

SEGMENTAL SEARCH FOR CONTINUOUS SPEECH RECOGNITION

P. Laface ★ and *L. Fissore* ◇ and *A. Maro* ◇ and *F. Ravera* ◇

★ Dipartimento di Automatica e Informatica - Politecnico di Torino
Corso Duca degli Abruzzi 24 - I-10129 Torino, Italy e-mail laface@polito.it
◇ CSELT - Centro Studi e Laboratori Telecomunicazioni
Via G. Reiss Romoli 274 - I-10148 Torino, Italy e-mail fissore@cse.lt.stet.it

ABSTRACT

The paper illustrates a search strategy for continuous speech recognition based on the recently developed Fast Segmental Viterbi Algorithm (FSVA) [5], a new search strategy particularly effective for very large vocabulary word recognition.

The FSVA search has been extended to deal with continuous speech using a network that merges a general lexical tree and a set of bigram subtrees generated on demand during the search.

Results are given for a 751-words speaker independent spontaneous speech recognizer of a railway timetable inquiry application, managed by a dialog system. Preliminary tests have been performed on the Wall Street Journal 5K words 1992 evaluation set.

1. INTRODUCTION

It is well known that a tree organization of the lexicon reduces both memory costs and search efforts because it makes many words share their initial units [4].

FSVA is a time synchronous left-to-right beam search Viterbi algorithm searching a lexical tree of subword units and developing time-dependent phonetic hypotheses. This latter feature is particularly suited for very large vocabulary recognition because it allows the likelihood of the previously computed units to be exploited. It is also effective for continuous speech systems using long span language models (LM) because these systems develop and keep separate many word theories which expand the same word (the same units) in different contexts.

Moreover, to reduce to a minimum the unit activation times that are likely to be incorrect, the FSVA algorithm explicitly takes into account the boundaries between units associated to the lexical tree arcs.

In [5] we have shown that $H_t(a_i)$, the log probability of the sequence of lexical tree arcs that generated the observation sequence $o_1 \dots o_t$, with a_i as last arc of the sequence, can be computed recursively using a time synchronous algorithm that develops a trellis of unit nodes rather than of state nodes. This technique,

tested with a vocabulary of around 190K directory entries, achieved the same results obtained with Viterbi algorithm, with a 35% speedup.

It is straightforward to perform the FSVA search for continuous speech with a unigram LM because a word terminal node reactivates the lexical tree from its root after adding the unigram LM probability to the score of a partial theory. This approach cannot be extended to bigram language models because a word pair is not known until a partial theory has reached a terminal node of the tree, where the identity of the second word is stored. Several solutions have been proposed to solve this problem that can be summarized into three categories: duplication of the lexical tree associated with a careful dynamic expansion of the search space [4, 8], use of a network that includes both a tree and a linear structure [7], offline generation of a static tree-based network that includes both the lexical tree and an interpolated bigram language model [1, 2].

The network solution merges a general *lexical tree* and the set of *bigram subtrees*. The static structure of the bigram subtrees offers the attractive properties of the tree organization, it allows to keep separate theories that differ for the last word and to reduce the delay of application of the LM probabilities. Moreover, since it is straightforward to associate a trellis node with a network node, accessing the successor nodes and their related information is direct, and pruning inactive trellis nodes is immediate. Although the number of the network nodes can be optimized [2], this structure is still rather demanding in terms of storage requirements for a very large vocabulary, especially if cross-word context-dependent units are used. In this work we propose, therefore, to build on demand the bigram subtrees of the words hypothesized during the search.

2. TREE REPRESENTATION

Since the identity of a word is available only when the decoding process reaches a leaf node of a lexical tree, the application of the LM probabilities will be delayed. To reduce this delay, the probabilities can be distributed among the tree nodes by assigning to a node

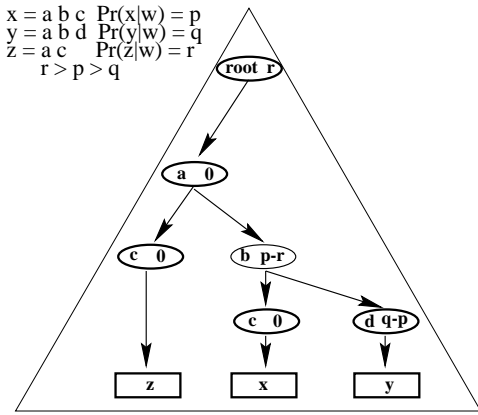


Figure 1: Log probability distribution in a tree

the maximum LM probability of all words sharing that node [8, 1].

The distribution of the probabilities can be effectively carried out during the generation of a new branch of a tree, assuring that the set of words to be included in the tree are provided in decreasing order of (log) probability.

A tree is grown assigning to its root node the maximum probability of the words in the set and then considering the phonetic transcriptions of words one at a time. Let's consider a transcription with associated log probability L , and let d be the current log probability to be distributed. The growth of a new branch is performed according to the following steps:

- initialize d to L ;
- while the current phonetic unit matches the current tree node identity
 - traverse the tree looking up the current phonetic unit and subtract the node log probability to d ;
- create a new tree node, and set its distributed log probability to d ;
- create the remaining nodes with associated distributed log probability of zero.

An example of log probability distribution, borrowed by [1], but performed according to our algorithm, is shown in Fig. 1 for a tree including words x , y , z . Notice that a set of words corresponds to a subtree, and that its leftmost branch always represents the word with maximum probability.

3. LM REPRESENTATION

To account for bigram events never observed in the training corpus an interpolated language model can be estimated using a discounted relative frequency $f(y|x)$

and the probability $\lambda(x)$ of words never occurred in the context of x according to:

$$Pr(y|x) = f(y|x) + \lambda(x) \cdot Pr(y) \quad (1)$$

The solution proposed in [1] to represent an interpolated language model is a network that merges a general *lexical tree* and a set of *bigram subtrees*. The network connects a bigram subtree for each word x . The subtree of x includes only the set of its successors appeared in the LM training corpus.

It is worth noting that trees with distributed probabilities have many nodes with associated zero log probability. In particular, the units of a word that are not shared by other words within the tree (word tails) are represented by a linear sequence of nodes: all these nodes, possibly excluding the first one, are zero log probability nodes (see Fig. 1). This property can be exploited to reduce the size of the network.

In our first approach, the nodes of a word tail of the general *lexical tree* are shared with the related nodes of every bigram subtree in which the word appears. Sharing is shown by dashed arcs leading to a “recombination node” in Fig. 2, while dotted arcs in Fig. 2 suggest that, if word y has been hypothesized, the tree of the successors of y and the general lexical tree are reactivated accounting for the interpolation log probability $\lambda(y)$. The use of the recombination node allows not only to reduce the size of the networks, but also to effectively perform at the same time both the Viterbi optimization and the LM probability interpolation.

Notice that the size of the bigram subtrees is generally small, even for very large vocabularies: for example, the average number of successor words in the 17118 baseline bigrams (bcb20cnp) of the Wall Street Journal is 72.2. If the 5K vocabulary words are transcribed using a set of 1678 context-dependent units, the average size of the bigram subtrees is 1540 nodes (the largest one has 15958 nodes). The size of a complete network statically built using this approach is 7.683.511 nodes which is about 80% the size of a network which does not share the lexical tree tails.

4. DYNAMIC TREE GENERATION

Unfortunately, although the number of the network nodes can be further reduced [2], a static structure is still rather demanding in terms of storage requirements for a very large vocabulary due to the large number of different bigrams occurring in a corpus, and to the distribution of the LM probabilities among the network nodes. To face this disadvantage, we dynamically build a network that includes only the bigram subtrees of the words hypothesized during the search. Since the number of the decoded words is controlled by the beam search and histogram pruning thresholds, a small fraction of the total network is generated for each sentence.

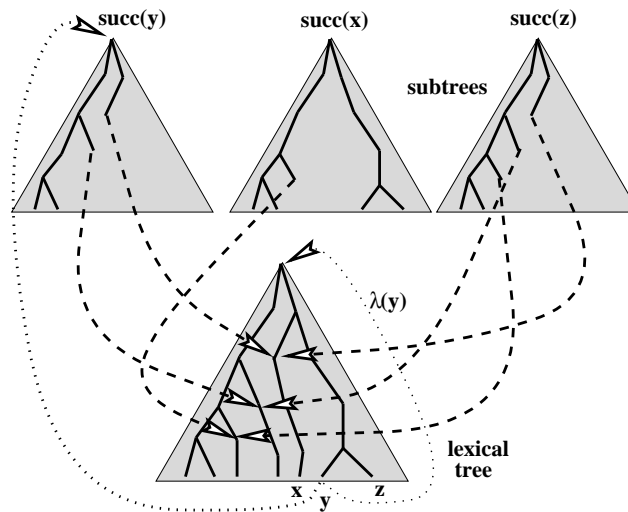


Figure 2: Tree-based bigram language model representation with the tails of the lexical tree shared

Notice that, since the trees are constructed on demand, their structures must be relocatable. To suit this requirement, in our tree data structure each pointer to a node is recorded as an offset with respect to the current node. To speed up the generation of a bigram subtree, and to allow the probability distribution to be carried out during the generation of a new branch of the subtree, the following operations are performed off-line:

- the set of successors of a word are ordered by decreasing probability;
- the recombination nodes in the lexical tree are identified and marked;
- the “updated transcription” of each vocabulary word is obtained eliminating the units belonging to the word tail and replacing them with a single pointer to the word recombination node in the lexical tree.

The decoding algorithm is simply modified to identify and expand arcs leading to recombination nodes. At startup, the lexical tree and the start-of-sentence subtree are loaded along with the updated transcriptions of each word. During the search, when a new word is hypothesized, its bigram subtree is built using the updated transcriptions of its successor words and the tree generation algorithm described in Section 2.

The generation of 1000 subtrees with an average of 1540 nodes takes less than 0.5 sec CPU time on a Digital ALPHA DecStation. For a large vocabulary, therefore, the overhead for dynamically constructing the subtrees is small with respect to the decoding complexity. Since in our 751 word application, the average size of the bigram subtrees is 8.2, the overhead for their generation is absolutely negligible.

In a second approach similar to [2] a dummy “linear tree” including the *linear transcription* of each word is off-line built. Since the tails of a given word appear both in the lexical tree and in many subtrees, they share their common nodes having zero log probability with the corresponding nodes of the linear tree.

The algorithm for generating shared tail trees makes use of a counter associated to each node which stores the number of words sharing that node, and follows these steps:

- A temporary distributed probability subtree is built as described in Section 2. During the tree traversal for the generation of a new branch the counter associated to each visited node is incremented.
- The final subtree is constructed visiting, by means of an inorder recursion, the temporary tree and replacing a node having a count equal to 1 and log probability equal to 0 (this condition identifies the beginning of the tail of a word) with a pointer to the appropriate node of the linear tree.

The decoding algorithm for this shared tail network loads, at startup, the linear tree along with the general lexical tree and the start-of-sentence subtree (generated off-line). Then, the bigram subtrees of a word are grown on demand using the shared tails algorithm.

A slightly greater overhead for subtree generation is experienced with this approach, but a complete network with shared tails includes 2.611.097 nodes only.

5. RESULTS

In the following we report some results of the evaluation of the FSVA search on a speaker independent spontaneous speech recognizer for a railway timetable

Bigram (Perplexity 15.7)	Word Accuracy %	Expanded Nodes
Viterbi	80.4	1598
FSVA	80.3	983
Trigram (Perplexity 13.3)	82.5	

Table 1: Recognition results and search complexity

inquiry application, managed by a dialog system [3]. For this application, the size of recognizer vocabulary is 751 word only, a complete network, thus, easily fits in memory. To test our approach, however, the decoder has been modified to dynamically generate the bigram subtrees. It has been then tested using the Wall Street Journal 5K word 1992 evaluation database for which we give preliminary results.

Table 1 summarizes the recognition results in terms of word accuracy (WA) obtained with the bigram and the trigram LM. To train the language models a database including the transcription of about 9,000 spontaneous speech sentences (about 60,000 words) collected from "naive" users through a PABX has been used. The test set bigram and trigram perplexity computed by non-linear interpolation are 15.7 and 13.3 respectively; for this task it is natural to observe many out of vocabulary words both in the training and in the test corpus. The test set includes a total of 858 sentences from 20 different users.

The recognizer employs 310 context-dependent subword units modeled by 3 state discrete density HMMs.

Since in the first experiments we experimented a 25% speedup using the FSVA with respect to the Viterbi algorithm, corresponding to the reported 40% reduction of the expanded trellis nodes, successive experiments were carried out using the FSVA.

The results with the trigram LM are obtained performing the bigram constrained FSVA decoding in a first pass and generating a word graph [7]. The word graph is used as a grammar that constrains the search space of a second pass search that performs a word pair approximated trigram acoustic decoding using, in our case, again the same acoustic models. The decoding time for the graph constrained second pass is 1 sec per sentence on the average.

As far as the WSJ is concerned, preliminary results using continuous density HMM models of a total number of 1678 units (biphones and triphones) trained without any tying, and not including function words or gender dependent models, give 89% word accuracy using an interpolated bigram language model. With the thresholds needed for obtaining this result, the average number of hypothesized words (and created bigram subtrees) per sentence is 358. Using our first approach for generating the trees, the average number of created nodes per sentence is 1.408.560 rather than 7.6 million,

while an average of 4013 unit nodes per frame are expanded with an average decoding time of 76.1 sec.

6. CONCLUSIONS

We described a search strategy for decoding large vocabulary continuous speech relying upon a time synchronous left-to-right beam search Viterbi algorithm searching a dynamically generated tree-based network of subword units and developing time-dependent phonetic hypotheses. Further experiments are in progress to evaluate a new set of transition units that can be easily integrated in the network to take into account cross-word coarticulation [6].

7. REFERENCES

- [1] G. Antoniol, F. Brugnara, M. Cettolo, and M. Federico, "Language Model Representations for Beam-Search Decoding", In Proc. ICASSP95, Detroit, pp. 560-563, 1995.
- [2] F. Brugnara, M. Cettolo, "Improvements in Tree-based Language Model Representation", In Proc. EUROSPEECH95, Madrid, pp. 1797-1800, 1995.
- [3] D. Clementino and L. Fissore, "A man-machine dialogue system for speech access to train table information", In Proc. EUROSPEECH 93, Berlin, September 1993.
- [4] R. Haeb-Humbach, H. Ney, "Improvements in Beam Search for 10000-Word Continuous-Speech Recognition", IEEE Transactions on Speech and Audio Processing, Vol.2, n.2, pp. 353-356, 1994.
- [5] P. Laface, C. Vair, and L. Fissore, "A Fast Segmental Viterbi Algorithm for Large Vocabulary Recognition", In Proc. ICASSP95, Detroit, pp. 560-563, 1995.
- [6] L. Fissore, P. Laface, G. Micca, F. Ravera, "Vocabulary Independent Acoustic-Phonetic Modeling for Continuous Speech Recognition", EUSIPCO96, Trieste, Sept. 1996.
- [7] H. Murveit, P. Monaco, V. Digalakis, J. Butzberger, "Techniques to Achieve an Accurate Real-Time Large-Vocabulary Speech Recognition System", In Proc. ARPA Human Language Technology Workshop, ARP, pp. 368-373, March 1994.
- [8] J.J. Odell, V. Vatchev, P.C. Woodland, S.J. Young, "A One Pass Decoder Design for Large Vocabulary Recognition", In Proc. ARPA Human Language Technology Workshop, ARPA, pp. 380-385, March 1994.