Weighted Rational Transductions and their Application to Human Language Processing

Fernando Pereira *Michael Riley* *Richard Sproat*

AT&T Bell Laboratories
600 Mountain Ave.
Murray Hill, NJ 07974

ABSTRACT

We present the concepts of weighted language, transduction and automaton from algebraic automata theory as a general framework for describing and implementing decoding cascades in speech and language processing. This generality allows us to represent uniformly such information sources as pronunciation dictionaries, language models and lattices, and to use uniform algorithms for building decoding stages and for optimizing and combining them. In particular, a single *automata join* algorithm can be used either to combine information sources such as a pronunciation dictionary and a context-dependency model during the construction of a decoder, or dynamically during the operation of the decoder. Applications to speech recognition and to Chinese text segmentation will be discussed.

1. Introduction

As is well known, many problems in human language processing can be usefully analyzed in terms of the "noisy channel" metaphor: given an observation sequence o, find which intended message w is most likely to generate that observation sequence by maximizing

$$P(w, o) = P(o|w)P(w),$$

where $P(o|w)$ characterizes the transduction between intended messages and observations, and $P(w)$ characterizes the message generator. More generally, the transduction between messages and observations may involve several intermediate stages

$$\begin{array}{l} P(s_0, s_k) = P(s_k|s_0)P(s_0) \\ P(s_k|s_0) = \sum_{s_1,\ldots,s_{k-1}} P(s_k|s_{k-1}) \cdots P(s_1|s_0) \end{array} \quad (1)$$

where $P(s_k|s_0)$ is the probability of transducing s_0 to s_k through the intermediate stages, assuming that each step in the cascade is conditionally independent from the previous ones. Each s_j is a sequence of units in an appropriate representation. For instance, in speech recognition some of the intermediate stages might correspond to sequences of units like phones or syllables. A straightforward but useful observation is that any such a cascade can be factored at any intermediate stage

$$P(s_i|s_j) = \sum_{s_k} P(s_i|s_k)P(s_k|s_j) \quad (2)$$

For computational reasons, sums and products in (1) are often replaced by minimizations and sums of negative log probabilities, yielding the approximation

$$\begin{array}{rcl} \tilde{P}(s_0, s_k) & = & \tilde{P}(s_k|s_0) + \tilde{P}(s_0) \\ \tilde{P}(s_k|s_0) & \approx & \min_{s_1,\ldots,s_{k-1}} \sum_{1 \le j \le k} \tilde{P}(s_j|s_{j-1}) \end{array} \quad (3)$$

where $\tilde{X} = -\log X$. In this formulation, assuming the approximation is reasonable, the most likely message s_0 is the one minimizing $\tilde{P}(s_0, s_k)$.

Finally, each transduction in such a cascade is often modeled by some finite-state device, for example a hidden Markov model.

Although the above approach is widely used in speech and language processing, usually the elements of the transduction cascade are built by "ad hoc" means, and commonalities between them are not exploited. We will here outline how the theory of weighted rational languages and transductions can be used as a general framework for transduction cascades. This theoretical foundation provides a rich set of operators for combining cascade elements that generalizes the standard operations on regular languages, suggests novel ways of combining models of different parts of the decoding process, and supports uniform algorithms for transduction and search at all levels in the cascade. In particular, we developed a generic *join* algorithm for combining any two consecutive levels of a cascade, a generic best-path search algorithm, and a generic interleaving of join and search for building pruned joins. In addition, general finite-state minimization techniques are also applicable to all levels of a cascade.

Weighted languages and transductions are generalizations of the standard notions of language and transduction in formal language theory [1, 2]. A weighted language is just a mapping from strings over an alphabet to weights. A weighted transduction is a mapping from pairs of strings over two alphabets to weights. For example, when weights represent probabilities and assuming appropriate normalization, a weighted language is just a probability distribution over strings, and a weighted transduction a joint probability distribution over string pairs. The weighted *rational* languages and transducers are those that can be represented by *weighted* finite-state acceptors (WFSAs) and weighted finite-state transducers (WFSTs), as described in more detail in the next section. In this paper we will be concerned with the weighted rational case, although some of the theory can be profitably extended beyond the finite-state case [3, 4].

The notion of weighted rational transduction arises from the combination of two ideas in automata theory: rational transductions, used in many aspects of formal language theory [2], and weighted languages and automata, developed in pattern recognition [5, 6] and algebraic automata theory [7, 8, 9]. Ordinary (unweighted) rational transductions have been successfully applied by researchers at Xerox PARC [10] and at the University of Paris 7 [11], among others, to several problems in language processing, including morphological analysis, dictionary compression and syntactic analysis. Hidden Markov Models and probabilistic finite-state language models can be shown to be equivalent to WFSAs. In algebraic automata theory, rational series and rational transductions [8] are the algebraic counterparts of WFSAs and WFSTs and give the correct generalizations to the weighted case of the standard algebraic operations on formal languages and transductions, such as union, concatenation, intersection, restriction and composition. We believe the work presented here is among the first to apply these generalizations to human-language processing.

Our first application is to speech recognition decoding. We show that a conventional HMM decoder can be naturally viewed as equivalent to a cascade of weighted transductions, and that our approach requires no modification whatsoever when context dependencies cross higher-level unit boundaries (for instance, cross-word context-dependent models).

Our second application is to the segmentation of Chinese text into words, and the assignment of pronunciations to those words. In Chinese orthography, most characters represent (monosyllabic) 'morphemes', and as in English, 'words' may consist of one or more morphemes. Given that Chinese does not use whitespace to delimit words, it is necessary to reconstruct the grouping of characters into words. This reconstruction can also be thought of as a transduction problem.

2. Theory

In the transduction cascade (1), each step corresponds to a mapping from input-output pairs (r, s) to probabilities $P(s|r)$. More formally, steps in the cascade will be *weighted transductions* $T : \Sigma^* \times \Gamma^* \rightarrow K$ where Σ^* and Γ^* the sets of strings over the alphabets Σ and Γ, and K is an appropriate set of weights, for instance the real numbers between 0 and 1 in the case of probabilities. We will denote by T^{-1} the *inverse* of T defined by $T(t, s) = T(s, t)$.

The right-most step of (1) is not a transduction, but rather an information source, in that case the language model. We will represent such sources as *weighted languages* $L : \Sigma^* \rightarrow K$.

Given two transductions $S : \Sigma^* \times \Gamma^* \rightarrow K$ and $T : \Gamma^* \times \Delta^* \rightarrow K$, we can define their *composition* $S \circ T$ by

$$(S \circ T)(r, t) = \sum_{s \in \Gamma^*} S(r, s)T(s, t) \qquad (4)$$

For example, if S represents $P(s_k|s_j)$ and T $P(s_j|s_i)$ in (2), is is clear that $S \circ T$ represents $P(s_k|s_i)$.

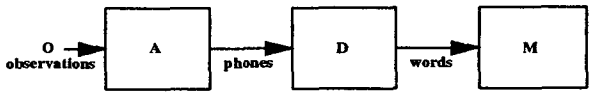

Figure 1: Recognition Cascade

A weighted transduction $S : \Sigma^* \times \Gamma^* \rightarrow K$ can be *applied* to a weighted language $L : \Sigma^* \rightarrow K$ to yield a weighted language over Γ. It is convenient to abuse notation somewhat and use $M \circ S$ for the result of the application, defined as

$$(L \circ S)(t) = \sum_{s \in \Gamma^*} L(s)S(s, t) \qquad (5)$$

Furthermore, if M is a weighted language over Γ, we can *reverse apply* S to M, written $S \circ M = M \circ (S^{-1})$. For example, if S represents $P(s_k|s_0)$ and M represents $P(s_0)$ in (1), then $S \circ M$ represents $P(p_0, p_k)$.

Finally, given two weighted languages $M, N : \Sigma^* \rightarrow K$ we define their *intersection*, also by convenient abuse of notation written $M \circ N$ as:

$$(M \circ N)(t) = M(s)N(s) \qquad (6)$$

In any cascade $R_1 \circ \cdots \circ R_m$, with the R_i for $1 < i < m$ appropriate transductions and R_1 and R_m transductions or languages, it is easy to see that the order of association of the $\circ$ operators does not matter. For example, if we have $L \circ S \circ T \circ M$, we could either apply S to L, apply T to the result and intersect the result with M, or compose S with T, reverse apply the result to M and intersect the result with L. We are thus justified in our use of the same symbol for composition, application and intersection, and we will in the rest of the paper use the term "(generalized) composition" for all of these operations.

For a more concrete example, consider the transduction cascade for speech recognition depicted in Figure 1, where A is the transduction from acoustic observation sequences to phone sequences, D the transduction from phone sequences to word sequences (essentially a pronunciation dictionary) and M a weighted language representing the language model. Given a particular sequence of observations o, we can represent it as the trivial weighted language O that assigns 1 to o and 0 to any other sequence. Then $O \circ A$ represents the acoustic likelihoods of possible phone sequences that generate o, $O \circ A \circ D$ the acoustic-lexical likelihoods of possible word sequences yielding o, and $O \circ A \circ D \circ M$ the combined acoustic-lexical-linguistic probabilities of word sequences generating o. The word string w with the highest weight $(O \circ A \circ D \circ M)(w)$ is precisely the most likely sentence hypothesis generating o.

Exactly the same construction could have been carried out with weights combined by min and sum instead of sum and product in the definitions of application and intersection, and

	Language	Transduction
singleton	$\{u\}(v) = 1$ iff $u = v$	$\{(u,v)\}(w,z) = 1$ iff $u = w$ and $v = z$
scaling	$(kL)(u) = kL(u)$	$(kT)(u,v) = kT(u,v)$
sum	$(L+M)(u) = L(u) + M(u)$	$(S+T)(u,v) = S(u,v) + T(u,v)$
concatenation	$(LM)(w) = \sum_{uv=w} L(u)M(v)$	$(ST)(t,w) = \sum_{rs=t,uv=w} S(r,u)T(s,v)$
power	$L^0(\epsilon) = 1$	$T^0(\epsilon,\epsilon) = 1$
	$L^0(u \neq \epsilon) = 0$	$T^0(u \neq \epsilon, v \neq \epsilon) = 0$
	$L^{n+1} = LL^n$	$T^{n+1} = TT^n$
closure	$L^* = \sum_{k \geq 0} L^k$	$T^* = \sum_{k \geq 0} T^k$

Table 1: Rational Operations

in that case the string w with the lowest weight $(O \circ A \circ D \circ M)(w)$ would the best hypothesis. More generally, the sum and product operations in (4), (5) and (6) can be replaced by any two operations forming an appropriate *semiring* [7, 8, 9], of which numeric addition and multiplication and numeric minimum and addition are two examples [1].

Generalized composition is thus the main operation involved in the construction and use of transduction cascades. As we will see in a moment, for rational languages and transductions, all instances of generalized composition are implemented by a uniform algorithm, the *join* of two weighted finite automata. In addition to those operations, weighted languages and transductions can be constructed from simpler ones by the operations shown in Table 1, which generalize in a straightforward way the regular operations well-known from traditional automata theory [1]. In fact, the rational languages and transductions are exactly those that can be built from singletons by applications of scaling, sum, concatenation and closure.

For example, assume that for each word w in a lexicon we are given a rational transduction D_w such that $D_w(p,w)$ is the probability that w is realized as the phone sequence p. Note that this crucially allows for multiple pronunciations for w. Then the rational transduction $\left(\sum_w D_w\right)^*$ gives the probabilities for realizations of word sequences as phone sequences (ignoring possible cross-word dependencies, which will be discussed in the next section).

Kleene's theorem states that regular languages are exactly those representable by finite-state acceptors [1]. Its generalization to the weighted case and to transducers states that weighted rational languages and transducers are exactly those that can be represented by finite automata [8]. Furthermore, all the operations on languages and transductions we have discussed have finite-automata counterparts, which we have implemented. Any cascade representable in terms of those operations can thus be implemented directly as an appropriate combination of the programs implementing each of the operations.

[1] Additional conditions to guarantee the existence of certain infinite sums may be necessary for certain semirings, for details see [7] and [8].

In the present setting, a *K-weighted finite automaton* $\mathcal{A}$ consists of a finite set of states $Q_\mathcal{A}$ and a finite set $\Delta_\mathcal{A}$ of *transitions* $q \xrightarrow{x/k} q'$ between states, where x is an element of the set of transition *labels* $\Lambda_\mathcal{A}$ and $k \in K$ is the transition weight. An associative concatenation operation $u \cdot v$ must defined between transition labels, with identity element $\epsilon_\mathcal{A}$. As usual, each automaton has an *initial state* $i_\mathcal{A}$ and a *final state* assignment, which we represent as column vector of weights $F_\mathcal{A}$ indexed by states[2]. A K-weighted finite automaton with $\Lambda_\mathcal{A} = \Sigma^*$ is just a *weighted finite-state acceptor* (WFSA). On the other hand, if $\Lambda_\mathcal{A} = \Sigma^* \times \Gamma^*$ with concatenation defined by $(r,s) \cdot (u,v) = (ru, sv)$, we have a *weighted finite-state transducer* (WFST).

As usual, we can define a *path* in an automaton $\mathcal{A}$ as a sequence of connected transitions $\bar{p} = (q_0, x_1, k_1, q_1), \ldots, (q_{m-1}, x_m, k_m, q_m)$. Such a path has *label* $L_\mathcal{A}(\bar{p}) = x_1 \cdot \cdots \cdot x_m$, weight $W_\mathcal{A}(\bar{p}) = k_1 \cdots k_m$ and *final weight* $W^F_\mathcal{A}(\bar{p}) = W_\mathcal{A}(\bar{p})F_\mathcal{A}(q_m)$. We call $\bar{p}$ *reduced* if it is the empty path or if $x_1 \neq \epsilon$, and we write $p \overset{u/k}{\rightsquigarrow} p'$ if k is the sum of the weights of all reduced paths with label u from q to q'.

The *language* of automaton $\mathcal{A}$ is defined as

$$[\![\mathcal{A}]\!](u) = \sum_{\bar{p} \in I_\mathcal{A}(u)} W^F_\mathcal{A}(\bar{p}) \quad ,$$

where $I_\mathcal{A}(u)$ is the set of paths in $\mathcal{A}$ with label u that start in the initial state $i_\mathcal{A}$. Obviously, if $\mathcal{A}$ is an acceptor, $[\![\mathcal{A}]\!]$ is a weighted language, and if $\mathcal{A}$ is a transducer $[\![\mathcal{A}]\!]$ is a weighted transduction. The appropriate generalization of Kleene's theorem to weighted acceptors and transducers states that under mild conditions on the weights (which for instance are satisfied by the min, sum semiring), weighted rational languages and transductions are exactly those defined by weighted automata as outlined here [8].

Weighted acceptors and transducers are thus faithful implementations of rational languages and transductions, and all

[2] The usual notion of final state can be encoded this way by setting $F_\mathcal{A}(q) = 1$ if q is final, $F_\mathcal{A}(q) = 0$ otherwise.

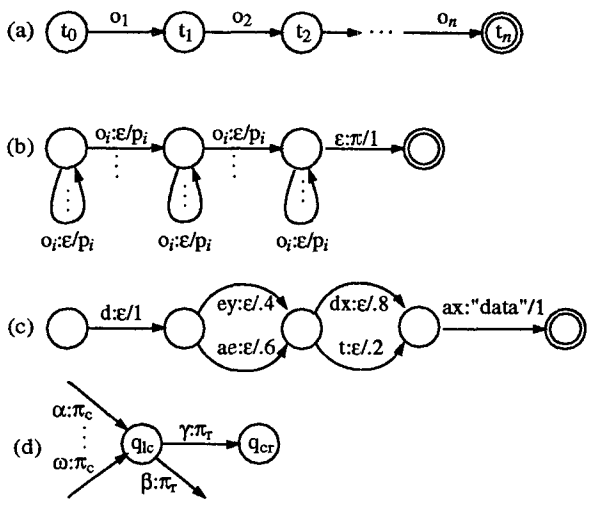

Figure 2: Models as Automata

the operations on these described above have corresponding implementations in terms of algorithms on automata. In particular, generalized composition corresponds to the join of two automata.

Given two automata $\mathcal{A}$ and $\mathcal{B}$ and a new label set J, and a partial *label join* function $\bowtie: \Lambda_{\mathcal{A}} \times \Lambda_{\mathcal{B}} \rightarrow J$, we define their *join* by $\bowtie$ as a new automaton $\mathcal{C}$ with label set J, states $Q_{\mathcal{C}} = Q_{\mathcal{A}} \times Q_{\mathcal{B}}$, initial state $i_{\mathcal{C}} = (i_{\mathcal{A}}, i_{\mathcal{B}})$, final weights $F_{\mathcal{C}}(q, q') = F_{\mathcal{A}}(q)F_{\mathcal{B}}(q)$ and transitions

$$(p, p') \stackrel{x/k}{\rightarrow} (q, q') \text{ iff } k = \sum_{x=y\bowtie z, p \stackrel{y/a}{\rightarrow} q, p' \stackrel{z/b}{\rightarrow} q'} ab \quad (7)$$

Different choices of $\bowtie$ correspond to the instances of generalized composition: for intersection, $\Lambda_{\mathcal{A}} = \Lambda_{\mathcal{B}} = \Sigma^*$, $x = y \bowtie z$ iff $x = y = z$; for composition, $\Lambda_{\mathcal{A}} = \Sigma^* \times \Gamma^*$, $\Lambda_{\mathcal{B}} = \Gamma^* \times \Delta^*$ and $(x, z) = (x, y) \bowtie (y, z)$; and for application $= \Lambda_{\mathcal{A}}\Sigma^*$, $\Lambda_{\mathcal{B}} = \Sigma^* \times \Gamma^*$ and $y = x \bowtie (x, y)$. Thus join is the automata counterpart of generalized composition, and we will use the composition symbol indiferently in what follows to represent either composition or join.

The operation between automata thus defined has a direct dynamic-programming implementation in which reachable join states (q, q') are placed in a queue and extended in turn using (7). By organizing this queue according to the weights of least-weight paths from the start state, we can combine join computation with search for lowest-weight paths, and subautomata of the join with states reachable by paths with weights within a beam of the best path.

3. Speech Recognition

In our first application, we elaborate on how to describe a speech recognizer as a transduction cascade. Recall we decompose the problem into a language, O, of acoustic observation sequences, a transduction, A, from acoustic observation sequences to phone sequences, a transduction, D, from phone sequences to word sequences and a weighted language, M, specifying the language model (see Figure 1). Each of these can be represented as a finite-state automaton (to some approximation).

The trivial automaton for the acoustic observation language, O, is defined for a given utterance as depicted in Figure 2a. Each state represents a fixed point in time t_i, and each transition has a label, o_i, drawn from a finite alphabet that quantizes the acoustic waveform between adjacent time points and is assigned probability 1.0.

The automaton for the acoustic observation sequence to phone sequence transduction, A, is defined in terms of phone models. A phone model is defined as a transducer from a subsequence of acoustic observation labels to a specific phone, and assigns to each subsequence a likelihood that the specified phone produced it. Thus, different paths through a phone model correspond to different acoustic realizations of the phone. Figure 2b depicts a common topology for such a phone model. A is then defined as the closure of the sum of the phone models.

The automaton for the phone sequence to word sequence transduction, D, is defined similarly to that for A. We define a word model as a transducer from a subsequence of phone labels to a specific word, which assigns to each subsequence a likelihood that the specified word produced it. Thus, different paths through a word model correspond to different phonetic realizations of the word. Figure 2c depicts a common topology for such a word model. D is then defined as the closure of the sum of the phone models.

Finally, the language model, M, is commonly an N-gram model, encodable as a WFSA. Combining these automata, $(O \circ A \circ D \circ M)(w)$ is thus an automaton that assigns a probability to each word sequence, and the highest-probability path through that automaton estimates the most likely word sequence for the given utterance.

The finite-state modeling for speech recognition that we have just described is hardly novel. In fact, it is equivalent to that presented in [12], in the sense that it generates the same weighted language. However, the transduction cascade approach presented here allows one to view the computations in new ways.

For instance, because composition, $\circ$, is associative, we see that the computation of $\max_w(O \circ A \circ D \circ M)(w)$ can be organized in several ways. A conventional integrated–search, speech recognizer computes $\max_w(O \circ (A \circ D \circ M))(w)$. In other words, the phone, word, and language models are, in effect, compiled together into one large transducer which is then applied to the input observation sequence [12]. On the other hand, one can use a more modular, staged computation, $\max_w(((O \circ A) \circ D) \circ M)(w)$. In other words, first the acoustic observations are transduced into a phone lattice represented as an automaton labeled by phones (phone recog-

nition). This lattice is in turn transduced into a word lattice (word recognition), which is then joined with the language model (language model application) [13].

The best approach may depend on the specific task, which determines the size of intermediate results and the whether finite-state minimization is fruitful. By having a general package to manipulate these automata, we have been able to experiment with various alternatives. For many tasks, the complete network, $O \circ A \circ D \circ M$, is too large to compute explicitly, regardless of the order in which the operations are applied. The solution that is usually taken is to interleave the best path computation with the composition operations and to retain only a portion of the intermediate results by discarding unpromising paths.

So far, our presentation has used context-independent phone models. In other words, the likelihoods assigned by a phone model in A assumed conditional independence from neighboring phones. However, it has been shown that context-dependent phone models, which model a phone in the context of its adjacent phones, are very effective for improving recognition performance [14].

We can include context-dependent models, such as triphone models, in our presentation by expanding our 'atomic models' in A to one for every phone in a distinct triphonic context. Each model will have the same form as in Figure 2b, but will have different likelihoods for the different contexts. We could also try to directly specify D in terms of the new units, but this is problematic. First, even if each word in D had only one phonetic realization, we could not directly substitute its spelling in terms of context-dependent units, since the cross-word units must be specified (because of the closure operation). In this case, a common approach is to either use left (right) context-independent units at the word starts (ends), or to build a fully context-dependent lexicon, but have special computations that insure the correct models are used at word junctures. In either case, this disallows use of phonetic networks as in Figure 2c.

There is, however, a natural solution to these problems using a a finite-state transduction. We leave D as defined before, but interpose a new transduction, C, between A and D, to convert between context-dependent and context-independent units. In other words, we now compute $\max_w(O \circ A \circ C \circ D \circ M)(w)$.

The form of C for triphonic models is depicted in Figure 2d. For each context-dependent phone model, γ, which corresponds to the (context-independent) phone π_c in the context of π_l and π_r, there is a state q_{lc} in C for the biphone $\pi_l\pi_c$, a state q_{cr} for $\pi_c\pi_r$ and a transition from q_{lc} to q_{cr} with input label γ and output label π_r. We have constructed such a transducer and have been able to easily convert context-independent phonetic networks into context-dependent networks for certain tasks. In those cases, we can implement full-context dependency with no special-purpose computations.

4. Chinese Text Segmentation

Our second application is to text processing, namely the tokenization of Chinese text into words, and the assignment of pronunciations to those words. In Chinese orthography, most characters represent (monosyllabic) *morphemes*, and as in English, *words* may consist of one or more morphemes. Given that Chinese does not use whitespace to delimit words, it is necessary to 'reconstruct' the grouping of characters into words. For example, we want to say that the sentence 日文章魚怎麼說 "How do you say octopus in Japanese?", consists of four words, namely 日文 *ri4-wen2* 'Japanese', 章魚 *zhang1-yu2* 'octopus', 怎麼 *zen3-mo* 'how', and 說 *shuo1* 'say'. The problem with this sentence is that 日 *ri4* is also a word (e.g. a common abbreviation for Japan) as are 文章 *wen2-zhang1* 'essay', and 魚 *yu2* 'fish', so there is not a unique segmentation.

The task of segmenting and pronouncing Chinese text is naturally thought of as a transduction problem. The Chinese dictionary[3] is represented as a WFST D. The input alphabet is the set of Chinese characters, and the output alphabet is the union of the set of Mandarin syllables with the set of part-of-speech labels. A given word is represented as a sequence of character-to-syllable transitions, terminated in an ϵ-to-part-of-speech transition weighted by an estimate of the negative log probability of the word. For instance, the word 章魚 'octopus' would be represented as the sequence of transductions 章:*zhang1*/0.0 魚:*yu2*/0.0 ϵ:noun/13.18. A dictionary in this form can easily be minimized using standard algorithms.

An input sentence is represented as an unweighted acceptor S, with characters as transition labels. Segmentation is then accomplished by finding the lowest weight string in $S \circ D^*$. The result is a string with the words delimited by part-of-speech labels and marked with their pronunciation. For the example at hand, the best path is the correct segmentation, mapping the input sequence 日文ϵ章魚ϵ怎麼ϵ說ϵ to the sequence *ri4 wen2* noun *zhang1 yu2* noun *zen3 mo* adv *shuo1* verb.

As is the case with English, no Chinese dictionary covers all of the words that one will encounter in Chinese text. For example, many words that are derived via productive morphological processes are not generally to be found in the dictionary. One such case in Chinese involves words derived via the nominal plural affix 們 *-men*. While some words in 們 will be found in the dictionary (e.g., 他們 *ta1-men* 'they'; 人們 *ren2-men* 'people'), many attested instances will not: for example, 將們 *jiang4-men* '(military) generals', 青蛙們 *qing1-wa1-men* 'frogs'. Given that the basic dictionary is represented as a finite-state automaton, it is a simple matter to augment the model just described with standard techniques from finite-state morphology ([15, 16], inter alia). For in-

[3]We are currently using the 'Behavior Chinese-English Electronic Dictionary', Copyright Number 112366, from Behavior Design Corporation, R.O.C.; we also wish to thank United Informatics, Inc., R.O.C. for providing us with the Chinese text corpus that we used in estimating lexical probabilities. Finally we thank Dr. Jyun-Sheng Chang for kindly providing us with Chinese personal name corpora.

stance, we can represent the fact that 們 attaches to nouns by allowing ϵ-transitions from the final states of noun entries, to the initial state of a sub-transducer containing 們. However, for our purposes it is not sufficient merely to represent the morphological decomposition of (say) plural nouns, since we also want to estimate the cost of the resulting words. For derived words that occur in our corpus we can estimate these costs as we would the costs for an underived dictionary entry. So, 將們 *jiang4-men* '(military) generals' occurs and we estimate its cost at 15.02; we include this word by allowing an ϵ-transition between 將 and 們, with a cost chosen so that the entire analysis of 將們 ends up with a cost of 15.02. For non-occurring possible plural forms (e.g., 南瓜們 *nan2-gua1-men* 'pumpkins') we use the Good-Turing estimate (e.g. [17]), whereby the aggregate probability of previously unseen members of a construction is estimated as N_1/N, where N is the total number of observed tokens and N_1 is the number of types observed only once; again, we arrange the automaton so that noun entries may transition to 們, and the cost of the whole (previously unseen) construction comes out with the value derived from the Good-Turing estimate.

Another large class of words that are generally not to be found in the dictionary are Chinese personal names: only famous names like 周恩來 'Zhou Enlai' can reasonably be expected to be in a dictionary, and even many of these are missing. Full Chinese personal names are *formally* simple, being always of the form FAMILY+GIVEN. The FAMILY name set is restricted: there are a few hundred single-character FAMILY names, and about ten double-character ones. Given names are most commonly two characters long, occasionally one-character long: there are thus four possible name types. The difficulty is that GIVEN names can consist, in principle, of any character or pair of characters, so the possible GIVEN names are limited only by the total number of characters, though some characters are certainly far more likely than others. For a sequence of characters that is a *possible* name, we wish to assign a probability to that sequence *qua* name. We use a variant of an estimate proposed in [18]. Given a potential name of the form F1 G1 G2, where F1 is a legal FAMILY name and G1 and G2 are Chinese characters, we estimate the probability of that name as the product of the probability of finding *any* name in text; the probability of F1 as a FAMILY name; the probability of the first character of a double GIVEN name being G1; the probability of the second character of a double GIVEN name being G2; and the probability of a name of the form SINGLE-FAMILY+DOUBLE-GIVEN. The first probability is estimated from a count of names in a text database, whereas the last four probabilities are estimated from a large list of personal names. This model is easily incorporated into the segmenter by building a transducer restricting the names to the four licit types, with costs on the transitions for any particular name summing to an estimate of the cost of that name. This transducer is then summed with the transducer implementing the dictionary and morphological rules, and the transitive closure of the resulting transducer computed.

References

1. M. A. Harrison, *Introduction to Formal Language Theory*. Reading, Massachussets: Addison-Wesley, 1978.
2. J. Berstel, *Transductions and Context-Free Languages*. No. 38 in Leitfäden der angewandten Mathematik and Mechanik LAMM, Stuttgart, Germany: Teubner Studienbücher, 1979.
3. R. Teitelbaum, "Context-free error analysis by evaluation of algebraic power series," in *Proc. Fifth Annual ACM Symposium on Theory of Computing*, (Austin, Texas), pp. 196–199, 1973.
4. B. Lang, "A generative view of ill-formed input processing," in *ATR Symposium on Basic Research for Telephone Interpretation*, (Kyoto, Japan), Dec. 1989.
5. A. Paz, *Introduction to Probabilistic Automata*. Academic, 1971.
6. T. R. Booth and R. A. Thompson, "Applying probability measures to abstract languages," *IEEE Trans. Computers*, vol. C-22, pp. 442–450, May 1973.
7. S. Eilenberg, *Automata, Languages, and Machines*, vol. A. San Diego, California: Academic Press, 1974.
8. W. Kuich and A. Salomaa, *Semirings, Automata, Languages*. No. 5 in EATCS Monographs on Theoretical Computer Science, Berlin, Germany: Springer-Verlag, 1986.
9. J. Berstel and C. Reutenauer, *Rational Series and Their Languages*. No. 12 in EATCS Monographs on Theoretical Computer Science, Berlin, Germany: Springer-Verlag, 1988.
10. R. M. Kaplan and M. Kay, "Regular models of phonological rule systems," *Computational Linguistics*, 1994. To appear.
11. E. Roche, *Analyse Syntaxique Transformationelle du Français par Transducteurs et Lexique-Grammaire*. PhD thesis, Université Paris 7, 1993.
12. L. R. Bahl, F. Jelinek, and R. Mercer, "A maximum likelihood approach to continuous speech recognition," *IEEE Trans. PAMI*, vol. 5, pp. 179–190, Mar. 1983.
13. A. Ljolje and M. D. Riley, "Optimal speech recognition using phone recognition and lexical access," in *Proceedings of ICSLP*, (Banff, Canada), pp. 313–316, Oct. 1992.
14. K.-F. Lee, "Context dependent phonetic hidden Markov models for continuous speech recognition," *IEEE Trans. ASSP*, vol. 38, pp. 599–609, Apr. 1990.
15. K. Koskenniemi, *Two-Level Morphology: a General Computational Model for Word-Form Recognition and Production*. PhD thesis, University of Helsinki, Helsinki, 1983.
16. E. Tzoukermann and M. Liberman, "A finite-state morphological processor for Spanish," in *COLING-90, Volume 3*, pp. 3: 277–286, COLING, 1990.
17. K. W. Church and W. Gale, "A comparison of the enhanced Good-Turing and deleted estimation methods for estimating probabilities of English bigrams," *Computer Speech and Language*, vol. 5, no. 1, pp. 19–54, 1991.
18. J.-S. Chang, S.-D. Chen, Y. Zheng, X.-Z. Liu, and S.-J. Ke, "Large-corpus-based methods for Chinese personal name recognition. (In Chinese])," *Journal of Chinese Information Processing*, vol. 6, no. 3, pp. 7–15, 1992.