# Word Pair Approximation for More Efficient Decoding with High-Order Language Models

*David Nolden[a], Ralf Schlüter[a], Hermann Ney[a,b]*

[a] RWTH Aachen University {nolden,schluter,ney}@cs.rwth-aachen.de

[b] Spoken Language Processing Group, LIMSI CNRS, Paris, France

## Abstract

The search effort in LVCSR depends on the order of the language model (*LM*); search hypotheses are only recombined once the LM allows for it. In this work we show how the LM dependence can be partially eliminated by exploiting the well-known word pair approximation. We enforce preemptive unigram- or bigram-like LM recombination at word boundaries. We capture the recombination in a lattice, and later expand the lattice using LM rescoring. LM rescoring unfolds the same search space which would have been encountered without the preemptive recombination, but the overall efficiency is improved, because the amount of redundant HMM expansion in different LM contexts is reduced. Additionally, we show how to expand the recombined hypotheses on-the-fly, omitting the intermediate lattice form. Our new approach allows using the full n-gram LM for decoding, but based on a compact unigram- or bigram search space. We show that our approach works better than common lattice rescoring pipelines, where a pruned lower-order LM is used to generate lattices; such pipelines suffer from the weak lower-order LM, which guides the pruning suboptimally. Our new decoding approach improves the runtime efficiency by up to 40% at equal precision when using a large vocabulary and high-order LM.

**Index Terms**: efficient, decoding, search, rescoring, word pair approximation, context approximation

## 1. Introduction

It is a popular belief that the order of the incorporated language model (LM) during LVCSR decoding is a major factor which degrades the efficiency. For some decoding algorithms this is certainly true. For Weighted finite state transducer (WFST) decoders which compose the LM and the lexicon statically, the static composition becomes too costly when the LM is too large; the LM needs to be pruned down for the composition, and lattices need to be generated, afterwards the lattices can be rescored using the unpruned LM [1, 2]. Dynamic on-the-fly composition is an alternative approach for situations where a static composition is not feasible [3], but the dynamic composition may become very costly, especially when using a large vocabulary [4]. To profit from a large high-order LM during initial decoding in a WFST based decoder, without the costs of composition, and without lattice rescoring, approaches to combine a unigram-based search space and a high-orer LM on-the-fly during decoding were proposed [5], thereby re-inventing the classical dynamic token-passing decoder [6, 2].

Dynamic network decoders, whether based on the token-passing or word-conditioned concept, combine an optimized single-word search network and the LM on-the-fly, and are able to use arbitrarily large LMs in a single decoding pass [2, 7, 8]. A powerful LM helps focussing the search, prevents search errors, and thus helps achieving better results at tighter pruning thresholds.

In [8] we have shown that a dynamic network decoder can outperform an efficient rescoring-based static WFST decoder when a large LM is used. Although dynamic decoders are able to deal with huge $n$-gram LMs, high order LMs still impose certain problems, which might be avoidable; firstly, hypothesis recombination is delayed until $n-1$ equal words were recognized, which increases the theoretical search space by orders of magnitude; secondly, full-order LM look-ahead [9] becomes more expensive with higher LM order, because one LM look-ahead table needs to be filled for each encountered LM context of length $n-1$. The first problem is relaxed by the fact that usually the overall search effort focusses on a small set of active LM histories; the second problem is avoided in our decoder, by exploiting the LM sparsity [9], and by activating full-order LM look-ahead only for such LM histories, which correspond to at least a certain minimum fraction of the active search space.

Nevertheless, equal words with equal acoustic realization and time alignment, can be encountered in many different LM contexts at the same time. Such words then incur redundant search costs, because the best time alignment needs to be determined for each LM context separately.

The time conditioned search framework [10, 11] is one way to avoid redundant evaluation of equal HMMs in different LM contexts; however, time conditioned search shifts some of the overall costs from within-word search to the cross-word recombination, and is not compatible with some state-of-the-art methods like full-order sparse LM look-ahead [9]. Thus we do not follow the time-conditioned approach directly.

Instead, we show how to modify a common dynamic network decoder to severely limit the redundant decoding of equal word realizations in different LM contexts. We derive our motivation from the well-known word pair approximation, which is commonly used for lattice generation. Something similar was tried in [12], but based on a more severe modification of the standard search strategy, which is not compatible with full-order LM look-ahead, and which introduces many approximation errors. Our approach avoids these approximation errors, by only focussing on word ends, which yields a certain amount of self-healing (this is discussed in Section 5).

In our approach, we modify the word end recombination to reduce the number of considered LM histories, and then we add a succeeding lattice rescoring pass, which expands those LM histories which were omitted during decoding.

In a second step, we integrate the context approximation directly into a word conditioned decoder, by expanding recombined hypotheses directly when encountering the next word end, instead of during lattice rescoring. This approach allows the decoder to correctly consider all expanded LM histories, and leads to more precise pruning and LM look-ahead. It yields a more elegant integrated framework, which does not require lattice generation and -rescoring. However, this second approach requires more severe modifications to the decoder, and induces some additional overhead during decoding.

## 2. Dynamic Network Decoding

A dynamic network decoder combines the LM and an optimized single-word search network dynamically [7, 8]. A state hypothesis $(h, s, q)$ consists of a LM history $h$, a state $s$ in the single-word search network, and a path probability $q$. Additionally a traceback is attached to each hypotheses, which tracks the full word history since sentence start. Hypotheses are propagated through the search network using dynamic programming, and if multiple hypotheses meet on the same history $h$ and state $s$, they are recombined (i.e. only the best one is preserved). At each timeframe, hypotheses on word end states $s = S_w$ with word label $w$ are expanded to word end hypotheses, by applying the LM probability $p(w|h)$, and extending the LM history $h$ by the next word $w$. When across word modelling is used [13], then the search network has many coarticulated root states, and

each word end state $S_w$ has a specific target root state $R(S_w)$, which is followed by the hidden-markov-model (HMM) representation of all correspondingly coarticulated successor words [7]. After expansion of word end hypotheses, they are first pruned, then their traceback is extended, and then they are recombined according to their new history $h'$ and their target root state $R(S_w)$. During word end recombination, we can straightforwardly generate a lattice, by simply linking the tracebacks of deleted word end hypotheses into the tracebacks of corresponding surviving ones, so that no traceback path is lost. According to the word pair approximation [14], this approach generates a correct lattice, when the length of histories $h$ is at least 1 (see Section 5).

## 3. Context Approximation (CA)

When using an $n$-gram LM, we can usually recombine two hypotheses after $n-1$ equal words were recognized. When using a higher-order LM, for example a 4-gram, this means that search paths are recombined only after 3 equal words were recognized. Word pair approximation tells us that we could generate a good lattice after a single equal word was recognized. This leads us to the idea to shorten the LM histories used during recombination according to word pair approximation, and expand full LM histories later in a lattice rescoring pass.

As we will see in Section 5, word pair approximation is just one specific level of *context approximation (CA)*. We will define and evaluate three CA levels $C(h)$ for histories $h$:

| | |
|---|---|
| word | $C(h)$ :=last word of history $h$ |
| $n$-phone | $C(h)$ :=last $n$ phones of last word of history $h$ |
| none | $C(h)$ :=nil |

The *word* CA corresponds to word pair approximation, and the *none* CA completely ignores the context.

During word end recombination, we usually recombine word end hypotheses $(h, s, q)$ and $(h', s', q')$ if:

$$h = h' \wedge R(s) = R(s') \tag{1}$$

With CA, we change the criterion to:

$$C(h) = C(h') \wedge R(s) = R(s') \tag{2}$$

For within-word dynamic programming, we retain the standard recombination criterion $h = h' \wedge s = s'$, based on the history $h$ which has won during the preceding word end recombination. Figure 1 shows an example 3gram lattice, and the corresponding lattices created using the *word* and *none* CA. Decoding with CA yields an approximate lattice with wrong context dependency, and highly increased single-best error rate. By applying LM rescoring, we can re-expand the lattice to its full-order form.

The advantage of lattice rescoring over handling the full histories during decoding is, that dead-end arcs can be removed before the LM rescoring, and that within-word HMM alignments don't need to be performed for each of the LM histories separately.

## 4. Integrated Context Approximation

The lattice rescoring approach from the previous section is easy to implement, because it requires only minimal changes to the recombination of a common dynamic network decoder. It suffers from two problems though; firstly, a lattice rescoring step is required, which forces us to generate lattices in the first place, even if we don't need them, and which yields a 2-pass strategy; secondly, the overall precision of the decoding might be reduced, because the LM histories used during decoding are constrained in a problematic way.

This leads us to the idea of re-expanding the preemptively recombined hypotheses directly when reaching the next word end, similarly to time conditioned search [10, 11]. We perform preemptive recombination as in the previous lattice rescoring

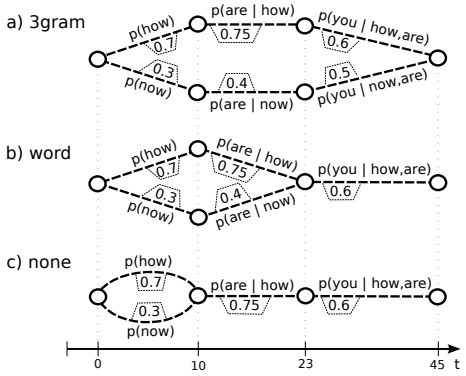


Figure 1: *Lattices generated with different levels of CA.*

approach, but we remember which tracebacks we preemptively combine together; then, when reaching the next word end, we expand a separate word end hypothesis for each of the tracebacks which were combined at the word start, each with corrected acoustic and LM score. This expansion is very frequent, and unlike the lattice based method, many dead-end paths are expanded, thus early pruning is required to limit the expansion; we sort the tracebacks by score during the preemptive recombination, and expand them in the same order, aborting the expansion as soon as an expanded hypothesis falls beyond the pruning threshold. We use a slightly relaxed word end pruning threshold for this pruning.

When following this integrated approach, CA only affects the within-word dynamic programming. During the handling of word ends, full LM histories are expanded, and no further LM rescoring is required.

## 5. Motivation

Why do we need word pair approximation for lattice generation at all? As mentioned in Section 2, we only link tracebacks to form a lattice during the word end recombination. When two state hypotheses are recombined during the within-word dynamic programming, then only the traceback of the better hypothesis is preserved, and the traceback of the worse hypothesis is lost.

According to word pair approximation, the *optimal* boundary time $t_{opt}$ between $w$ and any successor words $w'$ is *the same*

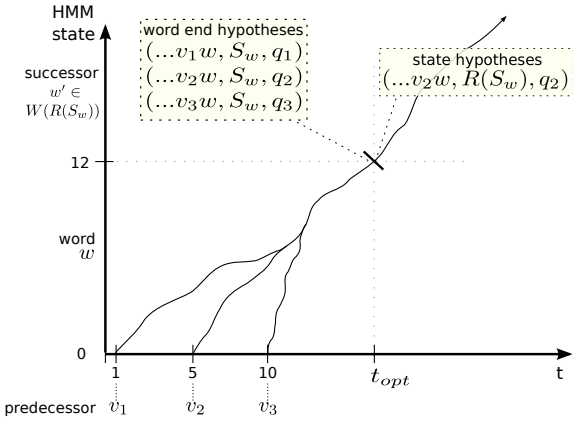


Figure 2: *Illustration of word pair approximation, with 3 predecessor words $v_1$, $v_2$, and $v_3$, successor word $w'$, and optimal boundary time $t_{opt}$. The path through $v_2$ is most likely.*

for all words $v$ that came before $w$. Thus, word end hypotheses corresponding to word $w$ following the optimal alignment path of predecessor words $v_1$ and $v_2$, are guaranteed to get recombined and linked together during the word end recombination at one specific optimal timeframe $t_{opt}$, and no optimal path to a predecessor word $v_i$ can get lost. The within-word dynamic programming only needs to optimize over the boundary time $t_{opt}$, which is the same for all preceding histories; thus we don't need to link a lattice during within-word dynamic programming, it is sufficient to do that during word end recombination.

Figure 2 illustrates the optimal alignment paths through word $w$ following 3 different predecessor words $v_1$, $v_2$, and $v_3$; $W(s)$ is the set of word ends reachable from state $s$. The word pair approximation is satisfied here, because the optimal alignment paths merge within the HMM corresponding to the common successor word $w$, and thus the ideal word boundary $t_{opt}$ between $w$ and a hypothetical successor word $w'$ is the same for all predecessors. Word pair approximation follows from the *monotonicity* and *convergence* assumptions as described in [15]; the critical condition for paths to converge is that they are aligned with a sufficiently long HMM.

### 5.1. Shorter Context Approximation

The word $w$ illustrated in Figure 2 consists of 12 HMM states. But what happens, if we apply CA from Section 3, and reduce the number of common HMM states over which we expect the optimal alignment paths to converge? Figure 3 shows an example of the ideal alignment paths of 3 different words $w_1$, $w_2$, and $w_3$, which we expect to recombine while aligning their last phoneme $a$, according to the 1-phone CA. In this example, the assumption fails for the predecessor word $w_1$; its ideal alignment path crosses the word boundary at $t_{opt_1} = 7$, instead of $t_{opt} = 13$ like for the other predecessor words.

CA with very short context lengths is likely not to satisfy the path recombination, as shown in Figure 3. But what does that mean for us? For example, consider that the most likely history at $t_{opt}$ is the one following $w_2$. In worst case, the ideal alignment path following $w_1$ is lost, because its followup hypothesis is overwritten by the path following $w_2$, during within-word dynamic programming of the next word. Since we retain the full-order LM histories for the within-word dynamic programming, this can only happen if the most likely word end LM history at $t_{opt_1}$ is the same as at $t_{opt}$ (i.e. it is ending with word $w_2$). Otherwise a different best LM history is selected during the word end recombination at $t_{opt_1}$, and the successor paths of $w_1$ and $w_2$ will never meet during within-word dynamic programming; thus the alignment path following $w_1$ won't get lost. It is not very likely that $w_2$ is the best hypothesis at timeframe $t_{opt_1}$, because $t_{opt_1}$ is quite far away from $t_{opt}$. Furthermore, even if the ideal alignment path following $w_1$ gets lost, there may still be a less optimal path following $w_1$ going through $t_{opt}$, if the beam pruning threshold is large enough (see the dotted path following $w_1$ in Figure 3). These two arguments are not independent: The closer $t_{opt}$ and $t_{opt_1}$ are, the more likely it is that both have the same dominant predecessor word $w_i$, and in turn, the more likely it is that a traceback is incorrectly overwritten. However, the potential loss arising from the overwriting declines with a reduced distance between $t_{opt}$ and $t_{opt_1}$, because the closer they are together, the less costly it is for the path following $w_1$ to take the detour over $t_{opt}$ instead of $t_{opt_1}$. Thus, in our proposed decoding framework, too short CA is *likely* to *slightly* degrade the quality of cross-word HMM alignment paths, by forcing the paths to take a slight detour.

The *none* CA, which completely ignores the context, can be motivated by the structure of the search network. We only recombine word end hypotheses which point to the same coarticulated root state. Due to the coarticulation, these root states are context dependent. When using triphonic acoustic models, then they usually depend on the left and right context phoneme.

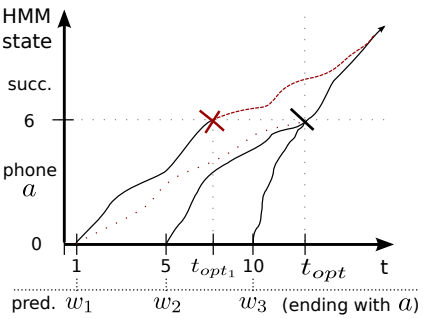


Figure 3: *Illustration of failed 1-phone CA, with 3 predecessor words $w_1$, $w_2$, and $w_3$, all ending with the same phone $a$.*

In our decoder, we completely minimize the root states, so a part of this context dependency is lost, but some context dependency certainly survives the minimization. If we wouldn't minimize the roots, then *1-phone* CA would have no effect, because it would already be an implicit part of the recombination.

## 6. Experimental Results

Our basic decoder was described in [7]. Acoustic look-ahead is used to focus the pruning [16], and we use an efficient batching feature scorer with density clustering. All real time factors (RTF) are measured on our computing cluster under full load. The cluster mostly consists of 16-core AMD Opteron machines with 2.6Ghz.

### 6.1. Results on Quaero Polish 600k

The Quaero Polish system uses a vocabulary of 600k words and a 4-gram LM consisting of 82M n-grams.

Figure 4 (left) compares the search space under varied global beam pruning, with different levels of integrated CA. The *word* CA yields only a minor gain. The *1-phone* CA gives a significant gain, but the *none* CA, which ignores the context altogether, works best; it achieves a reduction of 50 to 70% in comparison to the baseline. This shows that the context-dependency which is provided implicitly by the coarticulated root states is sufficient (see Section 5). When not using across-word modelling, at least the *1-phone* approximation is required, to avoid a large amount of recombination errors. In the following we will always use the *none* context-independent approximation.

Figure 4 (right) shows the effect of the different decoding strategies on the relationship between WER and search space. By using lattice-based CA followed by 4gram LM rescoring (i.e. *approx-lat-rescore*), we can reduce the search space by 50 to 75%. Integrated CA performs similarly, but surprisingly a little worse. The precision of the integrated CA suffers from

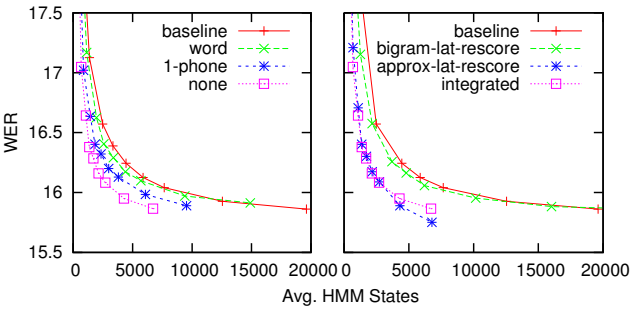


Figure 4: *WER vs. search space on Quaero Polish. Left: different levels of CA. Right: bigram decoding followed by rescoring, lattice-based* none *CA, and integrated* none *CA.*

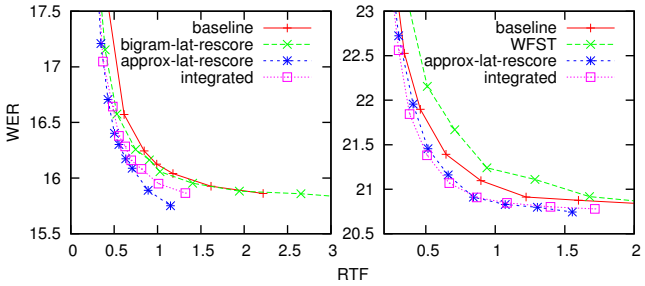


Figure 5: *WER vs. RTF on Quaero Polish (left) and -English (right), for baseline 4gram decoding, dynamic WFST decoding, bigram decoding followed by rescoring, lattice-based CA, and integrated CA.*

the pruning which we apply during the expansion of combined traces.

Figure 5 (left) shows the relationship between WER and RTF for the different strategies. We achieve a reduction of the RTF by 20 to 50% at equal precision. The lattice-based approach is considerably faster than the integrated approach on this task.

### 6.2. Results on Quaero English 180k

The following experiments are performed on the RWTH Aachen Quaero English ASR system [17]. The lexicon comprises 158k words with 180k pronunciations, and the 4-gram LM contains 50M n-grams.

Figure 5 (right) compares the RTF using the different decoding algorithms. We add our OpenFST based WFST decoder with on-the-fly composition to the comparison, which we introduced in [18] and extended with acoustic look-ahead in [19]. On top of the modifications mentioned in that paper, we recently added a new pragmatic cross-arc pruning method, which dramatically reduces the costs of the dynamic WFST composition; it applies a tighter pruning beam to HMM transitions that cross multiple WFST arcs, similar to the arc-pruning mentioned in [8]. Cross-arc pruning basically halves the RTF of the WFST based decoder at equal WER. However, it still can not fully match the dynamic decoder baseline, which profits much more from the acoustic look-ahead [7]. Both lattice-based and integrated CA reduce the RTF by around 20% at equal WER. On this task, we see the improved precision of the integrated approach, which generates slightly better WERs for the tighter pruning thresholds.

Figures 6 shows how the profiling of the baseline (left) and the integrated CA (right) changes with the beam size.

Figure 7 compares the efficiency of the different methods

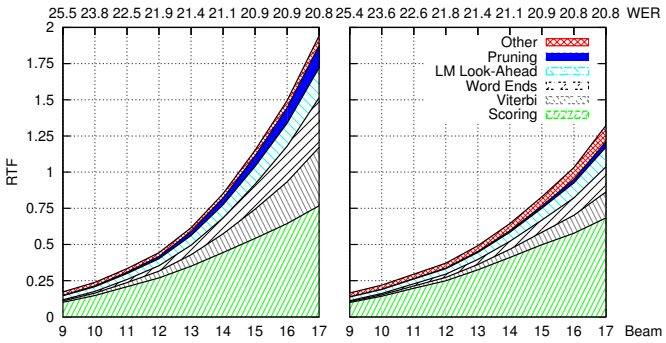


Figure 6: *Profiling of baseline (left), and integrated CA (right).*

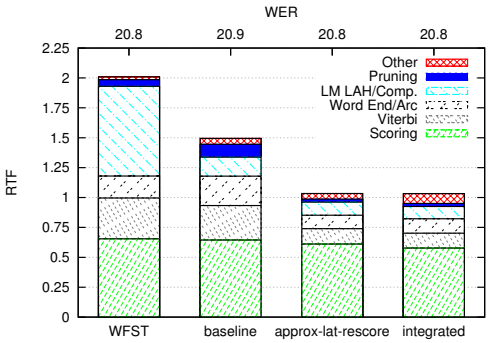


Figure 7: *Profiling on Quaero English.*

at beam size 16, which is the point at which most of the methods reach a WER close to the optimum. The portion spent with acoustic scoring is nearly constant for the different methods, and CA can impossibly reduce that portion. The remaining costs of the actual decoding are nearly halved by CA. We see that the WFST decoder spends more time on the on-the-fly composition than on acoustic scoring, despite its arc pruning. Even if we ignore the costs of on-the-fly composition, as if the search network was composed statically, the dynamic decoder with CA stays faster than the WFST decoder.

## 7. Conclusions

We have introduced two novel decoding strategies which considerably reduce the search effort of a dynamic network decoder, especially when using a high-order LM. The first method is a simple modification of the word end recombination, followed by a lattice rescoring step. The second integrated method uses the same approximative recombination, but expands it directly when encountering the next word end, instead of delaying the expansion to a lattice rescoring step. Both methods work similarly well, and allow reducing the RTF by up to 50% when using a high-order LM. The most efficient level of context approximation is the one which completely ignores the LM context during the word end recombination. This seems like a striking result at first, but it can be motivated by the across-word-modelling, which yields a context-dependency similar to the *1-phone* context approximation described in Section 5. We have compared our approach to a standard bigram-lattice based rescoring method, and found that our approach works better, because it allows integrating the full-order LM right during the initial decoding. Our approach allows to partially decouple the effort of decoding from the LM order.

## 8. Acknowledgements

This work was partly realized under the Quaero Programme, funded by OSEO, French State agency for innovation, and from the European Union Seventh Framework Programme EUBridge (FP7/2007-2013) under grant agreement no. 287658.

H. Ney was partially supported by a senior chair award from DIGITEO, a French research cluster in Ile-de-France.

Supported by the Intelligence Advanced Research Projects Activity (IARPA) via Department of Defense U.S. Army Re- search Laboratory (DoD / ARL) contract number W911NF- 12-C-0012.[1]

[1]The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon. Disclaimer: The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of IARPA, DoD/ARL, or the U.S. Government.

# 9. References


[1] G. Saon, D. Povey, and G. Zweig, "Anatomy of an Extremely Fast LVCSR Decoder," in *Interspeech*, 2005.

[2] H. Soltau and G. Saon, "Dynamic Network Decoding revisited." Merano: ASRU, December 2009, pp. 276 – 281.

[3] C. Allauzen, M. Riley, and M. Mohri, "A Generalized Composition Algorithm for Weighted Finite-State Transducers," in *Interspeech*, Brighton, U.K., September 2009, pp. 1203 – 1206.

[4] D. Nolden, R. Schluter, and H. Ney, "Advanced search space pruning with acoustic look-ahead for wfst based lvcsr," in *ICASSP*, May 2013, pp. 6734–6738.

[5] T. Hori, C. Hori, Y. Minami, and A. Nakamura, "Efficient wfst-based one-pass decoding with on-the-fly hypothesis rescoring in extremely large vocabulary continuous speech recognition," *Audio, Speech, and Language Processing, IEEE Transactions on*, vol. 15, no. 4, pp. 1352–1365, May 2007.

[6] S. J. Young, N. H. Russell, and J. H. S. Thornton, "Token Passing: a Simple Conceptual Model for Connected Speech Recognition," in *Tech. Report*, Cambridge University Engineering Department, 1989.

[7] D. Nolden, D. Rybach, R. Schlüter, and H. Ney, "Joining Advantages of Word-Conditioned and Token-Passing Decoding," in *ICASSP*, 2012.

[8] D. Nolden, H. Soltau, and H. Ney, "Progress in Dynamic Network Decoding," in *ICASSP*, May 2014.

[9] D. Nolden, H. Ney, and R. Schlüter, "Exploiting Sparseness of Backing-Off Language Models for Efficient Look-Ahead in LVCSR," in *ICASSP*, Prague, Czech Republic, May 2011, pp. 4684 – 4687.

[10] S. Ortmanns, H. Ney, F. Seide, and I. Lindam, "A Comparison Of Time Conditioned And Word Conditioned Search Techniques For Large Vocabulary Speech Recognition." ICSLP, 1996, pp. 2091 – 2094.

[11] D. Nolden, H. Ney, and R. Schlüter, "Time Conditioned Search in Automatic Speech Recognition Reconsidered," in *Interspeech*, Makuhari, Japan, September 2010.

[12] F. Seide, "The Use of Virtual Hypothesis Copies in Decoding of Large-Vocabulary Continuous Speech," *IEEE Transactions on Speech and Audio Processing*, vol. 13, no. 4, pp. 520–533, 2005.

[13] A. Sixtus, "Across-Word Phoneme Models for Large Vocabulary Continuous Speech Recognition," Ph.D. dissertation, RWTH Aachen, Germany, 2003.

[14] H. Ney and X. Aubert, "A Word Graph Algorithm for Large Vocabulary Continuous Speech Recognition," vol. 3. Yokohama, Japan: ICSLP, September 1994, pp. 1355 – 1358.

[15] D. Nolden, R. Schlüter, and H. Ney, "Search Space Pruning Based on Anticipated Path Recombination in LVCSR," in *Interspeech*, 2012.

[16] D. Nolden, R. Schlüter, and H. Ney, "Acoustic Look-Ahead for More Efficient Decoding in LVCSR," in *Interspeech*, Florence, Italy, August 2011.

[17] M. Sundermeyer, M. Nußbaum-Thom, S. Wiesler, C. Plahl, A. El-Desoky Mousa, S. Hahn, D. Nolden, R. Schlüter, and H. Ney, "The RWTH 2010 Quaero ASR Evaluation System for English, French, and German," in *ICASSP*, Prague, Czech Republic, May 2011, pp. 2212–2215.

[18] D. Rybach, R. Schlüter, and H. Ney, "A Comparative Analysis of Dynamic Network Decoding," in *ICASSP*.

[19] D. Nolden, R. Schluter, and H. Ney, "Advanced Search Space Pruning with Acoustic Look-Ahead for WFST Based LVCSR," in *ICASSP*, 2013, pp. 6734–6738.