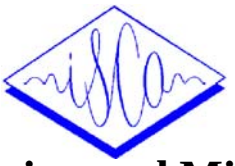

EUROSPEECH '97
5th European Conference on Speech Communication and Technology
Rhodes, Greece, September 22-25, 1997

Weighted Determinization and Minimization for Large Vocabulary Speech Recognition

Mehryar Mohri
mohri@research.att.com

Michael Riley
riley@research.att.com

AT&T Labs – Research, 180 Park Avenue, Florham Park, NJ 07932-0971, USA

ABSTRACT

Speech recognition requires solving many space and time problems that can have a critical effect on the overall system performance. We describe the use of two general new algorithms [5] that transform recognition networks into equivalent ones that require much less time and space in large-vocabulary speech recognition. The new algorithms generalize classical automata determinization and minimization to deal properly with the probabilities of alternative hypotheses and with the relationships between units (distributions, phones, words) at different levels in the recognition system.

10.21437/Eurospeech.1997-60

1. INTRODUCTION

The networks used in the search stage of speech recognition systems are often highly redundant. Many paths correspond to the same word contents (word lattices and language models), or to the same phonemes (pronunciation dictionaries) for instance, with distinct weights or probabilities. More generally, at a given state of a network there might be several thousand alternative outgoing arcs, many of them with the same label. This nondeterminism directly affects the speed of large vocabulary speech recognition systems. The determinization algorithm allows one to address exactly this problem by reducing the alternatives at each state to the minimum. In other words, the *deterministic* result of the algorithm contains at each state at most one transition labeled with a given element of the alphabet considered (words, phonemes, etc.).

Other related work has been done to reduce that redundancy using deterministic trees in particular lexical trees [9, 10, 11]. The general determinization algorithm that we present differs from those approaches by the following: it does not require that networks be constructed as trees, it applies to all networks used in speech processing, and it leads to deterministic networks that are in general much more compact than trees.

The determinization algorithm is not an approximation, a pruning or a heuristic. Its result is exact: for each string, the weight (likelihood) of the best path for that string is the same in the original and in the determinized network. We describe the application of determinization to several distinct tasks: the reduction of the size and the improvement of the speed of using word lattices, a very substantial increase of the speed of large vocabulary systems on the DARPA North American Business task (NAB), and a new task consisting of real-time discrimination among one million surnames (the One Million Names task). Our results show the benefits of using determinization in all of those tasks.

The size of the deterministic networks used in speech recognition can be reduced to the minimum using the weighted minimization algorithm. We also report the result of the application of this algorithm in the NAB and the One Million Names tasks.

2. ALGORITHMS

Weighted determinization and minimization are very general algorithms that apply to weighted automata and transducers. [1] We cannot give a detailed description of these algorithms here. In the following sections, we briefly illustrate these algorithms in the particular case of weighted automata. We have given elsewhere a detailed description of these algorithms, including their mathematical basis and proofs of their soundness [3, 4, 5, 6].

2.1. WEIGHTED DETERMINIZATION

A weighted automaton A is said to be *deterministic*[2] iff at each state of A there exists at most one transition labeled with any given element of the input alphabet. Figure 1 gives an example of a non-deterministic weighted automaton: at state 0, for instance, there are several transitions with the same label a.

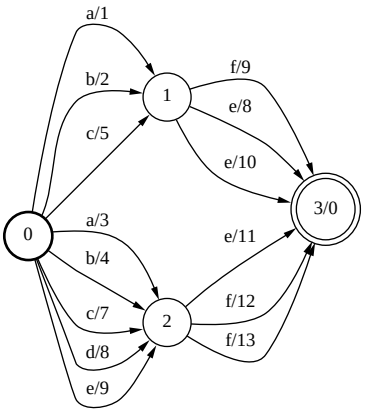

Figure 1: Non-deterministic weighted automaton A_1.

Weighted determinization applies to a weighted automaton and outputs an equivalent weighted automaton that is deterministic. Weighted determinization is a generalization of the classical determinization of automata [2]. Unlike the classical case, not all weighted automata can be determinized – we have determined the set of weighted automata that admit determinization [5]. However, most weighted automata used in speech processing can be determinized. In particular, any acyclic weighted automaton admits determinization.

The advantage of weighted determinization is clear: the use of the deterministic output is much more efficient than that of a non-deterministic one. A deterministic weighted automaton is not redundant. It contains at most one path labeled with any given input string. At each state, the choice of the transition to explore is deterministic, since at most one transition admits a

[1] Transducers are automata that are provided with an additional output label.

[2] The appropriate term used in theoretical computer science is *subsequential*.

label matching the input.

We consider here the common case in speech recognition, where the weights are interpreted as (negative) logarithms of probabilities. The weight of a path is obtained by adding the weights of its transitions. The output associated to an accepted input string is the minimum of the weights of all paths corresponding to that string. The case where weights are interpreted as probabilities can be treated similarly.

Figure 2 gives the result of weighted determinization for the input automaton A_1. The two automata A_1 and A_2 realize exactly the same function: they associate the same output weight to each input string. As an example, there are two paths corresponding to the input string ae in A_1. The corresponding weights are: $\{1 + 8 = 9, 3 + 11 = 14\}$. The minimum 9 is also the output associated by A_2 to the string ae. This is a general characteristic of determinization: the resultant weighted automaton is equivalent to the input.

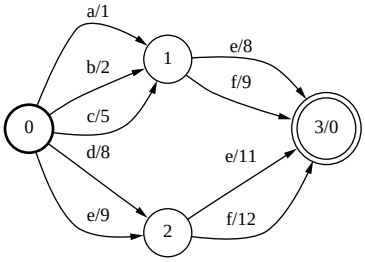

Figure 2: Equivalent weighted automaton A_2 obtained by weighted determinization from A_1.

The algorithm is close to the classical powerset construction for unweighted automata[3]. However, since transitions with the same input label can have different weights, one can only output the minimum of these weights and needs to keep track of leftover weight. Therefore, the states of the output of weighted determinization are subsets of the set of pairs (q, w), where q is a state of the input automaton and w the leftover weight.

The initial subset is $\{(i, 0)\}$, where i is the initial state of the input automaton. For example, for automaton A_1 the initial subset $\{(0, 0)\}$. Each new subset S is processed in turn. For each element of the alphabet a labeling at least one transition leaving a state of S, a new transition t leaving S is constructed in the output machine. The label of t is a and its weight is the minimum of the sums $w + l$ where w is the weight of an a-transition leaving a state s in S and l is s's leftover weight. The destination state of the t is the subset S' made of pairs (q, w), where q is a state reached by a transition labeled with a from a state of S, and w is the appropriate leftover weight.

As an example, the state 0 in A_2 corresponds to the initial subset $\{(0, 0)\}$ constructed by the algorithm. The transition leaving 0 in A_2 labeled with a is obtained by considering the two transitions labeled with a leaving the state 0 in A_1. Its weight is obtained by taking the minimum of the weight of these two transitions. Its destination state is the subset $S' = \{(1, 1 - 1 = 0), (2, 3 - 1 = 2)\}$ numbered 1 in A_2.

Note that the order of expansion of the output automaton does not affect the result. Further, the work done for a given subset S depends only on the elements of S and not on the previous or future work for other subsets. This is an interesting feature of weighted determinization because it makes it possible to give an on-the-fly implementation of the algorithm. Only states and transitions required by the search algorithm are expanded for a given input string. This plays an important role in the implementation of an efficient n-best decoder and in other applications where one does not wish to expand the entire automaton.

[3]The powerset construction is based on the idea that each state of the deterministic automaton corresponds to a set of states of the original non-deterministic one [2].

2.2. WEIGHTED MINIMIZATION

Any deterministic automaton can be minimized using classical algorithms [1, 13]. In the same way, any deterministic weighted automaton A can be minimized using our minimization algorithm [5].

The resulting weighted automaton B is equivalent to the automaton A. It has the minimal number of states and the minimal number of transitions among all equivalent deterministic weighted automata equivalent to A.

The weighted minimization algorithm is very efficient. Its time complexity is equivalent to that of the classical minimization, linear in the acyclic case ($O(|Q| + |E|)$), and in $O(|E| \log |Q|)$ in the general case, where Q is the set of states of A and E the set of transitions.

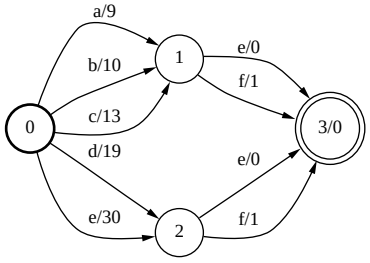

Figure 3: Equivalent weighted automaton B_2 obtained by *pushing* from A_2.

Consider the deterministic weighted automaton A_2. One can view it as an unweighted automaton by considering each pair (a, w), made of a label a and a weight w, as a single label, and then apply the classical minimization algorithm to it. But, since the pairs are all distinct, classical minimization would have no effect on the automaton A_2.

The size of A_2 can still be reduced using (true) weighted minimization. The algorithm works in two steps: the first *pushes* weight among arcs, and the second applies the classical minimization algorithm to the result with each distinct label-weight pair viewed as a distinct symbol, as described above.

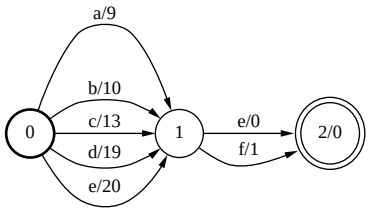

Figure 4: Equivalent weighted automaton A_3 obtained by weighted minimization from A_2.

The pushing step moves the weights of the input automaton towards the initial state as much as possible. This does not change the topology of the input automaton and produces an equivalent automaton. Figure 3 shows the result of pushing for the input A_2. Thanks to pushing, the size of the machine can then be reduced using classical minimization.

Figure 4 illustrates the result of the final step of the algorithm.

Table 1: Word lattices in the ATIS task.

	Determinization	Determinization + Minimization
Objects	Reduction factor	Reduction factor
States	≈ 3	≈ 5
Transitions	≈ 9	≈ 17
Paths	$> \mathbf{2^{32}}$	$> \mathbf{2^{32}}$

No approximation or heuristic is used: the resulting automaton A_3 is equivalent to A_2.

3. EXPERIMENTS AND RESULTS

We have given an efficient implementation of weighted determinization (on-the-fly) and weighted minimization. These programs are currently used in speech processing projects at AT&T Labs and at Lucent Bell Laboratories. In the following sections, we describe their use in several speech recognition tasks and report the corresponding results of our experiments. The results show their efficiency and the importance of their use in all these tasks.

3.1. WORD LATTICES

We applied weighted determinization to the word lattices obtained in the ARPA ATIS task. This not only made the use of the word lattices more efficient by removing their redundancy, but also led to an average reduction of their size by a factor of 9 (table 1).

Figures 5-6 illustrate the weighted determinization in a specific case. Figure 5 corresponds to a word lattice W_1 obtained in speech recognition for the 1,500-word ATIS task. It corresponds to the following utterance: *Show me the flights from Charlotte to Minneapolis on Monday.* Although it is one of the smallest word lattices obtained in this task, W_1 is very complex. It contains more than 151 million paths.

Weighted determinization applies to this lattice. This clearly improves the speed of the use of the lattice for search or matching purposes. It also reduces the size of the original network by reducing its redundancy. The resulting deterministic lattice W_2 only contains 18 paths (figure 6). As mentioned previously, the result is exact: the two lattices realize exactly the same function. The algorithm is very efficient: it took about $.06$s real time including I/O's (reading and writing the networks) to determinize the lattice W_1 on an SGI O2 174 MHz IP32.

The minimization of weighted automata allowed us to reduce further the size of the deterministic weighted automata. The total reduction factor corresponding to the use of determinization and minimization in the ATIS task was about 17 on the average for the word lattices that we used (table 1).

We also applied weighted minimization to deterministic word lattices obtained in the NAB task. On the average, it reduced by a factor of 3 the size of those lattices.

Figure 7 illustrates the use of the minimization algorithm applied to the deterministic network W_2. The resulting machine W_3 has the least number of states and transitions among all equivalent deterministic networks. Let us insist that the minimization algorithm is also an exact algorithm: it is not a heuristic or an approximation; the minimal network contains exactly the same best paths with exactly the same labels and total weights. The algorithm is very efficient: it took about $.07$s to minimize the word lattice W_2, including I/O (which take most of this time) on the same computer.

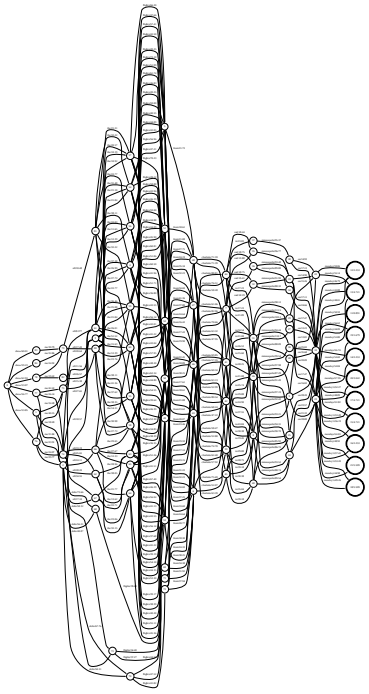

Figure 5: Word lattice W_1, ATIS task, for the utterance *Show me the flights from Charlotte to Minneapolis on Monday.*

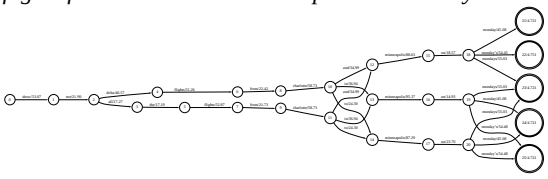

Figure 6: Equivalent word lattice W_2 obtained by determinization of W_1.

Figure 7: Equivalent word lattice W_3 obtained by minimization from W_2.

3.2. THE NAB TASK

In most recognition systems, the words in a language model are (in effect) substitued with their pronunciations to create a large phonemic network. The phonemic network created in this way can have a high degree of nondeterminism in large vocabulary systems. This remains true even when using the efficient finite-state composition techniques that lead to more compact results [7, 14], rather than simple substitution.

The phonemic network can contain states with as many (or more) outgoing arcs as the size of the vocabulary. This large number of alternatives can considerably reduce the speed of a recognition system. Weighted determinization (of weighted transducers) allows one to improve the recognition time by reducing the number of alternatives to the minimum. The number of outgoing arcs in the equivalent deterministic network is at most equal to the number of phones (about 40) versus the size of the vocabulary ($\approx 20,000$) in the original network[4].

[4] Strictly speaking, the phonemic network based on a n-gram model would have an inherent non-deterministism due to the backoff model and to fact that a word can be a prefix of another word, in particular homophones. We have solved this by treating epsilons as regular

Table 2: Use of determinization in the NAB task.

	Reduction factor
Size of network	3.3
Recognition time	91.7
Error rate	2.0
Memory used	8.8

Our experiments in the DARPA North American Business task (NAB) show that weighted determinization plays an important role in building a real-time large vocabulary speech recognition system: it reduces recognition time by a factor of 91.7, reduces the error rate in half, reduces the the size of the network by a factor of 3.3 and and the memory used by a factor of 8.8 (Table 2).

3.3. THE ONE MILLION NAMES TASK

The determinization of weighted automata also allowed us to build a *real-time* system for the recognition of the one million most frequent U.S. surnames found in the Donnelley direct marketing list. We used the frequency of the surnames to assign probabilities to the mapping of pronunciations to names. A priori, in some states such as the initial state of the pronunciation network, one million alternatives were possible and the recognition would be very slow. The use of the weighted determinization substantially reduced the number of alternatives by limiting it to the number of phones, and led to a real-time recognition system.

Table 3: Use of determinization and minimization in the One Million Names task.

	Reduction factor
Size of network	5.2
Recognition time	1000

We also used the weighted minimization algorithm after determinization in this task to reduce the size of the networks used by a factor of 5.

4. CONCLUSION

The use of weighted determinization and minimization in large vocabulary speech recognition leads to very substantial improvements in performance. In addition, it shows the true degree of redundancy in speech recognition networks, which may not have been fully appreciated previously. In our speech recognition systems we also use another algorithm, *local determinization*, that reduces the redundancy of networks only locally. We report elsewhere on the benefits of local determinization for weighted transducers [8].

The success of the use of these algorithms does not come as a surprise. While most ASR systems use *ad hoc* solutions, thereby limiting the possibility of further improvement and understanding, we believe that general algorithmic solutions based on a sound theoretical foundation can lead to substantially better results.

5. Acknowledgments

We thank Fernando Pereira, Enrico Bocchieri, and Giuseppe Riccardi for discussions, and Andrej Ljolje and Hiyan Alshawi for supplying the ATIS word lattices we used.

symbols, using word boundary symbols (a distinct word boundary for each homophone), and removing the boundary symbols removed after determinization.

6. REFERENCES

1. A. V. Aho, J. E. Hopcroft, and J. D. Ullman. *The design and analysis of computer algorithms*. Addison Wesley: Reading, MA, 1974.
2. A. V. Aho, R. Sethi, and J. D. Ullman. *Compilers, Principles, Techniques and Tools*. Addison Wesley: Reading, MA, 1986.
3. M. Mohri. Minimization of sequential transducers. *Lecture Notes in Computer Science*, 807, 1994.
4. M. Mohri. On some applications of finite-state automata theory to natural language processing. *Journal of Natural Language Engineering*, 2:1–20, 1996.
5. M. Mohri. Finite-state transducers in language and speech processing. *Computational Linguistics*, 23, 1997.
6. M. Mohri. Minimization algorithms for sequential transducers. *Theoretical Computer Science*, To appear, 1997.
7. M. Mohri, F. C. N. Pereira, and M. Riley. Weighted automata in text and speech processing. In *ECAI-96 Workshop, Budapest, Hungary*. ECAI, 1996.
8. M. Mohri, M. Riley, and R. Sproat. Finite-state transducers in language and speech processing. In *Tutorial at the 16th International Conference on Computational Linguistics (COLING-96), Copenhagen, Denmark*. COLING, 1996.
9. J. Odell, V. Valtchev, P. Woodland, and S. Young. A one pass decoder design for large vocabulary recognition. In *Proceedings of the ARPA Human Language Technology Workshop, March 1994*. Morgan Kaufmann, 1994.
10. S. Ortmanns, H. Ney, and A. Eiden. Language-model look-ahead for large vocabulary speech recognition. In *ICSLP'96*, pages 2095–2098. University of Delaware and Alfred I.duPont Institute, 1996.
11. S. Ortmanns, H. Ney, F. Seide, and I. Lindam. A comparison of time conditioned and word conditioned search techniques for large vocabulary speech recognition. In *ICSLP'96*, pages 2091–2094. University of Delaware and Alfred I.duPont Institute, 1996.
12. F. C. N. Pereira and M. Riley. *Finite State Devices in Natural Language Processing*, chapter Weighted Rational Transductions and their Application to Human Language Processing. The MIT Press, to appear, 1997.
13. D. Revuz. Minimisation of acyclic deterministic automata in linear time. *Theoretical Computer Science*, 92:181–189, 1992.
14. M. Riley, F. C. N. Pereira, and M. Mohri. Transducer composition for context-dependent network expansion. In *Proceedings of Eurospeech'97*. Rhodes, Greece, 1997.