interspeech 2008 incorporating SST 08

Implementation and Evaluation of Fast On-the-fly WFST Composition Algorithms

Tasuku Oonishi, Paul R. Dixon, Koji Iwano, Sadaoki Furui

Department of Computer Science, Tokyo Institute of Technology, Japan
{oonishi,dixonp,iwano,furui}@furui.cs.titech.ac.jp

Abstract

When using Weighted Finite State Transducers (WFSTs) in speech recognition, on-the-fly composition approaches have been proposed as a method of reducing memory consumption and increasing flexibility during decoding. We have recently implemented several fast on-the-fly techniques, namely *avoiding dead-end states*, *dynamic pushing* and *state sharing* in our decoding engine. The goal of this paper is to provide a unified study of how the different on-the-fly techniques and online composition combinations effect speech recognition performance. The evaluations were performed on a large spontaneous speech recognition task and the results show that when using on-the-fly composition with a fully dynamically composed language model component the performance degrades substantially even when avoiding dead-end states. We then show in these cases the recognition performance can be dramatically improved with the addition of dynamic pushing and state sharing.
Index Terms: LVCSR, WFST

1. Introduction

Within recent years it is evident that the Weighted Finite State Transducer (WFST) approach pioneered at AT&T [1] is becoming a very popular approach for building speech recognition systems. The elegant WFST framework allows for the search networks to be pre-compiled and heavily optimized often leading to much better recognition performance. The unified WFST representation brings many advantages when developing a high quality decoding engine, for example changes to any of the knowledge sources should not require extensive modifications of the decoder.

Currently at Tokyo Institute of Technology we are also developing a WFST based speech decoder with the aim of achieving state-of-the-performance and high flexibility [2].

When using the WFST framework for speech recognition the various knowledge sources are all first represented as an equivalent WFST, typically these will consist of:

- H maps HMM states to context-dependent phones.
- C represents a transduction from context-dependent phones to context-independent phones.
- L is lexicon converted to a WFST that will map context-independent phone sequences to words.
- G is a WFST that represents the language model, for example an N-gram model that maps words to N-gram weighted word sequences.

The composition operation (denoted by $\circ$) combines WFSTs together. To generate an integrated search network that will map from HMM states to language model weighted word sequences the multiple levels of information are combined together according to $H \circ C \circ L \circ G$. The optimization procedures are typically performed on the individual WFSTs as well as the final composed WFST to reduce computational resources both during composition and decoding.

A drawback of this WFST approach is access to original knowledge sources is lost once the final network has been composed and optimized. On-the-fly composition and optimization algorithms have been developed by others [3, 4, 5, 6] as a method of increasing flexibility within the WFST paradigm. However, one disadvantage with such on-line algorithms is some of the optimization powers available in the static equivalents are sacrificed.

In this paper we evaluate our fast on-the-fly implementations based on the schemes proposed by [4, 5, 6] that we have recently added to our decoding engine. When using on-the-fly composition there exists many different combinations of static and on-the-fly compositions and it is very useful to have knowledge of where the combination and optimization effects performance the most, today few studies have attempted to investigate this area. The results presented will illustrate the speed and accuracy of the decoder operating on a variety of transducer cascades.

The remainder of the paper is structured as follows: section 2 describes WFST evaluated in this paper, section 3 describes our fast on-the-fly composition implementations. Next in Section 4 we present experiment evaluations and then the paper is finished in section 5 with conclusions.

2. WFST combinations evaluated

In this paper we evaluate the different on-the-fly composition combinations as detailed below. With the exception of the fully static network, composition operations occurring within parenthesis indicate a static composition and composition operations occurring outside parenthesis are on-the-fly operations.

- $H \circ C \circ L \circ G$: Fully statically composed and optimized WFSTs.
- $(H \circ C) \circ (L \circ G)$: This combination was studied in [6], the lexicon and the language model are bound together and therefore can be changed dynamically as a unit but cannot be changed individually.
- $(H \circ C \circ L \circ G_{uni}) \circ G_{tri/uni}$: The WFST G_{uni} represents a uni-gram language model and WFST $G_{tri/uni}$ represents a tri-gram model divided by the uni-gram probability. This scheme as studied in [3] allows for the inclusion of some static language model information and the addition of this look-ahead information can improve the search performance.

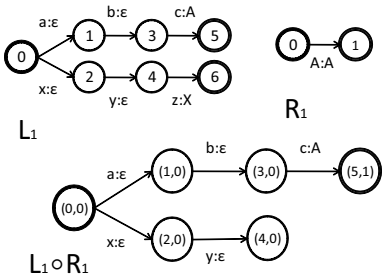

Figure 1: *Dead-end state generation when performing composition.*

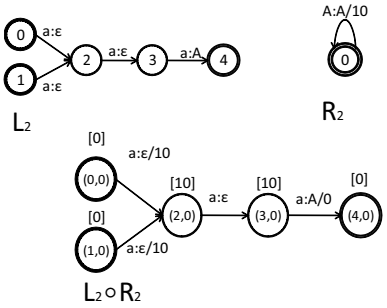

Figure 2: *Composed WFST with dynamic pushing.*

- $(H \circ C \circ L) \circ G$: The language model information can be changed dynamically and therefore it has the potential to be very flexible [4, 5].
- $(H \circ C) \circ L \circ G$: The lexicon and language model can be changed individually offering a great deal of potential flexibility, however, the two dynamic composition operations could incur a large overhead and generate sub-optimal search networks.

3. Fast on-the-fly composition

One of the biggest potential drawbacks when using on-the-fly transducer composition is the inability to fully optimize the final search graph used in the decoder. This will often lead to a degradation in performance and speed. On-the-fly composition algorithms have been proposed [4, 5, 6] that attempt to perform some type of on-line optimization thus giving both flexibility and incurring fewer performance penalties, examples of such algorithms are *avoiding dead-end states*, *dynamic pushing* and *state-sharing*.

3.1. Dead-end states

The composition of the transducers L_1 with R_1 to give the transducer $L_1 \circ R_1$ is shown in Figure 1. In $L_1 \circ R_1$ the path (0,0)-(2,0)-(4,0) will not reach a valid end state because it contains a non-coaccessible or *dead end* state (4,0). The dead-end state was produced because WFST L_1 did not have any output labels that matched an input label in R_1. A static composition operation would remove such a path from the final transducer, however, when performing search with on-the-fly composition, such states can be generated frequently and this can lead to a degradation in recognition performance and speed.

Caseiro [4] proposed an on-the-fly composition technique that could remove dead-end states and this technique was also extended by Cheng [5] and McDonough [6]. In this approach before composing WFSTs, L and R, a set of anticipated non ϵ output labels is pre-calculated for each of the states in L. During composition when the state pair (q_l, q_r) is generated, if the intersection of q_l anticipated output labels and q_r input labels is empty then the state does not produce a valid output transition and is culled. We have implemented a similar operation to remove dead-end states in our decoder.

3.2. Dynamic pushing

The weight pushing operations allow for the weights to be moved closer to the initial state [1]. If pushing is applied to search networks the decoder will encounter the weights earlier in search and this can lead to improvements in performance. The pushing operation is performed after static composition, therefore, when switching to on-the-fly composition it is no longer possible to perform the pushing and hence a decline of speech recognition performance is often observed.

Also proposed in Caseiro [4] is a pushing algorithm for use in on-the-fly composition known as *dynamic pushing* and this technique was also extended by Cheng [5] and McDonough [6].

In this algorithm a *look-ahead* score is stored at the composed states and when a transition is generated, the difference between destination and source look-ahead scores is added to the transition weight. For example the look-ahead score which is set at state (q_l, q_r) is the minimum value of a transition weight from q_r whose input label is included in an anticipated label set for q_l. Figure 2 shows an example of composition with dynamic pushing. The values inside the square brackets are the look-ahead scores stored inside the states. The figure shows how the weights are moved closer to the initial states.

The example in Figure 3 serves to illustrate a potential problem that can occur when using dynamic pushing. Two possible weightings for the composed transducer $L_3 \circ R_3$ are shown Figure 3 (1) and (2). The cause of the non-deterministic problem is the pair of ϵ and non-ϵ transitions in L_3 entering the common state 2.

In order to solve the problem, a *filter* WFST [1] (F) shown in Figure 4 is used in our dynamic pushing implementation. At the first step of the on-the-fly composition, the ϵ output symbols in WFST L_3 and ϵ input symbols in R_3 are substituted with "ϵ_2" and "ϵ_1", respectively and the resulting WFSTs are denoted by $L_3^{'}$ and $R_3^{'}$. The composition "$L_3^{'} \circ F \circ R_3^{'}$" is performed and in the process the pair of ϵ and non-ϵ transitions in L_3 entering state 2 becomes separated into two transitions with different destination states (2,2,0) and (2,0,0) in the final transducer (Figure 5). Since both transitions entering a common state (3,2,0) becomes ϵ transitions the weighting conflict is avoided.

3.3. State sharing

When the minimization [1] operation is applied to a deterministic WFST, it will produce an equivalent WFST that contains the minimum number of states. The practical effects of this operation are a potential reduction in decoder memory usage and speed-up due to the removal of redundant information.

To perform the minimization operation for a given state we need to know all the future states we can reach from it. However, during on-the-fly composition only neighboring state information is available and therefore it is not possible to run the minimization algorithm in an on-line manner. To simulate minimization operation, the state sharing algorithm in conjunction with the on-the-fly composition was proposed in [7]. In this

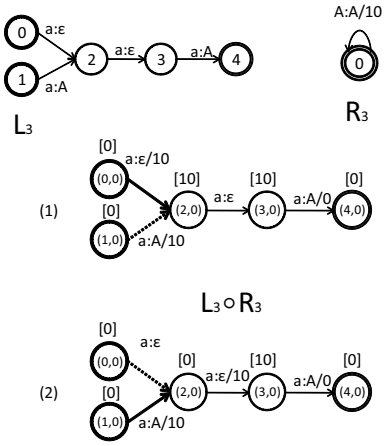

Figure 3: *Incorrect dynamic pushing of a WFST.*

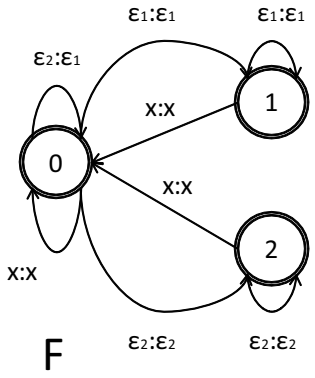

Figure 4: *Filter WFST.*

algorithm two states (ql, q_r) and $(q_l, q_r^{'})$ are shared when they enter a common destination state using arcs with equal input labels.

4. Experiments

4.1. Experimental setup

The speech waveforms were first converted to sequences of 38 dimensional feature vectors with 10 ms frame rate and 25 ms window size. Each feature vector was composed of 12 Mel-frequency cepstral coefficients (MFCCs) with delta and delta-deltas, augmented with delta and delta-delta energy terms.

The acoustic models were three state left-to-right HMM triphone models where each state output density was a 32 component Gaussian mixture model with diagonal covariance. EM training was performed utilizing the data from 967 lectures.

The language model was back-off tri-gram with a vocabulary of 55k words trained on 2,682 lectures of data.

The test set used for evaluations was composed of 2338 utterances which spanned 10 lectures. This yielded a total of 116 minutes of speech. The experiments were conducted on a 2.4 GHz Intel Core2 machines with 2GB of memory.

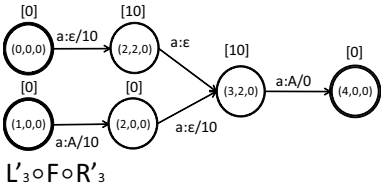

Figure 5: *WFST with dynamic pushing and filter.*

4.2. Experimental results

We considered how performing on-the-fly composition of different combinations would effect speed and accuracy by using the cascades listed in Table 1. The composition operator is denoted by ∘. With the exception of the fully composed network, any parts of a cascade occurring within parenthesis indicate an up-front static composition and in such cases determinization, pushing, minimization and factorization are performed in the manner described in [1]. When the composition operator occurs outside parenthesis, this means we are performing composition on-the-fly.

In Large Vocabulary Continuous Speech Recognition (LVCSR) tasks the WFST network often accounts for the largest part of the overall memory consumption during decoding. The parameter counts for each of the WFSTs cascades we considered are listed in Table 1. The table shows the total memory requirements of the $(H \circ C) \circ (L \circ G)$ cascade were greater than the $H \circ C \circ L \circ G$, which may be due to the additional optimizations applied after full composition. For consistency we factored the $(H \circ C)$ part of the $(H \circ C) \circ (L \circ G)$ cascades, and in accordance with our expectations no significant compression of this WFST was observed. A substantial reduction in the total number of WFST parameters was observed in the $(H \circ C \circ L \circ G_{uni}) \circ G_{tri/uni}$, $(H \circ C \circ L) \circ G$ and $(H \circ C) \circ L \circ G$ cascades, with all cases requiring approximately half of the parameters used in the $H \circ C \circ L \circ G$ network.

In Figure 6 we illustrate the performance of various WFST combinations. At 1xRTF, the performance of the $(H \circ C) \circ (L \circ G)$ and $(H \circ C \circ L \circ G_{uni}) \circ G_{tri/uni}$ networks is very close to the $H \circ C \circ L \circ G$ WFST. In the $(H \circ C \circ L) \circ G$ and $(H \circ C) \circ L \circ G$ configurations, a significant degradation of performance is observed, this is because the on-line composition does not perform any weight pushing on the language model WFSTs leading to inefficient pruning during decoding.

The $(H \circ C \circ L) \circ$ G and $(H \circ C) \circ L \circ G$ networks were utilized to consider the benefits of performing dynamic pushing and state sharing in addition to dead-end state avoidance. Figure 7 shows that the addition of these techniques can lead to a substantial improvement in performance, especially in the $(H \circ C \circ L) \circ G$ WFST which is now able to achieve a similar performance level as the $H \circ C \circ L \circ G$ network at the 1xRTF operating point. Even after the addition of dynamic pushing and state sharing, the performance of the $(H \circ C) \circ L \circ G$ network did not converge adequately, which is partly because of the overhead and inefficiencies present in the on-line composition of the network. Finding a solution to this problem is an issue we are currently investigating.

5. Conclusions and future work

Is this paper we have presented a combination of fast on-the-fly composition algorithms and evaluated the performance using

Table 1: *Size of WFSTs used in the evaluations.*

	1st WFST		2nd WFST		3rd WFST		Total	
	#state	#transition	#state	#transition	#state	#transition	#state	#transition
$H \circ C \circ L \circ G$	2,643,523	5,665,967					2,643,523	5,665,967
$(H \circ C) \circ (L \circ G)$	1,500	6,080	3,118,382	5,918,451			3,119,882	5,924,531
$(H \circ C \circ L \circ G_{uni}) \circ G_{tri/uni}$	38,000	112,907	752,185	3,784,320			790,185	3,897,227
$(H \circ C \circ L) \circ G$	38,000	112,907	752,185	3,784,320			790,185	3,897,227
$(H \circ C) \circ L \circ G$	1,500	6,080	39,666	95,057	752,185	3,784,320	793,351	3,885,457

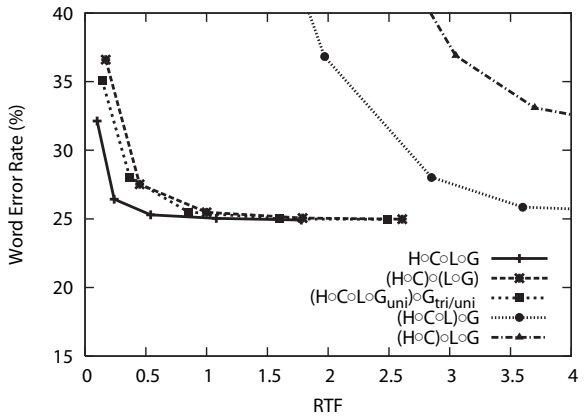

Figure 6: *Performance of various recognition cascades when performing on-the-fly composition with dead-end states avoidance.*

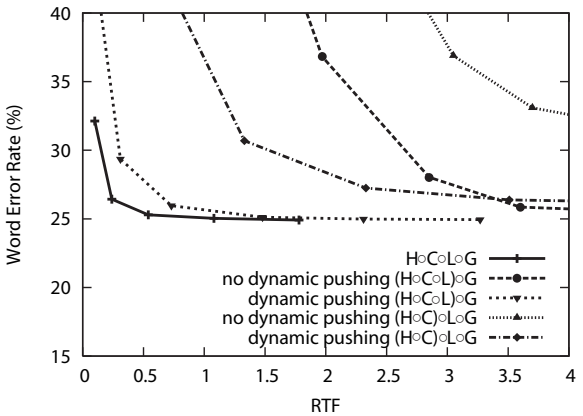

Figure 7: *The Effect on the performance after the addition of dynamic pushing and state sharing.*

various WFST combinations.

Our experimental results showed that by utilizing dead-end state removal in a recognition cascade that contains some form of static language model information a performance level nearly the same as the fully statically composed and optimized search network can be achieved.

Cascades which only contain a dynamically composed language model did not perform as well. However, we demonstrated that the performance of such combinations can be dramatically improved by the addition of dynamic pushing and state sharing operations in combination with dead-end state avoidance.

In future work we will investigate efficient techniques for performing on-the-fly composition of several layers of information. Hori [8] also proposed a fast on-the-fly composition algorithm and demonstrated high performance with low composition overhead. In our future work we would like to evaluate the possibility of combination of both the Hori [8] and Caseiro [4] approaches. Another area of future work is to evaluate the current on-the-fly implementation on various corpora and different types of models. For example a class base language model WFST or the interpolation of several higher order N-gram language models will be investigated.

6. Acknowledgments

This work was supported by the METI Project "Development of Fundamental Speech Recognition Technology".

7. References

[1] M. Mohri, F. Pereira, and M. Riley., "Weighted finite-state transducers in speech recognition," *Computer Speech and Language*, vol. 16, no. 1, pp. 69–88, 2002.

[2] P. R. Dixon, D. A. Caseiro, T. Oonishi, and S. Furui, "The TITECH large vocabulary WFST speech recognition system," *Proc. IEEE Workshop on ASRU*, pp. 443–448, 2007.

[3] H. J. Dolfing and I. Hetherington, "Incremental language models for speech recognition using finite-state transducers," *Proc. IEEE Workshop on ASRU*, pp. 194–197, 2001.

[4] D. A. Caseiro and I. Trancoso, "A specialized on-the-fly algorithm for lexicon and language model composition," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 14, no. 4, pp. 1281–1291, 2006.

[5] O. Cheng, J. Dines, and M. M. Doss, "A generalized dynamic composition algorithm of weighted finite state transducers for large vocabulary speech recognition," *Proc. ICASSP*, pp. 345–348, 2007.

[6] J. McDonough, E. Stoimenov, and D. Klakow, "An algorithm for fast composition of weighted finite-state transducers," *Proc. ASRU*, pp. 461–466, 2007.

[7] D. A. Caseiro and I. Trancoso, "A tail-sharing wfst composition algorithm for large vocabulary speech recognition," *Proc. ICASSP*, pp. 356–359, 2003.

[8] T. Hori and A. Nakamura, "Generalized fast on-the-fly composition algorithm for WFST-based speech recognition," *Proc. Interspeech*, pp. 847–805, 2005.