Preliminary Evaluations of a WFST Speech Decoder∗

△ Diamantino A. Caseiro†‡ △Paul R. Dixon† ◎ Tasuku Oonishi† Sadaoki Furui†
Tokyo Institute of Technology† INESC-ID, Portugal‡

1 Introduction

In this paper we present preliminary evaluations on the large vocabulary speech decoder we are currently developing at Tokyo Institute of Technology. Our goal is to build a scalable and flexible decoder to operate on *weighted finite state transducer* (WFST) search spaces. Even though the development of the decoder is still in its infancy we are already achieving good accuracy and speed on a large vocabulary spontaneous speech task.

2 Background

The basic speech recognition problem can be expressed as:

$$\hat{\mathbf{W}} = \arg\max_{\mathbf{W}} \{P(\mathbf{X} \mid \mathbf{W})P(\mathbf{W})\} \qquad (1)$$

The task of the decoder is to find the most likely word sequence $\hat{\mathbf{W}}$, given the observed sequence of speech feature vectors $\mathbf{X}$. The language model probability is denoted by $P(\mathbf{W})$, whilst $P(\mathbf{X} \mid \mathbf{W})$ is the score from the other knowledge sources such as the acoustic models.

Within the WFST paradigm all the knowledge sources in the search space are combined together to form a static search network [1]. The composition often happens off-line before decoding and there exists powerful operations to manipulate and optimise the search networks. Within this framework the decoder becomes agnostic to various knowledge sources, changes can be made to the information sources before the network compilation stage and should not require modifications to the decoder.

However there is a caveat with the WFST approach. The fully composed networks can often be very large and therefore the decoders can require large amounts of memory. To alleviate the memory burden on the decoder approaches such as *on-the-fly* composition of the network [2, 3], disk based search networks [4] and efficient in-memory representations [5] have been developed by others.

3 Decoder Description

We have so far developed a single pass time synchronous Viterbi beam search decoder using a token passing scheme. The decoder has been designed specifically with speech tasks in mind.

We are primarily considering a recogniser cascade which performs a transduction from phone arcs to word sequences, that is $C \circ L \circ G$ (CLevel) where C is the context dependency, L is the lexicon and G is the language model. During search the decoder dynamically expands the HMM arcs into state sequences. It is also possible to decode state level networks by composing acoustic models H into the recogniser cascade $H \circ C \circ L \circ G$ (HLevel), and then simulating the self-transitions via appropriate arc definition with length one.

The algorithm keeps a list of active search states and a list of active arcs. States and arcs are considered active when they contain tokens. The algorithm consists of creating a token for the empty hypothesis, associating it with the initial state of the search network and then iterating the three following steps for each parameter vector in the utterance. In the first step, expand active states, tokens are propagated from each state to the initial state of every arc leaving the state. In the second step, expand active arcs, the tokens held in each arc are advanced to the next frame. This step uses a specialised time synchronous Viterbi algorithm optimised to the arc topology. When a token reaches the last state in the arc, it is propagated to the following search state, activating it if necessary. Finally, in the third step, epsilon propagation, tokens are propagated across epsilon input edges. The best solution is recovered as the best hypothesis contained in a final state after processing the last frame of the utterance.

3.1 Multiprocessing

It is common for commodity processors to support some form of multiprocessing, either as multiple cores or as multi threads. To fully explore these architectures, our decoder divides the work in various threads, the approach is similar to [6]. Each thread has its own active state and arc lists, and its own memory management. States are assigned to a particular thread using a hash function of its numerical id and arcs are assigned to the same thread as its source state. Interaction between threads occurs when a token is propagated from the last state of an arc to the following search state, because this state can belong to a different thread. A pending activation list is used to delay this propagation so that it will be later done by the thread that owns the state. Synchronisation between threads is achieved using barriers at various points in the algorithm: before propagating pending lists, and before and after epsilon propagation. Epsilon propagation is still single threaded.

∗WFST による音声認識デコーダの検討
カセイロ・ディアマンティーノ、 ディクソン・ポール、 大西 翼、 古井 貞熙

4 Evaluation

Our evaluations were carried out using the corpus of spontaneous Japanese (CSJ) [7]. There are a total of 228 hours of training data from 953 lectures.

The raw speech was first converted to a sequence of 38 dimensional feature vectors with a 10ms frame rate and 25ms window size. Each feature vector was composed of 12 Mel-frequency cepstral coefficients with delta and delta–deltas, augmented with delta and delta–detla squared energy terms.

The state output densities were 16 component Gaussian mixture models with diagonal covariance. The language model was back-off trigram with a vocabulary of 25k words

The test set used for evaluations was comprised of 2328 utterances spanned across 10 lectures. This gave a total of 116 minutes of speech. On this testing data the language model perplexity was 57.8 and the out of vocabulary rate was 0.75%.

We evaluated our decoder operating on both CLevel and HLevel networks and compared the performance to the Julius decoder [8]. Several recognition experiments were run with the beam width (relative to the best hypothesis) varied from 100 to 250, the maximum number of hypotheses allowed at any one time during the decoding was capped to the best 10000. The experiments were conducted on a 2.40GHz Intel Core2 machine with 2GB of memory. The WFST decoder was additionally run using two threads to take advantages of both of the cores in the processor.

Figure 1 shows the recognition accuracy and its relationship to the real time factor (RTF). The WFST decoder was consistently able to achieve higher accuracy for the same RTF when compared with Julius. The HLevel decoder was slower than the CLevel decoder for the same beam width, this is partly because at the HLevel every time the decoder exits a state there is work involved manipulating the active arc and state lists. The multi-threaded decoder was faster than its single threaded sibling. However, during decoding in the parallel mode both cores did not fully saturate due to the non-parallel epsilon propagation step and waiting for threads to synchronise.

The CLevel network had approximately 2.1M states and 4.3M arcs. The decoder required between 240MBs and 260MBs of memory during search. As expected the HLevel network had substantially more states and arcs at approximately 6.2M and 7.7 respectively, this translated to between 570MBs $\sim$ 590MBs of memory usage in the decoder. Julius required 60MBs $\sim$ 100MBs of memory.

5 Conclusions

In this paper we have presented preliminary evaluations of our new large vocabulary speech decoder. Future work will include adding support for on-the-fly composition of the transducer network, more efficient memory usage and increasing recognition accuracy. There is still headroom in the current decoder to improve the parallelisation. We will conduct further research on ways to take advantage of multi-core processors.

Acknowledgements This work was supported by the Japanese government 21st century COE programme “Framework for Systematization and Application of Large-Scale Knowledge Resourcesre” and the METI Project “Development of Fundamental Speech Recognition Technology”.

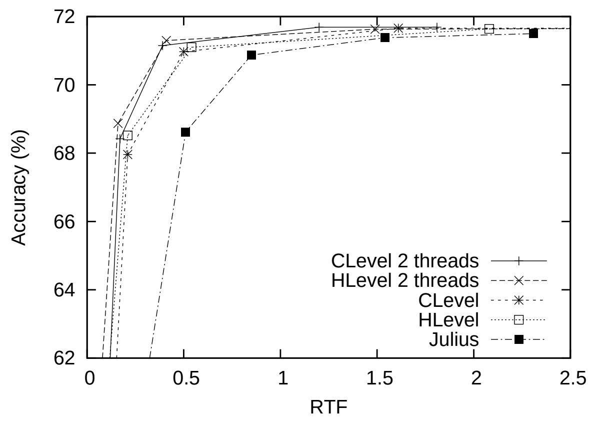

Figure 1: Recognition Accuracy vs. RTF.

References

[1] Mehryar Mohri, Fernando Pereira, and Michael Riley. Weighted finite-state transducers in speech recognition. *Computer Speech and Language*, 16(1):69–88, 2002.

[2] Takaaki Hori and Atsushi Nakamura. Generalized fast on-the-fly composition algorithm for WFST-based speech recognition. In *Proc. of INTERSPEECH*, pages 847–850, 2005.

[3] Diamantino A. Caseiro and Isabel 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.

[4] Daniel Willet, Erik McDermott, Yasuhiro Minami, and Shigeru Katagiri. Time and memory efficient Viterbi decoding for LVCSR using a precompiled search network. In *Proc. of EUROSPEECH*, pages 847–850, 2001.

[5] Diamantino A. Caseiro and Isabel Trancoso. Using dynamic WFST composition for recognizing broadcast news. In *Proc. ICSLP*, pages 1301–1304, 2002.

[6] Steven Phillips and Anne Rogers. Parallel speech recognition. *Int. J. Parallel Program.*, 27(4):257–288, 1999.

[7] Kikuo Maekawa. Corpus of spontaneous Japanese: Its design and evaluation. In *Proc. ISCA and IEEE Workshop on Spontaneous Speech Processing and Recognition*, pages 7–12, 2003.

[8] Akinobu Lee, Tatsuya Kawahara, and Kiyohiro Shikano. Julius an open source real-time large vocabulary recognition engine. In *Proc. of EUROSPEECH*, pages 1691–1694, 2001.