A SEARCH ORGANIZATION FOR LARGE-VOCABULARY RECOGNITION BASED ON N-BEST DECODING

Volker Steinbiss

Philips GmbH Forschungslaboratorium Aachen
Weisshausstr., D-5100 Aachen, Germany
Phone: ++49 (241) 6003-504; FAX: -518

ABSTRACT

This paper presents a time-synchronous search concept in which (a) the language model contexts are represented in local hypotheses lists, as in N-best decoding, and where (b) these hypotheses lists are rescored using a language model in a time-delayed manner. The concept is shown to work well on a real 12000-word continuous-speech recognition task with a phonetic-tree based search. A faster though approximative search algorithm is presented which compares favorably well with the exact ones. - The discussion starts with some new aspects on word-dependent N-best decoding.

Keywords: continuous speech recognition, search, N-best decoding, top N.

1. INTRODUCTION - A SHORT REVIEW ON N-BEST DECODING

Given a spoken utterance, a Viterbi-based speech-recognition system determines the word sequence given by the most likely state sequence. With N-best decoding, instead of one sentence only, the recognizer produces a list of the top-scoring word sequences, usually a pre-specified number of or all hypotheses within a given beam of log-likelihoods.

Although there exist several previous papers generalizing dynamic programming to an N-best dynamic programming [1] as well as describing the concept of 2-best or N-best decoding in speech recognition [2,3], N-best decoding has become increasingly popular in the speech-recognition community only around 1989/90 [4,5,6]. The algorithms published until then suffer from the computation being about linear in N, which is due to the fact that N theories have to be kept in parallel at each point in the search space.

New algorithms have significantly reduced the computational costs of N-best decoding. *Word-dependent N-best decoding* [7] allows us to work on n-best lists in the word interiors with n being substantially smaller than N, while the proper N-best sentence list is built up in a subsequent pass. The idea of delaying part of the effort of N-best decoding to a second pass is carried to an extreme in the *tree-trellis based search* of [8], which seems to be the most efficient algorithm for time-synchronous N-best decoding: A forward Viterbi decoding pass is used to build up a trellis; a subsequent time-asynchronous backward A*-search then uses the scores[1] stored in the trellis as an exact estimate of the remaining portion. Finally, a recognizer based on *stack decoding* [9] has the built-in capability of producing the top scoring sentences one after the other.

[1] The word "score" is used interchangeably for "negative log-likelihood" throughout this paper.

In this paper, we will first review *word-dependent N-best decoding* and highlight some new aspects. The word lattice, which is a more suitable data structure than the N-best sentence list, can be accessed in several ways. Throughout this paper, the term *word lattice* is used in such a way that each word has exactly one starting and one ending point in time, such that for each possible sentence there is at most one path through the lattice.

We then explain how local n-best lists can be used to represent language model contexts to the left (e.g. states in the stochastic finite-state network defined by the language model). This representation is appropriate for a time-delayed incorporation of a language model. I.e., instead of using the language model in a predictive manner, it is incorporated e.g. one word later by updating the local candidate lists. This concept fits into the N-best paradigm of incorporating several knowledge sources by rescoring N-best sentences lists. A faster though approximative search algorithm is then presented which compares favorably well with the exact one.

Experimental results are given for a speaker-dependent 12000-word continuous-speech recognition task with a tree-structured search space and a unigram and a bigram language model.

2. THE WORD-DEPENDENT N-BEST ALGORITHM

2.1 N-Best Decoding: An Overview

All algorithms mentioned in the introduction which can be used within an integrated time-synchronous search share general principles.

In exact N-best decoding, hypotheses with different histories (the word sequence so far) are kept separate while hypotheses with the same history are recombined. More generally, histories can be grouped into equivalence classes called *history classes* in this paper, and hypotheses are recombined if and only if they have the same history class. Two extreme cases of history classes are the history itself (exact N-best) and one class only (standard Viterbi search).

Table 1: *Comparison of different time-synchronous N-best decoding schemes.*

Algorithm	Local History Class	Data Structure Built up During 1st Pass	Remarks on Search in 2nd Pass	Exactness
Standard Viterbi	one class	tree	trivial	exact (1-best)
Lattice N-Best	one class	word lattice	cf. 2.4	poor approximation
Tree-Trellis Based	one class	trellis	incl. isolated-word alignment	exact
Word-Dep. N-Best	predecessor word	word lattice	cf. 2.4	good approximation
Exact N-Best	predecessor sentence	none (or tree)	not necessary	exact

To reduce computation, several algorithms split up the decoding effort into a time-synchronous part, using a local history class inside words, and a subsequent time-asynchronous part which performs a search within a structure stored in the previous pass. Table 1 gives a comparative overview.

2.2 The Independence Assumption

In the word-dependent N-best algorithm [7], the predecessor word is taken as local history class. This choice is based on the assumption that for a hypothesis inside word w_m with history $w_1,...,w_{m-1}$, the word boundary[2] between words w_{m-1} and w_m does not depend on the previous words $w_1,..., w_{m-2}$.

While the independence assumption is reasonable for large-vocabulary recognition, it might be doubtful for other tasks. As an example, consider a digit-string recognition with zeros pronounced 'oh': The two confusable strings "00" and "0" cannot both be recovered by the word-dependent N-best because both hypotheses are assigned the same history class "0" and are thus recombined when meeting within the subsequent word.

2.3 The Word Lattice

During the first pass of word-dependent N-best decoding, at the end of each word w, the following information is stored for each hypothesis in the n-best list:

- The score difference to the top hypothesis score S_{opt}
- The history class, i.e. the predecessor word.

A pointer to this information as well as score S_{opt} and a new history class identifier w (the word which ended) are propagated during the subsequent within-word search process.

Hence, essentially a lattice is built up, with arcs labelled with words and score differences to the locally optimal decision. This structure is more compact than a list of sentences. Indeed, the sentence-spanning N-best lists typically consist of (rather uninteresting) variations of local top candidates (cf. 4.3). If e.g. each of m words in the sentence is easily confusable with another word, this already can fill a 2^m-sized sentence list!

Since the optimization over the word boundaries is carried out during the first pass, the lattice size is kept small, and there is a one-to-one correspondence between sentences and paths through the lattice. Both properties can be helpful when using computationally more expensive knowledge sources or models such as a natural-language parser or cross-word triphones.

Hence, the word lattice provides a suitable data structure in many respects: It still contains all score information, the optimization over the word boundaries is already done, and it is more compact than a list of sentences.

2.4 Extracting the N-Best List from the Lattice

There are several ways of extracting an N-best list given the word lattice produced by the word-dependent N-best algorithm. We consider four ways (all start at the end of the sentence):

(a) Going backward in time, perform an N-best dynamic programming search (apply the general principle described e.g. in [5]).

(b) Perform a depth-first search through the lattice and merge the extracted sentences according to their scores into a list of length N until a stopping criterion is met [7].

(c) Perform an A*-search on the lattice, using the score information as an exact estimate for the remaining part. (The scores rather than the differences have to be stored in this case. This is [8] without the necessity of optimizing over the word boundaries and doing the acoustic word recognition.)

(d) Successively, extract the best sentence W_{opt} from the lattice L by just deciding for the best arc at each node; then modify the lattice such that

- The new lattice L' contains exactly the same sentences except sentence W_{opt} , and that
- All scores are appropriately updated.

The last algorithm (d) which needs no predetermined N in contrast to (a)-(c) is not outlined here. Any of these lattice search procedures takes considerably less time than the Viterbi decoding pass.

3. Language Model Context Representation Using the N-Best Paradigm

3.1 General Concept and Delayed Language Model Incorporation

In the one-stage dynamic programming search, both acoustic and language model (LM) constraints are expanded thus forming a huge search space in which both knowledge sources are incorporated simultaneously. E.g. using a vocabulary of size V and a stochastic word-based n-gram language model [10] (with $n \geq 2$), there are V^{n-1} different contexts to the left for each acoustic word w_m - namely its n-1 predecessors $w_{m-n+1}, ..., w_{m-1}$ - which have (potentially) to be realized with V^{n-2} separate copies of each of the V words of the vocabulary. Even for moderate values of n and V, the resulting search space may be prohibitively large.

Normally, the search space is built up as follows: First the language model network is expanded into word copies. The LM is a probabilistic finite-state grammar for both finite-state and stochastic n-gram LMs. Second, the word copies are expanded into stochastic finite-state networks called Hidden Markov models. But we can quite as well first build a structure representing the acoustic model - e.g. a phonetic tree - and then keep the syntactic left-contexts separate using hypotheses lists. When doing so, we gain some flexibility concerning the incorporation of language-model knowledge. - If the context to the left is finer than taking the predecessor word, the same trick as used in word-dependent N-best decoding can be applied.

The concept going hand in hand with the N-best representation of LM context is the concept of delayed LM incorporation. In the N-best paradigm, LM knowledge is used for recognition in two very different ways. The conditional probability of a word given a history can be incorporated in a predictive manner when entering the word during the search process. The LM can be used as well after the proper recognition process to rescore an N-best sentence list or the word lattice. For a full search, the result is not affected by the time when LM knowledge is incorporated - as long as all sentence hypotheses are kept separate, which is a number exponential in the sentence length.

But real large-vocabulary systems use pruning techniques in order to achieve acceptable decoding times, so it matters when LM knowledge is incorporated. While an incorporation of the LM after recognition reduces interaction with the probably computationally expensive LM, the drawback is that the LM cannot control the search process, such that the optimal sentence using this LM might be pruned during the recognition pass with a poorer LM.

[2] For the sake of clarity, subtleties concerning silence between words will not be discussed in this paper.

As a compromise, a delayed incorporation of the LM might be useful in certain cases. Instead of incorporating the conditional probability $p(w_m|w_1,...,w_{m-1})$ at the beginning of word w_m, it is taken into account during search with a certain delay, e.g. at the end of word w_m or two words later. The basic assumption when using one or a sequence of delayed LMs is that the hypotheses list of one LM still contains the candidates favored by the subsequent (more powerful) LM. The delay must be chosen with regard to the trade-off between the accuracy loss and a possible speed-up.

The next section gives an example of a search where the identity of a word is only known at its end but not at its beginning, such that a 1-word delay for incorporating the bigram LM is a necessity.

3.2 An Application: Tree-Based Acoustic Search and the Bigram Language Model

When all words are assumed equally probable (zerogram language model) or when only the unconditional word probabilities are taken into account (unigram language model), the acoustic search space can be organized very efficiently in a tree structure (partially described in [11]), taking advantage of the fact that many different words share the same initial phoneme sequences. The LM scores of a unigram LM are added to the partial scores at the word endings.

In our 12000-word system, the speed-up factor as compared to a "linear search organization" with a separate copy per word was about 5 for the search procedure, excluding the log-likelihood calculations.

In order to use this search-space architecture together with a bigram LM, the context to the left of a word (namely the predecessor word) was represented in a local hypothesis list, and the score $-\log p(w_m|w_{m-1})$ provided by the bigram LM was incorporated at the end of word w_m.

3.3 An Approximative Search Variant

In order to speed up the search process, the technique of delayed incorporation of a LM can be combined with some heuristic approximations [12]. Being in the final state of word w, we determine the optimal predecessor word v_{opt} using only the LM incorporated so far as well as its word boundary t_{opt}. Then all words ending at time t_{opt} are examined as potential predecessors of w, now using the finer LM. The outlined principle applies to other LMs as well as to other heuristic assumptions.

Algorithm:

LOOP 1: for every unpruned HMM state in the acoustic tree do
  if end of word w is reached with local score S_{loc}, do:
  LOOP 2: get word boundary t_{opt} to optimal predecessor word v_{opt}
    LOOP 3: Evaluate
    $v_{opt} = \underset{v}{\text{argmin}} \{S - \log p(w|v) \mid v \text{ ending at time } t_{opt}\}$
    Replace S by $S - \log p(w|v_{opt})$, and store all information necessary for trace-back in trace-back array
    LOOP CONTROL 3
  LOOP CONTROL 2
LOOP CONTROL 1

While the scores in this algorithm take account of the exact bigram LM scores, the decision on the word boundary of the optimal predecessor word are not conditioned on the bigram probabilities.

The algorithm reminds of a variant of N-best decoding (cf. [5,4]) called "Lattice N-Best" in [7] which uses a language-model node instead of the predecessor word as history class. But in contrast to this algorithm, the hypothesis in the final state of a word w_m following the word sequence $w_1, ..., w_{m-1}$ contains already all bigram LM information of this partial sentence. The approximation is that the LM knowledge is not available for the decision taken at the word beginning.

4. Experimental Results

4.1 General Framework

The tests were carried out on a current version of the Philips speaker-dependent large-vocabulary continuous-speech recognizer [14,13,11] which succeeds the one used in the SPICOS speech-understanding system. We use context-independent phone models, the output probability density functions are modelled as mixtures of Laplacian densities. The system was tested on a dictation task with a recognition vocabulary comprising about 12000 words. The utterances - read office correspondence in German - were recorded in a quiet office environment using a hand-held microphone. Speaker dependent training and testing were performed on 2200 and 1100 words, respectively.

Error rate is defined as

(deletions + insertions + substitutions) / spoken words.

The *N-best error rate* is calculated by choosing from sentence hypotheses 1 to N the one with the least errors, for each spoken sentence. The N-best error rate is useful for evaluation tests on N-best decoding but should be carefully interpreted. E.g. given a fixed word error rate and fixed N, the N-best word error rate improves with decreasing sentence length!

4.2 The Search Depth of the Word-Dependent N-Best Decoding

In order to get the top N sentences, word-dependent N-best decoding only requires a much smaller number n of hypotheses to be kept locally. In this pre-experiment, we kept this number n fixed in order to get an idea on how long the local n-best lists have to be. Table 1 shows that there is no significant decrease in performance between cases n = 3 and 5. Thus it seems that relatively short local n-best lists can be used without loss in accuracy, which is important for achieving short decoding times.

Table 1: *N-best word-error rate (defined in 4.1) for three values of n (speaker M-21, vocabulary 11425 words, test-set perplexity 1792).*

N	1	3	6	10	30	50
n=5	26.2%	23.6%	22.0%	21.3%	19.4%	18.9%
n=3	26.2%	23.6%	22.0%	21.3%	19.6%	18.9%
n=2	26.2%	23.7%	22.1%	21.5%	20.2%	19.8%

4.3 Interpretation of N-Best Error Rates: Sentence Length, Pruning and Homophones

Although the experiments were focused on the performance of the search process, we were at first disappointed about the improvement in the error rate for increasing N. One reason is that with 22.0 words per sentences on the average, the test sentences are rather long (cf. 4.1). We also ran two control experiments in order to check two other possible reasons: The effect of pruning and the high number of confusable words in our lexicon. In these and the following experiments, the local n-best list length was not fixed but depended on the scores of the competing hypotheses, and the test conditions were identical but different from those in 4.2.

We used two pruning parameters T_{glob} and T_{loc}. T_{glob} is the usual pruning threshold which compares every score at time t with the optimum at this time t; T_{loc} does the same only for the hypotheses within one list compared to the list optimum. Table 2 shows that tight pruning hurts but that pruning was not responsible for the performance in Table 1.

Table 2: *N-best word-error rates for three different pruning threshold pairs (T_{glob}, T_{loc}) (explained in the text) in logarithmic units: weak (145, 60), standard (130, 40) and strong (110, 40).*

Table 3: *Error rates for homophone classes instead of words and two different history classes (standard pruning).*

Speaker M-21, 12306 words, unigram LM of test-set perplexity 1831

	Word-Error Rate			Homophone-Error Rate	
top	Weak	Standard	Strong	History Class "Word"	History Class "Homophone"
1	22.9%	22.9%	24.1%	18.1%	18.1%
2	21.9%	21.9%	23.2%	16.9%	16.0%
3	21.2%	21.2%	22.5%	16.3%	15.8%
4	20.8%	20.7%	22.0%	15.9%	15.5%
5	20.4%	20.4%	21.6%	15.7%	15.1%
6	20.2%	20.2%	21.6%	15.6%	14.8%
8	19.8%	19.9%	21.2%	15.4%	14.6%
10	19.6%	19.7%	20.7%	15.2%	14.3%
15	19.2%	19.2%	20.2%	14.8%	14.0%
20	19.2%	19.1%	20.2%	14.8%	13.8%
30	19.1%	18.7%	19.4%	14.5%	13.5%
50	18.5%	17.7%	18.8%	13.8%	12.7%
75	18.2%	17.2%	18.5%	13.4%	12.5%
100	17.4%	16.7%	18.1%	12.8%	12.2%

Another reason is that there are 728 homophone pairs in the recognition lexicon of 12306 words. While for some of these clearing the ambiguities is obviously the task of the LM, this is often impossible even for a human, e.g. in the case of "zwei" ("two") and "2". So we used the predecessor homophone class of the predecessor word rather than the word itself as a history class in the word-(or should we say homophone-) dependent N-best decoding. This essentially amounts to eliminating homophonic variants of a higher-ranked sentences from the N-best list. The error rate is calculated on homophones (two homophones are considered the same token) instead of word spellings. Table 3 shows the effects of homophones to the N-best error rate for increasing N.

4.4 Delayed Language Model Incorporation: Using a Bigram Language Model Together with a Phonetic Tree

Table 4 shows that the delayed bigram LM incorporation of section 3.2 using a phonetic tree (c) works as well as the conventional linear search (e); (a) are the results for the unigram LM. Differences in the error rate between (c) and (e) should be due to the different pruning strategies and pruning parameters. The execution time of (c) lies between those of (a) and (e).

Table 4: *Word-error rates for four male speakers; 12306-word vocabulary.*
a) Unigram LM of test-set perplexity 1831
b) Bigram LM rescoring of 100 best sentences generated using the unigram LM
c) Delayed incorporation of bigram LM using N-best (cf. 3.3 and 4.4)
d) Approximation to method b) as described in 3.4, cf. 4.5
e) Standard (linear) search with bigram LM of test-set perplexity 1056

Speaker	(a) Unigram LM	(b) Bigram on Lattice	(c) Delayed Bigram	(d) Approxi-mation	(e) Bigram LM
M-21	22.9%	22.1%	20.7%	19.9%	20.0%
M-22	28.5%	27.5%	25.4%	24.1%	24.3%
M-24	35.3%	34.8%	30.6%	31.4%	31.7%
M-25	23.0%	21.9%	18.2%	17.6%	18.0%
average	27.4%	26.6%	23.7%	23.3%	23.5%

4.5 The Approximative Method

The approximative method proposed in 3.3 compares favorably well with both linear search with bigram LM and the exact delayed incorporation of the bigram LM (cf. Table 4, column (d)).

We ran a control experiment (b) in order to check whether we could also use the word boundary segmentation from the unigram LM and update the scores using the bigram LM after recognition. Hence, we did a 100-best recognition using the unigram LM and rescored the sentences according to the use of the bigram LM. This was clearly suboptimal. Method (d) outperforms (b) because a better LM incorporated with delay guides the search in spite of a few suboptimal decisions taken (cf. discussion in 3.3).

REFERENCES

[1] T. Bayer, M. Oberländer: "Ein erweiterter Viterbi-Algorithmus zur Berechnung der n besten Wege in zyklenfreien Modellgraphen", in *Mustererkennung 1986*, G. Hartmann (ed.), 8. DAGM-Symposium Paderborn, Sep./Oct. 1986, pp. 56-60: Springer Verlag, Berlin, 1986.

[2] L. R. Rabiner, S. E. Levinson: "Isolated and Connected Word Recognition - Theory and Selected Applications", IEEE Trans. on Communications COM-29(5), pp. 621-659, May, 1981. Also in: A. Waibel, K.-F. Lee (eds.): *Readings in Speech Recognition*, pp. 115-153, Morgan Kaufman Publishers, San Mateo, CA, 1990.

[3] C.-H. Lee, L. R. Rabiner: "A Network-Based Frame-Synchronous Level Building Algorithm For Connected Word Recognition", Proc. IEEE *ICASSP-88*, New York, pp. 410-413, 1988.

[4] J. B. Mariño, E. Monte: "Generation of Multiple Hypothesis in Connected Phonetic-Unit Recognition by a Modified One-Stage Dynamic Programming Algorithm", Proc. Eurospeech-89, Paris, Vol. 2, pp. 408-411, Sept. 1989.

[5] V. Steinbiss: "Sentence-Hypotheses Generation in a Continuous Speech Recognition System", Proc. *Eurospeech-89* Paris, Vol. 2, pp. 51-54, Sept. 1989.

[6] R. Schwartz, Y.-L. Chow: "The N-Best Algorithm: An Efficient and Exact Procedure for Finding the Most Likely Sentence Hypotheses", Proc. IEEE *ICASSP-90*, Albuquerque, NM, pp. 81-84, April 1990. Also in *Proceedings of the DARPA Speech and Natural Language Workshop*, Cape Cod, Oct. 1989.

[7] R. Schwartz, S. Austin: "A Comparison of Several Approximate Algorithms For Finding Multiple (NBEST) Sentence Hypotheses", Proc. IEEE *ICASSP-91*, Toronto, Canada, pp. 701-704, May 1991.

[8] F. Soong, E.-F. Huang: "A Tree-Trellis Based Fast Search for Finding the N-Best Sentence Hypotheses in Continuous Speech Recognition", Proc. IEEE *ICASSP-91*, Toronto, Canada, pp. 705-708, May 1991. Also in *Proceedings of the DARPA Speech and Natural Language Workshop*, pp. 709-712, Hidden Valley, June 1990.

[9] D. Paul: "Algorithms for an Optimal A* Search and Linearizing the Search in the Stack Decoder", Proc. IEEE *ICASSP-91*, Toronto, Canada, pp. 693-696, May 1991.

[10] F. Jelinek: "Continuous Speech Recognition by Statistical Methods", Proc. of the IEEE, Vol. 64, No. 10, pp. 532-556, April 1976.

[11] R. Haeb-Umbach, H. Ney: "A Look-Ahead Search Technique for Large Vocabulary Continuous Speech Recognition", elsewhere in these Proc. *Eurospeech -91*, Genova, Italy, Sep. 1991.

[12] H. Ney, Personal Communication, March 1991.

[13] H. Ney: "Acoustic Modeling of Phoneme Units for Continuous Speech Recognition", Proc. of the European Conf. on Speech Communication and Technology, Barcelona, Spain, Sep. 1990, pp. 65-72.