A Weight Pushing Algorithm for Large Vocabulary Speech Recognition

Mehryar Mohri, Michael Riley

AT&T Labs - Research
180 Park Avenue,
Florham Park, NJ 07932, USA
{mohri,riley}@research.att.com

Abstract

Weighted finite-state transducers provide a general framework for the representation of the components of speech recognition systems; language models, pronunciation dictionaries, context-dependent models, HMM-level acoustic models, and the output word or phone lattices can all be represented by weighted automata and transducers. In general, a representation is not unique and there may be different weighted transducers realizing the same mapping. In particular, even when they have exactly the same topology with the same input and output labels, two equivalent transducers may differ by the way the weights are distributed along each path.

We present a *weight pushing* algorithm that modifies the weights of a given weighted transducer in a way such that the transition probabilities form a stochastic distribution. This results in an equivalent transducer whose weight distribution is more suitable for pruning and speech recognition. We demonstrate substantial improvements of the speed of our recognition system in several tasks based on the use of this algorithm. We report a 45% speedup at 83% word accuracy with a simple single-pass $40,000$-word vocabulary North American Business News (NAB) recognition system on the DARPA Eval '95 test set. With the same technique, we report a 550% speedup at 88% word accuracy in rescoring NAB word lattices with more accurate 2nd-pass models. We finally report a 280% speedup at 68% word accuracy for $100,000$ first name-last name pairs recognition.

1. Introduction

Finite-state transducers are used in many areas of text and speech processing. A transducer is a finite-state device that encodes a mapping between input and output symbol sequences; a *weighted* transducer associates weights such as log probabilities, durations, penalties, or any other quantity that accumulates linearly along paths, to each pair of input and output symbol sequences [1, 2].

In earlier works, we have shown that weighted finite-state transducers provide a common and natural representation for HMMs, context-dependency, pronunciation dictionaries, and language models [3, 4, 5] and described general automata operations for combining these representations flexibly and efficiently [6]. These operations can be used to build a single transducer that integrates these components, directly mapping from HMM states to words [3].

We have previously described weighted determinization and minimization which optimize the time and space requirements of a transducer [7, 8] and reported their benefits for speech recognition [9, 7]. Once an automaton representing a particular transduction is so optimized, it is uniquely determined up to state renumbering and to any weight and output label redistribution that preserves the total path weights and output strings. Thus, the weight and output label distribution are the only degrees of freedom left in the transducer that can be optimized for a given use. For some applications, the weight and output label distribution is irrelevant, since it is only the total path weight and output strings that matters. However, for speech recognition, the weight distribution is paramount, since all practical automatic speech recognition systems employ pruning during recognition and the pruning is characteristically based on the combined weight from the acoustic, duration, pronunciation, and language model components accumulated so far along an explored path.

In this paper, we propose that the weights should be *pushed* as far as possible to the initial state in a way such that the sum of the probabilities for all transitions leaving a state is 1. This makes the transducer *stochastic*: the weight on individual transitions has the form of (log) transition probabilities in a Markov chain [10]. [1]

We first introduce the classical definitions of semirings and automata. We then develop pushing in detail, presenting a general algorithm that works over a large class of semirings. Finally, we give experimental results from the application of weight pushing to various speech recognition tasks and show its benefits.

2. Preliminaries

2.1. Semirings

The definitions in this section are given in the most general case and depend on the algebraic structure of a *semiring*, $(\mathbb{K}, \oplus, \otimes, \overline{0}, \overline{1})$ [1]. A semiring is a ring that may lack negation. It has two associative operations $\oplus$ and $\otimes$ that are closed over the set $\mathbb{K}$, they have identities $\overline{0}$ and $\overline{1}$, respectively. $\otimes$ 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 image by log of the semiring $(\mathbb{R}, +, \cdot, 0, 1)$ and is called 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 $-\log(0) = \infty$. 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. In the following, unless otherwise specified, we assume that an arbitrary semiring $(\mathbb{K}, \oplus, \otimes, \overline{0}, \overline{1})$ is given.

2.2. Weighted Acceptors and Transducers

Finite-state devices such as HMMs used in speech recognition are special cases of *weighted finite-state acceptors* (WFSAs). A WFSA $A = (\Sigma, Q, E, i, F, \lambda, \rho)$ over the semiring $\mathbb{K}$ is given by an alphabet or label set Σ, a finite set of states Q, a finite set of *transitions* $E \subseteq Q \times (\Sigma \cup \{\epsilon\}) \times \mathbb{K} \times Q$, an *initial state* $i \in Q$, a set of *final states* $F \subseteq Q$, an *initial weight* λ and a *final weight function* ρ.

A transition $t = (p[t], \ell[t], w[t], n[t]) \in E$ can be represented by an arc from the *source state* $p[t]$ to the *destination state* $n[t]$, with the *label* $\ell[t]$ and *weight* $w[t]$. In speech recognition, the transition weight $w[t]$ often represents a probability

[1] The overall probability mass in the transducer, which may not sum to 1, is set as the initial weight.

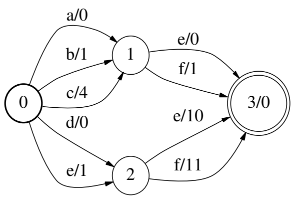

Figure 1: Weighted acceptor A_1.

or a log probability. For each state $q \in Q$, $E[q]$ denotes the set of transitions leaving q.

A path in A is a sequence of consecutive transitions $t_1 \cdots t_n$ with $n[t_i] = p[t_{i+1}]$, $i = 1, \ldots, n-1$. Transitions labeled with the *empty string* ϵ consume no input. A *successful path* $\pi = t_1 \cdots t_n$ is a path from the initial state i to a final state $f \in F$. The label of the path π is the result of the concatenation of the labels of its constituent transitions: $\ell[\pi] = \ell[t_1] \cdots \ell[t_n]$. The weight associated to π is the $\otimes$-product of the initial weight, the weights of its constituent transitions and the final weight $\rho(n[t_n])$ of the state reached by π:

$$w[\pi] = \lambda \otimes w[t_1] \otimes \cdots \otimes w[t_n] \otimes \rho(n[t_n])$$

A symbol sequence x is accepted by A if there exists at least one successful path π labeled with x: $\ell[\pi] = x$. The weight associated by A to the sequence x is then the $\oplus$-sum of the weights of the successful paths π labeled with x. Thus, a WFSA provides a mapping from symbol sequences to weights [11, 1, 2].

Weighted finite-state transducers (WFSTs) generalize WFSAs by replacing the single transition label by a pair (i, o) of an input label i and an output label o. A weighted transducer associates associates pairs of symbol sequences and weights, that is, it represents a weighted binary relation between symbol sequences [2, 11, 1].

Two weighted acceptors are *equivalent* if they associate the same weight to each input string; weights may be distributed differently along the paths of two equivalent acceptors. Two weighted transducers are equivalent if they associate the same output sequence and weights to each input sequence; the distribution of the weight or output labels along paths need not be the same in the two transducers.

For simplicity, our discussion of weight pushing in the next section will be illustrated with weighted acceptors. The more general weighted transducer case can be shown to be equivalent to this case by interpreting weight-output label pairs as new *weights* combined by the appropriate semiring [7].

3. Algorithm

3.1. Weight Pushing

We have already presented elsewhere weighted determinization and minimization of transducers [7, 8]. A weighted transducer is *sequential* or *deterministic* if and only if each of its states has at most one transition with any given input label. It can be minimized using a general weighted minimization algorithm.

The transition weights of a weighted automaton A over a field can be changed without modifying path weights. Indeed, let $V : Q \rightarrow \mathbb{K} - \{\overline{0}\}$ be an arbitrary function, called a *potential* function on states. Update the initial weight, the transition weights and the final weights by:

$$\lambda \leftarrow \lambda \otimes V(i) \quad (1)$$

$$\forall e \in E, w[e] \leftarrow [V(p[e])]^{-1} \otimes (w[e] \otimes V(n[e])) \quad (2)$$

$$\forall f \in F, \rho(f) \leftarrow [V(f)]^{-1} \otimes \rho[f] \quad (3)$$

It is easy to see that this reweighting does not affect the total weight of a successful path and that the resulting automaton is equivalent to the original since the potentials along any successful path cancel each other.

Weight pushing consists of reweighting an automaton with the potential V defined in such a way that for each state $q \in Q$,

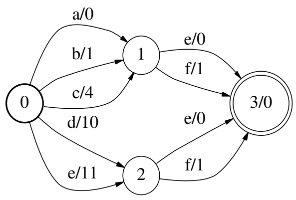

Figure 2: Weighted automaton A_2 obtained from A_1 by pushing weights in the tropical semiring.

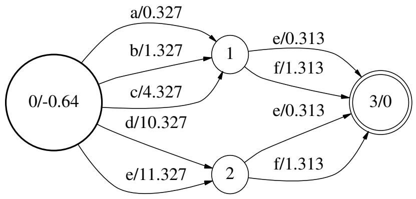

Figure 3: Weighted automaton A_3 obtained from A_1 by pushing weights in the log semiring.

$V(q) = d[q]$, the *shortest distance* from q to the final states F defined by:

$$d[q] = \bigoplus_{\pi \in P(q)} w[\pi]$$

where $P(q)$ represents the set of all paths from q to F. To extend weight pushing to the case of semirings, we assume that the semiring $\mathbb{K}$ is *divisible* [12], that is for any $a, b \in \mathbb{K}$ such that $a \oplus b \neq \overline{0}$ there exists $a_1 \in \mathbb{K}$ such that:

$$a = (a \oplus b) \otimes a_1$$

When $\mathbb{K}$ is divisible, for each pair (a, b) such that $a \oplus b \neq \overline{0}$, we select a fixed element a_1 satisfying this equation. We say that a_1 is *the remainder of the division of* a *by* $a \oplus b$ and write:

$$a_1 = (a \oplus b)^{-1} \otimes a$$

Figure 2 shows the result of pushing for the input automaton A_1 of figure 1 defined over the tropical semiring. As can be seen from figure 2, the shortest path from each state to a final state has weight zero, this is a general consequence of pushing in the tropical semiring. Figure 3 shows the result of pushing in the log semiring applied to A_1. At each state, the probability weights of outgoing transitions sum to one. Thanks to pushing, the size of the automata A_2 and A_3 can be reduced using classical minimization.

One key step that has yet to be described is how to compute the potential function. In the case of the tropical semiring, the shortest distance from each state to final states can be computed efficiently using classical single-source shortest-paths algorithms [13].

In the probability semiring or the log semiring, under certain general conditions, an extended version of the Floyd-Warshall algorithm can be proved to effectively compute shortest distances [12]. Unfortunately, this later algorithm has time complexity $O(|Q|^2)$ and space complexity $O(|Q|^3)$. This makes its application to the large transducers in speech recognition impossible in practice. We have devised a generic single-source shortest-distance algorithm that works with any *k-closed* semiring [12], that is any semiring for which there exists k such that:

$$\forall a \in \mathbb{K}, \bigoplus_{n=0}^{k+1} a^n = \bigoplus_{n=0}^{k} a^n$$

It is easy to verify that the tropical semiring is 0-closed. The log semiring is not k-closed for any $k \in \mathbb{N}$. However, it can be proved that if the equality tests in our algorithm are replaced by approximate equalities modulo a small number ϵ, then our algorithm can also cover the case of the log semiring by producing ϵ-approximations of the correct shortest distances. Smaller ϵ's result in better approximations and in practice, the resulting approximation is in the order of or less than that of the approximations already made in speech recognition. In the case of the log semiring, our algorithm is several orders of magnitude faster than the extension of the Floyd-Warshall algorithm and is very practical even for very large recognition transducers. In the case of tropical semiring, our algorithm coincides with classical shortest-distance algorithms.

3.2. Pseudocode

GENERIC-SINGLE-SOURCE-SHORTEST-DISTANCE (A)

1 **for** $j \leftarrow 1$ **to** $|Q|$
2   **do** $d[j] \leftarrow r[j] \leftarrow \overline{0}$
3 $d[i] \leftarrow r[i] \leftarrow \overline{1}$
4 $S \leftarrow \{i\}$
5 **while** $S \neq \emptyset$
6   **do** $q \leftarrow head(S)$
7   DEQUEUE(S)
8         $R \leftarrow r[q]$
9         $r[q] \leftarrow \overline{0}$
10   **for** each $e \in E[q]$
11   **do if** $d[n[e]] \neq d[n[e]] \oplus (R \otimes w[e])$
12   **then** $d[n[e]] \leftarrow d[n[e]] \oplus (R \otimes w[e])$
13                     $r[n[e]] \leftarrow r[n[e]] \oplus (R \otimes w[e])$
14   **if** $n[e] \notin S$
15   **then** ENQUEUE($S, n[e]$)
16 $d[i] \leftarrow \overline{1}$

Figure 4: Generic single-source shortest-distance algorithm.

Figure 4 gives the pseudocode of the algorithm. We use a queue S to maintain the set of vertices whose leaving edges are to be *relaxed*. S is initialized to $\{i\}$ (line 4). For each vertex $q \in Q$, we maintain two attributes: $d[q] \in \mathbb{K}$ an estimate of the shortest distance from i to q, and $r[q] \in \mathbb{K}$ the total weight added to $d[q]$ since the last time q was extracted from S. Lines $1-3$ initialize arrays d and r. After initialization, $d[q] = r[q] = \overline{0}$ for $q \in Q - \{i\}$, and $d[i] = r[i] = \overline{1}$.

Given a vertex $q \in Q$ and an edge $e \in E[q]$, a relaxation step on e is performed by lines 11-13 of the pseudocode. Each time through the **while** loop of lines 5-15, a vertex q is extracted from S (lines 6-7). The value of $r[q]$ just after extraction of q is stored in R, and then $r[q]$ is set to $\overline{0}$ (lines 8-9). Lines 11-13 relax each edge leaving q. If the tentative shortest distance $d[n[e]]$ is updated during the relaxation and if $n[e]$ is not already in S, the vertex $n[e]$ is inserted in S so that its leaving edges are later relaxed (lines 14-15). $r[n[e]]$ is updated whenever $d[n[e]]$ is, to keep track of the total weight added to $d[n[e]]$ since $n[e]$ was last extracted from S or since the time after initialization if $n[e]$ has never been extracted from S. Finally, line 16 resets the value of $d[i]$ to $\overline{1}$. The algorithm works with any queue discipline chosen for S.

Interestingly, it can be proved that using either tropical or the log semiring pushing in the minimization step results in equivalent machines with the same number of states and transitions. What differs, often radically, is how the weights are distributed along a path. We will demonstrate in the next section that pushing in the log semiring benefits speech recognition pruning while using the tropical semiring can, in fact, be harmful in some cases.

4. Experiments and Results

We now describe several applications of weight pushing to speech recognition. In particular, we describe experiments that show that pushing over the log semiring makes pruning more effective. We conjecture that this is because during pruning the acoustic likelihoods and the automata probabilities representing the language model and (possibly) pronunciation model probabilities are now *synchronized* to obtain the optimal likelihood ratio test for sequential decisions. We further conjecture that

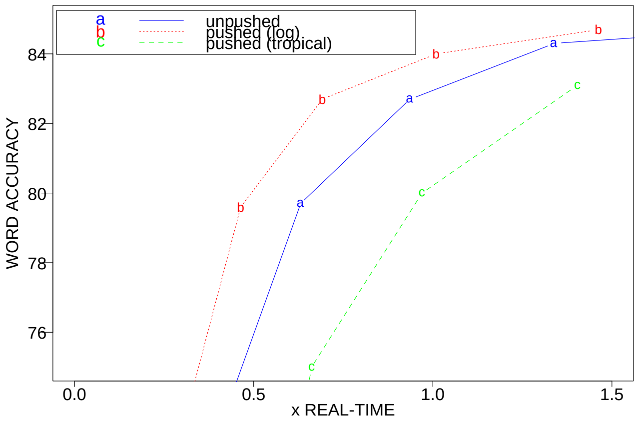

Figure 5: 40K-word NAB 1st-pass recognition, the effect of weight pushing algorithm [14].

this reweighting is the best possible for pruning. A proof of these conjectures would, however, require careful mathematical analysis of pruning. Note the asymptotic (very wide beam) accuracy for each of the experiments below will be identical in the pushed and unpushed cases (although this may not be obvious from the plots), only the real-time performance differs.

4.1. 40K-word NAB First-Pass Recognition

We applied the techniques outlined in the previous sections and in [14] to build an integrated, optimized recognition transducer for a $40,000$-word vocabulary North American Business News (NAB) task. We used a 4-million-transition trigram language model automaton and a 4-Gaussians-per-mixture triphonic acoustic model. We describe the models used in this experiment in detail in Section 3 of [14], which we do not repeat here to conserve space. We used these automata in a simple, general-purpose, one-pass Viterbi decoder applied to the DARPA NAB Eval '95 test set. Figure 5 shows the speed of recognition on a Compaq Alpha 21284 processor with and without weight-pushing the 6,108,907-transition determinized and factored HMM-to-word transducer. In particular, we see that pushing the transducer weights over the log semiring gives a substantial speedup – 45%, for example, at 83% word accuracy. However, if we push over the tropical semiring, performance is significantly degraded.

4.2. 160K-word NAB Rescoring Recognition

We have also tested weight-pushing in lattice rescoring for a $160,000$-word vocabulary NAB task. We used a 40-million-transition 6-gram language model automaton and an MLLR-adapted, 12-Gaussians-per-mixture triphonic acoustic model to rescore lattices generated in a 1st-pass using models similar to the previous section. The models used and the rescoring paradigm are precisely those described in detail in Section 3 of [14], which we do no repeat here to conserve space.[2]

Figure 6 shows the speed of second-pass recognition on a Compaq Alpha 21284 processor with and without weight-pushing with determinized HMM-to-word transducer lattices on the DARPA Eval '95 test set.[3] We see that pushing the recognition transducers over the log semiring very substantially speeds up recognition – 550%, for example, at 88% word accuracy. In this case, pushing over the tropical semiring also improves performance almost as much.

4.3. 100K Names Recognition

As a final test, we applied the weight-pushing algorithm to a 100K first-name, last name recognition system. The system details are described in [15], which we do not repeat to con-

[2] The number of Gaussians per mixture was incorrectly stated as four not twelve in this experiment in [14].

[3] The recognition speed excludes the offline automata construction time.

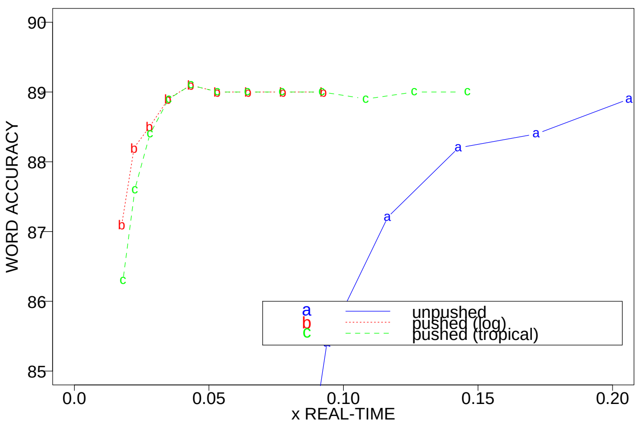

Figure 6: 160K-word NAB lattice rescoring recognition, the effect of weight pushing algorithm [14].

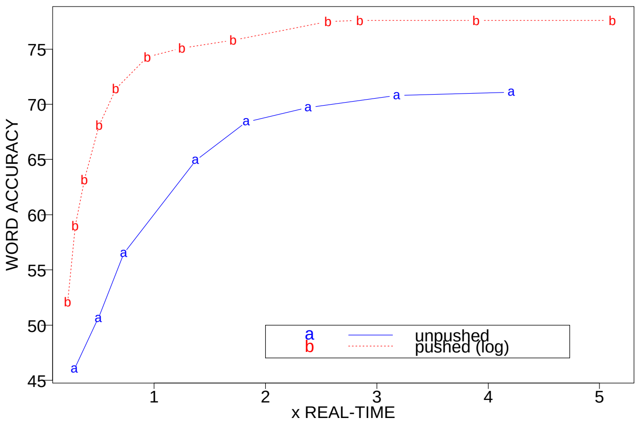

Figure 7: 100K names recognition, the effect of weight pushing algorithm [15].

serve space. The names grammar is unweighted, meaning each name is given the same prior likelihood. What does pushing the weights of the determinized HMM-to-word transducer do in this case? Consider, for example, that without weight pushing, the first HMM label of each name is weighted equally, while after pushing over the log semiring, these HMM labels will be weighted according to their frequency of starting a name. Figure 7 shows that weight pushing over the log semiring substantially speeds up recognition – 220%, for example, at 68% word accuracy. In this unweighted case, pushing over the tropical semiring will have identical performance to the unpushed case.

5. Conclusion

The combination of determinization, minimization and weight-pushing *standardizes* a transduction's representation since a deterministic, minimal, pushed transducer implementing that transduction is unique up to state renumbering. Therefore, if one accepts that these are desired properties of a transducer, then our methods compute the *optimal* choice among all equivalent transducers.

However, not all transductions admit this standardization. For example, ambiguous transductions can not be determinized. We have discussed elsewhere how we ensure that the transducers we use in speech recognition can be determinized [14]. As another example, a language model transducer originally stochastic may not be pushable over the probability semiring when free (probability $= 1.0$) silence-loops are added at each of its states because of the excess probability mass carried by the silences. The use of word insertion penalties often more than compensates for this problem and make the machine pushable. A more accurate solution consists of suitably including silences in the stochastic estimation of the language or pronunciation model itself.

6. Acknowledgments

We thank Andrej Ljolje for providing the acoustic models, Don Hindle and Richard Sproat for providing the language models used in our experiments, and Fernando C. N. Pereira for various discussions about this work.

7. References

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

[2] Arto Salomaa and Matti Soittola, *Automata-Theoretic Aspects of Formal Power Series*, Springer-Verlag: New York, 1978.

[3] Mehryar Mohri and Michael 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.

[4] Mehryar Mohri, Michael Riley, Don Hindle, Andrej Ljolje, and Fernando C. N. Pereira, "Full Expansion of Context-Dependent Networks in Large Vocabulary Speech Recognition," in *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP '98)*, Seattle, Washington, 1998.

[5] Fernando C. N. Pereira and Michael Riley, *Finite State Language Processing*, chapter Weighted Rational Transductions and their Application to Human Language Processing, The MIT Press, 1997.

[6] Mehryar Mohri, Fernando C. N. Pereira, and Michael Riley, "The design principles of a weighted finite-state transducer library," *Theoretical Computer Science*, vol. 231, pp. 17–32, January 2000.

[7] Mehryar Mohri, "Finite-state transducers in language and speech processing," *Computational Linguistics*, vol. 23:2, 1997.

[8] Mehryar Mohri, "Minimization algorithms for sequential transducers," *Theoretical Computer Science*, vol. 234, pp. 177–201, March 2000.

[9] Mehryar Mohri and Michael Riley, "Network optimizations for large vocabulary speech recognition," *Speech Communication*, vol. 25:3, 1998.

[10] J. W. Carlyle and A. Paz, "Realizations by stochastic finite automaton," *Journal of Computer and System Sciences*, vol. 5, pp. 26–40, 1971.

[11] Jean Berstel and Christophe Reutenauer, *Rational Series and Their Languages*, Springer-Verlag: Berlin-New York, 1988.

[12] Mehryar Mohri, "General Algebraic Frameworks and Algorithms for Shortest-Distance Problems," Technical Memorandum 981210-10TM, AT&T Labs - Research, 62 pages, 1998.

[13] T. Cormen, C. Leiserson, and R. Rivest, *Introduction to Algorithms*, The MIT Press: Cambridge, MA, 1992.

[14] Mehryar Mohri, Fernando C. N. Pereira, and Michael Riley, "Weighted Finite-State Transducers in Speech Recognition," in *Proceedings of the ISCA Tutorial and Research Workshop, Automatic Speech Recognition: Challenges for the new Millenium (ASR2000)*, Paris, France, September 2000.

[15] B. Buntschuh, C. C. Kamm, G. DiFabbrizio, A. Abella, M. Mohri, S. Narayan, I. Zeljkovic, R. Sharp, J. Wright, S. Marcus, J. Shaffer, R. Duncan, and J. G. Wilpon, "VPQ: A Spoken Language Interface to Large Scale Directory Information," in *Proceedings of the 1998th International Conference on Spoken Language Processing (ICSLP '98), Sydney, Australia*. 1998, IEEE.