In *Proc. of the ESSLLI Workshop on Finite-State Methods in NLP (FSMNLP)*, Helsinki, August 2001.

Expectation Semirings: Flexible EM for Learning Finite-State Transducers*

Jason Eisner, Johns Hopkins University (jason@cs.jhu.edu)

1 Motivation

Most recent work on finite-state transducers (FSTs) falls into two camps according to how the transducers are constructed. The algebraic camp employs experts who write (possibly weighted) regular expressions by hand, using an ever-growing language of powerful algebraic operators [10, 7]. The statistical camp, which prefers to extract expertise automatically from data, builds transducers with much simpler topology so that their arc probabilities can be easily trained (e.g., [17, 12, 11]).[1]

This paper offers a clean way to combine the two traditions: an Expectation-Maximation (EM) [4] algorithm for training arbitrary FSTs. First the human expert uses domain knowledge to specify the topology and parameterization of the transducer in any convenient way. Then the EM algorithm automatically chooses parameter values that (locally) maximize the joint likelihood of fully or partly observed data.

Unlike previous specialized methods, the EM algorithm here allows transducers having ϵ's and arbitrary topology. It also allows parameterizations that are independent of the transducer topology (hence unaffected by determinization and minimization). But it remains surprisingly simple because all the difficult work can be *subcontracted to existing algorithms* for semiring-weighted automata. The trick is to use a novel semiring.

To combine the two traditions, a domain expert might build a weighted transducer by using *weighted* expressions in the full finite-state calculus. Weighted regexps can also optionally refer to machines that are specified directly in terms of states, arcs, and weights, or even derived by approximating PCFGs [16]. But the various weights are written in terms of unknown parameters and estimated automatically. The semiring approach ensures that the method works even if the transducer is built with operations such as composition, directed replacement, and minimization, which distribute the parameters over the arcs in a complex way.

This parameter estimation yields a transducer that models the joint probability $P(x, y)$, where x and y are input and output strings. As usual, this transducer can easily be manipulated to obtain the marginal distributions $P(x), P(y)$ and the conditional distributions $P(x \mid y), P(y \mid x)$, and to find strings maximizing these distributions. For example, $\text{argmax}_{x \in X} P(x \mid y \in Y) = \text{argmax}_{x \in X} P(x, (y \in Y))$ is the most likely underlying string in regular set X when the corresponding surface string is known to be in regular set Y.

Learning weights for such transducers has many applications. Some examples:

- Simple special cases include parameter estimation for common probability models—alignment models [23], HMMs [2], n-gram models for segmentation or classification, and transformation models [5]—as well as the usual variations of these to allow variable-length history, epsilons, partly-observed input or output, tied parameters, mixtures, etc. Treating all these models in a unified framework makes them easy to implement and modify, as well as providing a testbed for improved learning algorithms.

- One could stochasticize hand-built machines. For example, given a nondeterministic transducer for morphological analysis, one could learn parameters that predict the probabilities of different analyses.

- Existing statistical finite-state models could be trained "end-to-end." [12] contains a noisy-channel component that maps English phonology ep to Japanese phonology jp, trained solely on hard-to-obtain (ep, jp) pairs. Composing this channel with other transducers yields [12]'s full model that transliterates English text et to Japanese text jt. EM can use (et, jt) pairs to help train all the parameters at once.

- One could try to learn underlying representations with little or no supervision (as with HMMs). Building on [8], one could construct a stochastic transducer that concatenates randomly-selected possible stems and affixes, then passes the result through randomly selected hand-built phonological rules or constraints. Observations of unanalyzed (or analyzed) words could then be used to set the parameters that govern the random choices—thereby inducing a morphological lexicon and a phonology.

*Thanks to Gideon Mann, Charles Schafer, and Mehryar Mohri for useful conversations. Apologies for these 4 pages' density.

[1]The typical statistical approach is to compose an n-gram source model with a cascade of noisy-channel models that distort the source locally. Each of these component models handles one level of intermediate representation, and is implemented as a transducer that is either hand-built or of an especially simple and trainable form. This lets the component models be trained separately (i.e., before composition). The n-gram source model can be trained by standard ML estimation with backoff [1]. The channel models are *one-state* transducers with arc labels in $\Sigma^* \times \Sigma^*$, and can be trained by an "alignment" EM algorithm that is folklore in computational linguistics and biology; see [23] for a finite-state presentation as weighted edit distance.

In short, finite-state devices are so broadly applicable that any learning method should be extremely useful.

2 The Problem: EM for Parameterized Probabilistic Relations

A **(real-)weighted regular relation** is a function $f : \Sigma^* \times \Delta^* \to \mathbb{R}_{\geq 0}$ that can be computed by a weighted finite-state transducer—or, equivalently, by a regular expression (regexp) built up by concatenation, union, and closure from atomic expressions of the form $p \in \mathbb{R}$ and $a\!:\!b \in (\Sigma \cup \{\epsilon\}) \times (\Delta \cup \{\epsilon\})$.

Such an f is called **probabilistic** if $\sum_{x,y} f(x,y) = 1$, so that it is a probability distribution over string pairs. We offer two theorems:[2] (1) f is probabilistic iff it can be computed by a **Markovian transducer**, i.e., one in which each state has total outgoing probability of 1 (out-arc probabilities plus stopping probability) and the expected path length is finite. (2) f is probabilistic iff it can be expressed as a **probabilistic regexp**, that is, a regexp built up from atomic expressions $a\!:\!b$ using concatenation, probabilistic union $+_p$, and probabilistic closure ^{*_p}. Here $f +_p g \stackrel{\text{def}}{=} pf + (1-p)g$, and $f^{*_p} \stackrel{\text{def}}{=} (pf)^*(1-p)$, for $0 \leq p < 1$.

Probabilistic regexps can be extended with other operators (crossprod, multi-way $+_p$, PCFG approximation [16], …). In particular, probabilistic relations are closed under compositions of the form $f \circ \text{condit}(g)$ (if $\text{range}(f) \subseteq \text{domain}(g)$). Here $\text{condit}(g)$ is the conditional probability distribution $g(x,y)/\sum_{y'} g(x,y')$: it is a weighted regular relation expressible as $\text{recip}(\text{weighted-domain}(g)) \circ g$, where $\text{recip}(h)$ is constructed by taking the reciprocal of all nonzero weights in a *determinized* acceptor for h. Conditional-distribution transducers for use in composition can also be built by stochastic variants of directed replacement, including directed replacement by another conditional transducer [18, 7]. Finally, notice that since any weighted relation can be normalized to a probabilistic one, reweighting or log-linear constructions are possible: e.g., let $f_1, \ldots f_n$ be weighted acceptors that separately score a string, and consider the transducer $\text{normalize}(f_1 \odot f_2 \odot \cdots \odot f_n) \circ \text{condit}(g)$, where $\odot$ is weighted intersection (Hadamard product) so that $(f_1 \odot f_2)(x) \stackrel{\text{def}}{=} f_1(x) f_2(x)$.

A **parameterized** probabilistic relation f is a recipe for turning a parameter vector θ into a probabilistic relation, denoted f_θ. For example, the recipe might use θ to obtain arc probabilities for a transducer of fixed topology, or to obtain operator probabilities for an otherwise fixed probabilistic regexp.

It is important not to simply identify parameters with transducer arcs. For example, if f is defined as the composition of g and h, then we wish its parameter set to be the union of g's and h's (few parameters), even though its arc set is the product of g's and h's (many arcs). The fewer parameters, the easier learning is. Indeed it often pays to reuse parameters explicitly: a single parameter for gemination might determine, or at least affect, the probabilities of arcs mapping $t\!:\!tt$, $k\!:\!kk$, etc. in a transducer from English to Japanese phonology. We particularly like log-linear (maximum-entropy) parameterizations, since then the weights of several meaningful domain-specific features (like gemination) can interact to determine an arc probability or operator probability. In [5] we suggest several log-linear parameterizations of arc weights for both Markovian and non-Markovian probabilistic transducers, including one inspired by current flow in electrical networks.

Given a sample of pairs (x_i, y_i) from the unknown distribution $f_{\hat{\theta}}$, we want to find θ that maximizes the likelihood $\prod_i f_\theta(x_i, y_i)$.[3] The EM method [4] starts with some guess f_θ and repeatedly reestimates θ, converging to a local optimum. Roughly, the **E step** of reestimation figures out what paths were probably used to generate the sample if it came from the current f_θ, and the **M step** updates θ to make those paths more likely. The M step depends on the parameterization and the E step serves the M step's needs.

In the simplest case, f_θ is a Markovian transducer and the parameters θ are the arc probabilities. Then the E step needs to find the expected count of visits to each arc; the M step sets an arc's new probability as proportional to its count. But if f_θ is implemented as a (possibly minimized) composition of two such Markovian transducers, $T_1 \circ T_2$, the M step will have to compute new probabilities for arcs in the original T_1 and T_2—*so the E step must implicitly "undo" the composition and minimization,* and find the expected count of visits to each arc *in* T_1 *and* T_2.[4] We now see how to make the *composed* machine carry out such an E step without "undoing" anything. The idea also works for machines built using other operators.

[2] To prove (2)⇒, get a machine for f and apply the Kleene-Schützenberger (= Floyd-Warshall) construction, taking care to write each regexp in the construction as a constant times a probabilistic regexp. To prove (1)⇒, just convert this probabilistic regexp into a Markovian transducer. Full proofs are straightforward but suppressed here for space reasons.

[3] Perhaps times a prior probability $P(\theta)$, which EM allows if the M step does. Or one can choose to maximize the conditional probability $\prod_i f_\theta(y_i \mid x_i)$ instead of the joint: just make the M step improve the likelihood of the expected paths *given* x_i.

[4] As defined by a distribution over *pairs* of paths that could have generated (x_i, y_i) via some intermediate representation z_i.

3 The Technique: Expectation Semirings

A unifying framework for finite-state computation is given in [3]. A finite-state function is represented by a finite-state machine over an input alphabet Σ, with arc weights in a semiring $(K, \oplus, \otimes)$.[5] The machine maps each input $x \in \Sigma^*$ to a weight in K, found by using $\oplus$ to sum over the weights of all paths that accept x. The weight of each path is found by multiplying its arcs' weights using $\otimes$. If the machine has ϵ-cycles, it may be necessary to sum over infinitely many paths; then the semiring also needs a closure operation *, interpreted as $k^* = \bigoplus_{i=0}^{\infty} k^i$, which makes it possible to eliminate all ϵ arcs.

To obtain weighted transducers from Σ^* to Δ^*, one labels each arc with an output string as well as an input symbol and a weight. Then the machine assigns a weight to each string pair $(x, y) \in \Sigma^* \times \Delta^*$. This is found by summing the weights of all paths that accept x while outputting y.[6]

A probabilistic regular relation can be computed by a transducer weighted with the semiring $(\mathbb{R}_{\geq 0}, +, \times)$.[7] The key idea of this paper is to augment this semiring with expectation information. Then a weight records not only a probability but also (e.g.) the degree to which each arc has contributed to that probability.

Let us define the problem abstractly. Suppose we assign each arc in the stochastic transducer a **value** drawn from a set V. The **total value** $\mathrm{val}(\pi)$ of a path π is the sum of its arcs' values. If Π is a set of paths, the **expected value** given that we took a path in Π is $\mathbf{E}[\mathrm{val}(\pi) \mid \Pi] \stackrel{\text{def}}{=} \sum_\pi P(\pi \mid \Pi) \cdot \mathrm{val}(\pi)$. For instance, if every arc has value 1, this is the expected length of a random path in Π. We will see in the next section that, given an appropriate assignment of values to arcs, the E step of EM can be formulated as computing $\mathbf{E}[\mathrm{val}(\pi) \mid \Pi]$ where Π is the set of paths compatible with an observation (x_i, y_i).

As always, there is also an algebraic view. Values in V can be interspersed through a probabilistic regexp. They may be regarded as outputs onto an extra tape—or rather, into an accumulator, since they are summed rather than concatenated. The regexp can then be regarded as defining $P(x, y, v)$. We desire the expected value of the accumulator when x_i is transduced nondeterministically to y_i, namely $\sum_v P(v \mid x_i, y_i) \cdot v$.

How to compute these expected values by finite-state methods? Notice $\mathbf{E}[\mathrm{val}(\pi) \mid \Pi] = \mathbf{E}[\mathrm{val}(\pi) \& \Pi]/P(\Pi)$, where $\mathbf{E}[\mathrm{val}(\pi) \& \Pi] \stackrel{\text{def}}{=} \sum_{\pi \in \Pi} P(\pi) \cdot \mathrm{val}(\pi)$ (and $P(\Pi) \stackrel{\text{def}}{=} \sum_{\pi \in \Pi} P(\pi)$). We will now modify the automaton to use a semiring in which the total weight of the paths in Π is the ordered pair $t = (P(\Pi), \mathbf{E}[\mathrm{val}(\pi) \& \Pi]) \in \mathbb{R}_{\geq 0} \times V$. Once we obtain the pair t, just dividing one element by the other will give $\mathbf{E}[\mathrm{val}(\pi) \mid \Pi]$ as desired.

The semiring we introduce for this purpose is the **V-expectation semiring**, $(\mathbb{R}_{\geq 0} \times V, \oplus, \otimes)$, where

$$(p_1, v_1) \oplus (p_2, v_2) \stackrel{\text{def}}{=} (p_1 + p_2, v_1 + v_2) \qquad (p_1, v_1) \otimes (p_2, v_2) \stackrel{\text{def}}{=} (p_1 p_2, p_1 v_2 + v_1 p_2) \qquad (p, v)^* \underset{\text{if } p^* \text{ defined}}{\stackrel{\text{def}}{=}} (p^*, p^* v p^*)$$

We augment the machine's arc weights to fall in this semiring. We want an invariant that the total weight of any pathset Π is $(P(\Pi), \mathbf{E}[\mathrm{val}(\pi) \& \Pi])$. If an arc has probability p and value v, we give it the weight (p, pv), so that the invariant holds if Π consists of a single length-0 or length-1 path. The above definitions preserve the invariant when we build up longer paths with $\otimes$ and sets of paths with $\oplus$ or *, as discussed earlier. For example, concatenating paths π_1, π_2 with weights $(P(\pi_1), P(\pi_1) \cdot \mathrm{val}(\pi_1))$ and $(P(\pi_2), P(\pi_2) \cdot \mathrm{val}(\pi_2))$ gives a longer path π with weight $(P(\pi_1)P(\pi_2), P(\pi_1)P(\pi_2) \cdot (\mathrm{val}(\pi_1) + \mathrm{val}(\pi_2))) = (P(\pi), P(\pi) \cdot \mathrm{val}(\pi))$.

So to find the expected value, we need the total weight t_i of paths in machine T compatible with observation (x_i, y_i). Standard methods now apply. t_i is just the total weight of *all* accepting paths in $x_i \circ T \circ y_i$. Finding such totals is the **single-source algebraic path** problem (= semiring "shortest" path), which can be solved approximately by relaxation [13]. Indeed, the generalization to **all-pairs algebraic path** (= semiring transitive closure)—where various exact $O(n^3)$ algorithms are known [6]—is already implemented inside any WFSA ϵ-elimination routine [15]. So for convenience, one could simply minimize the machine $(\epsilon \times x_i) \circ T \circ (y_i \times \epsilon)$, yielding a one-state machine that maps (ϵ, ϵ) to t_i, and then just read t_i off the machine.

Thus, if weights are augmented with expectations, the entire E step reduces to "compose + minimize"!

In general x_i and y_i can be regular languages rather than strings, representing incomplete observations. E.g., in the traditional EM algorithm for HMMs [2], $y_i = \Sigma^*$ (wholly unobserved), though [22] allows any y_i.

When is the V-expectation semiring really a semiring? Notice that to take expected values, we had to assume that linear combinations of V were defined. Indeed, V must have the structure of a module

[5] Such a function can be described as a formal power series over non-commuting variables Σ with coefficients in $(K, \oplus, \otimes)$. Semiring axioms: $(K, \otimes)$ is a monoid (meaning $\otimes : K \times K \to K$ is associative) with identity $\underline{1}$, $(K, \oplus)$ is a *commutative* monoid with identity $\underline{0}$, $\otimes$ distributes over $\oplus$ from both sides, and $\underline{0} \otimes k = k \otimes \underline{0} = \underline{0}$. Also $k^* = \underline{1} \oplus k \otimes k^*$ if defined.

[6] By treating the (weight,output) pair on each arc as a kind of complex weight, such a transducer can be regarded as simply a weighted automaton. However, in this paper, K denotes just the ordinary weight without the output string.

[7] The closure operation is defined for $p \in [0, 1)$ as $p^* = 1/(1 - p)$, so cycles are allowed if they have weights in $[0, 1)$.

(a generalization of a vector space). If R is the semiring of probabilities (any semiring will do), then $K = (R \times V, \oplus, \otimes)$ as defined above is a semiring provided that V is a two-sided R-semimodule.[8] Its additive and multiplicative identities are $(\underline{0}, \underline{0})$ and $(\underline{1}, \underline{0})$. K is commutative iff R is commutative and commutes with V ($r \otimes v = v \otimes r$). K has additive inverses iff R does: $-(r, v) = (-r, (-\underline{1}) \otimes v)$. If so, K has multiplicative left inverses iff R does: $(r, v)^{-1} = (r^{-1}, -r^{-1} \otimes v \otimes r^{-1})$. These properties hold for $R = \mathbb{R}$, and the last (as we will show elsewhere) nontrivially leads to an efficient minimization algorithm for our K-weighted transducers (cf. [14]). We have not yet investigated the effect on determinizability of replacing semiring R with K.

4 Computing the Expectation in Practice

As an example, we use an expectation semiring to perform the E step for composed Markovian transducers $T_1 \circ T_2$, as promised at the end of section 2. Number the arcs in (T_1, T_2) from 1 to m. Let $V = \mathbb{R}^m$ (length-m vectors) and let the value of arc i be the i^{th} basis vector. Now proceed as in section 3. The arc values are used to define arc weights in $\mathbb{R} \times V$ for T_1 and T_2, and hence also for their composition T. Then for each pair (x_i, y_i), compute $t_i \in \mathbb{R} \times V$ and thence the expected value, i.e., the expected # of traversals of each arc, as desired. (A similar technique gets the expected traversals of each subexpression in a probabilistic regexp.)

As another example, consider a log-linear parameterization where each arc has a vector of features that determines its probability. The M step is now a convex maximization problem traditionally solved by iterative scaling [19], which needs to know how many times each feature was observed in the sample.[9] To reconstruct the expectation of this (the E step), simply let each arc's value be its feature vector and proceed as before.

Now for some important remarks on efficiency. Section 3 runs algebraic path on $T_i \stackrel{\text{def}}{=} x_i \circ T \circ y_i$ in order to find $t_i = w_{0n}$, the total semiring weight of paths from initial state 0 to final state n (assumed WLOG to be unique and unweighted). An exact answer takes $O(n^3)$ time in the worst case. But when T_i is acyclic[10]—e.g., it is often an HMM-style trellis—w_{0n} can easily be found in $O(m)$ time (for m arcs) by computing w_{0j} for all j in topologically sorted order. As a generalization, one can partition the transducer graph into strongly connected components (in linear time) and run all-pairs algebraic path on each component separately to find some of the w_{jk} values, after which all the w_{0j} can be computed in $O(m)$ time roughly as before [13].

When V counts arcs and T_i is a trellis, this method is a variant of the forward-backward algorithm [2] with no backward pass: instead of pushing cumulative probabilities backward to the arcs, it pushes cumulative arcs (more generally, values in V) forward to the probabilities. This is slower because our $\oplus$ and $\otimes$ are vector operations, and the vectors rapidly lose sparsity as they are added together. Luckily, the forward-backward idea *can* be used here—indeed in a fully general form allowing any V and (perhaps cyclic) T_i. It exploits our semiring structure to avoid $\oplus$ and $\otimes$ (though they are still crucial to construct T_i!). Write w_{jk} as (p_{jk}, v_{jk}), and let w^1_{jk} denote the total weight of length-1 paths from j to k. The forward probabilities p_{0j} can be computed just like w_{0j} in the previous paragraph, but using only operations on $(\mathbb{R}, +, \times)$ without V, and the backward probabilities p_{kn} similarly. (Indeed this just solves a $\mathbb{R}$-linear system, a well-studied problem at $O(n)$ space, $O(nm)$ time, approximation faster [9].) Then $w_{0n} = (p_{0n}, v_{0n})$ is found as $(p_{0n}, \sum_{j,k} p_{0j} v^1_{jk} p_{kn})$.

5 Conclusions

We know of no previously published EM algorithm for general FSTs (with ϵ), even for the simplest parameterization. Our expectation-semiring solution seems general enough to work with many parameterizations, though details will vary. It may also be a starting point for faster or more robust parameter estimation methods. Conjugate gradient is an easy variation because of the linearity of transducers (vs. neural nets): the gradient of the mapping from arc weights θ to $f_\theta(x_i, y_i)$ is exactly the v_{0n} vector computed above, with the change that arc i must be given weight (p, v) instead of (p, pv) (where $v = i^{\text{th}}$ basis vector). To avoid local maxima, one might try deterministic annealing [20], or randomized methods, or place a prior on θ. Finally, parameter estimation enables experiments with FSA model merging methods [24] to learn FST topology.

[8] Module axioms: V is a commutative monoid with operation denoted by $\oplus$ and identity by $\underline{0}$ (note the overloaded notation!). It is equipped with a left scalar multiplication $\otimes : R \times V \to V$ that satisfies $\underline{0} \otimes v = \underline{0} = r \otimes \underline{0}$, $\underline{1} \times v = v$, $r \otimes (s \otimes v) = (r \otimes s) \otimes v$, $(r \oplus s) \otimes v = r \otimes v \oplus s \otimes v$, $r \otimes (v \oplus w) = r \otimes v \oplus r \otimes w$. Similarly it has a right scalar multiplication $\otimes : V \times R \to V$.

[9] This is for a log-linear distribution $P(\text{path})$. For a Markovian machine, we need a *conditional* log-linear distribution $P(\text{arc} \mid \text{state})$; then the M step needs the E step to collect counts of states as well as features [5], which is straightforward. In either case, iterative scaling needn't be run to convergence on each M step; [21] runs just one iteration to get a GEM algorithm.

[10] In particular, if x_i and y_i are acyclic (e.g., fully observed strings), then eliminating $\epsilon : \epsilon$ cycles from T ensures that the composition will "unroll" T into an acyclic machine T_i. If only x_i is acyclic, then T_i is still acyclic if domain(T) has no ϵ cycles.

References

[1] Lalit R. Bahl, Frederick Jelinek, and Robert L. Mercer. A maximum likelihood approach to continuous speech recognition. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 5(2):179–190, 1983.

[2] L. E. Baum. An inequality and associated maximization technique in statistical estimation of probabilistic functions of a Markov process. *Inequalities*, 3, 1972.

[3] Jean Berstel and Christophe Reutenauer. *Rational Series and their Languages*. Springer-Verlag, 1988.

[4] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *J. Royal Statist. Soc. Ser. B*, 39(1):1–38, 1977. With discussion.

[5] Jason Eisner. *Smoothing a Probabilistic Lexicon Via Syntactic Transformations*. PhD thesis, University of Pennsylvania, 2001.

[6] Eugene Fink. A survey of sequential and systolic algorithms for the algebraic path problem. Technical Report CS-92-37, Department of Computer Science, University of Waterloo, 1992.

[7] D. Gerdemann & G. van Noord. Transducers from rewrite rules with backreferences. *Proc. EACL*, 1999.

[8] John Goldsmith. Unsupervised learning of the morphology of a natural language. *Comp. Ling.*, 2001.

[9] Anne Greenbaum. *Iterative Methods for Solving Linear Systems*. Number 17 in Frontiers in Applied Mathematics. Society for Industrial and Applied Mathematics (SIAM), 1997.

[10] Lauri Karttunen, Jean-Pierre Chanod, Gregory Grefenstette, , and Anne Schiller. Regular expressions for language engineering. *Journal of Natural Language Engineering*, 2(4):305–328, 1996.

[11] Kevin Knight and Yaser Al-Onaizan. Translation with finite-state devices. In *Proc. of 4th AMTA*, 1998.

[12] Kevin Knight and Jonathan Graehl. Machine transliteration. *Computational Linguistics*, 24(4), 1998.

[13] Mehryar Mohri. General algebraic frameworks and algorithms for shortest-distance problems. Technical Memorandum 981210-10TM, AT&T Labs—Research, 1998.

[14] Mehryar Mohri. Minimization algorithms for sequential transducers. *Theoretical CS*, 324:177–201, 2000.

[15] Mehryar Mohri. Generic Epsilon-Removal and Input Epsilon-Normalization Algorithms for Weighted Transducers. *International Journal of Foundations of Computer Science*, to appear 2001.

[16] Mehryar Mohri and Mark-Jan Nederhof. Regular approximation of context-free grammars through transformation. In Jean-Claude Junqua and Gertjan van Noord, editors, *Robustness in Language and Speech Technology*, chapter 9, pages 153–163. Kluwer Academic Publishers, The Netherlands, 2001.

[17] Mehryar Mohri, Fernando Pereira, and Michael Riley. Weighted automata in text and speech processing. In *Workshop on Extended Finite-State Models of Language (ECAI-96)*, pages 46–50, Budapest, 1996.

[18] Mehryar Mohri and Richard Sproat. An efficient compiler for weighted rewrite rules. In *Proceedings of the 34th Annual Meeting of the ACL*, pages 231–238, Santa Cruz, 1996.

[19] S. Della Pietra, V. Della Pietra, and J. Lafferty. Inducing features of random fields. *IEEE Transactions Pattern Analysis and Machine Intelligence*, 19(4), 1997.

[20] A. V. Rao, K. Rose, and A. Gersho. A deterministic annealing approach to discriminative hidden Markov model design. In *IEEE Workshop on Neural Networks and Signal Processing*, September 1997.

[21] Stefan Riezler. *Probabilistic Constraint Logic Programming*. PhD thesis, Universität Tübingen, 1999.

[22] Eric Sven Ristad. Hidden markov models with finite state supervision. In András Kornai, editor, *Extended Finite State Models of Language*. Cambridge University Press, 1998.

[23] Eric Sven Ristad and Peter N. Yianilos. Learning string edit distance. Princeton CS-TR-532-96, 1996.

[24] Andreas Stolcke and Stephen M. Omohundro. Best-first model merging for hidden Markov model induction. Technical Report ICSI TR-94-003, ICSI, Berkeley, CA, 1994.