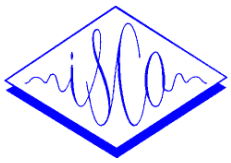
ISCA Archive
http://www.isca-speech.org/archive



3rd European Conference on
Speech Communication and Technology
EUROSPEECH'93
Berlin, Germany, September 19-23, 1993

Using Grammars in Forward and Backward Search

L. Fissore ◇ E. Giachin ◇ P. Laface ⋆ P. Massafra ⋆

◇ CSELT- Centro Studi e Laboratori Telecomunicazioni
Via G. Reiss Romoli 274 – I–10148 Torino, Italy
E-mail fissore@cselt.stet.it giachin@cselt.stet.it

⋆ Dipartimento di Automatica e Informatica – Politecnico di Torino
Corso Duca degli Abruzzi 24 – I–10129 Torino Italy
E-mail laface@polito.it

ABSTRACT

This paper focuses on the use of linguistic constraints in continuous speech decoding. In particular, it aims at verifying the advantages of the delayed use of linguistic constraints with respect to their use during the forward search. We evaluate the accuracy loss of the best "backward" derived sentence with respect to the one obtained by forward decoding using grammars. A "backward" derived sentence is obtained growing a complete path starting from the end of the sentence using a time asynchronous A^* search which exploits the word lattice scores computed during a classical forward search without grammar.
Results are evaluated on a 718 word, speaker independent recognition task. Applying linguistic constraints during forward decoding gives slightly better results than delaying them to the backward search, but the difference is small, and it has to be traded with the advantage of having N solutions instead of just one.

1 INTRODUCTION

A recently developed approach to continuous speech recognition [1] consists of a forward search technique, during which inexpensive knowledge sources are applied, followed by a backward search, where more complex language models are introduced. It is known that, in principle, some accuracy is lost when linguistic constraints are applied after forward decoding. On the other hand, the potential complexity reduction of this approach remains attractive. Therefore it is worth evaluating the accuracy loss of the best "backward" derived sentence with respect to the best "forward" decoded one, with or without grammars, in order to verify the advantages of the delayed use of linguistic constraints with respect to their use during the forward search.
The effectiveness of linguistic constraints has been assessed by evaluating word and sentence accuracy resulting from these different recognition approaches:

- forward decoding without grammars
- forward decoding using grammars
- grammar driven N-best decoding

The first approach constitutes a benchmark for a baseline system and for obtaining an inexpensive best solution without linguistic constraints.
In the second approach, the recognizer executes a Viterbi decoding on a large network of Markov states where acoustic, phonetic, lexical and grammatical constraints are embedded. The grammars tested in our experiments include Word Pairs and Finite-State Networks designed to represent habitable subsets of natural language.
The third approach executes a grammar driven traceback search, based on N-best decoding; the process works on the information gathered during the previous decoding procedure which does not make use of any linguistic constraint. A lattice of word hypotheses is produced during forward decoding without grammar, then N-best sentences are recovered backward by a Tree-Trellis A^* search. The latter approach has been further developed in order to obtain more accurate word hypotheses generating backward a *word dependent* lattice and finally performing the N-best decoding in the forward direction.
The approaches are evaluated on a test database of 600 sentences referring to a train timetable inquiry task, with a vocabulary of 718 words, collected from 10 speakers in a controlled environment through a local PABX [3].

2 GRAMMARS

Two types of grammars of increasing complexity have been used for testing: a Word Pair grammar and a Finite-State Network. The former one provides less constraining power than the latter, but is simpler to be implemented. Both grammars are tested on the same database used to develop them. This study, in fact, is designed to compare the performance of the decoding algorithms, and not to assess the effectiveness of the grammars themselves.

2.1 Word Pairs

Word pairs, expressing the legal pairs of words that may follow each other, are the simplest means for introducing

*This work has been partially supported by the Esprit II project 2218 – "SUNDIAL"

linguistic constraints in a recognizer. They are easy to define, train, and integrate in the acoustic decoding algorithm. Their main disadvantage is an overcoverage: in order to cover a habitable subset of language, the resulting word pairs will also admit many wrong word sequences.
To define word pairs, words have first been partitioned into equivalence classes; words within a class belong to the same linguistic category. Classes have been defined taking into account grammatical categories (noun, verb, etc.), morphological features (gender, tense, etc.), and semantic aspects (e.g., the words *station* and *hour*, both singular nouns, belong to different classes). 172 such classes have been created. Class pairs are then computed by looking at adjacent classes appearing in a training corpus of 600 sentences. Finally legal class pairs have been extended to all the words belonging to each class. The resulting word pair grammar has a perplexity of 122.

2.2 Finite-State Networks

Finite-State Networks (FSN) are a formalism equivalent to Regular Grammars and have been used since the early days of speech recognition as a means for constraining the search during acoustic-phonetic decoding. FSNs would become too large, however, if they were to capture the complexities of natural language. It has nevertheless been shown [4] that, for specific task domains, it is possible to construct *local* FSNs that succeed in modeling most of the linguistic expressions typical of the task, without increasing too much their size. FSNs are "local" since they do not represent the ways in which constituents may be ordered in a well formed sentence, rather they only model the individual constituents of the language; only loose models are used for phrases connecting such constituents. This approach limits the network size at expense of an increased perplexity. At least for languages (like Italian) in which the order of constituents is relatively free, a limited perplexity increase has been observed which does not impair system performance. Moreover, some redundancy may turn even useful in that it permits to parse linguistic constructs that were not included in the database used to develop the FSN.
A detailed description of the local-FSN approach is beyond the scope of this paper, thus only summary information is provided here. The general FSN structure is similar to the "ergodic" network of [4], with each constituent modeled by a subnetwork and all the subnetworks connected in parallel; a null arc connects the last node to the first one, thus permitting any sequence of constituents to be parsed. Constituents are generally related to individual semantic concepts of the task domain. In the present domain (train timetable inquiry), constituents are for example time expressions ("between two thirty and four in the afternoon", "on Thursday March 2", "late at evening hours", etc.), arrival and departure places ("from Torino", "with Milan as the arrival city", etc.), expressions for inquiries on travel fares, services available on trains, etc. Moreover, a set of words has been identified, which are placed as loop arcs around the initial node of subnetworks; by traversing these arcs, the recognizer can get through inter-constituent phrases, which are often semantically irrelevant or redundant ("I would like to know", etc.).
The FSN used in these experiments has been manually developed starting from a context-free like formalism which is then compiled in finite-state form; this gives rise to more "structured" constituent subnetworks than those of [4]. Though designed to keep FSN size small, this approach still produces FSNs of the order of some ten thousands word arcs in order to represent habitable subsets of natural language. This size is still large enough to require several optimizations to be introduced in the recognizer to limit time and memory resources as will be detailed in the next Section. The network used in the present experiments includes 6457 word arcs and 684 nodes.

3 FORWARD SEARCH

FSNs are integrated into a Viterbi-based recognizer in a classical way. Some implementation optimizations have been necessary due to the still large size of grammars. Here they are briefly reviewed:

- *Grammar compilation.* Given a lexicon and a grammar, they are compiled into a structure of phonetic units. This structure permits an immediate access to the successor nodes of any given node.
- *Lexical subtrees.* In the no grammar setup, the lexicon is compiled into a tree of phonetic units in which the initial common part of different words are shared. This reduces decoding time considerably. This feature can no longer be used with a grammar, because words appearing in different arcs of the grammars have to be separately represented and cannot share parts. However, words pertaining to parallel arcs that connect the same pair of left and right node can be organized into a tree. In the present implementation, a subtree has been created for every group of parallel arcs. The degree of sharing (and hence the benefits) with respect to the no grammar case is lower, but still significant.
- *Backtracking information.* With no grammar, backtracking requires to keep a record, for each time frame, of the best-scoring path among those that have reached the end of a word. With a grammar, this information has to be recorded for each node of the grammar, because words ending at different nodes have in general different successors, and therefore much more memory is consumed. Memory requirements are reduced by designing suitable structures in which only information relative to alive grammar nodes is stored.

A grammar has two effects on analysis time. On the one hand, since it provides more constraints, it reduces the number of competing paths and hence the analysis time. This advantage, however, is usually masked by the use of larger beam search thresholds which are needed to reach the best accuracy. Moreover, the use of grammars makes the storage, retrieval, and search operations more complex, thus slowing down the process. The total effect is that analysis times increase, depending on the complexity

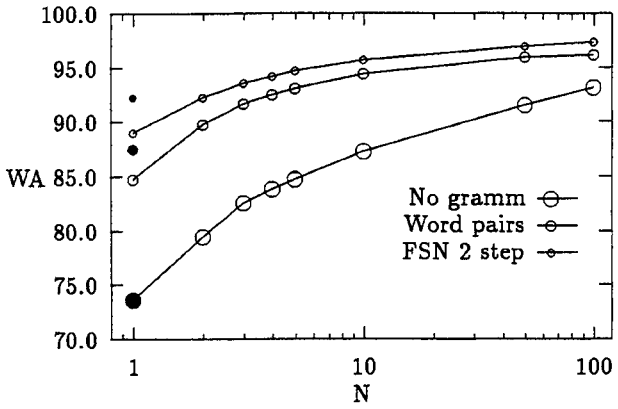


Figure 1: Word Accuracy for N-best decoding, DDHMM

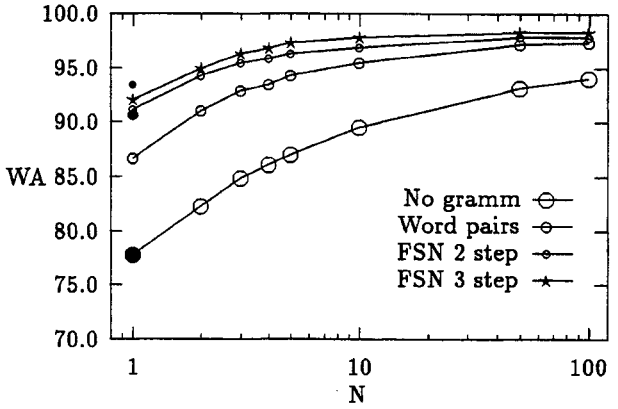


Figure 2: Word Accuracy for N-best decoding, CDHMM

of the grammar. For the grammar used in these experiments, the analysis time is about 2.8 times greater than without grammar.

4 BACKWARD SEARCH

The N-Best approach, briefly recalled here, is based on the observation that it is cost effective to use a two-step search strategy that first produces a set of hypotheses exploiting inexpensive knowledge sources (for example simple acoustic and statistical language models) and then applies more complex models and strategies for the evaluation of the resulting alternatives [6, 2].

For the sake of efficiency we use a modified Tree-Trellis search [7] where the first step is a time synchronous Viterbi decoding, that produces a lattice of word hypotheses. At every time frame, the alternative words ending at that time are stored in a list together with their score and traceback pointers to their optimal predecessors. As no grammatical constraints are used in this phase, the best scoring word hypothesis is propagated to the root node of the application lexicon organized as a tree.

The second step is a time asynchronous A^* search working backward from the end of the lattice, which grows partial paths using the word lattice scores computed during the first step to exactly estimate the remaining portion of the complete path. The modification introduced to the Tree-Trellis strategy is related to the direct use of the word lattice scores rather than acoustic word scores obtained by an expensive decoding process in the backward step.

As the A^* search can easily take into account syntactic constraints, we experimented with FSNs to verify the degree of accuracy of the delayed use of these knowledge sources with respect to their use during the forward search. This approach will be referred to in the following as "two-step strategy".

This two-step strategy working on the lattice scores, however, can underestimate or completely miss good scoring hypotheses [5, 8] because only one search path is kept for each state within a word in the forward step. As a word ending at a given time frame may have only one start time and thus only one best predecessor word, alternative theories with different preceding words are lost. Therefore, a *word dependent* N-best strategy has been proposed in [5] where multiple theories are kept within a word assuming that the best starting time of a word does not depend on the whole preceding theory, but on the preceding word only. Thus, theories are allowed to reach a given word at the same time frame arriving from different preceding words.

Since our lexicon is organized in terms of a tree structure to reduce decoding time, at each time frame in which a set of words ends, rather than activating the tree root node with the best scoring word ending at that frame, different theories are kept by propagating to the root node the best theory for each word along with the word identifier.

The number of theories "local" to each node is controlled by a state beam-search threshold that typically maintains three to four theories.

As the computational load increases with the number of theories kept, we devised a "three-step" strategy along the line suggested in [1]:

1. Time synchronous Viterbi decoding without grammar working forward in time and producing a set of scored word hypotheses for each frame. This is also the first step of the two-step strategy.

2. Word dependent refinement of the lattice hypotheses, using again a time synchronous Viterbi decoding without grammar, but working backward in time.

   This step requires much less computational effort because it uses the information gathered in the lattice produced during the previous, "forward", step. In particular, when the initial state of a word model is activated working backward in time, we can find in the lattice the set of words which end at the previous frame and their scores. This information is exploited by activating the final states of this set of models only. From the set, moreover, are excluded words which are not worth to be expanded because they would

be pruned on the basis of the beam search threshold value [1].

3. Finally the A^* can be performed forward in time to generate the N-best sentences using the knowledge offered by the grammar.

After Step 1, an alternative approach could consists of using the grammar directly in Step 2 which would increase the benefit due to pruning, and produce "backward" the best grammatically correct sentence, at the expense of a slightly more complicated process.
It is worth noting that working backward in time keeping the tree as lexical knowledge requires, in Step 2, that the beginning node of each word activates the leaf nodes of the tree corresponding to the words hypothesized in Step 1, and that a different theory is expanded for each ending word. An effective solution to this problem uses different instances of the tree nodes, dynamically included in the trellis when needed by a growing theory. By using a hashing addressing scheme it is easy to identify a theory - by means of the current and preceding word - and to perform Dynamic Programming for the nodes towards which a theory converges from different paths.

5 RESULTS

Both the "backward" N-best and the "forward" decoding procedures have been assessed on 600 test sentences referring to a train information service task, collected from 10 speakers in a controlled environment from a local PABX. The results obtained using both Discrete and Continuous Density HMM, Viterbi decoding and applying linguistic constraints of increasing complexity during forward decoding are given in Tab. 1 (WA and SA refer to word accuracy and sentence accuracy respectively). The corresponding results for the two-step N-best scheme are presented in Tab. 2, as far as the best sentence is concerned (the first column, referring to the no grammar case, is obviously the same for both tables.) Results for different values of N are plotted in Fig. 1 and Fig. 2 for the Discrete and Continuous Density HMM respectively (filled circles refer to the results presented also in Tab. 1).
The results of the Continuous Density HMM three-step strategy with grammar are presented in Fig. 2 where the upper curve shows that word-dependent decoding allows a slight improvement of the word and sentence accuracy to be achieved.
The word-pair grammar, not surprisingly, gives a lower recognition accuracy than the FSN grammar. However, the word accuracy vs. N grows more rapidly, thus decreasing the performance difference between these two models for large Ns.
In agreement with expectations, the application of linguistic constraints during forward decoding provides slightly better results than delaying them to the backward search. The difference, however, is small, and it has to be traded with the advantage of having N solutions instead of just one. Moreover, the forward step allows to use a fast recognition decoding process, e.g. one using simple Discrete Density HMMs, and to delay to the backward step the use of the more computational expensive Continuous Density HMMs.

	No gram		Word Pairs		FSN	
	WA	SA	WA	SA	WA	SA
DDHMM	73.6	24.7	87.5	55.7	92.3	70.3
CDHMM	77.8	32.2	90.7	63.5	93.5	72.8

Table 1: Linguistic constrained "forward" decoding

	No gram		Word Pairs		FSN	
	WA	SA	WA	SA	WA	SA
DDHMM	73.6	24.7	84.8	51.6	89.0	64.3
CDHMM	77.8	32.2	86.7	52.2	91.2	67.1

Table 2: Linguistic constrained "two-step" decoding

References

[1] S. Austin, R. Schwartz, and P. Placeway, "The Forward-Backward Search Algorithm". *Proc. of the ICASSP*, pp. 697-700, 1991.

[2] M. Bates *et al.*, "Design and Performance of HARC, the BBN Spoken Language Understanding System". *Proc. of 1992 International Conference on Spoken Language Processing*, Banff, Canada, 1992, pp. 241–245.

[3] D. Clementino and L. Fissore, "A man-machine dialogue system for speech access to train timetable information". *This Conference.*

[4] E. Giachin, "Automatic Training of Stochastic Finite-State Language Models for Speech Understanding". *Proc. of the ICASSP*, San Francisco, CA, pp. I-173–I-176, 1992.

[5] R. Schwartz and S. Austin, "A Comparison of Several Approximate Algorithms for Finding Multiple (N-BEST) Sentence Hypotheses". *Proc. of the ICASSP*, pp. 701–704, 1991.

[6] R. Schwartz *et al.*, "New Uses of the N-Best Sentence Hypotheses Within the BIBLOS Speech Recognition System". *Proc. of the ICASSP*, San Francisco, CA, pp. I-1–I-4, 1992.

[7] F. K. Soong and E.-F. Huang, "A Tree-Trellis Fast Search for Finding the N-Best Sentences Hypotheses in Continuous Speech Recognition". *Proc. of the ICASSP*, Toronto, Canada, pp. 705-708, 1991.

[8] V. Steinbiss, "A Search organization for Large Vocabulary Recognition Based on N-Best Decoding". *Proc. of Eurospeech 91*, Genova, Italy, pp. 1217–1220, 1991.