A COMPARISON OF DYNAMIC WFST DECODING APPROACHES

Paul R. Dixon   Chiori Hori   Hideki Kashioka

National Institute of Information and Communications Technology
Kyoto, Japan

ABSTRACT

In this paper we perform a comparison of lookahead composition and on-the-fly hypothesis rescoring using a common decoder. The results on a large vocabulary speech recognition task illustrate the differences in the behaviour of these algorithms in terms of error rate, real time factor, memory usage and internal statistics of the decoder. The evaluations were performed when the decoder was operated at either the state or arc level. The results show the dynamic approaches also work well at the state level even though there is greater dynamic construction cost.

***Index Terms*—** Speech recognition, WFST, on-the-fly composition

1. INTRODUCTION

1.1. WFSTs for Speech Recognition

In recent years the Weighted Finite State Transducer (WFST) based approach to speech recognition search space construction and decoding [1] has become very popular. One of the advantages of the WFST framework is the formal manner each of the knowledge sources can be represented in a consistent fashion. These knowledge sources can then be composed together and optimized for speed and size ahead of decoding.

Often the recognition cascade is constructed from the following components; the Language Model G which represents the recognition grammar, the lexicon L which is built from the pronunciation dictionary and maps from phoneme sequences to words, a transducer C that converts context-dependent phonemes to context-independent phonemes, and optionally the acoustic models H. In the experiments described in this paper, the decoder searches a cascade that is constructed according to:

$$\pi(C \circ det(L \circ G))$$

where det is the determinization operation, $\circ$ is the composition operation and the π operator is a procedure that removes auxiliary symbols. In section 2.3 the cascade is extended by composing in the acoustic models H.

The static search network approach allows for the construction of very fast decoders. However, many modern speech recognition tasks require extremely large vocabularies and language models (LMs). Even though large memory machines and advances in composition [2] can make the static construction possible in most cases. There is still the problem that the compiled search networks may exceed the memory limitations of the deployment machines. Moreover, static composition of large class models is still infeasible and many applications need to change or adapt the components online. This cannot be done efficiently when using large pre-compiled networks.

The dynamic composition approach builds the search space online during decoding. The most common approach is to decouple the language model from the other components. That is, to perform an offline composition of the lexicon and context-dependency transducers such as $C \circ \det(L)$ (and possible the acoustic models H). Then in the recognition phase perform an additional on-demand composition of the $(C \circ \det(L))$ machine with language model G during the decoding.

However, attempting to compose $C \circ \det(L)$ with G will lead to a substantially amount of dead-end paths. The problem is the determinized CL will have many null output transitions that occur before an output symbol. The composition algorithm must follow each of these paths looking for a word symbol to match against the input symbols of the current state in the LM. Every lexicon path that is expanded and not matched will be costly in terms both of memory and CPU time. To circumvent this problem there has been much research in different constructions and specialized composition approaches [3, 4, 5, 6, 7, 8, 9, 10, 2, 11].

The goals of this work were to compare lookahead composition [2] with on-the-fly hypothesis rescoring [5] using a common WFST decoder. In particular we wanted to determine the strengths of each approach, gain more insight in the manner the algorithms behave, and to investigate the scope for combination of the methods. This is different to much of the previous work where comparison are normally made between static WFST and traditional decoders.

Among the work focusing on decoder architectures comparisons most recently, Rybach et al [12] compared a dynamic search decoder with dynamic WFST decoder. In [13] a comparison was made between tree search decoder and a static WFST based decoder. The authors of [14] implemented a very fast dynamic decoder and provided detailed comparisons with a state level WFST decoder. In [15] there was a comparison of two tree search decoders and two different WFST based decoders. The papers where dynamic WFST algorithms have been proposed usually contain comparisons with a static baseline [3, 4, 5, 6, 7, 8, 9, 10, 2, 11].

1.2. On-the-fly Composition Approaches

The starting points of WFSTs in speech recognition can be found in the early publications from the researchers at AT&T[16, 17]. Some of the these early publications briefly mentions on-the-fly composition of the language model and or context-dependency transducers, but give very few details or experimental results[18, 19, 16].

The first published works with a focus on on-the-fly composition were [3] and [20]. This was later followed by [21] which features a similar method to [20]. Caseiro's [3] approach was to specialize the composition algorithm for the acyclic nature of the lexicon transducer. In [20] and [21] the common theme was to factor the language model into two components and apply a standard composition algorithm.

1.3. On-the-fly Rescoring

The on-the-fly rescoring approach proposed by Hori et al [5, 6, 8] factors the language model into two components G_1 and G_{3-1} . Here the decoder performs a standard beam search on CLG_1 and generates a recognition lattice (possibly virtually). In this method though, instead of waiting for the recognition to complete the rescoring is performed during the first-pass. As the lattice is constructed a set of *rescoring* tokens or *co-hypotheses* are maintained that are associated with a token in the active search graph of CLG_1 and a state in the second language model G_{3-1}. When lattice arcs are generated from the first transducer they are rescored using the full LM and the updated scores are to readjust the token scores in the first transducer. The advantages of this approach are no composition dead-end paths are generated and the method has the potential to decode faster than a fully composed search network[8]. Later in [6] an extended approach was described that allowed for the dynamic cascading of multiple transducers.

The rescoring implementation we used in the experiments in this paper is slightly different to the original scheme proposed in Hori[6]. Our implementation is somewhat similar to the scheme described in [11]. In our rescoring tokens we keep track of both the accumulated acoustic and total costs. By tracking the acoustic contribution this removes the requirement to factor the LM and instead we can use G_3 directly for rescoring.

1.4. Lookahead Composition

Caseiro [3, 4, 7] described a composition algorithm tailored to the structure of the lexicon transducer that avoids the generation of dead-end states. The algorithm also applied dynamic weight pushing to improve recognition speed and dynamic label pushing that helps reduce the size of the final search space.

Similar approaches were extended by others[22, 23, 9] to allow the composition to operate on more general topologies of transducers. In [10] the author proposed the use of a composition filter that interacts with a more general composition algorithm. The Lookahead composition proposed by [2] uses a very flexible generalized filter mechanism.

A further advantage of the generalized composition approach is it allows for the $C \circ \det(L) \circ G$ to be constructed directly and this greatly reduces the amount of memory required for constructing static networks.

2. EXPERIMENTS

We performed experiments on the Corpus of Spontaneous Japanese (CSJ) using the training and testing protocols described in [24]. The test set consisted of a total of 116 minutes of speech which spanned 10 lectures. Each of the lectures was segmented to give a total of 2338 test utterances.

The training speech was converted to sequences of 39 dimensional feature vectors with 10 ms frame rate and 25 ms window size. Each featur e vector was composed of 12 MFCCs with deltas and delta-deltas, augmented with log energy, log delta and log delta-delta energy terms. The acoustic models were comprised of 8000 left-to-right HMM tri-phone models each with three states. In total there were 3000 tied states each with 32 Gaussian densities . The Kneser-Ney smoothed trigram language model (G) was constructed using the MITLM toolkit [25]. The final language model contained a 90k word vocabulary and approximately 2M n-grams.

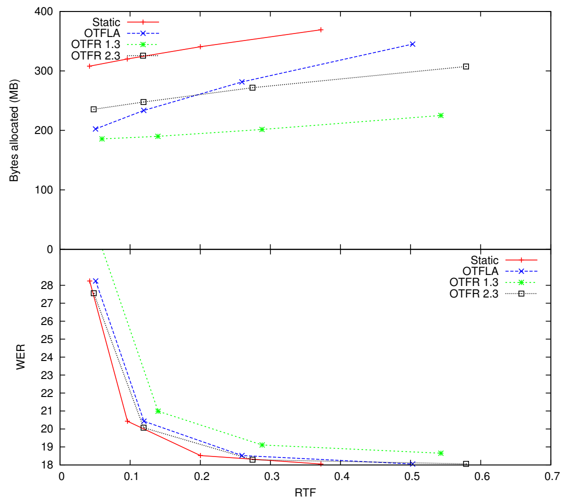

Fig. 1. Comparison of different dynamic composition approaches on the CSJ task

The experiments were conducted on an Intel Core i7 machine with 12GB of memory. The operating system was Ubuntu 10.04 (64bit) and the decoder was compiled for 32bit using g++ 4.4.

We used our in-house decoder called *SprinTra*, the architecture is very similar to the scheme described in [26]. We make use of the OpenFst libary [27] for the WFST representations, IO, and fundamental algorithms.

In the following section we compare the following composition schemes: Statically composed cascade (Static), on-the-fly composed using lookahead composition (OTFLA), on-the-fly rescored unigram/trigram combination (OTFR 1.3) and on-the-fly rescored bigram/trigram combination (OTFR 2.3).

2.1. Accuracy, Speed and Memory Performance

The results in Figure 1 show the performance of the difference approaches. The Real Time Factor (RTF) is computed by dividing the wall time by audio length. The results show as expected the static search network performing best. From the dynamic approaches we notice nearly identical performance when comparing the OTFLA approach and OTRF bigram approach. For comparison, a one-pass bigram language model gave a lower absolute accuracy by around 2% and worse RTF characteristics.

We found the performance for the OTRF with unigram lookahead did not perform as well as expected. The best error rate was 0.6 worse than the asymptotic best error rates of the other systems. This could be slightly improved by increasing the number of co-hypotheses at a greater RTF cost. To verify the implementation we also tried a standard two-pass rescoring approach and observed the same asymptotic accuracy. These results may be due to differences in our models, decoder or construction process. A similar drop in accuracy was reported in [20] when using a standard composition algorithm with a unigram/trigram factorization. The transducers were optimized by determinization and pushing in (or equivalent to) the log semiring for all the cascades. We found minimization or other optimizations did not make any significant changes to the operating curves. We did observe the unigram factorization performance converged faster when we estimated a separate lower order n-gram rather than re-using the unigram probabilities from a trigram model.

In the speed experiments the default OpenFst garbage collection was disabled and the decoder and `ComposeFst` along with its composition cache were completely destroyed and re-instantiated after processing 32 utterance.

The memory measurements were performed under the following conditions. All of the transducers where converted to the OpenFst `ConstFst` class, this holds all the arcs for all the states in a single contiguous array. The `ConstFst` is more memory efficient then the default OpenFst `VectorFst` type. The actual memory usage was determined by hooking into the global memory allocation functions and tracking the number of bytes requested by the decoder. The memory usage was recorded before resetting the caches and this value was averaged over the entire testset.

The upper part of Figure 1 shows the memory consumption at the different RTFs. The reason the rescored implementations achieve low memory usage is because we found that the algorithm could perform well without any caching. The rescoring transducer was simply input sorted and a binary search was used to find matching arcs as needed.

The lookahead composition `ComposeFst` uses an underlying cache that is similar to the OpenFst `VectorFst` type which increases the memory consumption (To fully expand the `ComposeFst` alone required around 1G).

For a given beam the static and OTFLA configurations alwas achieved indentical accuray. The OTFR2.3 was nearly identical to the static network, and larger difference were observed for the OTFR1.3 configuration.

2.2. Search Analysis

Figure 2 shows the average number of Gaussian mixtures and active arcs per frame at the various beam widths. We focused on these characteristics as there are often the most computationally expensive parts of the decoding process. The results show in both of the OTFR cases fewer active arcs are required and this verifies the claims in [8]. It may be possible to obtain further speed-ups with more aggressive band pruning. Better caching strategies may give performance curves more similar to those described in [8] at the cost of increased memory usage.

The results show for a given beam the OTFR setups require more Gaussian mixture evaluations per frame. One reason is because the pruning of active arcs with the full LM score can only be performed when a token leaves an arc. In the case of the static network or lookahead composition the full LM score can be applied immediately once an arc is activated. In the OTRF bigram factorization the increase in acoustic scoring costs and the reduction in active arcs combined with a increase in accuracy for a given beam balances out to give nearly identical speed characteristics to the lookahead composition system.

2.3. State Level Decoding

In the next set of experiments we explore how the dynamic approaches operate when adding the acoustic model(H) transducer into the static search network. Figure 3 shows that operating at the state level gives very similar static vs dynamic performance characteristics. In the state level setup approximately 50% more requests are made to the main search WFST or to the lattice construction. We expected the increase in these calls to causes larger slow downs in the dynamic approaches. However, we observed the delta between the static and dynamic approaches to remained largely the same except for the wider beam settings.

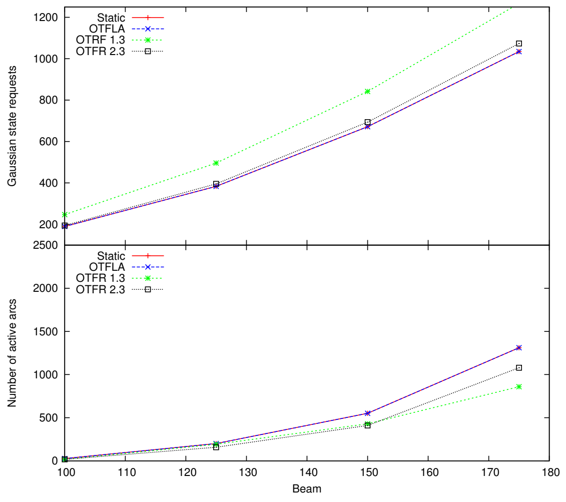

Fig. 2. State computations and number of active arcs for a given beam width when decoding at the arc level.

When comparing equivalent state and arc level systems, the state level requires around 20% less Gaussians to be computed per frame. There is a 30% increase in the average number of active arcs per frame for the static and OTFLA approaches and a 20% increase for the OTFR approaches. Overall the re-distribution in computation cancels out to very similar WER vs RTF when comparing an arc or state level network. This result shows that either of the composition approaches are suitable for state level decoders. However, the larger percentage of time spent by the Gaussian computations in the arc level decoder suggests that the arc level decoding can benefit more from Gaussian speed-up schemes and overall the arc level could be much more memory efficient.

3. CONCLUSIONS

In this paper we have presented an experimental comparison of lookahead composition and on-the-fly hypothesis rescoring.

The experiments have shown both algorithms display different run-time characteristics. On-the-fly rescoring explores a smaller search space in the first transducers but requires more Gaussian state evaluations. Lookahead composition is the opposite, requiring a larger search space but fewer unique Gaussian state evaluations.

We found in our experiments that on-the-fly rescoring could achieve good performance when there was a bigram model in the static search network component. Overall we found lookahead composition and on-the-fly rescoring (bigram) performed approximately the same in terms of speed and accuracy. We have shown both algorithms perform well when operating at either the arc or state level. In both of these cases the slow down in comparison to a static baseline is similar for both methods.

Areas for future work are to investigate how the algorithms perform on different tasks and other models. In particular more comparison should be performed with larger language models to clearly show the benefits of using on-the-fly techniques to reduce memory consumption during decoding. In this task the language model was too small to fully appreciate the memory saving gains from using delayed composition. It would be especially interesting to see if the unigram rescored system can be made to perform as fast as in [5, 6, 8] as this approach required the least memory of all the systems.

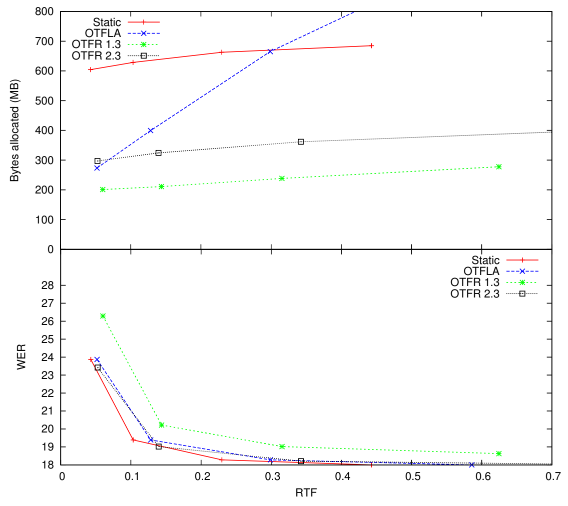

Fig. 3. Comparison of different dynamic composition approaches when operating on the state level.

4. ACKNOWLEDGMENTS

Thanks to Takaaki Hori and Michael Riley for useful discussions.

5. REFERENCES

[1] M. Mohri, F. C. N. Pereira, and M. Riley, "Speech recognition with weighted finite-state transducers," *Springer Handbook of Speech Processing*, pp. 1–31, 2008.

[2] C. Allauzen, M. Riley, and J. Schalkwyk, "A generalized composition algorithm for weighted finite-state transducers," in *Proc. Interspeech*, 2000, pp. 1203–1206.

[3] D. Caseiro and I. Trancoso, "Transducer composition for on-the-fly lexicon and language model integration," in *Proc. ASRU*, 2001, pp. 393–396.

[4] D. Caseiro and I. Trancoso, "Using dynamic WFST composition for recognizing broadcast news," in *Proc. ICSLP*, 2002, pp. 1301–1304.

[5] T. Hori, C. Hori, and Y. Minami, "Fast on-the-fly composition for weighted finite-state transducers in 1.8 million-word vocabulary continuous speech recognition," in *Proc. Interspeech*, 2004, pp. 289–292.

[6] T. Hori and A. Nakamura, "Generalized fast on-the-fly composition algorithm for WFST-based speech recognition," in *Proc. Interspeech*, 2005, pp. 847–850.

[7] 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.

[8] 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," *IEEE Transactions on Audio, Speech and Language Processing*, vol. 15, pp. 1352–1365, 2007.

[9] T. Oonishi, P. R. Dixon, K. Iwano, and S. Furui, "Implementation and evaluation of fast on-the-fly WFST composition algorithms," in *Proc. Interspeech*, 2008, pp. 2110–2113.

[10] T. Oonishi, P. R. Dixon, K. Iwano, and S. Furui, "Generalization of specialized on-the-fly composition," in *Proc. ICASSP*, 2009, pp. 4317–4320.

[11] H. Sak, M. Saraclar, and T. Gungor, "On-the-fly lattice rescoring for real-time automatic speech recognition," in *Proc. Interspeech*, 2010, pp. 2450–2453.

[12] D. Rybach, R. Schluter, and H. Ney, "A comparative analysis of dynamic network decoding," in *Proc. ICASPP*, 2011, pp. 5184–5187.

[13] S. Kanthak, H. Ney, M. Riley, and M. Mohri, "A comparison of two LVR search optimization techniques," in *Proc. ICSLP*, 2002, pp. 1309–1312.

[14] H. Soltau and G. Saon, "Dynamic network decoding revisited," in *Proc. ASRU*, 2009, pp. 276–281.

[15] J. R. Novak, P. R. Dixon, and S. Furui, "An empirical comparison of the t3, juicer, hdecode and sphinx3 decoders," in *Proc. Interspeech*, 2010, pp. 1890–1893.

[16] M. Riley, F. Pereira, and M. Mohri, "Transducer composition for context-dependent network expansion," in *Proc. Eurospeech*, 1997, pp. 1427–1430.

[17] M. Mohri, M. Riley, D. Hindle, A. Ljolje, and F. Pereira, "Full expansion of context-dependent networks in large vocabulary speech recognition," in *Proc. ICASSP*, 1998, pp. 393–396.

[18] M. Riley, A. Ljolje, D. Hindle, and F. Pereira, "The ATT 60,000 word speech-to-text system," in *Proc. ESCA*, 1995, pp. 207–210.

[19] F. Pereira and M. Riley, "Speech recognition by composition of weighted finite automata," Tech. Rep., AT&T Research, 1996.

[20] H. J. G. A. Dolfing and I. L. Hetherington, "Incremental language models for speech recognition using finite-state transducers," in *Proc ASRU*, 2001, pp. 194–197.

[21] D. Willett and S. Katagiri, "Recent advances in efficient decoding combining on-line transducer composition and smoothed language model incorporation," in *Proc. ICASSP*, 2002, pp. 713–716.

[22] O. Cheng, J. Dines, and M. M. Doss, "A generalized dynamic composition algorithm of weighted finite state transducers for large vocabulary speech recognition," in *Proc ICASSP*, 2007, pp. 348–351.

[23] J. McDonough, E. Stoimenov, and D. Klakow, "An algorithm for fast composition of weighted finite-state transducers," in *Proc. ASRU*, 2007, pp. 1–4.

[24] K. Maekawa, "Corpus of spontaneous Japanese: Its design and evaluation," in *Proc. ISCA and IEEE Workshop on Spontaneous Speech Processing and Recognition*, 2003, pp. 7–12.

[25] H. Bo-June and J. Glass, "Iterative language model estimation: Efficient data structure & algorithms," in *Proc. Interspeech*, 2008, pp. 841–844.

[26] P. R. Dixon, D. A. Caseiro, T. Oonishi, and S. Furui, "The Titech large vocabulary WFST speech recognition system," in *Proc. ASRU*, 2007, pp. 1301–1304.

[27] C. Allauzen, M. Riley, J. Schalkwyk, W. Skut, and M. Mohri, "OpenFst: A general and efficient weighted finite-state transducer library," in *Proc. of CIAA 2007*, 2007, pp. 11–23.