An Investigation into Lattices Generated From WFST Based Decoders*

◎ Paul R. Dixon Chiori Hori Hideki Kashioka
NICT, Kyoto, Japan
paul.dixon@nict.go.jp

1 Introduction

In recent years the Weighted Finite State Transducer (WFST) based approach to speech recognition search space construction and decoding [8] 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 manner, combined together and then optimized. Hoffmeister [4] draws an analogue between WFST operations and high-level programming languages. He describes how the WFST framework provides a crucial part of other aspects of a speech recognition system such as confidence scoring, active learning, system combination and discriminative training. Many of these procedures are based around the processing and manipulation of lattices.

A speech recognition lattice is a directed graph which encodes a compact representation of the decoders search space for a given utterance. A lattice will often contains the same word sequence many times with different weightings to correspond to the various way an audio signal can be transcribed to the same word sequence. For many applications such as n-best generation and system combination it is essential to remove this ambiguity and keep only the best scoring path for any given word sequence.

In this paper we investigate optimization methods for lattices generated from WFST decoders. Our end goal is to obtain accurate word level lattices that are unambiguous and have correct score and time alignments. Towards achieving this goal we describe a construction method and algorithm to correctly time align word and acoustic costs. The other main contribution of this paper is an experimental study into the behavior of a recently proposed disambiguation [7] algorithm. The disambiguation algorithm is used to remove duplicate paths from automata and we compare the algorithm with the standard determinization based approach.

2 Conversion to Word Level Lattices

Our decoder *SprinTra* [3] generates phone-to-word lattices based on the phone-pair approximation [5]. The standard way to convert phone lattices to word lattices is to apply a combination of the projection and epsilon removal operations. However, during construction of the search network the optimizations will move the labels and this makes the acoustic scores accumulated on the word labels in the lattice incorrectly time aligned.

To recover the correct acoustic score alignments we use a modified construction and a specialized synchronization algorithm [6]. During the search network construction phase we first introduce a set of virtual *word-end* phone alternatives which prevents word-end and word-internal tri-phones from becoming merged and allows us to recover word end boundaries.

After recognition we apply a specialized algorithm similar to synchronization [6] to the lattices. This algorithm traverses the lattice and synchronizes word labels and costs with their corresponding word-end phones. During this synchronization we also collect the input phones until we reach a word-end phone. We then replace the word-end phone with a special factored phone sequence label. After the synchronization all other epsilon transition are removed to give the final lattice. In the final processed lattices each transition carries a word label and an input label that identifies the phone sequence for the word.

3 Disambugiation

The determinization algorithm is used to compute an equivalent deterministic automata which will contain the best scoring path for each label sequence (assuming the tropical semiring). There are two issues with determinization. Firstly, for weighted automata the algorithm may never terminate, lattices are acyclic graphs and therefore determinization is guaranteed to terminate because there are only a finite number of possible subset. However, even for the acyclic case determinization can take exponential time in the worst case. Secondly, determinization will move costs and therefore time alignment of scores becomes impossible.

In practice the determinization of lattices is problematic as the memory usage can *blow-up*. This is due to the addition of weights in the subset construction causing a increases in the number of states in the determinized automata. This problem was studies in depth by Buchsbaum [2] who looked at the properties that cause a lattice to become significantly larger or smaller than the

---

WFST デコーダから生成されたラティスに関する研究
ディクソン・ポール、堀智織、 柏岡秀紀

original. Buchsbaum proposed an approximate determinization algorithm that did not suffer the same blow-up at the cost of assigning possible incorrect scores to some paths.

The common way to counter this problem is to prune the lattices before determinization [4]. However, pruning requires a heuristic parameter which if too large may remove too much information, or if the parameter is too small the determinization may terminate efficiently. Pruning also has the undesired effect of removing the amount of variation in the lattice and this could cause better word sequences to be lost. For certain applications it is clearly more beneficial to maintain all the alternative word sequences and instead find a different method to remove the lower scoring repeated paths.

Disambiguation is a promising new algorithm recently proposed by Mohri [7]. Briefly the algorithm uses the intersection of an automata with itself to determine which states share a *common future*, that is a common string to a final state. The unambiguous machine is constructed by maintaining a relationship on the new states to indicate which states are related to each other. This relation and lookahead information can then be used to veto the creation of arcs arriving in a state if there already exists a transition that has related source and common future. The reader is referred to Mohri [7] for a detailed explanation of the algorithm.

4 Experiments

To investigate the characterizes of the disambiguation algorithm. We first generated 1288 lattices from the corpus of spontaneous Japanese for various beam widths. The lattices were synchronized and the epsilons removed to generate compact lattices using the technique described in section 2. Before applying disambiguation the weights were removed and the output labels were projected to give a word level automata. The resulting unambiguous lattices were compared to deterministic lattices computed from the same un-weighted processed lattices using the implementation in OpenFst [1].

Lattice	Beam	# of states	# of arcs	Density
Word	125	223	950	41.9
Determinized	125	129	643	28.4
Disambiguated	125	211	864	38.1
Word	150	899	8837	390
Determinized	150	452	10504	463
Disambiguated	150	1006	15604	688
Word	175	3951	89473	3945
Determinized	175	1808	190045	8379
Disambiguated	175	5011	325998	14374

The results show the effect on the lattices sizes and *lattice density* after applying the disambiguation and determinization algorithms. The lattice density is number of arcs in the lattice divided by the number of words in the reference transcription and is often considered to be a measure of the lattice *complexity*.

The results show that the for beams 150 and 175 the unambiguous and deterministic lattices have a higher lattice density owning to an increase in the number of arcs. However, the un-processed lattices have substantially more paths.

To verify the correctness of our implementation each processed lattice was converted to the real semiring and the total number of paths in the lattice counted using a shortest distance algorithm. The optimized lattices had fewer overall paths than the raw word lattices and the disambiguated and determinized lattices always agreed on the total number of paths.

Finally, we considered the queue discipline used to traverse the states of the unambiguous automata. The choice of queue can lead to different but equally correct unambiguous automata [7]. In practice we found that using a stack required more processing time than using a First In First Out (FIFO) queue.

5 Conclusions

In this paper we have described algorithms for optimizing WFST lattices. We presented empirical results using disambiguation to remove redundant paths from lattices. Currently we are investigating ways to extend disambiguation to weighted automata. Although Mohri [7] mentions the existence of weighted disambiguation algorothm no details are given in his paper [7].

Hoffmeister [4] observed that determinism was essential to allow for efficient use of the lattices in post-processing stages. The unambiguous lattices have non-determinism so further work is needed to ascertain how much effect this non-determinism has in practical applications.

References

[1] 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*, pages 11–23, 2007.

[2] A.L. Buchsbaum, R. Giancarlo, and J.R. Westbrook. On the determinization of weighted finite automata. *SIAM Journal on Computing*, 30(5):1502–1531, 2000.

[3] P.R. Dixon, C. Hori, and H. Kashioka. A comparison of dynamic WFST decoding approaches. In *Acoustics, Speech and Signal Processing (ICASSP), 2012 IEEE International Conference on*, pages 4209–4212. IEEE, 2012.

[4] Bjorn Hoffmeister. *Bayes Risk Decoding and its Application to System Combination*. PhD thesis, 2011.

[5] A. Ljolje, F. Pereira, and M. Riley. Efficient general lattice generation and rescoring. In *Sixth European Conference on Speech Communication and Technology*, 1999.

[6] M. Mohri. Edit-distance of weighted automata: general definitions and algorithms. *International Journal of Foundations of Computer Science*, 14(06):957–982, 2003.

[7] M. Mohri. A disambiguation algorithm for finite automata and functional transducers. *Implementation and Application of Automata*, pages 265–277, 2012.

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