generation

Given the finite state network for guiding the one–pass search algorithm, the several active zones on the finite state network drift time–synchronously towards the final state. If it is possible to dynamically extend the succeeding sub–networks at any node, to which the active zone has reached, it does not need to initially provide the full–network such as context–free grammars.

3.2 An algorithm for sub–network generation

In order to direct the one–pass search algorithm under the constraints of the context–free grammar, it is needed to design a framework for generating the suitable sub–network which can deal with the recursive rules and sub–sentence alternatives.

Active chart parser

The active chart parser has a data structure that is called the chart which is suitable for an Earley–style, top–down parsing. Chart has two kinds of edges and vertices: The well–formed, sub–string table (WFST) which is used to deal with structural ambiguities and to avoid duplicative parsing, is represented by passive edges; and the predictive information at any vertex is stored in active edges.

Our attempt is to embed an active chart parser into the one–pass search algorithm based on the generated sub–network using top–down prediction of the parser.

Sub–sentence alternatives

The conventional active chart parsing algorithm was designed to parse a sequence of words, and word lattice (sets of word hypotheses).

The formalization of the extended chart parser for unknown spoken sentences makes it possible to incrementally generate both new vertices and multiple lexical edges controlled by a data–driven beam search technique.

The parsing process is started with an initial vertex and an active edge associated with start symbol (for example, "S" which stands for sentence). Then, the phrasal and lexical active edges which are located at initial vertex, are derived from the start symbol. The words which can be located at the head of sentences, are predicted by these derived lexical active edges at the initial vertex. And then, the one–pass search is performed for each predicted word in a time–synchronous process. If an active zone can reach to the end of the frame of a word belonging to a lexical edge, a lexical passive edge with an new vertex is created from that active edge. The chart parser moreover tries to execute the extension and prediction process using new added passive edge for providing the active lexical edges at the newly created vertex.

Due to the recognition uncertainty, a lot of lexical passive edges may be derived from one vertex as alternatives.

Bundling edges and merging vertices

If the end vertices of each lexical passive edge are independently created, the number of the vertices included in the chart can be combinationally quite large. To avoid those combinational explosions, it is easy to use pruning strategies to reduce the search space further. However, it requires large beam width for finding out a solution with high optimality. Moreover, the duplicative extensions of the chart are repeatedly attempted in tree structured search space.

The algorithm proposed here merges the end vertex of passive edges with the same start vertex and the same syntactic functions in order to avoid tree structure formation in the search space. For the rewriting rules of context–free grammar the new passive edge adds to the chart if the edge which has the same left–hand side symbol of rule and the same start vertex already exists, then the end vertex of both the new passive edge and its daughter edges is merged with the end vertex of the edge already presented. This merging process will have an important role in achieving efficient parsing of an ambiguous spoken language. And, it makes the junctions of lexical passive edges on the chart, which has an important role for the one–pass search algorithm as between–word transition rules.

Left–recursive rules

One of the most difficult problems in sub–network generating algorithms is to deal with recursive rules in context–free grammar.

The active chart parser itself is able to avoid the nontermination problem in left–recursive rules using the well–formed, sub–string table. On the other hand, the chart parser with merging process has the passive edges with loops for left–recursive rules. For example, the two rules, $\underline{A \rightarrow A\ B}$ and $\underline{A \rightarrow a}$, are alternatives for the left–hand side symbol A. The passive edge for phrase B has the same vertex for both start and end vertex.

The vertices with the self–looped edges are obviously independent of the time axis of input an utterance, and are corresponding to the states in a grammatical network.

The sub–networks for guiding the one–pass algorithm are constructed by the vertices and the lexical edges of the chart. There are two types of alternatives in the sub–networks. First, the paths which span the two vertices i, j, represent the most normal alternatives such as { abc, d, efg, ...}. Second, the paths which have the same vertex i, and form the loops, can represent alternatives in word sequences such as {a, ab, abb, abbb, abbbb, ...}.

The optimal word sequence and its syntactic structure are found out using the traceback mechanism after the frame–synchronous parsing and searching process, because the chart includes the alternative edges.

3.3 Chart parsing algorithm for generating sub-networks

The outline of the chart parsing algorithm for generating sub-networks is shown in Fig.1.

Initialization of Chart :
for each rules $S \Rightarrow \alpha_1\alpha_2...\alpha_n$ do
  $Edge \leftarrow < 0, 0, S \Rightarrow \cdot\alpha_1\alpha_2...\alpha_n >$;
  Add_item_to_agenda($Edge$,$Agenda$);

Updating Chart with new edge :
while not $Agenda = null$ do
  $Edge \leftarrow$ Get_item_from_agenda($Agenda$);
  case $Edge$ of
  $< i, j, A \Rightarrow \alpha \cdot B\beta >$: /* for active edge */
    if not subsuming?($Edge$,$Chart$) then
      Add_item_to_chart($Edge$,$Chart$);
      for each rules $B \Rightarrow \gamma_1...\gamma_n$ do
        $Edge \leftarrow < j, j, B \Rightarrow \cdot\gamma_1...\gamma_n >$;
        Add_item_to_agenda($Edge$,$Agenda$);
      for each $< j, k, B \Rightarrow \gamma \cdot C\delta >$ on $Chart$ do
        $Edge \leftarrow < i, k, A \Rightarrow \alpha B \cdot \beta >$;
        Add_item_to_agenda($Edge$,$Agenda$);
  $< i, j, B \Rightarrow \alpha\cdot >$: /* for passive edge */
    if merging?($Edge$,$Chart$) then
      merge_vertex_in_chart($Edge$,$Chart$);
    else
      Add_item_to_chart($Edge$,$Chart$);
      for each $< k, i, A \Rightarrow \beta \cdot B\gamma >$ on $Chart$ do
        $Edge \leftarrow < k, j, A \Rightarrow \beta B \cdot \gamma >$;
        Add_item_to_agenda($Edge$,$Agenda$);
  {end of case}
{end of while}

Fig.1 An algorithm for generating sub-networks.

A new pending edge, which is taken from the agenda, is first checked whether it can be added to the chart. If that edge is not yet subsumed in the chart, it is added. Also, the algorithm attempts to create the edges derived from the new added edge. If it is possible to merge the end vertex of the new passive edge which can be added, the edge with merged end vertex is added to the chart, and the attempt to derive the new edges is abandoned. The agenda is a kind of priority queue used for putting the pending edges in order.

3.4 An extended one-pass search algorithm

A one-pass search algorithm which is guided by the generated sub-network is shown in Fig.2. It is regarded as a server which can receive the requests from the parser to find out new lexical passive edges. This one-pass algorithm updates the cumulative frame-synchronously likelihoods in the active zone of each lexical vertex, and proposes new passive lexical edges for the parser.

Traceback procedure

The associated sequence of words can be uniquely recovered. And, it is easy to reconstruct the syntactic structure with the structural ambiguity from the recovered word sequence and chart.

Integration of two algorithms

These two algorithms mentioned above can be integrated in a simple procedure. It is needed to receive and pass the lexical edges between the chart parser and the one-pass search algorithm. As such this architecture provides a convenient mechanism for integrating work in a modular way. For example, the acoustic-phonetic process can run on array processors, while the linguistic parsing system can be implemented on LISP machines.

Initialization :
$Chart \leftarrow$ Make_chart_system($Start\_symbol$);
for each outgoing lexical edge $LE_{k,1}$
  at initial vertex do
$w_{k,1} \leftarrow$ Lexical_item($LE_{k,1}$);
for each frames j of $w_{k,1}$ do
 $D(j,k,1) \leftarrow$ Initial_cumulative_score($j,k,1$);
 $B(j,k,1) \leftarrow 0$; /* Backpointers */

Frame-synchronous process :
for input speech frame $i = 1$ to N do
 for each vertex $v \in Chart$ do
  for each outgoing lexical edge $LE_{k,v}$ do
   $w_{k,v} \leftarrow$ Lexical_item($LE_{k,v}$);
   $(AZ_s(k,v), AZ_e(k,v)) \leftarrow$ Active_zone($LE_{k,v}$);
   for $j = AZ_s(k,v)$ to $AZ_e(k,v)$ do
     if start_frame?($j, w_{k,v}$)
     then /* between-unit transition rule */
       $k' \leftarrow$ {incoming lexical edges at v};
       $D_i(j,k,v) \leftarrow max_{k'}\{D(L(k',v),k',v)\}$;
       $B_i(j,k,v) \leftarrow i$ - 1;
     else /* within-unit transition rule */
       $j^* \leftarrow argmax\{D(j,k,v); D(j-1,k,v);$
                $D(j-2,k,v);\}$;
       $D_i(j,k,v) \leftarrow D(j^*,k,v) + d(i,j,k,v)$;
       $B_i(j,k,v) \leftarrow B(j^*,k)$;
   {end of loop}
   if active_edge?($LE_{k,v}$) and
      $AZ_e(k,v) = L(k,v)$ then
     $Edge \leftarrow$ create_passive_edge($LE_{k,v}$,$Chart$);
     Add_item_to_agenda($Edge$,$Agenda$);
     $Chart \leftarrow$ update_chart($Chart$);
 {end of loop}
 $k \leftarrow$ {incoming lexical edges};
 $U(i,v) \leftarrow argmax_k\{D_i(L(k,v),k,v)\}$; /*unit*/
 $V(i,v) \leftarrow$ start vertex of $U(i,v)$; /*vertex*/
 $F(i,v) \leftarrow B_i(L(U(i,v),v),U(i,v),v)$; /*frame*/
{end of loop}
for all j,k,v do
  $D(j,k,v) \leftarrow D_i(j,k,v)$; $B(j,k,v) \leftarrow B_i(j,k,v)$;
{end of loop}

Traceback :
$Solutions \leftarrow$ Trace_back($Chart, U, V, F$);

Fig.2 An extended one-pass search algorithm.

Computational complexity

Fig.3 shows the illustration of the active zones on the sub-network dynamically generated by the chart parser. The locations of these active zones are independently moving to the final vertex like a data-flow process.

These active zones are updated for all the frames of input speech. Thus, the overall number of operations for requiring the cumulative scores, is proportional to the number of input frames, the average width of active zones and the average number of active zones on the sub-network.

The computational complexity is independent of the number of words and the length of words, because it also directly takes account of predictive information in sentence parsing.

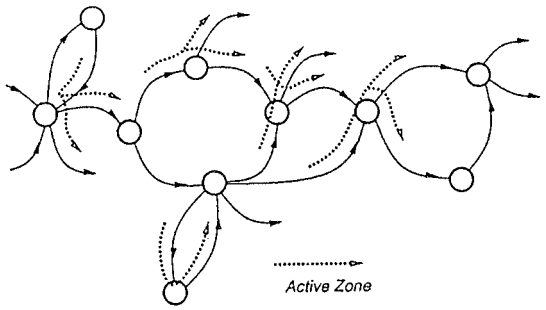

Fig.3 The active zones on the sub-network.

4 Some extensions of this algorithm

HMM formalism

It is well known that it is possible to extend several DTW-based formalisms for connected word recognition to HMM-based formalisms with Viterbi-style decoding algorithms.

N best search algorithm

The one-pass search algorithm can find out only optimal solution. However, it is not difficult to achieve the N best solutions[14], using multiple buffers to store both the likelihoods and back-pointers for the N best solutions in each vertex (node). Consequently, the algorithm finds the N most likely sentences with its computational complexity being proportional to the N.

If this integrated search algorithm can be given all the possible knowledge as a constraint for input speech, the N best solutions should not be needed, except for the purpose of evaluation of the whole system.

Algorithm for Unification-based Grammar

The above algorithm should be useful for both continuous speech recognition and speech understanding systems using all the possible linguistic knowledge.

In order to perform a tightly-coupled, top-down search, an algorithm is needed that uses all the possible knowledge sources to determine which word can come next. The constraints based on these knowledge sources and predictive information can be represented by the feature structure using a unification-based phrase structure grammar formalism[13].

Extension of the algorithm intending incorporation of the constraints based on the unification grammar into the one-pass search process requires the following modifications of the conventional unification algorithm. (1) For merging sub-sentence hypotheses with the same syntactic functions, the algorithm must identify the difference between two feature structures using a filter[15] for extracting prime features needed for predicting the succeeding phrases and words. (2) For reducing parsing time, the algorithm must avoid going through the full unification process for feature structures with minimal features being partially unified.

5 Conclusion

The proposed parsing algorithm efficiently finds the N most likely sentence hypotheses under the constraints of the unification-based phrase structure grammar.

The proposed algorithm has the following features: (1) It can find out semi-optimal word sequences by taking account of context-free phrase structure grammar. (2) The computational complexity of this algorithm is proportional to the number of the length of input speech frames, but it is independent of the number of words and the length of words. (3) The search process in the speech recognition phase can directly take account of the predictive information in the sentence parsing. (4) The procedure of this algorithm is formulated in a frame-synchronous process.

This framework can be extended to speech understandings system with dynamically varying constraints in dialogue situation. This algorithm leads to a unified scheme of a unification grammar based approach for speech understanding systems.

We have so far done only rather limited experiments using small vocabularies. The preliminary results show that this algorithm realizes an efficient parsing for continuously spoken language.

References:

(1) J.S.Bridle, M.D.Brown and R.M.Chamberlain : "An Algorithm for Connected Word Recognition", Proc. ICASSP-82, pp.899-902 (1982).

(2) C.S.Myers and S.E.Levinson : "Speaker-Independent Connected Word Recognition Using a Syntax-directed Dynamic Programming Procedure", Trans. ASSP-30, 4, pp.561-565 (1982).

(3) C.H.Lee and L.R.Rabiner : "A Network-based Frame-Synchronous Level Building Algorithm for Connected Word Recognition", Proc. ICASSP-88, pp.410-413 (1988).

(4) H.Ney, D.Mergel, A.Noll and A.Paeseler : "Data-Driven Organization of a Dynamic Programming Beam Search for Continuous Speech Recognition", Proc. ICASSP-87, pp.833-836 (1987).

(5) A.Averbuch, L.Bahl, et al. : "A real-time isolated-word speech recognition system for dictation transcription", Proc. ICASSP-85, pp.858-861 (1985).

(6) Y.L.Chow, M.O.Dunham, et al. : "BYBLOS: The BBN Continuous Speech Recognition System", Proc. ICASSP-87, pp.89-92 (1987).

(7) K-F.Lee, H-W.Hon, et al. : "The SPHINX Speech Recognition System", Proc. ICASSP-89, pp.445-448 (1989).

(8) M.Tomita : "An efficient word lattice parsing algorithm for continuous speech recognition", Proc. ICASSP-86, pp.1569-1572 (1986).

(9) H.Ney : "Dynamic Programming Speech Recognition using a Context-free Grammar", Proc. ICASSP-87, pp.69-72 (1987).

(10) H.Murveit and R.Moore : "Integrating Natural Language Constraints into HMM-based Speech Recognition", Proc. ICASSP-90, pp.573-576 (1990).

(11) S.R.Young and W.H.Ward : "Towards Habitable Systems: Use of World Knowledge to Dynamically Constrain Speech Recognition", The Second Symposium on Advanced Man-Machine Interface Through Spoken Language, 30 (1988).

(12) M.Kay : "Algorithm Schemata and Data Structures in Syntactic Processing", Technical Report CSL-80-12, Xerox PARC (1980).

(13) S.M.Shieber : "A Uniform Architecture for Parsing and Generation", Proc. of 12th Coling, pp.614-619 (1988).

(14) R.Schwartz and Y-L. Chow : "The N-Best Algorithm: An Efficient and Exact Procedure for Finding the N Most Likely Sentence Hypotheses", Proc. ICASSP-90, pp.81-84 (1990).

(15) S.M.Shieber : "Using Restriction to Extend Parsing Algorithms for Complex-Feature-Based Formalisms", Proc. of 23th ACL, pp.145-152 (1985).