3-2 Development of the SprinTra WFST Speech Decoder

DIXON Paul Richard, HORI Chiori, and KASHIOKA Hideki

In this paper we describe the NICT Weighted Finite State Transducer (WFST) based speech decoder named SprinTra. The paper starts with a brief introduction to WFSTs and the accompanying mathematical notation. This is followed by an introduction to the use of WFSTs in speech recognition, here we give a brief description of the WFST components used in a typical speech recognition system, and explain how they are combined and optimized to yield very efficient decoder search spaces. After describing these preliminaries we move on to a high level description of the features and architecture of SprinTra. Our focus was to design and implement an engine suitable for research and deployment usage. To bring the state-of-the art speech recognition technology to as many users as possible SprinTra can run on many platforms and be additionally accessed through various programming interfaces and scripting layers. The supporting tools are also designed with portability and good usability, and this allow users and non-speech recognition experts to easily construct state-of-the-art speech recognition systems. The description of SprinTra's core features includes a description our on-the-fly algorithm we have proposed to allow for memory efficient composition of class N-gram models.

Keywords
WFST, Speech recognition, Decoder, On-the-fly composition

1 Introduction

In this paper we introduce *SprinTra* a modern state-of-the-art speech decoder that is currently under development at NICT. Some of the main goals in development of the SprinTra were to:

- Create a recognition engine which would provide a state-of-the-art research platform for implementing and exploring new ideas.
- Develop a commercial quality engine that could be used in live services such as VoiceTra and licensed as a standalone engine.

The engine is designed to operate on Weighted Finite State Transducers (WFSTs) [1] which are a type of finite state machine that can provide a mapping between strings with an optional weight to represent uncertainty. Recently, the use of WFSTs in speech recognition has become extremely popular, one of the main advantages of the approach is the unified manner all of the models can be optimized and combined together. Furthermore, performing the optimization ahead of decoding allows for the development of speech recognition engines that can often deliver faster recognition speeds when compared to more traditional dynamic decoders [2].

However, there are several drawbacks to the unified approach, firstly large amounts of memory can be required to hold the fully composed network during recognition and the off-line memory usage during composition and optimization may become prohibitively large. After the composition access to the information source is lost and therefore changing the models on-line becomes much more difficult. To mitigate these issues various on-the-fly composition approaches have been proposed [3]-[12]. Later in the paper we describe a specialized algorithm we have developed to allow memory efficient *three-way* composition of class based N-gram language models [13].

The rest of this paper is structured as follows: In Section **2** we give an introduction to WFSTs and the accompanying notation. Section **3** brifley describes the use of WFSTs in speech recognition. Section **4** gives a high level description of the features and architecture of SprinTra. This is followed by a section describing our memory efficient class N-gram composition scheme. The paper is finished with a summary in Section **6**.

2 Weighted Finite State Transducers (WFST)

We first start with a brief overview of WFSTs that will cover the theoretical foundation needed to describe the algorithms presented later in the paper. For a more in depth description of WFSTs in speech recognition the reader is referred to [1][14][15].

A WFST is a generalized type of finite automata where each of the transitions has an output label and optional weight in addition to the input label. Formally a transducer T is defined as the 8 tuple [14][15].

$$T = (\Sigma,\ \Delta,\ Q,\ I,\ F,\ E,\ \lambda,\ \rho) \quad (1)$$

Where:

- Σ is a finite input alphabet.
- Δ is a finite output alphabet.
- Q is a finite set of states.
- $I \subseteq Q$ is the set of initial states.
- $F \subseteq Q$ is the set of final states.
- $E \subseteq Q \times (\Sigma \cup \{\epsilon\}) \times (\Delta \cup \{\epsilon\}) \times \mathbb{K} \times Q$ is a finite set of transitions.
- $\lambda : I \rightarrow \mathbb{K}$ the initial weight function.
- $\rho : F \rightarrow \mathbb{K}$ the final weight function.

3 WFSTs for speech recognition

We follow the construction scheme described in [14] where 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.

To give the reader a feel for based WFST representations a simple N-gram language model G is shown in Fig. 1. In this Figure the arc labels are of the form *label/weight*. To efficiently represent the N-gram language model as an WFST, each state is used to denote a N-gram history and each arc represents a N-gram probability or back-off score. For example the arc $a/P(a|b)$ gives the bigram probability ab. The input label a indicates we consume a from the input tape, the source state labeled b shows the current unigram history and the destination state label ba gives the new history. When no high order N-grams transitions match the current symbol we encode the back-off scores as epsilon transitions as described in [16]. These epsilon arcs consume no input and allow us to follow back-off transitions to lower order N-gram histories until the current input symbol can be matched.

The G, L and C transducers are combined according to:

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

where det is the determinization operation, min is the minimization operation, $\circ$ is the composition operation and the π operator is a procedure that removes auxiliary symbols. The determinization operation is similar to a prefix sharing operation, whilst the minimization operation is equivalent to a tail sharing operation.

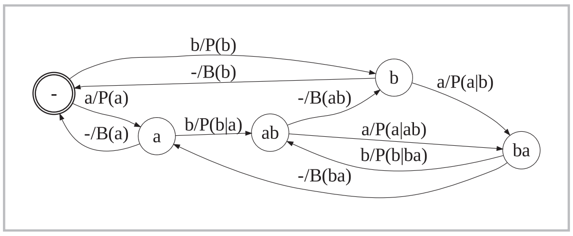

Fig.1 *A simple backoff N-gram model represented as a WFST*

To illustrate these operations, a very small speech recognition lexicon *L* is shown in Fig. 2. Here each word is a linear chain of arcs and states, the input side of each chain is sequence of phone that maps to a word output symbol. The lexicon shown here is constructed as the union of a set of such chains. In Figure 3 the effects of the determinization and minimization can be seen, here we can see common prefixes and suffixes are shared to give more efficient WFST but equivalent functional representation.

4 Engine features overview

4.1 Core features

The core feature set includes all of the functionality one would expect from a modern recognition engine:

● **Input** - There is an integrated front end

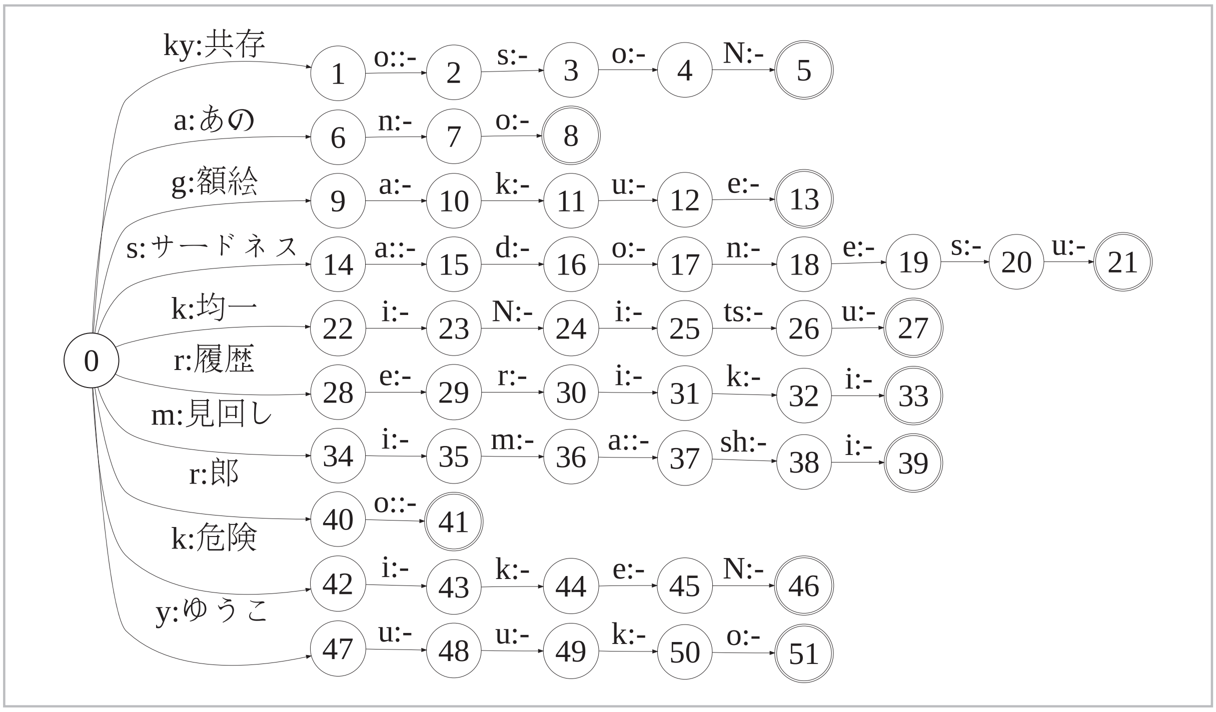

Fig.2 A simple lexicon transducer

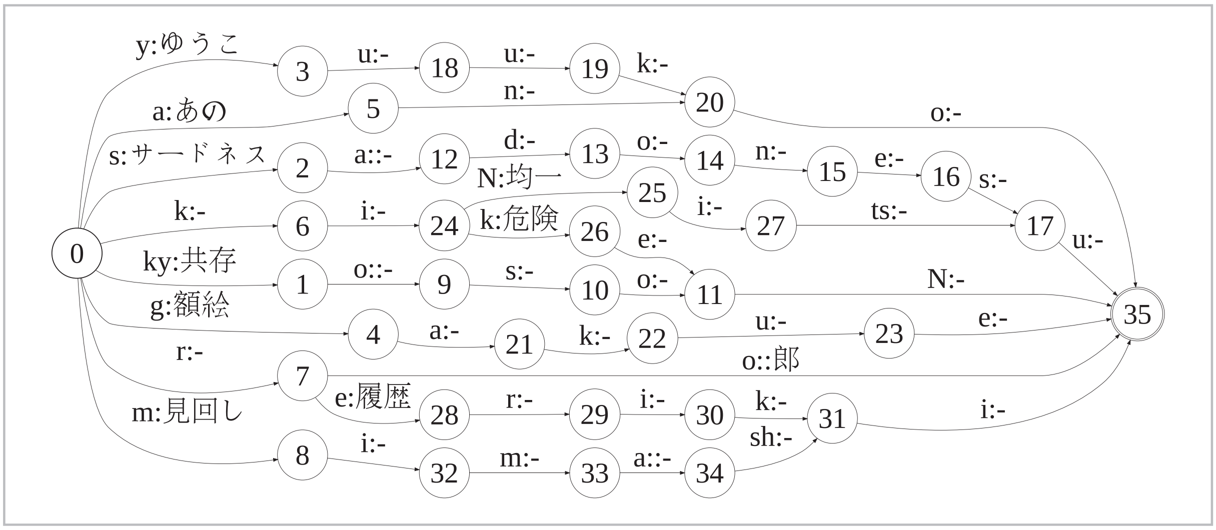

Fig.3 A deterministic and minimalistic equivalent lexicon

for speech signal feature extraction or we can optional consume pre-computed features from files or network sockets.

- **Output** - SprinTra can output 1-best recognition results or a lattice which is a representation of the many alternative speech hypotheses. From the lattice useful information such n-best lists or confusion networks can be extracted.
- **Flexibility** - For flexibility we can operate on static search networks, compose language dynamically or use our specialized class model composition algorithm.

4.2 Portability

One of the main requirements of SprinTra was to bring the technology to as many platforms as possible. To achieve this we have no binary library dependencies and the decoder is currently able to run as the following:

- The decoder can be run as command line application for batch mode processing and used in offline research setting.
- We also provide a developer level programming interface on both Windows and Unix platforms.
- Finally we provide a high-level Python scripting interface for rapidly developing server applications.

4.3 Supporting tools

Constructing the integrated and component WFSTs requires much skill and experience. In particular certain composition or optimization operations will never terminate and easily exhaust huge memory machines. The SprinTra toolbox includes several integrated build tools that convert the raw models to WFSTs. The tools will perform all the necessary composition and optimization steps based on the applications requirements. Using the tools allows for non-domain experts and developers to rapidly and safely build high performance WFST speech recognition systems. There are many subtle choices which can have a profound effect on the final search network. In just one command we use our experience to select the best optimization options.

Like the decoder itself the build tools are portable and run on all supported platforms. For seamless integration into NICTs current architectures we support standard and legacy formats such as HTK [17] or ATRASR acoustic models and ARPA format language models. For the WFST representations we use the ATT text format and the OpenFst binary format.

5 Fused composition algorithm

A class N-gram model can be represented as two transducers, an N-gram model of class labels G, and a transducer T that maps from class labels to word labels. The expansion of G with T can be performed using standard composition and projection operations [1]:

$$GT = Sort_1(Proj_2(G \circ T)) \qquad (2)$$

Here the subscripts 1 and 2 denote input or output labels respectively. $Proj_2$ replaces all output labels in GT with word labels. The final $Sort_1$ is necessary to make the subsequent composition with CL more efficient. During speech decoding the entire cascade is:

$$CL \circ (Sort_1(Proj_2(G \circ T))) \qquad (3)$$

Where CL is shorthand for $C \circ det(L)$. The static expansion of a class N-gram will often require substantial memory as each class label in G could potentially be multiplied by every transition in T. Therefore the expansion of GT is performed on-the-fly.

Our proposed algorithm exploits the limited topology of the G and T WFSTs. We assume T has a single state and each arc represents a mapping from a class label to a single word label. Therefore the WFST resulting from the composition $G \circ T$ will have the same number of states as G. Under these constraints the composition and sort is equivalent to merging a set of sorted lists, and it is a simple addition to also fuse the projection operation into the merging process.

The fused GT WFST type aggregates G and T and works as follows: T is represented as a set of lists each sorted by word labels and indexed by the class labels. During decoding given a state s in GT, the first step is to iterate over the k arcs leaving state s in G and create a list of k class-to-word lists. A min-heap keyed on the output labels is used to merge the k lists with a total running time of $O(n\log(k))$ [18], where k is the total number of arcs leaving s and n is the sum of arcs across the k class lists. The fused algorithm also removes the memory requirement of the lazy project and sort operations.

6 Summary

In this paper we have introduced the SprinTra decoder. We have achieved the important goal of providing NICT with a state-of-the-art engine for future speech recognition and speech translation endeavors. In addition our highly portable engine and easy to use toolkit allow for non-domain experts and developers to easily make use of cutting edge WFST based speech recognition. In current and future versions we are working on more efficient algorithms that allow for dynamic vocabulary, this is very important for certain applications. In addition our current research is focused researching new methods for combining SprinTra with WFST based dialogue and translation systems. On a development front we have shown the portability of the SprinTra and would like to bring standalone versions to smaller platforms such as smartphones and tablets.

References

1 M. Mohri, F. Pereira, and M. Riley, "Weighted finite-state transducers in speech recognition," Computer Speech and Language, 16(1): 69–88, 2002.

2 S. Kanthak, H. Ney, M. Riley, and M. Mohri, "A comparison of two LVR search optimization techniques," In Proc. ICSLP, pp. 1309–1312, 2002.

3 Diamantino Caseiro and Isabel Trancoso, "Transducer composition for on-the-fly lexicon and language model integration," In Proc. ASRU, pp. 393–396, 2001.

4 Diamantino Caseiro and Isabel Trancoso, "Using dynamic WFST composition for recognizing broadcast news," In Proc. ICSLP, pp. 1301–1304, 2002.

5 Takaaki Hori, Chiori Hori, and Yasuhiro Minami, "Fast on-the-fly composition for weighted finite-state transducers in 1.8 millionword vocabulary continuous speech recognition," In Proc. Interspeech, pp. 289–292, 2004.

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

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, 14(4): 1281–1291, 2006.

8 Takaaki Hori, Chiori Hori, Yasuhiro Minami, and Atsushi 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, 15: 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, pp. 2110–2113, 2008.

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

11 C. Allauzen, M. Riley, and J. Schalkwyk, "A generalized composition algorithm for weighted finite-state transducers," In Proc. Interspeech, pp. 1203–1206, 2000.

12 Hasim Sak, Murat Saraclar, and Tunga Gungor, "On-the-fly lattice rescoring for real-time automatic speech recognition," In Proc. Interspeech, pp. 2450–2453, 2010.

13 Peter F. Brown, Peter V. deSouza, Robert L. Mercer, Vincent J. Della Pietra, and Jenifer C. Lai, “Class-based n-gram models of natural language,” Computer Linguistics, 18(4): 467–479, Dec 1992.
14 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.
15 M. Mohri, “Weighted automata algorithms,” Springer Handbook of weighted automata, (to appear) 2009.
16 C. Allauzen, M Mohri, and B. Roark, “Generalized algorithms for constructing statistical language models,” In Proc. of 41st Annual Meeting of the Assocication for Computational Linguistics, pp. 40–47, 2003.
17 S. Young, G. Evermann, D. Kershaw, G. Moore, J. Odell, D. Ollason, V. Valtchev, and P. Woodland, “The HTK Book (for HTK Version 3.2),” Cambridge University Engineering Department, 2006.

(Accepted June 14, 2012)

***DIXON Paul Richard**, Ph.D.*

Researcher, Spoken Language Communication Laboratory, Universal Communication Research Institute

Speech Recognition and Langauge Processing Technologies

***HORI Chiori**, Ph.D.*

Senior Researcher, Spoken Language Communication Laboratory, Universal Communication Research Institute

Speech Recognition, Speech Translation, Spoken Dialog Technologies

***KASHIOKA Hideki**, Ph.D.*

Director, Spoken Language Communication Laboratory, Universal Communication Research Institute

Spoken Language Processing, Speech Translation, Spoken Dialogue