IDIAP RESEARCH REPORT

A Generalized Dynamic Composition Algorithm of Weighted Finite State Transducers for Large Vocabulary Speech Recognition

Octavian Cheng [a] [b]   John Dines [a]
Mathew Magimai Doss [c]

IDIAP–RR 06-62

October 2006

submitted for publication

[a] IDIAP Research Institute, Martigny, Switzerland
[b] Department of Electrical and Computer Engineering, The University of Auckland, New Zealand
[c] International Computer Science Institute, Berkeley, California, USA

IDIAP Research Institute **www.idiap.ch**
Rue du Simplon 4 P.O. Box 592 1920 Martigny – Switzerland
Tel: +41 27 721 77 11 Fax: +41 27 721 77 12 Email: info@idiap.ch

IDIAP Research Report 06-62

A Generalized Dynamic Composition Algorithm of Weighted Finite State Transducers for Large Vocabulary Speech Recognition

Octavian Cheng John Dines Mathew Magimai Doss

October 2006

submitted for publication

Abstract. We propose a generalized dynamic composition algorithm of weighted finite state transducers (WFST), which avoids the creation of non-coaccessible paths, performs weight look-ahead and does not impose any constraints to the topology of the WFSTs. Experimental results on Wall Street Journal (WSJ1) 20k-word trigram task show that at 17% WER (moderately-wide beam width), the decoding time of the proposed approach is about 48% and 65% of the other two dynamic composition approaches. In comparison with static composition, at the same level of 17% WER, we observe a reduction of about 60% in memory requirement, with an increase of about 60% in decoding time due to extra overheads for dynamic composition.

1 Introduction

Recently, the use of Weighted Finite State Transducers (WFST) for Large Vocabulary Continuous Speech Recognition (LVCSR) has become an attractive approach [1, 2]. In simple terms, a WFST is a finite state machine which maps sequences of input symbols to sequences of output symbols with an associated weight. In the application of WFST in LVCSR, the idea is to represent each individual knowledge source by a WFST and fully integrate them into a unified WFST by the composition algorithm [2]. The fully integrated WFST provides weighted mappings from HMM state sequences to word sequences. Thus the speech recognition problem becomes searching for the mapped sequence with the lowest associated weight (cost). The composition of knowledge sources is a one-off process and is done offline. Therefore it is often referred to *static* composition.

There are two main advantages with the static approach. First the decoder design is simple because all the knowledge sources are integrated into one compact WFST. The knowledge sources are *decoupled* from the Viterbi search and therefore the decoder does not need to perform any combination of knowledge sources during decoding. The second advantage is that the fully integrated transducer can be further optimized by algorithms, such as, determinization, minimization and weight-pushing [1, 3].

Despite the above advantages, there are several drawbacks with the static approach. They include:

- The composition and optimization of the fully integrated WFST has prohibitively high memory requirement when the constituent WFSTs are large and complex;
- The size of the fully integrated WFST can be very large, resulting in large memory requirement during decoding;
- It does not allow on-line modification of knowledge sources once they have been fully integrated.

One way of addressing these issues is to perform *dynamic* transducer composition during decoding. Instead of representing the entire search space by an optimized transducer, it is possible to factorize the search space into two or more transducers. These component transducers are built statically and optimized separately. The combination is done dynamically during decoding.

In this paper, we investigate several existing dynamic composition approaches and propose our improved algorithm, which avoids the creation of non-coaccessible transitions, performs weight look-ahead and does not impose any constraints to the topology of component WFSTs. The paper is organized as follows. Section 2 briefly describes static WFST composition and how a fully integrated WFST is generated. Section 3 gives a general overview on current approaches to dynamic WFST composition. Section 4 describes our dynamic composition algorithm. Experimental results on different composition methods are shown in Section 5. Finally, Section 6 concludes the paper.

2 Static WFST Composition

Static WFST composition involves integration of all the knowledge sources. It can be represented by the following expression [2].

$$N = \pi_\epsilon(min(det(\tilde{H} \circ det(\tilde{C} \circ det(\tilde{L} \circ G))))) \tag{1}$$

In the above expression, $\tilde{H}$ represents the HMM topology; $\tilde{C}$ is a WFST which maps context-dependent phones to context-independent phones; $\tilde{L}$ is the lexicon WFST and G is the language model (LM) WFST. The symbol $\circ$ is the composition operator. Transducer optimization algorithms, for example determinization and minimization, are represented by *det* and *min* operators respectively. The $\tilde{\cdot}$ symbol means that the WFST is augmented with auxiliary symbols which are necessary for the success of transducer optimization. The π_ϵ operation replaces the auxiliary symbols by ϵ (null) symbols. The final transducer N is a fully integrated transducer which maps HMM state sequences to word sequences.

3 Current Approaches to Dynamic WFST Composition

Several groups of researchers have proposed different approaches to dynamic WFST composition. They include Dolfing [4], Willett [5], Caseiro [6] and Hori [7]. The first step of any dynamic composition algorithm is to factorize the entire search space into two or more component WFSTs before decoding. Approaches include:

1. Separating the entire G from other knowledge sources, resulting in two WFSTs [6];

2. Separating only part of the G (G_i or so called the *incremental* LM) from other knowledge sources. The remaining part of the LM (G_s or the *smearing* LM) is statically composed with other knowledge sources, resulting in two WFSTs [4, 5];

3. Factorizing the entire search space into multiple WFSTs [7].

During decoding, the component WFSTs are composed on-the-fly. There are two main approaches for combining component transducers dynamically, namely *with no look-ahead* and *with look-ahead*.

The *no look-ahead* approach is basically the dynamic version of static WFST composition. When two WFSTs, for example $\tilde{L}$ and G, are composed, the ϵ-output labels of $\tilde{L}$ are treated as "free-entries". They are not mapped with the input labels of G. These transitions are duplicated into the composite transducer, which is $\tilde{L} \circ G$ in this example.

There are two problems with this approach. The first problem is the creation of non-coaccessible transitions or so called "dead-end" transitions [6]. They are the transitions which will not reach the final state of a transducer. The second problem is the delay of the application of transducer weights. Weights in G are not applied to the composite transducer until there is an actual mapping between the output symbols and the input symbols of the component transducers. For pruning efficiency, it is beneficial to introduce G weights as early as possible before the actual mapping of symbols occurs, hence the motivation for the *incremental* approach.

The *look-ahead* approach proposed by Caseiro [6] addresses the above problems. He subdivides $\tilde{L}$ into two regions, a *prefix* region and a *suffix* region. The prefix region is the region between the initial state of $\tilde{L}$ and a non-ϵ output transition. In Figure 1, the prefix region is bounded by the grey rectangle. The region between the non-ϵ output transitions and the final state is the suffix region, which is bounded by the white rectangle. A set of *anticipated* output labels for each ϵ-output transitions is built inside the prefix region. The function of the anticipated label sets is to provide some look-ahead information. An ϵ-output transition in $\tilde{L}$ will be expanded in the composition only if there is a match between its anticipated label set and the input labels of G.

The early application of G weights before encountering the actual non-ϵ output labels in $\tilde{L}$ can also be done by finding the *semiring-sum* ($\oplus$) of the weights of the matched G transitions. In a *tropical* semiring, the $\oplus$ operator is *min*. Thus, it is very similar to language model look-ahead [8], where partial language model weights are applied to tokens before reaching the leaf nodes (word-end nodes) of a lexical tree.

4 Proposed Approach to Dynamic WFST Composition

We base our approach on that of Caseiro. Specifically, two component WFSTs are built: $(\tilde{C}_{opt} \circ \tilde{L}_{opt})$ and G, where $\tilde{C}_{opt}$ and $\tilde{L}_{opt}$ are $min(det(\tilde{C}))$ and $min(det(\tilde{L}))$ respectively. Component transducers are combined with *look-ahead*, avoiding the creation of "dead-end" transitions. Early application of G weights is also performed.

There are however two major differences between our approach and Caseiro's approach. In [6], he presented a specialized algorithm to compose $\tilde{L}$ and G. He made two assumptions (or constraints) about his approach. They are:

- $\tilde{L}$ is an acyclic graph, apart from the loop which connects the final state of $\tilde{L}$ to the initial state (Figure 1)

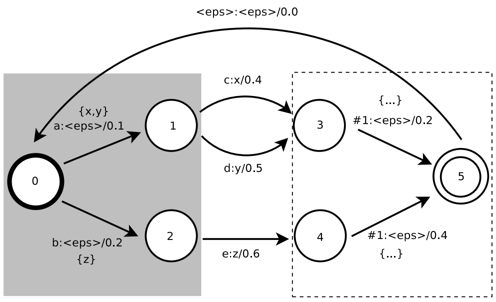

Figure 1: The lexicon WFST ($\tilde{L}$) in Caseiro's approach. {} indicates an anticipated output label set. {...} means the set containing all symbols. #1 is the word-end marker.

- No weight look-ahead is performed in the suffix region.

While the first assumption holds for a typical lexicon, it is not true for an arbitrary WFST. For example, the ($\tilde{C}_{opt} \circ \tilde{L}_{opt}$) WFST is cyclic in general. For the second assumption, no weight look-ahead is performed in the suffix region. However, in order to achieve better pruning efficiency, weights should be distributed or "pushed" to the initial state as far as possible. Hence, look-ahead of weights, as well as the avoidance of non-coaccessible transitions, should also be performed in the suffix region.

In the following subsections, we describe how the anticipated output label sets are found in the ($\tilde{C}_{opt} \circ \tilde{L}_{opt}$) transducer. We also describe how this transducer is dynamically composed with G during decoding.

4.1 Finding the Anticipated Output Labels

The entire ($\tilde{C}_{opt} \circ \tilde{L}_{opt}$) transducer is subdivided into *prefix* regions. Each prefix region is terminated with non-ϵ output label transitions. All the other transitions are ϵ-output transitions.

Figure 2 illustrates an example of a cyclic ($\tilde{C}_{opt} \circ \tilde{L}_{opt}$) transducer. For simplicity, only the output labels are shown. The transducer is segmented into three prefix regions. Each of them is ended with non-ϵ output label transitions. The anticipated output label set can be found by a simple depth-first traversal algorithm.

Since the transducer is subdivided into segments of prefix regions, word-end markers are now inside each region. When look-ahead is carried out within each region during dynamic composition, it implies that transducer weights of G could be pushed forward across the word-end markers, that is, across the word boundaries.

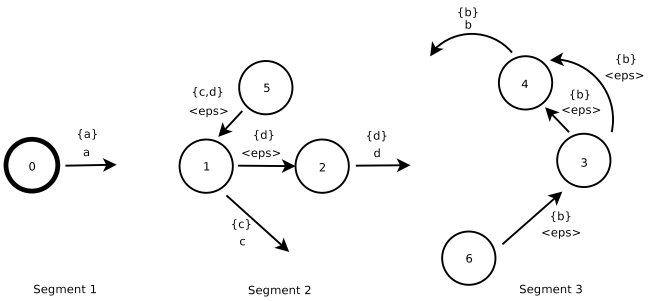

Figure 2: A ($\tilde{C}_{opt} \circ \tilde{L}_{opt}$) WFST is segmented into *prefix* regions, where label and weight look-ahead is performed.

4.2 The Dynamic Composition Algorithm

The dynamic composition algorithm follows a token passing paradigm [9]. Each token holds an alignment of hypothesized words together with the corresponding accumulated cost (*accCost*). In dynamic composition, tokens reside in the $(\tilde{C}_{opt} \circ \tilde{L}_{opt})$ transducer. Each token also has a reference to the G transducer. This reference is necessary for distinguishing two tokens when they arrive at the same transition in the $(\tilde{C}_{opt} \circ \tilde{L}_{opt})$ transducer, but have different word histories (i.e. different state number in the G transducer). Hence, two additional attributes are required for each token. They are S_G and *pushedCost*, where S_G is the state number of the G transducer to which the token is referencing and *pushedCost* is the accumulated look-ahead weight that has already been applied to the token.

Table 1 shows the pseudocode of the dynamic composition algorithm. The following points highlight the important parts of the algorithm.

Step 1 - 2 Update S_G and reset *pushedCost* if the token is leaving a prefix region and entering a new prefix region.

Step 5a Avoid tokens entering non-coaccessible transitions.

Step 5b - 5e Perform weight look-ahead.

Step 5f Organize tokens in lists. The UD list allows multiple tokens with different S_G on the same transition. The D list arranges tokens according to their $next\_S_G$ references. This enables early recombination of tokens with the same $next\_S_G$ but different current S_G. It simulates suffix sharing as in WFST minimization and it is similar to [10].

5 Experimental Results

The aim of this experiment is to compare the performance and the resource requirements of our dynamic composition algorithm with static composition and other dynamic composition approaches. The following list briefly describes the different approaches under test.

Static Perform decoding on the integrated $(opt(\tilde{C}_{opt} \circ \tilde{L}_{opt} \circ G))$.

Dynamic (Incremental, no look-ahead) Introduce unigram probabilities to build $(\tilde{C}_{opt} \circ \tilde{L}_{opt} \circ G_{uni})$. Dynamically compose this WFST with $G_{tri-uni}$, which is a trigram deviation from unigram, during decoding *without look-ahead* (i.e. no control on non-coaccessible paths and no weight look-ahead).

Dynamic (Caseiro) Build $(\tilde{C}_{opt} \circ \tilde{L}_{opt})$ and G WFSTs. Dynamically compose them during decoding. Since the topology of $(\tilde{C}_{opt} \circ \tilde{L}_{opt})$ is different from $\tilde{L}$ as in his approach, there is no direct comparison. To simulate his method, the control of non-coaccessible paths and weight look-ahead is prohibited until the token reaches a word-end marker inside a prefix region. This *no look-ahead* region can be considered as the *suffix* region as in his method. Look-ahead resumes after the token has passed the word-end marker.

Dynamic (Our approach) Build $(\tilde{C}_{opt} \circ \tilde{L}_{opt})$ and G WFSTs. Dynamically compose them as described in Section 4.

The performance of different approaches was assessed using the Wall Street Journal (WSJ1) corpus [11]. Cross-word triphone HMM models were trained on the "si_tr_s" set of 38275 utterances using 39-dimensional PLPs. A trigram LM, with 19979 unigrams, 3484372 bigrams and 2949590 trigrams, was used to test the 20k development test set "si_dt_20" from WSJ1 database, consisting of 503

1. A token resides on Transition $(i : o/w)$ between States $q1$ and $q2$ in $(\tilde{C}_{opt} \circ \tilde{L}_{opt})$. If o is non-ϵ, the token reaches the end of a prefix region. Go to Step 2. Otherwise, the token is still within the prefix region. Go to Step 3.
2. Set $S_G = next\_S_G$ (See Step 5f for details about $next\_S_G$). Reset $pushedCost = 0.0$.
3. Retrieve S_G from the token.
4. Get the set of transitions leaving from State $q2$.
5. For each transition t,
   (a) Get the anticipated label set of t. Also get a set of input labels from all the transitions leaving from State S_G of G. Find the intersection between these two sets. If there is no intersection, it means that t is a non-coaccessible transition, the token will not enter t. Go back to Step 5 for the next t. Otherwise, go to Step 5b.
   (b) Go through all the matched transitions at State S_G. Accumulate the *semiring-sum* ($\oplus$) of the weights of all the matched transitions. This is the *look-ahead weight*.
   (c) $\Delta cost = (pushedCost)^{-1} \otimes (\textit{look-ahead weight})$.
   (d) Update $accCost = (accCost) \otimes (\Delta cost)$.
   (e) Update $pushedCost = \textit{look-ahead weight}$.
   (f) Check the number of matched labels in the intersection.
      - > 1 match, put the token (indexed by S_G) in the UD (UnDecided) list of t. If there is a token with the same key, keep the lower $accCost$ token.
      - Only 1 match, the next S_G ($next\_S_G$) is the destination state of the matched G transition. Put the token in the D (Decided) list of t. The token is indexed by the pair $(next\_S_G, matched_O)$ where $matched_O$ is the matched symbol in the intersection. Keep the token with the lower $accCost$ if there is a token with the same key.

Table 1: Pseudocode of the proposed algorithm. States $q1$ and $q2$ are any arbitrary states in $(\tilde{C}_{opt} \circ \tilde{L}_{opt})$. The symbols $\oplus$ and $\otimes$ are *semiring-add* and *semiring-multiply* respectively.

utterances. The experiment was carried out using Juicer [12, 13], which is a WFST-based LVCSR decoder developed here at IDIAP.

Figure 3 shows the word error rate (WER) against the real-time factor (RTF) of different approaches. Amongst all dynamic composition approaches, the proposed method shows better WER versus RTF characteristics. One important observation is that the proposed method significantly outperforms the other two dynamic approaches at narrow and moderately-wide beam widths. At the level of 17% WER (moderately-wide beam), the RTF of our approach is about 65% and 48% of the *no look-ahead* approach and Caseiro's method respectively. This confirms that look-ahead is necessary for good accuracy-time tradeoff in narrow and moderately-wide beam width scenarios.

Comparing our approach with static composition, the WERs are similar at the same pruning settings, which suggests that our approach is close to the WFST optimization performed during static composition. At the same level of 17% WER, the RTF of the proposed approach is about 60% more than the RTF of static composition. This is due to the overhead, for example, finding the set intersection, searching tokens in a list, etc, required during dynamic composition. Figure 4 illustrates the RTF against the average number of tokens per frame. Our approach has a steeper slope in the figure, which indicates that it requires more time to process each token than the static case. Also it can be seen that the other two dynamic approaches have a lot more tokens per frame than both our approach and the static approach, which shows that the avoidance of non-coaccessible transitions in

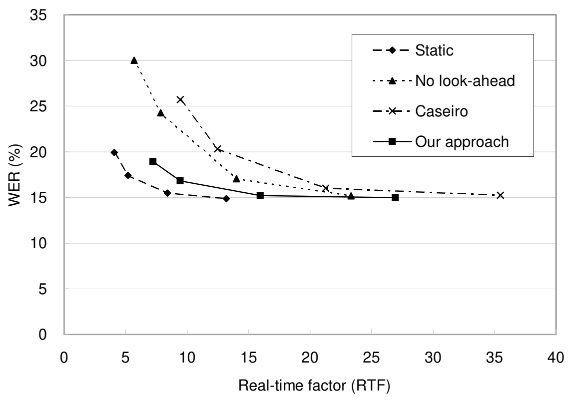

Figure 3: WER versus RTF of different approaches at various pruning beam-widths (150, 160, 180 and 200).

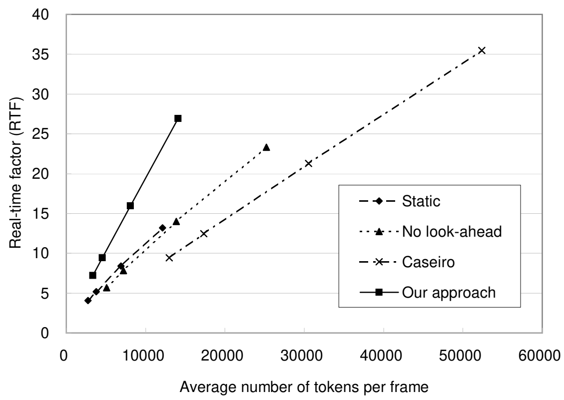

Figure 4: RTF versus Average number of tokens per frame of different approaches at various pruning beam-widths (150, 160, 180 and 200).

our approach helps to reduce the number of redundant tokens.

One of the major reasons to perform dynamic composition is the reduction in memory requirement. Table 2 compares the maximum memory usage (in MB) of our approach and the static approach. It shows a reduction of about 60% in memory usage.

6 Conclusions

We have proposed a generalized dynamic WFST composition algorithm, which avoids the creation of non-coaccessible transitions, performs weight look-ahead and does not impose any constraints to the topology of the WFSTs. Experimental results show that our weight look-ahead approach gives better WER versus RTF characteristics than other dynamic composition approaches. Comparing with static composition, it shows a significant reduction in memory usage.

Beam width	Static	Dynamic (Our approach)	% reduction
150	1774	679	61.7
160	1775	697	60.7
180	1965	722	63.3
200	1966	762	61.2

Table 2: Maximum memory usage (in MB) during decoding

7 Acknowledgements

This work was supported by the EU 6th FWP IST integrated project AMI and the Swiss National Science Foundation through the National Center of Competence in Research (NCCR) on "Interactive Multimodal Information Management (IM2)".

References

[1] M. Mohri, "Finite-state transducers in language and speech processing," *Computational Linguistics*, vol. 23, no. 2, pp. 269–311, 1997.

[2] M. Mohri, F. Pereira, and M. Riley, "Weighted finite state transducers in speech recognition," in *ISCA ITRW Automatic Speech Recognition: Challenges for the Millennium*, 2000.

[3] M. Mohri and M. Riley, "A weight pushing algorithm for large vocabulary speech recognition," in *Proc. Eurospeech*, 2001.

[4] H. Dolfing and I. Hetherington, "Incremental langauge models for speech recognition using finite-state transducers," in *Proc. ASRU*, 2001.

[5] D. Willett and S. Katagiri, "Recent advances in efficient decoding combining on-line transducer composition and smoothed language model incorporation," in *Proc. ICASSP*, 2002.

[6] D. Caseiro and I. Trancoso, "A specialized on-the-fly algorithm for lexicon and language model composition," *IEEE Tran. on Audio, Speech and Language Processing*, vol. 14, no. 4, 2006.

[7] T. Hori and A. Nakamura, "Generalized fast on-the-fly composition algorithm for WFST-based speech recognition," in *Proc. Interspeech*, 2005.

[8] S. Ortmanns, H. Ney, and A. Eiden, "Language-model look-ahead for large vocabulary speech recognition," in *Proc. ICSLP*, 1996.

[9] S. Young, N. Russell, and J. Thornton, "Token passing: A simple conceptual model for connected speech recognition systems," Tech. Rep. CUED/F-INFENG/TR38, Cambridge University Engineering Department, July 1989.

[10] D. Caseiro and I. Trancoso, "A tail-sharing WFST composition algorithm for large vocabulary speech recognition," in *Proc. ICASSP*, 2003.

[11] D. Paul and J. Baker, "The design for the Wall Street Journal-based CSR corpus," in *Proc. ICSLP*, 1992.

[12] D. Moore, "The Juicer LVCSR Decoder - user manual for Juicer version 0.5.0," IDIAP-COM 03, IDIAP, 2006.

[13] D. Moore, J. Dines, M. Magimai Doss, J. Vepa, O. Cheng, and T. Hain, "Juicer: A weighted finite-state transducer speech decoder," in *MLMI'06*, 2006.