GENERALIZED OPTIMIZATION ALGORITHM FOR SPEECH RECOGNITION TRANSDUCERS

Cyril Allauzen and Mehryar Mohri

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

ABSTRACT

Weighted transducers provide a common representation for the components of a speech recognition system. In previous work, we showed that these components can be combined off-line into a single compact recognition transducer that maps directly HMM state sequences to word sequences [11]. The construction of that recognition transducer and its efficiency of use critically depend on the use of a general optimization algorithm, *determinization*. However, not all weighted automata and transducers used in large-vocabulary speech recognition are determinizable. We present a general algorithm that can make an arbitrary weighted transducer determinizable and generalize our previous optimization technique for building an integrated recognition transducer to deal with arbitrary weighted transducers used in speech recognition. We report experimental results in a large-vocabulary speech recognition task, *How May I Help You* (HMIHY), showing that our generalized technique leads to a recognition transducer that performs as well as our original solution in the case of classical n-gram models while inserting less special symbols, and that it leads to a substantial improvement of the recognition speed, factor of 2.6, in the same task when using a class-based language model.

1. MOTIVATION

Weighted transducers are finite-state transducers in which each transition carries a weight in addition to the usual input and output symbols [14, 7]. They provide a common representation for several components of a speech recognition system: language models, pronunciation dictionaries, context-dependency and HMM models [11].

General weighted transducer algorithms can be used to combine and optimize these representations and to build off-line a single efficient recognition transducer that integrates all of these components, directly mapping from HMM state sequences to word sequences [11]. The size of that integrated recognition transducer is practical since it has been shown empirically to be close to that of the language model used.

The construction of that recognition transducer and its efficiency of use critically depend on the use of a general optimization algorithm, *determinization* [9, 12]. Determinization outputs a transducer equivalent to the input that is *deterministic*, i.e., one that has a unique initial state and that has no two transitions leaving the same state with the same input label. This considerably reduces the number of paths needed to be explored by the decoder and thus substantially improves the efficiency of recognition.

However, not all weighted automata and transducers used in speech recognition are determinizable. A solution to this problem was provided in the special case where n-gram statistical language models are used [11]. But that solution does not cover some important cases such as that of class-based language models which can lead to non-determinizable weighted transducers. Other language models created either directly or by approximation of weighted context-free grammars can also create similar non-determinism that could make our previous technique inapplicable.

We present a general algorithm that makes an arbitrary weighted transducer determinizable by inserting in it transitions labeled with special symbols just when needed and at the optimal positions to fully benefit from the application of determinization. Those special symbols can be removed, or mapped to the empty string after application of determinization. The algorithm generalizes our previous optimization technique for building an integrated recognition transducer to deal with arbitrary weighted transducers used in speech recognition.

Our experiments in a large-vocabulary speech recognition task, *How May I Help You* (HMIHY), show that our new and generalized technique leads to a recognition transducer that performs as well as our original solution in the case where classical n-gram models are used, while inserting less special symbols. We also report experiments with a class-based language model in the same task using our generalized optimization technique. The experiments show an improvement of the recognition speed by a factor of 2.6 over the system used without application of determinization.

We first introduce some preliminary definitions and notation necessary for the presentation of our symbol insertion algorithm and experimental results.

2. PRELIMINARIES

2.1. Semirings

Weighted automata are automata in which the transitions carry in addition to the usual alphabet symbols some weights elements of a *semiring* [7]. A semiring is a ring that may lack negation. It has two associative operations $\oplus$ and $\otimes$ with identity elements $\overline{0}$ and $\overline{1}$. $\otimes$ distributes over $\oplus$ and $\overline{0}$ is an annihilator. For example, $(\mathbb{N}, +, \cdot, 0, 1)$ is a semiring.

The weights used in speech recognition often represent probabilities. The appropriate semiring to use is then the *probability semiring* $(\mathbb{R}, +, \cdot, 0, 1)$. For numerical stability, implementations often replace probabilities with log probabilities. The appropriate semiring to use is then the *log semiring*: $(\mathbb{R} \cup \{\infty\}, \oplus_l, +, \infty, 0)$, with: $\forall a, b \in \mathbb{R} \cup \{\infty\}, a \oplus_l b = -\log(exp(-a) + exp(-b))$, where by convention $exp(-\infty) = 0$, and $-\log(0) = \infty$. The log semiring is the image by log of the semiring $(\mathbb{R}, +, \cdot, 0, 1)$. When log probabilities are used and a Viterbi approximation is assumed, $\oplus_l$ is replaced by $\min$ and the appropriate semiring is the *tropical semiring* $(\mathbb{R}_+ \cup \{\infty\}, \min, +, \infty, 0)$. The semiring abstraction permits a generic definition of representations and algorithms independent of the underlying algebra.

2.2. Weighted Transducers

A *weighted transducer* $T = (\Sigma, \Delta, Q, I, F, E, \lambda, \rho)$ over $\mathbb{K}$ is a 7-tuple where Σ is a finite input alphabet, Δ is a finite output alphabet, Q is a finite set of states, $I \subseteq Q$ the set of initial states, $F \subseteq Q$ the set of final states, $E \subseteq Q \times \Sigma \times \Delta \times \mathbb{K} \times Q$ a finite set of transitions, $\lambda : I \to \mathbb{K}$ the initial weight function mapping I to $\mathbb{K}$, and $\rho : F \to \mathbb{K}$ the final weight function mapping F to $\mathbb{K}$ [14, 7]. *Weighted automata* can be defined in a similar way by simply omitting the output labels.

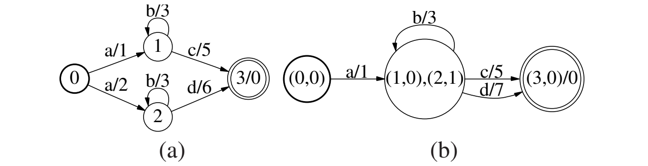

Fig. 1. Determinization of weighted automata. (a) Weighted automaton over the tropical semiring A. (b) Equivalent weighted automaton B obtained by determinization of A.

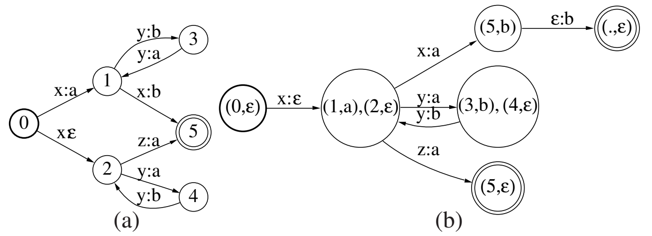

Fig. 2. Determinization of finite-state transucers. (a) Finite-State transducer T. (b) Equivalent transducer T' obtained by determinization of T.

Given a transition $e \in E$, we denote by $i[e]$ its input label, $p[e]$ its origin or previous state and $n[e]$ its destination state or next state, $w[e]$ its weight (weighted automata case), o[e] its output label (transducer case). Given a state $q \in Q$, we denote by $E[q]$ the set of transitions leaving q.

A *path* $\pi = e_1 \cdots e_k$ in A is an element of E^* with consecutive transitions: $n[e_{i-1}] = p[e_i]$, $i = 2, \ldots, k$. We extend n and p to paths by setting: $n[\pi] = n[e_k]$ and $p[\pi] = p[e_1]$. We denote by $P(q, q')$ the set of paths from q to q' and by $P(q, x, y, q')$ the set of paths from q to q' with input label $x \in \Sigma^*$ and output $y \in \Delta^*$. These definitions can be extended to subsets $R, R' \subseteq Q$, by: $P(R, x, y, R') = \bigcup_{q \in R,\, q' \in R'} P(q, x, y, q')$. The labeling functions i (and similarly o) and the weight function w can also be extended to paths by defining the label of a path as the concatenation of the labels of its constituent transitions, and the weight of a path as the $\otimes$-product of the weights of its constituent transitions: $i[\pi] = i[e_1] \cdots i[e_k]$, $w[\pi] = w[e_1] \otimes \cdots \otimes w[e_k]$. We also extend w to any finite set of paths Π by setting: $w[\Pi] = \bigoplus_{\pi \in \Pi} w[\pi]$. A transducer is *functional* or *single-valued* if it associates at most one string to any input string x.

A *successful path* in a weighted transducer T is a path from an initial state to a final state. A state q of T is *accessible* if q can be reached from I. It is *coaccessible* if a final state can be reached from q. A weighted transducer T is *trim* if there is no transition with weight $\overline{0}$ in T and if all states of T are both accessible and coaccessible. T is *cycle-unambiguous* if for any state q and any string x there is at most one cycle in q labeled with x. T is *unambiguous* if for any string $x \in \Sigma^*$ there is at most one successful path with input label x. Thus, an unambiguous transducer is functional.

2.3. Elements of Combinatorics on Words

We briefly introduce some elementary notions of combinatorics on words [13] necessary for the definitions given in the next section. Given two strings x and y in Σ^*, we say that y is a *suffix* of x if there exists $z \in \Sigma^*$ such that $x = zy$ and similarly that y is a *prefix* of x if there exists z such that $x = yz$. We extend the alphabet Σ by associating to each symbol $a \in \Sigma$ a new symbol denoted by a^{-1} and define Σ^{-1} as: $\Sigma^{-1} = \{a^{-1} : a \in \Sigma\}$. $X = (\Sigma \cup \Sigma^{-1})^*$ is then the set of strings written over the alphabet $(\Sigma \cup \Sigma^{-1})$. If we impose that $aa^{-1} = a^{-1}a = \epsilon$, then X forms a group called the *free group generated by* Σ and is denoted by $\Sigma^{(*)}$. Note that the inverse of a string $x = a_1 \cdots a_n$ is then simply $x^{-1} = a_n^{-1} \cdots a_1^{-1}$.

3. ALGORITHMS

Our main symbol insertion algorithm is based on an efficient algorithm for testing the *twins property* used to check determinizability [2]. This section first briefly describes and illustrates the determinization algorithm for weighted automata and transducers. It then describes the twins property and our symbol insertion algorithm for making transducers determinizable.

3.1. Determinization

A weighted automaton or transducer is said to be *deterministic* if it has a unique state and if no two transitions leaving the same state have the same input label. There exists a general determinization algorithm for weighted automata and transducers [8, 9]. The algorithm is a generalization of the classical subset construction [1].

Figure 1 illustrates the determinization of a weighted automaton. The states of the output weighted automaton correspond to *weighted subsets* of the type $\{(q_0, w_0), \ldots, (q_n, w_n)\}$ where each $q_k \in Q$ is a state of the input machine, and w_k a remainder weight. The algorithm starts with the subset reduced to $\{(p, 0)\}$ where p is an initial state and proceeds by creating a transition labeled with $a \in \Sigma$ and weight w leaving $\{(q_0, w_0), \ldots, (q_n, w_n)\}$ if there exists at least one state q_k admitting an outgoing transition labeled with a, w being defined by: $w = \min\{w_k + w[e] : e \in E[q_k],\ i[e] = a\}$.

Similarly, figure 2 illustrates the determinization of a finite-state transducer. Here, the states of the resulting transducer are *string subsets* of the type $\{(q_0, x_0), \ldots, (q_n, x_n)\}$, where each $q_k \in Q$ is a state of the input machine, and x_k a remainder string. We refer the reader to [8, 9] for a more detailed presentation of these algorithms.

Unlike the unweighted automata case, not all weighted automata or finite-state transducers are *determinizable*, that is the determinization algorithm does not halt with some inputs. Figure 3 (a) shows an example of a non-determinizable weighted automaton and figure 3 (b) a non-determinizable finite-state transducer. Note that the automaton of figure 3 (a) differs from that of figure 1 only by the weight of the self-loop at state 2. The difference between that weight and that of the similar loop at state 1 is the cause of the non-determinizability. There exists a general characterization property and an efficient testing algorithm for checking that property in very general cases [9, 2].

3.2. The twins property

The algorithm for testing the determinizability of finite-state transducers is based on a *twins property* introduced by [5, 6, 4]. The twins property provides a characterization of determinizable finite-state transducers [3]. The definition of the twins property and the charaterization results were extended by [9] to the case of cycle-unambiguous weighted automata.

Two states q and q' are said to be *siblings* when they can be reached from the initial states I by paths sharing the same input label and when there exists a cycle at q and a cycle at q' labeled with the same input. For example, states 1 and 2 are siblings in the automata of Figure 1 (a) and figure 3 (a) since they can be reached from the initial state 0 by paths labeled with a and admit cycles with the same input label b. Similarly, states 1 and 2, and states 3 and 4 are siblings in the transducer of figure 2 (a) and that of figure 3 (b).

Two sibling states q and q' of a weighted finite-state transducer are said to be *twins* if the two following conditions:

$$o[\pi]^{-1} o[\pi'] = o[\pi c]^{-1} o[\pi' c'] \quad (1)$$
$$w[c] = w[c'] \quad (2)$$

hold for any paths π from I to q and π' from I to q' and for any

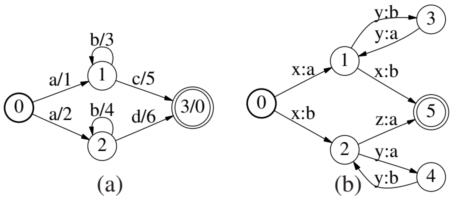

Fig. 3. Non-determinizable cases. In both examples, states 1 and 2 are non-twin siblings. (a) Non-determinizable weighted automaton defined over the tropical semiring. (b) Non-determinizable functional transducer.

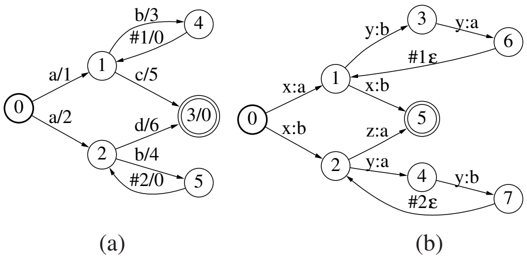

Fig. 4. Insertion of special symbols to guarantee determinizability. The insertion of the new transition labeled with $\#_1$ or the one labeled with $\#_2$ is sufficient to guarantee the determinizability of: (a) The automaton of figure 4 (a). (b) The finite-state transducer of figure 4 (b).

cycles c in q and c' in q' such that $i[\pi] = i[\pi']$ and $i[c] = i[c']$. T is said to have the *twins property* if any two siblings in T are twins. Note that in that definition q can be equal to q' and that we may have $\pi = \pi'$ or $c = c'$, or that π or π' can be the empty path if q, or q', is the initial state.

In the case of a weighted automaton, only the condition on the equality of the cycle weights (condition 2) is required, and in the case of an unweighted transducer, only the condition on the output labels (condition 1).

The twins property is a sufficient condition for the determinizability of weighted automata or weighted transducers over the tropical semiring [5, 9]. It is a necessary and sufficient condition for the determinizability of unweighted transducers and that of unambiguous weighted automata or weighted transducers over the tropical semiring [9, 2].

3.3. Symbol Insertion

We have designed an efficient algorithm for testing the twins property for weighted automata and transducers [2]. The algorithm can be used to detect non-twin sibling states such as those of figures 3 (a)-(b) and to locate the transitions that cause non-determinizability.

The key idea behind our algorithm for making an arbitrary transducer determinizable is to insert transitions labeled with new symbols along the paths corresponding to non-twin sibling states. The definition of the twins property is based on the existence of two distinct paths sharing the same input label. Thus, when the conditions of the twins property, equations 1 and 2, do not hold for these paths, we can insert a transition labeled with a special symbol not in Σ along one of these paths. The paths are then labeled with distinct input labels, which ensures that the twins property holds and guarantees the determinizability of the transducer.

Figures 4 (a)-(b) illustrate this symbol insertion. The weighted automaton of figure 3 (a) does not have the twins property because of the two cycles with input label b at states 1 and 2 which have different weights. These are detected by our algorithm for testing the twins property. If we insert a transition with input label $\#_1$ in the cycle at state 1 or a new transition with input label $\#_2$ in the cycle at state 2, the cycles have different labels and the resulting

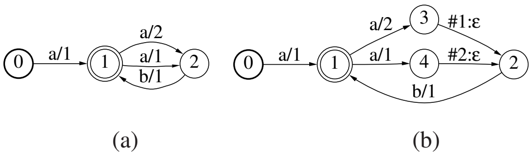

Fig. 5. (a) Weighted automaton wihout the twins property: state 1 is a non-twin sibling with itself because of the cycles at state 1 with the same input label ab but with distinct weights 2 and 3. (b) The farthest possible positions for symbol insertion are indicated.

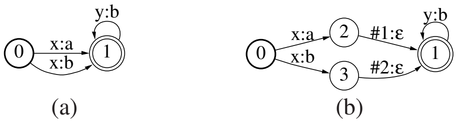

Fig. 6. (a) Non-determinizable finite-state transducer. State 1 is a non-twin sibling with itself because there are two paths π and π' from the initial state to state 1 with the same input label x and distinct outputs a and b, and because there is a cycle c at state 1 that violates condition 1 of the twins property: $o[\pi]^{-1}o[\pi'] = a^{-1}b \neq o[\pi c]^{-1}o[\pi' c] = (ab)^{-1}bb = b^{-1}a^{-1}bb$. (b) The only possible positions where new symbols can be inserted along π or π' are indicated in the figure.

automaton has the twins property. The finite-state transducer of figure 3 (b) can be made determinizable in the same way.

The examples show that there are some degrees of freedom in the choice of the position at which a new symbol is inserted: it can be inserted on either one of the cycles and at any position along that cycle. Our algorithm inserts new symbols only when needed to ensure that the conditions of the twins property hold. The position at which a transition labeled with a new symbol is inserted is chosen carefully so that we can merge the longest possible prefixes during the application of determinization and thereby benefit the most from this optimization for decoding and other purposes. To do so, we compute the level of the transitions where a new symbol can be inserted with respect to the output of determinization and choose to insert the symbol at the transition with the highest level.

It may appear at first sight that the best position for inserting a new symbol is the end of a cycle as in figures 4 (a)-(b), however, this is not always possible since two cycles may share that last transition. Figure 5 illustrates that situation. Similarly, in the case of transducers, the cycles may coincide, thus the new symbol cannot be inserted along the cycles. It can be inserted on the paths with same input label that lead to the state $q = q'$, see figure 6.

In all cases, the new symbols inserted can be removed or replaced by the empty string after determinization to preserve the weighted transduction defined by the original transducer.

3.4. Optimization of Recognition Transducers

In previous work, we showed that a single recognition transducer T can be built from a weighted automaton G representing an n-gram language model, a transducer L representing a pronunciation dictionary, a context-dependency transducer C and an HMM transducer H mapping sequences of distribution indices to context-dependent phones [11]. Here, we generalize that construction to the case of arbitrary transducers.

In general, the pronunciation transducer L is not determinizable both because of the presence of homonyms and because the pronunciation of a single word may coincide with that of a sequence of words. The language model G may also be non-determinizable, this is in fact typical in the case of class-based language models. Other transducer components may be non-determinizable as well. But this is no longer a problem with our generalized algorithm.

Using the symbol insertion algorithm described in the previous section, we can make an arbitrary language model automaton G determinizable without any prior knowledge about it. We denote

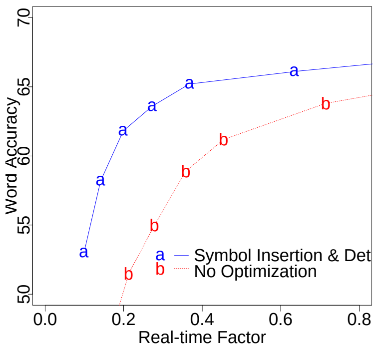

Fig. 7. Comparison of the technique based on symbol insertion and determinization versus no optimization in the 5,500-word vocabulary HMIHY 0300 task with a class-based language model.

by $\hat{G}$ the result of the application of our symbol insertion algorithm to G, and by $\hat{L}$ the result after application to L. For any transducer M, we also denote by $I(M)$ the transducer M augmented with extra transitions such that each new symbol on its input side is mapped to some new and distinct output symbol. Then, our generalized recognition transducer T is constructed according to the following formula:

$$\hat{T} = det(I(H) \circ det(I(C) \circ det(I(\hat{L}) \circ \hat{G})))$$

where $\circ$ denotes the composition [10] and det the determinization of weighted transducers [9]. The recognition transducer T is obtained by replacing the new symbols inserted with the empty string. This construction is similar to our previous construction in the specific case of n-gram language models [11]. We use composition to combine the transducer components and apply determinization after each composition step. But our symbol insertion algorithm based on the twins property generalizes this construction to the case of an arbitrary language model automaton.

4. EXPERIMENTAL RESULTS

We used the technique outlined in the previous section to construct integrated optimized recognition transducers for a 5,500-word vocabulary HMIHY 0300 task.

We first compared our new construction technique to the one we had introduced for classical n-gram language models [11] using exactly the same recognition components. Our experiment showed that the performance of the new construction matches exactly that of the old one in that case: the plots showing accuracy versus real-time factor with the old and new construction coincide. However, with the new construction, we are inserting 65% less symbols in the lexicon transducer L.

We also measured the improvement of our new construction technique over the construction of the integrated transducer with no optimization in the same task. The same components were used in both cases: a class-based trigram katz's backoff language model shrunk with an epsilon of 0 using the method of Seymore and Rosenfeld [15] based on a class of 13 sentences and 42 words – note that our previous optimization technique could not be used with these non-determinizable class-based language models; an acoustic model with 4,652 distinct HMM states, each associated to a four-Gaussian mixture model; a triphonic context-dependency transducer C with 1,333 states and 10,2641 transitions.

Our experiments with the HMIHY 0300 5,500-word vocabulary task using a simple general-purpose one-pass Viterbi decoder show that the integrated optimized recognition transducer T substantially speeds up recognition when using that class-based language model. Figure 7 gives recognition accuracy as a function of recognition time, in multiples of real-time on a single processor of a 1GHz Intel Pentium III linux cluster with 256 KB of cache and 2 GB of memory. With our new construction method, the accuracy achieved by the old non-optimized integrated transducer at .4 times real-time is reached by the new system using our optimization at about .15 times real-time, that is more than 2.6 times faster.

5. CONCLUSION

A general algorithm was presented that can make an arbitrary weighted transducer determinizable by inserting special symbols just when needed and at the most favorable positions for the application of determinization. The algorithm generalizes our previous optimization technique for building an integrated recognition transducer to deal with arbitrary weighted transducers in speech recognition [11] and was used to substantially improve the recognition speed in a task where class-based language models are used. The algorithm is general, it can be used in any other application where weighted automata and transducers are used, in particular other speech processing applications such as speech synthesis.

6. ACKNOWLEDGEMENTS

We thank Brian Roark for providing the class-based language models used in our experiments and Michael Riley for discussion and help with the experiments.

7. REFERENCES

[1] A. V. Aho, R. Sethi, and J. D. Ullman. *Compilers, Principles, Techniques and Tools*. Addison Wesley: Reading, MA, 1986.

[2] C. Allauzen and M. Mohri. Efficient Algorithms for Testing the Twins Property. *Journal of Automata, Languages and Combinatorics*, to appear, 2002.

[3] C. Allauzen and M. Mohri. p-Subsequential Transducers. In Jean-Marc Champarnaud and Denis Maurel, editor, *Seventh International Conference, CIAA 2002*, volume to appear of *Lecture Notes in Computer Science*, Tours, France, July 2002. Springer-Verlag, Berlin-NY.

[4] J. Berstel. *Transductions and Context-Free Languages*. Teubner Studienbucher: Stuttgart, 1979.

[5] C. Choffrut. Une caractérisation des fonctions séquentielles et des fonctions sous-séquentielles en tant que relations rationnelles. *Theoretical Computer Science*, 5:325–338, 1977.

[6] C. Choffrut. *Contributions à l'étude de quelques familles remarquables de fonctions rationnelles*. PhD thesis, (thèse de doctorat d'Etat), Université Paris 7, LITP: Paris, France, 1978.

[7] W. Kuich and A. Salomaa. *Semirings, Automata, Languages*. Number 5 in EATCS Monographs on Theoretical Computer Science. Springer-Verlag, Berlin, Germany, 1986.

[8] M. Mohri. On some Applications of Finite-State Automata Theory to Natural Language Processing. *Journal of Natural Language Engineering*, 2:1–20, 1996.

[9] M. Mohri. Finite-State Transducers in Language and Speech Processing. *Computational Linguistics*, 23(2), 1997.

[10] M. Mohri, F. C. N. Pereira, and M. Riley. Weighted Automata in Text and Speech Processing. In *Proceedings of the 12th biennial European Conference on Artificial Intelligence (ECAI-96), Workshop on Extended finite state models of language, Budapest, Hungary*. ECAI, 1996.

[11] M. Mohri and M. Riley. Integrated Context-Dependent Networks in Very Large Vocabulary Speech Recognition. In *Proceedings of the 6th European Conference on Speech Communication and Technology (Eurospeech '99)*, Budapest, Hungary, 1999.

[12] M. Mohri and M. Riley. Network Optimizations for Large Vocabulary Speech Recognition. *Speech Communication*, 28(1):1–12, 1999.

[13] D. Perrin. Words. In M. Lothaire, editor, *Combinatorics on words*, Cambridge Mathematical Library. Cambridge University Press, 1997.

[14] A. Salomaa and M. Soittola. *Automata-Theoretic Aspects of Formal Power Series*. Springer-Verlag: New York, 1978.

[15] K. Seymore and R. Rosenfeld. Scalable backoff language models. In *Proceedings of ICSLP*, Philadelphia, Pennsylvania, 1996.