A COMPARISON OF TIME CONDITIONED AND WORD CONDITIONED SEARCH TECHNIQUES FOR LARGE VOCABULARY SPEECH RECOGNITION

S. Ortmanns, H. Ney, F. Seide, I. Lindam*

Lehrstuhl für Informatik VI, RWTH Aachen – University of Technology,
D-52056 Aachen, Germany

*Philips GmbH Forschungslaboratorien, D-52066 Aachen, Germany

ABSTRACT

In this paper, we compare the search effort of the word conditioned and the time conditioned tree search methods. Both methods are based on a time-synchronous, left-to-right beam search using a tree-organized lexicon. Whereas the word conditioned method is well known and widely used, the time conditioned method is novel in the context of 20 000–word vocabulary recognition. We extend both methods to handle trigram language models in a one–pass strategy. Both methods were tested on a train schedule inquiry task (1 850 words, telephone speech) and on the North American Business (Nov.'94) development corpus (20 000 words).

1. INTRODUCTION

The paper describes two variants of the time-synchronous beam-search algorithm using a tree-organized pronunciation lexicon [6]. Both methods are based on a one-pass concept and utilize a (prefix) tree organization of the pronunciation lexicon [3]. The difference between the two search methods is the organization of the search space. In the word conditioned method, we structure the search space by using word conditioned copies of the pronunciation tree, i.e. for each word history required by the language model (LM), there is a separate copy of the lexical tree. In the time conditioned method, there is a separate copy of the lexical tree for each potential starting time. To analyze both search methods, experimental tests were carried out.

The paper is organized as follows. In Section 2, we review the word conditioned search along with an extension for handling trigram language models. In Section 3, we present the time conditioned search. In particular, we focus on the incorporation of bigram and trigram language models in the time conditioned method and an efficient implementation of the language model recombinations. In the last section, we present recognition experiments performed on a 1850-word telephone-based train schedule inquiry task and on the North American Business (Nov.'94) development set (a speaker-independent 20 000–word task).

2. WORD CONDITIONED SEARCH

In this section, we review the word conditioned search [3] and describe an extension for handling trigram language models. The search algorithm is based on the time-synchronous beam-search strategy using a tree-organized pronunciation lexicon. When using m-gram language models, such as bigram ($m = 2$) or trigram ($m = 3$) language models, we face the problem that during the search the word identity is only known when a leaf of the lexical tree is reached. To solve this problem, we have to keep a separate copy of the lexical tree for each of the $(m-1)$ predecessor words.

For a bigram language model [3], a separate copy of the lexical tree is needed for each predecessor word v. When going from a bigram to a trigram language model, we have to take into account that, for a trigram language model, the probabilities are conditioned on the previous two predecessor words (u, v) rather than one predecessor word v in the bigram case [6, 7]. Therefore, we have to make the copies dependent on the *two* predecessor words. To formulate the dynamic programming approach, we introduce the auxiliary quantity $Q_{uv}(t, s)$ and the backpointers $B_{uv}(t, s)$. These quantities must be made dependent on the two predecessor words (u, v) to account for the delayed application of the trigram language model:

> $Q_{uv}(t, s)$:= score of the best path up to time t that ends in state s of the lexical tree for the two-word history (u, v).
>
> $B_{uv}(t, s)$:= starting time of the best path up to time t that ends in state s of the lexical tree for the two-word history (u, v).

The dynamic programming recursion within the tree copies remains the same as for the bigram case. So we have the usual dynamic programming recursion for the word interior:

$$\begin{aligned} Q_{uv}(t,s) &= \max_{\sigma} \{ q(x_t, s|\sigma) \cdot Q_{uv}(t-1, \sigma) \} \\ B_{uv}(t,s) &= B_{uv}(t-1, \sigma_{uv}^{max}(t,s)) \ , \end{aligned}$$

where $q(x_t, s|\sigma)$ is the product of transition and emission probabilities of the underlying Hidden Markov Model. $\sigma_{uv}^{max}(t, s)$ denotes the optimum predecessor state for the hypothesis (t, s) and two-word history (u, v). The backpointers $B_{uv}(t, s)$ are propagated according to the dynamic programming decision. The recombination equation at the word boundaries can be written as:

$$H(v, w; t) := \max_{u} \{ p(w|u, v) \cdot Q_{uv}(t, S_w) \} \ ,$$

where $p(w|u, v)$ is the conditional trigram probability for the word triple (u, v, w) and S_w denotes a terminal state of the lexical tree

for word w. To start up new words, we have to pass on the score and the time index:

$$\begin{aligned} Q_{uv}(t-1,0) &= H(u,v;t-1) \\ B_{uv}(t-1,0) &= t-1\ , \end{aligned}$$

where we have introduced the fictitious state $s=0$ for initialization. For a trigram language model, we have to keep track of the two-word history rather than one-word history in the bigram case. Thus, the tree copies must be determined uniquely according to their two-word history. This can be efficiently achieved by using a hash table. The hash index of this table can be interpreted as the word pair index of the two-word history (v,w). To get a unique word pair index, a bijective function is used $f(v,w)$, e.g. $f(v,w)=W\cdot v+w$ where W is the vocabulary size. To allow for intraphrase silence, we use the same concept as for the bigram language model [3], which requires a separate copy of the silence model for each predecessor word. In particular, each intraphrase silence hypothesis is associated with its immediate non-silence predecessor wordpair.

3. TIME CONDITIONED SEARCH

Another natural way is to structure the search space by using time conditioned tree copies. Unlike the word conditioned method, the search hypotheses are organized according to the starting times of the word hypotheses. To take the language model into account, an additional effort is required for the recombination of word sequence hypotheses at potential word boundaries. The concept of time conditioned copies has already been used in other contexts, namely connected digit recognition [10] and word lattice generation [8]. We will describe the method first for a bigram language model and then for a trigram language model.

3.1. Bigram Language Models

To describe the time conditioned search algorithm, we define the following quantities as introduced in [6]:

> $h(w;\tau,t)$ = probability that word w produces the acoustic vectors $x_{\tau+1}...x_t$.
>
> $H(v;\tau)$ = probability that the acoustic vectors $x_1...x_\tau$ are generated by a word/state sequence with v as the last word and τ as the word boundary.

Stretching notation, we will use the symbol $Q_\tau(t,s)$ to denote tree internal search hypotheses with respect to starting time τ. To start up a new tree, i.e. new words, we initialize this quantity as follows:

$$Q_{t-1}(t-1,s)=\begin{cases} \max\limits_u H(u;t-1) & \text{if } s=0 \\ 0 & \text{if } s>0 \end{cases}$$

Again, the state $s=0$ is used for initialization. The usual dynamic programming recursion for the word interior is:

$$Q_\tau(t,s)=\max_\sigma\ [\ q(x_t,s|\sigma)\ Q_\tau(t-1,\sigma)\ ]\ .$$

Note that in the time conditioned search method the hypotheses $Q_\tau(t,s)$ are organized according to starting times τ and are started by using the best predecessor word. Therefore, backpointers [5] as employed in the word conditioned search method are not needed. However, the language model recombination across word boundaries is more complicated. For an efficient organization of the language model recombination, it is useful to introduce an additional auxiliary quantity:

$$\hat{H}(v,w;t)\ :=\ \max_\tau\left\{\frac{H(v;\tau)}{\max\limits_u H(u;\tau)}\cdot Q_\tau(t,S_w)\right\}\ .$$

To select the best predecessor word for each pair $(w;t)$, i.e. word w with ending time t, two optimization steps have to be performed. The first step is to maximize over all possible starting times τ of word w. This optimization results in a list of predecessor words v of word w. In the equation above, a normalization is necessary since each tree hypothesis is started with the best predecessor word. Second, we have to maximize the hypothesis score over the predecessor words v for each w:

$$H(w;t)=\max_v\left\{\ p(w|v)\cdot\hat{H}(v,w;t)\right\}\ .$$

The best of the scores $H(w;t)$ is used to start up the new tree for time $(t+1)$.

3.2. Extension to Trigram Language Models

To extend the time conditioned search algorithm from a bigram LM to a trigram LM, we have to distinguish the word sequence hypotheses according to their last *two* words. Therefore, we replace the quantity $H(v;\tau)$ by $H(u,v;\tau)$:

> $H(u,v;\tau)$ = probability that the acoustic vectors $x_1...x_\tau$ are generated by a word/state sequence with uv as the last *two* words and τ as the word boundary.

The definition of $Q_\tau(t,s)$, i.e. the state hypotheses within a tree, remains unchanged. To start up a new tree, we initialize the quantity by using the best wordpair:

$$Q_{t-1}(t-1,s)=\begin{cases} \max\limits_{(u,v)} H(u,v;t-1) & \text{if } s=0 \\ 0 & \text{if } s>0 \end{cases}$$

For the word (or tree) interior, we have the usual dynamic programming equation:

$$Q_\tau(t,s)=\max_\sigma\ \{q(x_t,s|\sigma)\ Q_\tau(t-1,\sigma)\}$$

and the recursive equation at word level can be expressed as:

$$H(v,w;t)=\max_u\left\{\ p(w|u\,v)\cdot\hat{H}(u,v,w;\tau)\right\}\ ,$$

where

$$\hat{H}(u,v,w;t)\ :=\ \max_\tau\left\{\frac{H(u,v;\tau)}{\max\limits_{(u',v')} H(u',v';\tau)}\cdot Q_\tau(t,S_w)\right\}\ .$$

Table 1: Search effort and recognition results for the word conditioned (WC) and time conditioned (TC) tree search methods for the train schedule task using a bigram language model ($PP_{bi} = 15.2$).

search method	average number of states	arcs	trees	word ends	LM recomb.	DEL – INS – SUB = TOTAL	WER [%]
WC	384	113	12	19	19	141 – 258 – 615 = 1014	14.5
	867	249	17	34	34	134 – 202 – 559 = 895	12.8
	4701	1255	28	130	130	132 – 184 – 541 = 857	12.3
	10649	2664	33	266	266	133 – 174 – 536 = 843	12.1
TC	1191	335	37	62	243	155 – 229 – 645 = 1029	14.8
	2143	596	39	86	458	147 – 184 – 564 = 895	12.8
	4488	1201	43	144	1009	140 – 175 – 539 = 854	12.3
	14824	3545	50	374	3408	142 – 167 – 533 = 842	12.1

The term $p(w|u\,v)$ denotes the trigram probability for the word triple (u, v, w).

The algorithm using a trigram language model can be described as follows. At the acoustic level, the algorithm is similar to the one in the bigram case. However, due to the recombination at the word level, additional computation steps are needed. In a first step we determine for each word w with ending time t all possible start times τ. This information is collected in a list. Then we can calculate for each fixed word w the best valuation for each word triple (u, v, w) denoting by the term $\hat{H}(u, v, w; t)$. The corresponding starting time τ of w and the two-word history (u, v) are stored in temporary lists. In a second step, the recombination by applying the conditional trigram probabilities $p(w|uv)$ can be carried out. Finally, we cash the tree start-up scores $\hat{H}(u, v, w; \tau)$ associated with the best word trigram for normalization. This allows us to determine efficiently the best two-word history of a word w by hashing. As in the word conditioned search, special care must be taken for intraphrase silence, because it has to be treated differently for acoustic search and for LM recombinations.

4. EXPERIMENTAL RESULTS

To analyze the search effort, recognition experiments were carried out on a German train schedule inquiry corpus [1] and on the ARPA North American Business (NAB'94) H1 development corpus [4]. In the word conditioned search, we used the three pruning techniques, namely acoustic pruning, language model pruning and histogram pruning as described in [9]. The acoustic pruning is refined by a unigram language model look-ahead [11]. In the time conditioned search, the language model pruning is modified so that only the most likely word end hypotheses are retained in the word lists.

4.1. Train Schedule Inquiry Corpus

First, the two methods were compared in the context of the Philips automatic telephone-based train timetable information system [1]. Its speaker-independent recognizer uses 3502 strongly tied context-dependent phonemes that share 703 emission probability distributions. They have been trained on $33\,081$ spontaneous utterances recorded over the telephone network during a field test, comprising a total of 12.1 hours of speech. The test corpus contains 7078 spoken words (2278 utterances) with $PP_{bi} = 15.2$, the lexicon size is 1850. The out-of-vocabulary word rate is 2.7 % (191 words).

4.2. NAB'94 Corpus

Second, the two methods were compared on the North American Business (Nov.'94) H1 development data. In this task 4688 context dependent phoneme models were used sharing 4623 emission probability distributions [2]. The training of the emission probability distributions was performed on WSJ 0 and WSJ 1 training data. For this experiment we used about 290 000 mixture densities for each gender. The test set contained 310 sentences with 7387 spoken words. In the experiments, a $20\,000$–word vocabulary and the standard language model with a test set perplexity of $PP_{bi} = 205.4$ and $PP_{tri} = 135.5$ for the bigram and for the trigram case, respectively. 2.7% (199 words) of the spoken words were out-of-vocabulary words.

4.3. Comparison: Word Conditioned vs. Time Conditioned Search

Tables 1 and 2 summarize the results of both recognition tasks. The tables show the word error rate (WER[%]) as a function of the search space which is given in terms of average number of active states, arcs, trees and word ends per time frame. In addition, the tables show the recombination effort (averaged over time frames) at the word level which is measured as the average number of active word end hypotheses considered in the language model recombinations (LM recomb.).

In the bigram case, it can be seen that there is nearly no significant difference in the size of search space for the NAB'94 task (Tab. 2). For high word error rates (i.e. 12.8 % to 14.8 %) in the case of the train schedule inquiry task, the word conditioned method requires $2 - 3$ times less states than the time conditioned method (Tab. 1). For this task, the small vocabulary size (1850 words) and the low perplexity ($PP_{bi} = 15.2$) result in a significantly smaller number of word conditioned tree copies in comparison with the NAB'94 task.

When using a trigram language model, a small advantage in favour of the time conditioned method can be observed. In particular, for the same number of active states, the average number of active trees per time frame is typically much lower than in the word conditioned method. The reason for this is that a tree is started for every time frame at which at least one word end hypothesis is produced, which holds for nearly every time frame. This effect is fairly independent of perplexity and vocabulary size. In both tasks, the number of trees

Table 2: Search effort and recognition results for the word conditioned (WC) and the time conditioned (TC) tree search methods on the NAB'94 H1 development data.

LM type	search method	average number of					DEL – INS – SUB = TOTAL	WER [%]
		states	arcs	trees	word ends	LM recomb.		
bigram	WC	5022	1416	20	86	86	185 – 198 – 879 = 1262	17.1
$PP_{bi} = 205.4$		9335	2593	28	156	156	181 – 197 – 860 = 1238	16.8
		26493	7222	45	540	540	179 – 195 – 846 = 1220	16.5
		52786	13968	60	560	560	179 – 193 – 842 = 1214	16.4
	TC	6175	1781	27	117	1539	177 – 217 – 884 = 1278	17.3
		11280	3101	31	197	2976	180 – 199 – 857 = 1236	16.7
		30019	7998	37	520	8761	179 – 195 – 845 = 1219	16.5
		60692	15877	43	1075	18081	179 – 193 – 840 = 1212	16.4
trigram	WC	4714	1392	29	80	80	128 – 211 – 757 = 1096	14.8
$PP_{tri} = 135.5$		18734	5430	70	294	294	120 – 206 – 721 = 1047	14.2
		48940	13877	112	702	702	120 – 204 – 717 = 1041	14.1
		86772	23688	145	1223	1223	121 – 200 – 709 = 1030	13.9
	TC	8573	2459	28	136	1477	119 – 216 – 735 = 1070	14.5
		15103	4194	31	234	2806	118 – 205 – 720 = 1043	14.1
		24914	6780	34	388	4858	118 – 201 – 717 = 1036	14.0
		71757	18953	42	1202	14293	118 – 199 – 715 = 1032	14.0

is between 27 and 50, which seems to correlate with the average word duration.

So far, we have not considered the computational effort for the language model recombination. This computational effort is much greater in the time-conditioned search, where, for each word triple (or word pair) under consideration, a separate optimization over the unknown word boundary has to be carried out. In contrast, in the word conditioned method, the optimization over the word boundaries is already taken into account when starting up the trees. Typically in the time conditioned search, the effort for the language model recombination is higher by a factor of approximately 10 for both bigram and trigram language models. All in all, the experiments indicate that both search methods are suitable for large vocabulary recognition and are comparable in terms of efficiency.

Acknowledgment. This work was partly carried out within the project MAIS and supported by the Commission of the European Community (MLAP LRE-63-036).

REFERENCES

1. H. Aust, M. Oerder, F. Seide, V. Steinbiss: "Experience with the Philips Automatic Train Timetable Information System", Proc. Second IEEE Workshop on Voice Technology for Telecommunications Applications, pp. 67-72, Kyoto, September 1994.

2. C. Dugast, R. Kneser, X. Aubert, S. Ortmanns, K. Beulen, H. Ney: "Continuous Speech Recognition Tests and Results for the NAB'94 Corpus", Proc. ARPA Spoken Language Technology Workshop, Austin, TX, pp. 156-161, January 1995.

3. R. Haeb-Umbach, H. Ney: "Improvements in Time-Synchronous Beam Search for 10,000-Word Continuous Speech Recognition", IEEE Trans. on Speech and Audio Processing, Vol. 2, pp. 353-356, April 1994.

4. F. Kubala: "Design of the 1994 CSR Benchmark Tests", Proc. ARPA Spoken Language Technology Workshop, Austin, TX, pp. 41-46, January 1995.

5. H. Ney, D. Mergel, A. Noll, A. Paeseler: "Data Driven Organization of the Dynamic Programming Beam Search for Continuous Speech Recognition", IEEE Trans. on Signal Processing, Vol. SP-40, No. 2, pp. 272-281, Feb. 1992.

6. H. Ney: Search Strategies for Large-Vocabulary Continuous-Speech Recognition. NATO Advanced Studies Institute, Bubion, Spain, June-July 1993, pp. 210–225, in A.J. Rubio Ayuso, J.M. Lopez Soler (eds.): "Speech Recognition and Coding – New Advances and Trends", Springer, Berlin, 1995.

7. J.J. Odell, V. Valtchev, P.C. Woodland, S.J. Young: "A One-Pass Decoder Design for Large Vocabulary Recognition", Proc. ARPA Spoken Language Technology Workshop, Plainsboro, NJ, pp. 405-410, March 1994.

8. M. Oerder, H. Ney: "Word Graphs: An Efficient Interface Between Continuous Speech Recognition and Language Understanding", Proc. Int. Conf. on Acoustics, Speech and Signal Processing, Minneapolis, MN, Vol.II, pp. 119-122, April 1993.

9. S. Ortmanns, H. Ney, A. Eiden: "Language-Model Look-Ahead for Large Vocabulary Speech Recognition", Proc. Int. Conf. on Spoken Language Processing, Philadelphia, PA, October 1996.

10. H. Sakoe: "Two-Level DP Matching – A Dynamic Programming-Based Pattern Matching Algorithm for Connected Word Recognition", IEEE Trans. on Acoustics, Speech and Signal Processing, Vol. ASSP-27, pp. 588-595, December 1979.

11. V. Steinbiss, B.-H. Tran, H. Ney: "Improvements in Beam Search", Proc. Int. Conf. on Spoken Language Processing, Yokohama, pp. 2143-2146, September 1994.