Generalized Fast On-the-fly Composition Algorithm for WFST-Based Speech Recognition

Takaaki Hori, Atsushi Nakamura

NTT Communication Science Laboratories, NTT Corporation
2-4 Hikaridai, Seika-cho, Soraku-gun, Kyoto, Japan
{hori,ats}@cslab.kecl.ntt.co.jp

Abstract

This paper describes a Generalized Fast On-the-fly Composition (GFOC) algorithm for Weighted Finite-State Transducers (WFSTs) in speech recognition. We already proposed the original version of GFOC, which yields fast and memory-efficient decoding using two WFSTs. GFOC enables fast on-the-fly composition of three or more WFSTs during decoding. In many cases, it is actually difficult or impossible to organize an entire transduction process of speech recognition using only one or two WFST(s) since some types of models considerably enlarge after written in WFST form. For example, a class n-gram model often results in a large WFST which is several times larger than a word n-gram model for the same vocabulary. GFOC makes it possible to use such a model after decomposing it into small multiple WFSTs. In a spontaneous speech transcription task, we evaluated the size of WFSTs, decoding speed, and word accuracy of several decoding approaches. The results show that GFOC with three or more WFSTs is an efficient algorithm when using a class-based language model.

1. Introduction

Recently, the Weighted Finite-State Transducer (WFST) approach has become a promising alternative formulation to traditional decoding approaches in speech recognition. It offers a unified framework representing various knowledge sources and produces a full search network optimized up to the HMM states [1]. The optimization step minimizes search space and accelerates decoding. However, when using a very large vocabulary lexicon, a detailed language model, or complicated transduction model, an enormous huge transducer is usually generated by composing all the components. Accordingly, both a high computational cost and large memory usage are necessary for decoding even if the WFST is optimized. Especially for memory usage, this problem is serious because states and transitions will be made for all possible combinations of the input-output relation between each adjacent pair of the cascaded WFSTs. Thus, the size of such a composite transducer easily exceeds the limitations of standard personal computers. To solve this problem, we proposed a Fast On-the-fly Composition algorithm [4] which is a fast version of the standard on-the-fly composition [2][3].

On-the-fly composition is a practical alternative to using a fully composed transducer. The set of WFSTs are separated into two groups, and in each group, one WFST is composed and optimized before decoding. Composition between the two WFSTs is performed as necessary during decoding. A lot of memory is saved by using on-the-fly composition, but search efficiency was decreased due to composition overhead. The Fast On-the-fly Composition (FOC) algorithm accelerates decoding by performing a one-pass Viterbi search based only on the first transducer of the two. The second transducer is only used to rescore the hypotheses generated during the search with minimal overhead. In our evaluation experiments, FOC achieved real-time 1.8 million-word vocabulary continuous speech recognition.

However, there are actually many cases when it is difficult or impossible to organize an entire transduction process of speech recognition using only one or two WFST(s) since some types of models considerably enlarge after written in WFST form. For example, a class-based language model, a combined language model, or a long-span language model often results in a much larger transducer than a simple word n-gram model for the same vocabulary. For this problem, on-the-fly composition of multiple WFSTs are efficient since such a huge model can be used after decomposing it into small WFSTs.

In this paper, we propose an extension of the FOC algorithm: *the Generalized Fast On-the-fly Composition (GFOC) algorithm*. GFOC enables efficient on-the-fly composition of three or more WFSTs during decoding while only two WFSTs can be composed by the original algorithm. Thus GFOC simplifies incorporating different language models and complicated constraints in the framework of WFST since it can deal with decomposed multiple WFSTs.

We conducted experiments in a benchmark test of Corpus of Spontaneous Japanese [6] (CSJ). In addition to a word n-gram model, we introduced a class n-gram model and a word-class combined n-gram model. We evaluated the performance of a simple search method using a single transducer, the original FOC, and GFOC using different language models.

2. Weighted Finite-State Transducers in Speech Recognition

WFST is a finite state network associating input and output symbols on each arc that can be weighted with a log probability value. Speech recognition is a transduction process from speech input to the corresponding word sequence. The process can be represented as a cascade of several transductions, each of which can be written in WFST form. Those WFSTs can then be combined by using a composition operator, leading to the integration of the underlying knowledge sources into a single input-output relation. An integrated WFST for speech recognition can be composed as

$$H \circ C \circ L \circ G, \tag{1}$$

where H, C, L, and G are WFSTs for a state network of triphone HMMs, a set of connection rules for triphones, a pronunciation lexicon, and a trigram language model, respectively; "$\circ$" represents the composition operator. As a result, decoding is a one-pass search problem in a single integrated network including cross-word triphones and a trigram language model. Once the network is further optimized by proceeding to weighted de-

terminization and minimization, search efficiency dramatically increases.

When a single WFST is composed of all knowledge sources for LVCSR, the number of states and transitions often becomes so large that an enormous amount of memory is required during decoding. To avoid this problem, on-the-fly composition is available. Generally, in on-the-fly composition, two WFSTs are prepared and composed during decoding.

In [3], the WFSTs are divided into two groups:

$$(H \circ C \circ L \circ G_1) \diamond G_{3/1}, \tag{2}$$

where we have introduced an on-the-fly composition operator "$\diamond$" to distinguish it from the general composition operator "$\circ$"; G_1 and $G_{3/1}$ are WFSTs for a unigram model and for a trigram model adjusted by dividing each trigram probability by the unigram probability of G_1. By including G_1 in the first group, search efficiency improves since G_1 yields prospection to each hypothesis during the search.

By using on-the-fly composition, memory usage is usually greatly saved since only a part of the entire composite search space is organized as necessary during the search. However, the entire space itself is larger than the full composition method since it is difficult to optimize the search space during decoding. Furthermore, on-the-fly composition overhead also increases the amount of computation required for decoding.

3. GFOC for Multiple WFSTs

First we show the concept of the original FOC. We assume that the decoder searches for the minimal cost path, where *cost* means an accumulated weight defined under *Tropical semiring*. Thus the weight of each transition is based on minus log probabilities.

Suppose two transducers A and B can be composed. In standard on-the-fly composition, given an input symbol sequence X, the decoder finds $\hat{Z}$ such that

$$W(X \rightarrow \hat{Z}) = \min_{Y,Z} \{W_A(X \rightarrow Y) + W_B(Y \rightarrow Z)\}, \tag{3}$$

where $W_A(X \rightarrow Y)$ is the accumulated weight when A translates the symbol sequence X to Y; $W_B(Y \rightarrow Z)$ is also the weight for $Y \rightarrow Z$ by B. The recognition result is the output symbol sequence $\hat{Z}$ that derives $W(X \rightarrow \hat{Z})$.

Equation (3) can be rewritten as

$$W(X \rightarrow \hat{Y}) = \min_{Y}\{W_A(X \rightarrow Y) + \min_{Z} W_B(Y \rightarrow Z)\}. \tag{4}$$

This means that the algorithm for finding $\hat{Y}$ can be applied to obtain $W(X \rightarrow \hat{Z})$, which is equal to $W(X \rightarrow \hat{Y})$, where $\min_Z W_B(Y \rightarrow Z)$ can be assumed to be the compensation weight.

A one-pass Viterbi search is performed based only on A but not based on $A \circ B$. B is only used to compensate accumulated weights of hypotheses generated by A with minimal overhead. Hence FOC is fast and memory-efficient compared to standard on-the-fly composition.

In frame-synchronous processing, hypotheses are generated by A, each of which represents an individual state transition process in A. If a new hypothesis h is generated by adding a new transition e to an existing hypothesis, h can be compensated with

$$\min_{f} W_B(o[h] \rightarrow o[f]),$$

by B only when e has a non-epsilon output, where f indicates a hypothesis generated by B accepting $o[h]$ that denotes the output symbol sequence of h. $o[f]$ denotes the output symbol sequences of f as well. By associating each hypothesis h with a list of hypotheses $g[h]$ produced by B, the compensation process can be efficiently performed. Here, $g[h]$ means a set of hypotheses generated by B when $o[h]$ is given as an input symbol sequence for B. We call the hypotheses in $g[h]$ "*cohypotheses*" to distinguish them from the hypotheses produced by A.

```
1   h = NewHyp(φ)
2   α(h) = λ
3   Insert(H, h)
4   while the next input exists do
5     x = ReadSym()
6     H' = φ
7     for each pair of h ∈ H and e ∈ E(n[h], x) do
8       h' = NewHyp(h · e)
9       α(h') = α(h) + w[e]
10      Insert(H', h')
11    end for
12    H = H'
13  end while
14  BestHyp = min_{h∈H:n[h]∈F} α(h) + ρ(n[h])
```

Figure 1: Decoding Algorithm Using a Single WFST.

GFOC is an extended version of FOC for composing multiple WFSTs. Before describing the extended algorithm, we show a basic procedure in cases using a single WFST. We introduce some symbols, quantities, and functions as follows.

Notation related to WFST	
F	set of final states
λ	initial weight
$\rho(s)$	final weight of state s ($s \in F$)
$E(s, x)$	set of transitions accepting label x from state s
$o[e]$	output label of transition e
$w[e]$	weight of transition e
Variables in the search	
h	hypothesis indicating a path
H	list of hypotheses
$\alpha(h)$	accumulated weight for h.
$n[h]$	state reached by h.
Basic functions for the search	
$ReadSym()$	return the next input symbol
$NewHyp(h)$	generate a hypothesis and initialize it as h
$Insert(H, h)$	insert h into H (If there is a hypothesis $\bar{h}$ that has reached $n[h]$ in H, either h or $\bar{h}$ having a smaller weight is retained).

Figure 1 shows an example of the algorithm. At line 1, initial hypothesis h is generated. At line 2 and 3, h is weighted with the initial weight and stored in H. In lines 4 - 13, the main search step is performed until the last input symbol is processed. The main step iterates the generation of hypothesis by $NewHyp()$ and their recombination by $Insert()$. Finally at line 14, the best hypothesis *BestHyp* is chosen among the hypotheses that have reached one of the final states. In the algorithm, however, transitions by ϵ symbol are not assumed for simplification. Pruning of hypotheses is also omitted.

Next, we explain the GFOC algorithm illustrated in Fig. 2. Given a set of M WFSTs, the algorithm searches for the minimal cost path of the M-th WFST while applying on-the-fly

```
1  h_1 = NewHyp(φ)
2  for m = 2 to M do
3     h_m = NewHyp(φ)
4     Insert(g[h_{m-1}], h_m)
5  end for
6  α_m(h_m) = Σ_{k=1}^{M} λ_k   for all m(1 ≤ m ≤ M)
7  Insert(H, h_1)
8  while the next input exists do
9     x = ReadSym()
10    H' = φ
11    for each pair of h ∈ H and e ∈ E(n[h], x) do
12       h' = NewHyp(h · e)
13       if o[e] ≠ ε then
14          α̃ = α_1(h) + w[e]
15          < α_1(h'), g[h'] >
                = FastCompose(α̃, g[h], o[e], 2)
16       else
17          α_1(h') = α_1(h) + w[e]
18          g[h'] = g[h]
19       end if
20       Insert(H', h')
21    end for
22    H = H'
23 end while
24 ĥ_1 = min_{h∈H:n[h]∈F_1} α_1(h)
25 for m = 2 to M do
26    ĥ_m = min_{h∈g[ĥ_{m-1}]:n[h]∈F_m} α_m(h) + ρ_m(n[h])
27 BestHyp = ĥ_M
```

Figure 2: GFOC Algorithm.

composition. The symbols and quantities in Fig. 2 are extended by subscript m indexing the corresponding m-th WFST.

At line 1, initial hypothesis h_1 for the first WFST is generated. In lines 2 - 5, a cohypothesis h_m for the m-th WFST is generated and associated with the *parent* hypothesis by inserting it into the cohypothesis list $g[h_{m-1}]$. At line 6, each h_m is weighted as $\sum_{k=1}^{M} \lambda_k$. At line 7, h_1 is stored in H.

The main search step is performed in lines 8 - 23. The main step is similar to the search with a single WFST, except for lines 13 - 16 where on-the-fly composition is performed by function $FastCompose()$ only when transition e has a non-epsilon output label. A compensated weight and a new cohypothesis list are prepared in this function. If computation for $FastCompose()$ can be ignored, the complexity of this algorithm is equal to the algorithm in Fig. 1. In lines 24 - 27, the best hypothesis *BestHyp* is chosen as the best complete hypothesis of the M-th transducer.

Finally we show the algorithm of $FastCompose()$ in Fig. 3. The function takes arguments α, L, y, and m given by a *parent* hypothesis in the procedure calling this function. α stands for the weight of the hypothesis before compensation. L is a cohypothesis list associated with the parent hypothesis. y is an output symbol. m is an index number of a WFST composed in this call. The function returns a compensated weight α' and a new list L' by expanding each cohypothesis in L with y.

This function includes a recursive call at line 7 by which multiple WFSTs can be composed in the same manner. This recursive call does not exist in the original FOC algorithm. β calculated at line 2 means a weight accumulated along the *parent* hypothesis from the last call to the current call. β is used to

```
1  function FastCompose(α, L, y, m)
2     β = α − min_{f∈L} α_m(f)
3     for each pair of f ∈ L and r ∈ E(n[f], y) do
4        f' = NewHyp(f · r)
5        if o[r] ≠ ε and m < M then
6           α̃ = α_m(f) + β + w[r]
7           < α_m(f'), g[f'] >
                = FastCompose(α̃, g[f], o[r], m + 1)
8        else
9           α_m(f') = α_m(f) + β + w[r]
10          g[f'] = g[f]
11       end if
12       Insert(L', f')
13    end for
14    α' = min_{f'∈L'} α_m(f')
15    return < α', L' >
16 end function
```

Figure 3: Function $FastCompose()$.

weight new cohypotheses at lines 6 and 9.

3.1. Application to Class-based Language Models

We describe the efficiency of GFOC in cases of a class-based language model. We assume only a simple case in which each word belongs to just one class.

Using a class trigram model, the probability of a word w occurring in the context of words u, v is calculated as

$$P(w|uv) = P(w|C_w)P(C_w|C_uC_v), \tag{5}$$

where C_u, C_v, and C_w indicate the classes to which u, v, and w belong, respectively.

As mentioned in [5], a WFST of a class n-gram model can be composed as

$$G^C = \pi_2(D \circ T) = \pi_1(T^{-1} \circ D), \tag{6}$$

where D is a WFST of a n-gram model of a class sequence weighted by $P(C_w|C_uC_v)$; T is a WFST that translates a class sequence to a word sequence with $P(w|C_w)$. T^{-1} is the inversion of T; π_1 and π_2 are the first and the second projection operators, respectively.

However, the WFST of such a class n-gram model is usually much larger than a word n-gram model since each transition with a class label is multiplied depending on words belonging to the class. This problem can be solved by GFOC as

$$(H \circ C \circ L \circ G_1) \diamond \tilde{\pi}_1(T_0^{-1} \diamond D_{3/1}), \tag{7}$$

where T_0^{-1} is made by removing transition weights from T^{-1}; $D_{3/1}$ is made by adjusting each transition weight of D as well as $G_{3/1}$. $\tilde{\pi}_1()$ means on-the-fly projection to derive a word sequence rather than a class sequence as a recognition result, which must be implemented in the decoder. This on-the-fly implementation is based on the fact that Eq. (5) can be rewritten as

$$P(w|uv) = P(w) \times P(C_w|C_uC_v)/P(C_w). \tag{8}$$

4. Experiments

We evaluated the GFOC algorithm in a 30k-word spontaneous speech transcription task based on the corpus of spontaneous

Table 1: Sizes of WFSTs

	#WFST	#state	#transition
FULL (WLM)	1	456,460	1,323,860
FULL (CLM)	1	5,068,336	13,764,994
FOC (WLM)	2	271,774	815,043
FOC (CLM)	2	189,638	4,023,119
GFOC (CLM)	3	193,318	732,324
GFOC (WLM+CLM)	4	441,928	1,454,029
G	-	248,611	751,458
T	-	1	30,000
B	-	170,154	638,986
G^C	-	166,475	3,959,781

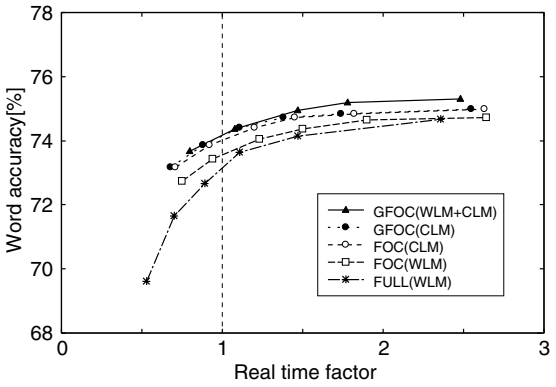

Figure 4: WACC vs. RTF using several decoding approaches

Japanese (CSJ) [6], which is mostly comprised of monologues such as lectures, presentations, and news commentaries.

Evaluation data were limited to lectures in academic fields. Feature vectors had 38 elements consisting of 12 MFCCs, their delta and delta-delta components, and delta and delta-delta log energies. Tied-state triphone HMMs with 3,000 states and 32 Gaussians per state were made by using utterances of male speakers (approximately 187 hours). Language models were estimated using transcription of 2,496 talks (6,203,011 words). Benchmark test 1 was used for evaluation, which consists of ten academic talks (26,176 words) presented by male speakers. The out-of-vocabulary rate is 1.1%, and the test-set perplexity is 118.1 for a trigram model. A standard PC (with a Xeon 3.0 GHz processor) was used to measure the decoder speed.

First we evaluated the size of the WFSTs. Table 1 shows the number of states and transitions of each WFST. In the table, FULL denotes a fully composed transducer ($H \circ C \circ L \circ G$). The optimization step was performed after each composition step. FOC and GFOC indicate fast on-the-fly composition when using only two WFSTs and when using three or more WFSTs, respectively. #WFST denotes the number of WFSTs composed during decoding. Each number in the column of #state or #transition is derived by counting all states or transitions of WFSTs used in decoding. WLM and CLM indicate a word model and a class model. In this experiment, a set of classes is derived using a bottom-up clustering of words based on mutual information. We set the number of classes to 2,000.

WLM+CLM indicates an interpolated model of word and class n-grams. The WFST can be composed as

$$(H \circ C \circ L \circ G_1) \diamond G_{3/1} \diamond \tilde{\pi}_1(T_0^{-1} \diamond D_{3/1}). \qquad (9)$$

In this case, each transition weight is derived by interpolating the weights of $G_{3/1}$ and $G^C_{3/1}$. We multiplied each weight in $G_{3/1}$ and $G^C_{3/1}$ by 0.5. This means linear interpolation in log-domain.

In table 1, it is shown that FOC and GFOC need less transitions than FULL. FOC(CLM), however, needs about 5 times transitions of FOC(WLM) because $T^{-1} \circ D$ results in a huge G^C. In GFOC(CLM), the total transitions is less than FOC(WLM) because D and T are not composed beforehand.

Second, we performed speech recognition using each WFST set. Figure 4 shows the relationship between word accuracy (WACC) and real-time factor (RTF) in each method when changing the beam width parameter of the decoder. It is shown that CLM yielded higher accuracies than WLM in all decoding methods, and both FOC and GFOC outperformed FULL(WLM). We could not evaluate FULL(CLM) in speech recognition because of the limitations of memory. In CLM, GFOC worked 2 - 7% faster than FOC, and the process memory size was less than a half of FOC. Finally GFOC(WLM+CLM) using four separated transducers worked well with small memory usage, and marked the best performance.

5. Conclusions

In this paper, we proposed a Generalized Fast On-the-fly Composition (GFOC) algorithm for WFST-based speech recognition. In a 30k-word vocabulary spontaneous speech transcription task, GFOC yielded fast and memory efficient decoding especially when using class-based language models. GFOC is also very convenient when off-line composition is difficult due to time and/or memory problems. GFOC has the potential to be successfully applied to other speech-input language processing. In the future, we would like to apply this technique to speech translation, speech summarization and so on.

6. Acknowledgments

We thank MIT's SLS group for providing the tools for composing and optimizing WFSTs.

7. References

[1] M. Mohri, F. Pereira, M. Riley, "Weighted finite-state transducers in speech recognition," in *Proc. ASR2000*, pp. 97–106, 2000.

[2] H. J. G. A. Dolfing, I. L. Hetherington, "Incremental language models for speech recognition using finite-state transducers," in *Proc. ASRU2001*, 2001.

[3] D. Willett, S. Katagiri, "Recent advances in efficient decoding combining on-line transducer composition and smoothed language model incorporation," in *Proc. ICASSP2002*, Vol. I, pp. 713–716, 2002.

[4] T. Hori, C. Hori, Y. Minami, "Fast on-the-fly composition for weighted finite-state transducers in 1.8 million-word vocabulary continuous-speech recognition," *Proc. ICSLP2004*, Vol. I, pp. 289–292, 2004.

[5] C. Allauzen, M. Mohri, B. Roark, "Generalized algorithms for constructing statistical language models," in *Proc. ACL2003*, pp. 40–47, 2003.

[6] T. Kawahara, H. Nanjo, T. Shinozaki, S. Furui, "Benchmark test for speech recognition using the corpus of spontaneous Japanese," *Proc. SSPR2003*, pp. 135–138, 2003.