Time Conditioned Search in Automatic Speech Recognition Reconsidered

D. Nolden, H. Ney, R. Schlüter

Human Language Technology and Pattern Recognition Group
RWTH Aachen University, Aachen, Germany
{nolden, ney, schlueter}@cs.rwth-aachen.de

Abstract

In this paper we re-investigate the time conditioned search (TCS) method in comparison to the well known word conditioned search (WCS), and analyze its applicability on state-of-the-art large vocabulary continuous speech recognition tasks. In contrast to current standard approaches, time conditioned search offers theoretical advantages particularly in combination with huge vocabularies and huge language models, but it is difficult to combine with across word modelling, which was proven to be an important technique in automatic speech recognition. Our novel contributions for TCS are a pruning step during the recombination called *Early Word End Pruning*, an additional recombination technique called *Context Recombination*, the idea of a *Startup Interval* to reduce the number of started trees, and a mechanism to combine TCS with across word modelling. We show that, with these techniques, TCS can outperform WCS on current ASR tasks.

Index Terms: speech recognition, search, word conditioned, time conditioned

1. Introduction

Both WCS and TCS are time-synchronous one-pass beam-search algorithms utilizing a prefix tree organization of the pronunciation lexicon [2, 1, 4]. The difference between the two search methods is the organization of the search space. In WCS, the search space is structured by creating virtual *word conditioned* copies of the pronunciation tree, while in TCS one virtual tree copy is created at each *time frame*. Depending on the vocabulary size and on the language model (LM), much less trees are started in TCS than in WCS, at the cost of a more expensive recombination step when hypothesizing ending words.

Recent research focuses either on approaches equivalent to WCS [7], or on weighted finite state transducer networks, where the whole word-conditioned search network is expanded and minimized statically in a preprocessing step [9]. The problem about statically compiled networks is that they can become huge, depending on the used vocabulary and LM, and that the knowledge sources are integrated statically, thus they cannot be changed without re-building the whole search network. In [7] the authors have shown that their dynamic search algorithm comes close in efficiency to their FST approach, which shows that dynamic decoders remain interesting.

Across word modelling [6] is a technique that allows incorporating the coarticulation at the word ends and word starts, which allows a more precise acoustic modelling. It is an essential technique in order to reach a low word error rate (WER). In dynamic search based on a prefix tree, the context is incorporated by inserting additional fan-out arcs at the leafs in the search tree, and fan-in arcs at the beginning of the tree, which, strictly speaking, transforms the tree into a directed acyclic graph (for simplicity we will keep up the "tree" terminology).

TCS and WCS have been compared in [5] and have been found comparable, however the comparison was done with a 20000 word vocabulary, without across word modelling and using a 3-gram LM, and the runtime was not analyzed. In the following experiments, we evaluate the algorithms regarding real time factor (RTF) and word error rate (WER) on the NAB'94 H1 development corpus for the simple experiments using within-word modelling, and on the EPPS English 2006 evaluation corpus from the TC-STAR project [1] for the experiments including across word modelling. On the EPPS English corpus we use a 60k word vocabulary, across word modelling, and a 4-gram LM with a perplexity of 129, which is the combination that has been chosen as the best one for single-pass recognition during the 2006 TC-STAR evaluation. The corpus consists of 192 minutes of speech, recorded from plenary sessions of the european parliament. On the NAB'94 H1 dev corpus we use a 60k or 20k word vocabulary, a 3-gram LM with a perplexity of 144 and within-word modelling. We perform all experiments on a variant of the RWTH Aachen open source speech recognition system [8].

2. Review of Time Conditioned Search

In TCS, at every time frame τ a new virtual tree copy Ω_τ is created. The search paths leading to all word end hypotheses $E(\tau)$ are combined into one time conditioned tree Ω_τ, and are expanded into separate word end hypotheses when a word end state S_w is reached within the tree, by applying the relative acoustic probability $p(x_\tau^t|w)$ of ending words w to each context hypothesis $e \in E(\tau)$.

Like in WCS, *acoustic pruning* is used to reduce the number of active state hypotheses $S(t)$, by applying a beam and removing all state hypotheses that have a probability lower then $Q_{AC}(t) \cdot f_{AC}$, where $Q_{AC}(t)$ is the probability of the best state hypothesis, and f_{AC} is the acoustic pruning threshold. Additionally, *histogram pruning* is used to limit the total number of active state hypotheses, using a dynamic acoustic pruning threshold that is automatically tuned to a value so that the number of state hypotheses stays below a specified maximum.

The virtual root HMM state $s = 0$ of each time conditioned tree is activated exactly once at the time τ when the tree is started, so we can extract the relative acoustic probability of a word on a specific interval:

$$p(x_\tau^t|w) = \frac{Q_\tau(t, S_w)}{Q_\tau(\tau, s = 0)} \tag{1}$$

Where $Q_\tau(t, s)$ is the probability of the most probable path ending in the state s at time t and leading through a word start

at time τ, and is propagated through dynamic programming as seen in earlier publications.

Then at time t new word end hypotheses $E_X(t)$ are expanded by correcting the acoustic probability relative to the context probability q, and by applying the corresponding LM probability relative to the word history $h = u_1^{m-1}$ for each context word end hypothesis $(h, q) \in E(\tau)$:

$$E_X(t) = \{(\tau, hw, q \cdot p(x_\tau^t|w) \cdot p(w|h))|(S_w, \tau) \in S(t), (h, q) \in E(\tau)\} \quad (2)$$

Where $(s, \tau) \in S(t)$ are the state hypotheses active at time t with state s and start-time τ, and S_w is the word end state for word w. We include the word start time τ in the expanded word end hypotheses, so the effort required during word end handling can be measured through $|E_X(t)|$.

LM pruning is used to reduce the number of word end hypotheses, by applying a pruning threshold $f_{LM} < 1$ to the expanded word end hypotheses with start-time τ, history h and probability q:

$$E_P(t) = \{(\tau, h, q)|(\tau, h, q) \in E_X(t), q > f_{LM} \cdot Q_{LM}(t)\} \quad (3)$$

Where $Q_{LM}(t) = \max_{(\cdot,q)\in E_X(t)} q$ is the probability of the most probable word end hypothesis of the time frame.

After pruning the hypotheses are recombined according to the LM, and the word start times τ are discarded:

$$E(t) = \{(u_2^m, \max_{(\cdot,\cdot u_2^m, q)\in E_P(t)} \{q\})\} \quad (4)$$

The new tree Ω_t is then started based on the probability q of the best word end hypothesis:

$$Q_t(t, s = 0) = \max_{(h,q)\in E(t)} \{q\} \quad (5)$$

2.1. LM Look-Ahead

In WCS, LM look-ahead has been shown to be of high importance to improve the efficiency [2]. Each time conditioned tree has multiple LM contexts $E(\tau)$, which makes it hard to apply context-dependent (bigram or higher) LM look-ahead efficiently. Computing the LM look-ahead probability in each state $(s, \tau) \in S(t)$ requires an effort that is linear in the number of word end hypotheses $|E(\tau)|$, since a maximization needs to be carried out over each context history h and probability q for $(h, q) \in E(\tau)$. Such a high effort does not pay out, so we have tried several heuristics to reduce it, but until now the context-independent unigram look-ahead has shown to be the most efficient option in TCS.

3. Extensions to Time Conditioned Search

3.1. Early Word End Pruning

A major problem of TCS is the number of word end hypotheses $E_X(t)$ that appear during the expansion (see Equation 2). To reduce that number, we apply an additional pruning step during the expansion.

Early Word End Pruning consists of two steps:

1. *Acoustic Word End Pruning*: Apply acoustic pruning to the *expanded* word end hypotheses, before computing the LM probability. In contrast to the normal acoustic pruning, which only respects the *best* context, this pruning is applied relative to *all* contexts of a state hypothesis.
2. *Anticipated LM Pruning* After adding the LM probability, apply an anticipated LM pruning based on the word end hypotheses that were hypothesized until now.

The set of expanded word end hypotheses with applied acoustic word end pruning is:

$$E_{XAC}(t) := \{(\tau, hw, q \cdot p(x_\tau^t|w) \cdot p(w|h))| (S_w, \tau) \in S(t), (h, q) \in E(\tau), q \cdot p(x_\tau^t|w) > Q_{AC}(t) \cdot f_{AC}\} \quad (6)$$

If the word end hypotheses $(h, q) \in E(\tau)$ are processed in an order sorted by probability q, the processing for time τ and word end state S_w can be stopped as soon as the expanded acoustic probability goes below $Q_{AC}(t) \cdot f_{AC}$, eliminating a large part of the overall effort.

The anticipated LM pruning is applied during the expansion as well:

$$E_{XLM}(t) := \{(\tau, h, q)|(\tau, h, q) \in E_{XAC}(t), q > Q_{LMP}(t) \cdot f_{LM}\} \quad (7)$$

Where $Q_{LMP}(t) = \max_{(\cdot,\cdot,q)\in E_{XLM}(t)}\{q\}$ based on the part of $E_{XLM}(t)$ that has been computed until the current hypothesis for this time frame.

Based on the expanded hypotheses $E_{XLM}(t)$, the normal word end handling follows (pruning and recombination, see Equation 3).

Table 1 shows a comparison of the word end hypotheses during different processing steps at the word boundary. The majority of potential word end hypotheses is discarded during early pruning. Since the LM pruning is applied already during the expansion (see Equation 7), LM pruning only reduces the amount of hypotheses slightly in TCS, while in WCS it removes the majority of hypotheses. As expected, a large part of the word end hypotheses in TCS are then discarded during the recombination step, when equal hypotheses from different trees with different start times are recombined. Still, although the same pruning values are used, a significantly higher amount of word end hypotheses stays active in TCS. This leads to a larger effective search space in TCS, and is the reason why TCS usually reaches a better WER at equal pruning constraints.

On the NAB'94 H1 dev corpus with a 20k words vocabulary, early word end pruning in TCS typically reduces the fraction of the overall runtime effort taken by the word end expansion from around 20% to only around 5%, thus the early word end pruning is essential for TCS to be competitive with WCS regarding the runtime.

3.2. Context Recombination

In TCS, each tree is entered at exactly one point in time, and has no further entries. Trees that are started at similar times are started with similar word end hypotheses. Adjacent time conditioned trees often contain equal *virtual* state hypotheses (state hypotheses expanded by their contexts) with different probabilities, that would be contained in only one word conditioned tree in WCS, and would most probably be recombined early on. Under specific circumstances, we can perform a similar kind of recombination in TCS, by removing word end hypotheses

Table 1: *The average number of word end state hypotheses $|(S_w, \cdot) \in S(t)|$, word end hypotheses without early pruning $|E_X(t)|$, with early pruning $|E_{XLM}(t)|$, after pruning $|E_P(t)|$, and after recombination $|E(t)|$. Computed on a small subset of the NAB'94 H1 dev corpus with a 20k word vocabulary, with relaxed pruning constraints.*

	WCS	TCS
Word end states $(S_w, \cdot) \in S(t)$	5k	5.5k
Without early pruning $E_X(t)$	5k	1150k
With early pruning $E_{XLM}(t)$		11.3k
After pruning $E_P(t)$	1.6k	8.3k
After recombination $E(t)$	0.77k	1.5k

from specific timeframes if, for a fixed word history h, all expanded state hypothesis probabilities within one tree (and thus the reachable word end hypotheses) have a lower probability than the equivalent expanded state hypotheses within another tree.

Formally, the word end hypothesis $(h, q) \in E(\tau)$ with history h and probability q can be removed from $E(\tau)$ if:

$$\forall s : q \cdot \frac{Q_\tau(t,s)}{Q_\tau(\tau,0)} < \max_{\tau', q' : (h,q') \in E(\tau')} \{q' \cdot \frac{Q_{\tau'}(t,s)}{Q_{\tau'}(\tau',0)}\} \quad (8)$$

The efficiency of this recombination is basically problematic: Actions need to be performed for every state and every context. As it can be expected that the relationships checked by this recombination do not change much over time, the recombination does not need to be performed at every time but just at a specified interval, allowing for a speedup. This recombination will be abbreviated by *CRi* where i is the interval in time frames.

3.3. Startup Interval

In practice, it usually does not lead to a high error when few very specific HMM transitions are forbidden. For example, we usually do not allow a skip from the previous-to-last HMM state of one word into the first non-virtual HMM state of the following word, because that would lead to an additional effort during the recombination, without a practical benefit. The HMM can easily compensate for such forbidden transitions, by performing the skip transition either before entering the new word, or afterwards.

Going one step further, we could also forbid the entry of a word at specific time frames, which the HMM could also compensate up to some level, by performing loops and skips accordingly at the word starts and ends. Since in TCS we start one tree at each time frame, we can save a significant part of the overall effort by forbidding the entry of words at specific times.

We can define an interval at which we start time conditioned trees, and at all time frames that don't match the interval, we define the set of word end hypotheses to be empty:

$$\text{if } \tau \bmod i \neq 0 \text{ then } E(\tau) = \emptyset \quad (9)$$

TCS with an interval of 1 equals the normal TCS, and an interval of 2 means that we start time conditioned trees every second time frame. WCS on the other hand will not benefit much from such an interval, as it starts trees based on word histories.

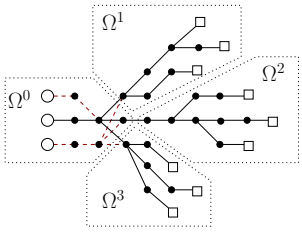

Figure 1: *Illustration of a partitioned simplified coarticulated search tree. Only 3 root states are illustrated (the circles at the left). The rectangles represent word end states. The central root state is the non-coarticulated root, the recombination paths behind coarticulated roots are illustrated in red.*

3.4. Time Conditioned Search with Across Word Modelling

When using across word modelling, the search network contains multiple fan-in arcs, and thus has no unique root state. The classical TCS algorithm can only be applied if there is exactly one root state, relative to which the acoustic probability $p(x_\tau^t|w)$ can be computed at the word ends (see Equation 1). Figure 1 illustrates a simplified coarticulated search tree with multiple roots. Notably all paths behind the coarticulated root states are recombined at latest after the first phoneme, and after that the search graph spans multiple subtrees with a fully valid tree structure (each subtree has exactly one root, and the fan-in is always 1). The whole coarticulation happens within the root tree Ω^0, and the subtrees Ω^1 to Ω^3 each have a tree structure that is compatible with TCS. TCS can be combined with across word modelling by partitioning the search tree as illustrated, and instantiating all subtrees as time conditioned trees $\Omega_\tau^{r>1}$ for time τ, and by instantiating the coarticulated root network as word conditioned tree Ω_h^0 with history h.

4. Experimental Results

4.1. Partitioned State Tree

The search tree of the EPPS English [1] recognition system was partitioned into a total of 944 subtrees $Q^{r>0}$ that together contain about 4971k states. 566k states are contained by the coarticulated root tree Ω^0 that represents the HMM of the first phonemes of the words. While Ω^0 is quite large, only on average about $16k$ of those states are reachabe within one word conditioned tree instance, as only those root states are activated that match the last phoneme of the predecessor word.

In TCS at a moderate acoustic pruning threshold of 300, on average 738 time conditioned subtrees $\Omega_\tau^{r>0}$ are active and each contains 42 state hypotheses. On average 76 word conditioned root trees Ω_h^0 are active, and each of those contains 215 state hypotheses. That means that 16.3k state hypotheses are contained by the word conditioned root trees, which makes up 34% of the search space, and 31.4k state hypotheses are contained by the time conditioned subtrees.

4.2. Startup Interval

Figure 2 shows the effect of different tree startup intervals in TCS with across word modelling. The RTFs were measured on one single Intel Core2 Duo 2.6 GHZ machine with 4 GB of memory, on one half of the corpus. Increasing the startup interval from 1 to 2 clearly improves the performance of TCS, for example for the WER of 13.1% a reduction of the RTF by a

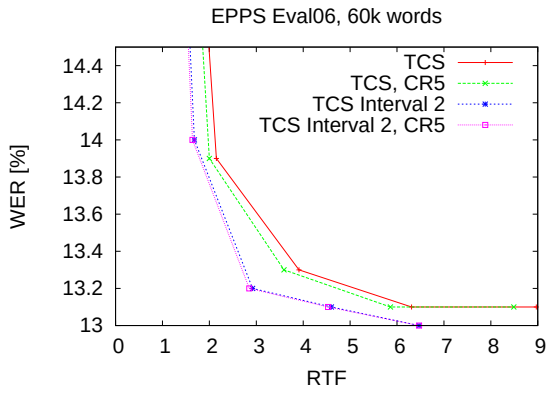

Figure 2: *WER and RTF in word- and time conditioned search with varied acoustic pruning threshold. Measured on the EPPS Eval06 English corpus with across word modelling.*

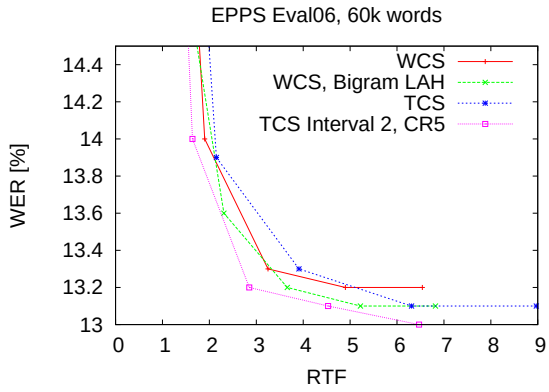

Figure 3: *WER and RTF in word- and time conditioned search with varied acoustic pruning threshold. Measured on the EPPS Eval06 English corpus with across word modelling.*

factor of 1.3 is achieved. We have also tested the interval 3, but it seems to be too large, and induces a significant error.

4.3. Context Recombination

Figure 2 illustrates the effect of context recombination. For a WER of 13.1%, context recombination reduces the RTF by 7%. However in combination with a tree startup interval of 2, the effect is only 1%.

4.4. Comparison with WCS

Figure 3 shows the comparison between WCS with unigram look-ahead, WCS with bigram look-ahead, TCS, and TCS with context recomination and tree startup interval 2. TCS reaches a slightly better WER than WCS with unigram look-ahead, but is mostly slightly slower for the same WER than WCS with unigram look-ahead. WCS with bigram look-ahead allows reaching the same best WER of 13.1% as TCS on this corpus, and reaches that WER at a significantly better RTF than TCS, otherwise it mostly performs similar to WCS with unigram look-ahead. TCS with context recombination and a tree startup interval of 2 clearly outperforms all other methods on this corpus.

5. Conclusions

We have shown that TCS can compete with WCS on state-of-the-art LVCSR tasks, and we have introduced several techniques that together even allow TCS to outperform WCS on our test corpus, despite WCS having the advantage of the bigram LM look-ahead. The structure of the search space in TCS is largely independent of the size of the vocabulary and the LM, which may specifically be an advantage when dealing with complex LMs with long-span history dependences.

6. Future Work

Several questions stay open for TCS.

- The way we have integrated TCS with across word modelling actually produces a kind of hybrid search, since the coarticulated root network is always started as a word conditioned tree. A future goal will be to make the search space *fully* time conditioned.
- The integration of context-dependent (bigram) LM look-ahead into TCS needs further investigation, as it typically leads to a significant advantage in WCS.
- The tree startup interval could be formulated more elegantly as a TCS-specific pruning problem.

7. Acknowledgements

This work was partly realized under the Quaero Programme, funded by OSEO, Frech State agency for innovation.

8. References

[1] J. Lööf, M. Bisani, C. Gollan, G. Heigold, B. Hoffmeister, C. Plahl, R. Schlüter, and H. Ney. *The 2006 RWTH Parliamentary Speeches Transcription System*. TC-STAR Workshop on Speech-to-Speech Translation, pp. 133-138, Barcelona, Spain, 2006.

[1] H. Ney and S. Ortmanns. *Progress in Dynamic Programming Search for LVCSR*. Proceedings of the IEEE, Vol. 88, No. 8, pp. 1224-1240, Aug. 2000.

[2] S. Ortmanns and H. Ney. *The Time-Conditioned Approach in Dynamic Programming Search for LVCSR*. IEEE Trans. on Speech and Audio Processing, Vol. 8, No. 6, pp. 676-687, Nov. 2000.

[3] S. Ortmanns and H. Ney. *Look-Ahead Techniques for Fast Beam Search*. Computer Speech & Language, Vol. 14, No. 1, Jan. 2000.

[4] X. Aubert. *An overview of decoding techniques for large vocabulary continuous speech recognition*. Computer Speech & Language, vol. 16, no. 1, pp. 89-114, January 2002.

[5] S. Ortmanns, H. Ney, F. Seide and I. Lindam. *A Comparison Of Time Conditioned And Word Conditioned Search Techniques For Large Vocabulary Speech Recognition*. Proc. Int. Conf. on Spoken Language Processing, pp. 2091-2094, 1996.

[6] A. Sixtus. *Across-Word Phoneme Models for Large Vocabulary Continuous Speech Recognition*. Ph. D. thesis, RWTH Aachen, 2003.

[7] H. Soltau and G. Saon. *Dynamic Network Decoding revisited*. Proc. 2009 IEEE Workshop on Automatic Speech Recognition & Understanding, Dec. 2009.

[8] D. Rybach, C. Gollan, G. Heigold, B. Hoffmeister, J. Lööf, R. Schlüter, and H. Ney. *The RWTH Aachen University Open Source Speech Recognition System*. Interspeech, pp. 2111-2114, Brighton, U. K., Sep. 2009.

[9] C. Allauzen, M. Mohri, M. Riley, and B. Roark. *A Generalized Construction of Integrated Speech Recognition Transducers*. ICASSP, 2004.