Learning Bias and Phonological-Rule Induction

Daniel Gildea[*]
International Computer Science Institute
& University of California at Berkeley

Daniel Jurafsky[†]
International Computer Science Institute
& University of Colorado at Boulder

A fundamental debate in the machine learning of language has been the role of prior knowledge in the learning process. Purely nativist approaches, such as the Principles and Parameters model, build parameterized linguistic generalizations directly into the learning system. Purely empirical approaches use a general, domain-independent learning rule (Error Back-Propagation, Instance-based Generalization, Minimum Description Length) to learn linguistic generalizations directly from the data.

In this paper we suggest that an alternative to the purely nativist or purely empiricist learning paradigms is to represent the prior knowledge of language as a set of abstract learning biases, *which guide an empirical inductive learning algorithm. We test our idea by examining the machine learning of simple Sound Pattern of English (SPE)-style phonological rules. We represent phonological rules as finite-state transducers that accept underlying forms as input and generate surface forms as output. We show that OSTIA, a general-purpose transducer induction algorithm, was incapable of learning simple phonological rules like flapping. We then augmented OSTIA with three kinds of learning biases that are specific to natural language phonology, and that are assumed explicitly or implicitly by every theory of phonology:* faithfulness *(underlying segments tend to be realized similarly on the surface),* community *(similar segments behave similarly), and* context *(phonological rules need access to variables in their context). These biases are so fundamental to generative phonology that they are left implicit in many theories. But explicitly modifying the OSTIA algorithm with these biases allowed it to learn more compact, accurate, and general transducers, and our implementation successfully learns a number of rules from English and German. Furthermore, we show that some of the remaining errors in our augmented model are due to implicit biases in the traditional SPE-style rewrite system that are not similarly represented in the transducer formalism, suggesting that while transducers may be formally equivalent to SPE-style rules, they may not have identical evaluation procedures.*

Because our biases were applied to the learning of very simple SPE-style rules, and to a non-psychologically-motivated and nonprobabilistic theory of purely deterministic transducers, we do not expect that our model as implemented has any practical use as a phonological learning device, nor is it intended as a cognitive model of human learning. Indeed, because of the noise and nondeterminism inherent to linguistic data, we feel strongly that stochastic algorithms for language induction are much more likely to be a fruitful research direction. Our model is rather intended to suggest the kind of biases that may be added to other empiricist induction models, and the way in which they may be added, in order to build a cognitively and computationally plausible learning model for phonological rules.

---

[*] 1947 Center Street, Berkeley, CA 94704. E-mail: gildea@cs.berkeley.edu
[†] Department of Linguistics, Boulder, CO 80302

1. Introduction

A fundamental debate in the machine learning of language has been the role of prior knowledge in the learning process. Nativist models suggest that learning in a complex domain like natural language requires that the learning mechanism either have some previous knowledge about language, or some learning bias that helps direct the formation of correct generalizations. In linguistics, theories of such prior knowledge are referred to as Universal Grammar (UG); nativist linguistic models of learning assume, implicitly or explicitly, that some kind of prior knowledge that contributes to language learning is innate, a product of evolution. Despite sharing this assumption, nativist researchers disagree strongly about the exact constitution of this Universal Grammar. Many models, for example, assume that much of the prior knowledge that children bring to bear in learning language is not linguistic at all, but derives from constraints imposed by our general cognitive architecture. Others, such the influential Principles and Parameters model (Chomsky 1981), assert that what is innate is linguistic knowledge itself, and that the learning process consists mainly of searching for the values of a relatively small number of parameters. Such nativist models of phonological learning include, for example, Dresher and Kaye's (1990) model of the acquisition of stress-assignment rules, and Tesar and Smolensky's (1993) model of learning in Optimality Theory.

Other scholars have argued that a purely nativist, parameterized learning algorithm is incapable of dealing with the noise, irregularity, and great variation of human language data, and that a more empiricist learning paradigm is possible. Such data-driven models include the stress acquisition models of Daelemans, Gillis, and Durieux (1994) (an application of Instance-based Learning [Aha, Kibler, and Albert 1991]) and Gupta and Touretzky (1994) (an application of Error Back-Propagation), as well as Ellison's (1992) Minimum-Description-Length-based model of the acquisition of the basic concepts of syllabicity and the sonority hierarchy. In each of these cases a general, domain-independent learning rule (BP, IBL, MDL) is used to learn directly from the data.

In this paper we suggest that an alternative to the purely nativist or purely empiricist learning paradigms is to represent the prior knowledge of language as a set of abstract **learning biases**, which guide an empirical inductive learning algorithm. Such biases are implicit, for example, in the work of Riley (1991) and Withgott and Chen (1993), who induced decision trees to predict the realization of a phone in its context. By initializing the decision-tree inducer with a set of phonological features, they essentially gave it a priori knowledge about the kind of phonological generalizations that the system might be expected to learn.

Our idea is that abstract biases from the domain of phonology, whether innate (i.e., part of UG) or merely learned prior to the learning of rules, can be used to guide a domain-independent empirical induction algorithm. We test this idea by examining the machine learning of simple Sound Pattern of English (SPE)-style phonological rules (Chomsky and Halle 1968), beginning by representing phonological rules as finite-state transducers that accept underlying forms as input and generate surface forms as output. Johnson (1972) first observed that traditional phonological rewrite rules can be expressed as regular (finite-state) relations if one accepts the constraint that no rule may reapply directly to its own output. This means that finite-state transducers (FSTs) can be used to represent phonological rules, greatly simplifying the problem of parsing the output of phonological rules in order to obtain the underlying, lexical forms (Koskenniemi 1983; Karttunen 1993; Pulman and Hepple 1993; Bird 1995; Bird and Ellison 1994). The fact that the weaker generative capacity of FSTs makes them easier to

learn than arbitrary context-sensitive rules has allowed the development of a number of learning algorithms including those for deterministic finite-state automata (FSAs) (Freund et al. 1993), deterministic transducers (Oncina, García, and Vidal 1993), as well as nondeterministic (stochastic) FSAs (Stolcke and Omohundro 1993; Stolcke and Omohundro 1994; Ron, Singer, and Tishby 1994). Like the empiricist models discussed above, these algorithms are all general-purpose; none include any domain knowledge about phonology, or indeed natural language; at most they include a bias toward simpler models (like the MDL-inspired algorithms of Ellison [1992]).

Our experiments were based on the OSTIA (Oncina, García, and Vidal 1993) algorithm, which learns general **subsequential finite-state transducers** (SFSTs; formally defined in Section 2). We presented pairs of underlying and surface forms to OSTIA, and examined the resulting transducers. Although OSTIA is capable of learning arbitrary SFSTs in the limit, large dictionaries of actual English pronunciations did not give enough samples to correctly induce phonological rules.

We then augmented OSTIA with three kinds of learning biases, which are specific to natural language phonology, and are assumed explicitly or implicitly by every theory of phonology: **faithfulness** (underlying segments tend to be realized similarly on the surface), **community** (similar segments behave similarly), and **context** (phonological rules need access to variables in their context). These biases are so fundamental to generative phonology that they are left implicit in many theories. But explicitly modifying the OSTIA algorithm with these biases allowed it to learn more compact, accurate, and general transducers, and our implementation successfully learns a number of rules from English and German. The algorithm is also successful in learning the composition of multiple rules applied in series. The more difficult problem of decomposing the learned underlying/surface correspondences into simple, individual rules remains unsolved.

Our transducer induction algorithm is not intended as a cognitive model of human phonological learning. First, for reasons of simplicity, we base our model on simple segmental SPE-style rules; it is not clear what the formal correspondence is of these rules to the more recent theoretical machinery of phonology (e.g., optimality constraints). Second, we assume that a cognitive model of automaton induction would be more stochastic and hence more robust than the OSTIA algorithm underlying our work.[1]

Rather, our model is intended to suggest the kind of biases that may be added to empiricist induction models to build a learning model for phonological rules that is cognitively and computationally plausible. Furthermore, our model is not necessarily nativist; these biases may be innate, but they may also be the product of some other earlier learning algorithm, as the results of Ellison (1992) and Brown et al. (1992) suggest (see Section 5.2). So our results suggest that assuming in the system some very general and fundamental properties of phonological knowledge (whether innate or previously learned) and learning others empirically may provide a basis for future learning models.

Ellison (1994), for example, has shown how to map the optimality constraints of Prince and Smolensky (1993) to finite-state automata; given this result, models of

---

1 Although our assumption of the simultaneous presentation of surface and underlying forms to the learner may seem at first glance to be unnatural as well, it is quite compatible with certain theories of word-based morphology. For example, in the word-based morphology of Aronoff (1976), word-formation rules apply only to already existing words. Thus the underlying form for any morphological rule must be a word of the language. Even if this word-based morphology assumption holds only for a subset of the language (see e.g., Orgun [1995]) it is not unreasonable to assume that a part of the learning process will involve previously-identified underlying/surface pairs.

automaton induction enriched in the way we suggest may contribute to the current debate on optimality learning. This may obviate the need to build in every phonological constraint, as for example nativist models of OT learning suggest (Prince and Smolensky 1993; Tesar and Smolensky 1993; Tesar 1995). We hope in this way to begin to help assess the role of computational phonology in answering the general question of the necessity and nature of linguistic innateness in learning.

The next sections (2 and 3) introduce the idea of representing phonological rules with transducers, and describe the OSTIA algorithm for inducing such transducers. Section 4 shows that the unaugmented OSTIA algorithm is unable to induce the correct transducer for the simple **flapping** rule of American English. Section 5 then describes each of the augmentations to OSTIA, based on the faithfulness, community, and context principles. We conclude with some observations about computational complexity and the inherent bias of the context-sensitive rewrite-rule formalism.

2. Transducer Representation

Rule-based variation in phonology has traditionally been represented with context-sensitive rewrite rules. For example, in American English an underlying t is realized as a **flap** (a tap of the tongue on the alveolar ridge) after a stressed vowel and zero or more r's, and before an unstressed vowel. In the rewrite-rule formalism of Chomsky and Halle (1968), this rule would be represented as in (1).

(1) $t \rightarrow dx / \acute{V} r^* \_ V$

Since Johnson's (1972) work, researchers have proposed a number of different ways to represent such phonological rules by transducers. The most popular method is the **two-level** formalism of Koskenniemi (1983), based on Johnson (1972) and the (belatedly published) work of Kaplan and Kay (1994), and various implementations and extensions (summarized and contrasted in Karttunen [1993]). The basic intuition of two-level phonology is that a rule that rewrites an underlying string as a surface string can be implemented as a transducer that reads from an underlying tape and writes to a surface tape. Figure 1 shows an example of a transducer that implements the flapping rule in (1). Each arc has an input symbol and an output symbol, separated by a colon. A single symbol (such as t or V) is a shorthand for a symbol that is the same in the input and output (i.e., t:t or V:V). Either the input or the output symbols can be null; a null input symbol is used for an insertion of a phone; a null output symbol for a deletion. A transduction of an input string to an output string corresponds to a path through the transducer, where the input string is formed by concatenating the input symbols of the arcs taken, and the output string by concatenating the output symbols of the arcs. The transducer's input string is the phonologically underlying form, while the transducer's output is the surface form. A transduction is valid if there is a corresponding path beginning in state 0 and ending in an **accepting state** (indicated by double circles in the figure). Table 1 shows our phone set—an ASCII symbol set based on the ARPA-sponsored ARPAbet alphabet—with the IPA equivalents.

More recently, Bird and Ellison (1994) show that a one-level finite-state automaton can model richer phonological structure, such as the multitier representations of autosegmental phonology. In their model, each tier is represented by a finite-state automaton, and autosegmental association by the synchronization of two automata. This synchronized-automata-based rather than transducer-based model generalizes over the two-level models of Koskenniemi (1983) and Karttunen (1993) but also the three-level models of Lakoff (1993), Goldsmith (1993), and Touretzky and Wheeler (1990).
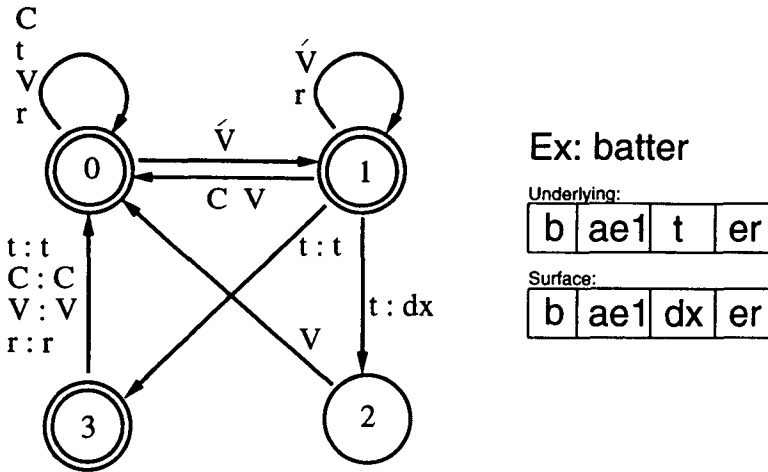
Ex: batter

Underlying:

| b | ae1 | t | er |

Surface:

| b | ae1 | dx | er |

Figure 1
Nondeterministic transducer for English flapping. Labels on arcs are of the form (input symbol):(output symbol). Labels with no colon indicate identical input and output symbols. "V" indicates any unstressed vowel, "V́" any stressed vowel, "dx" a flap, and "C" any consonant other than "t", "r" or "dx".

In order to take advantage of recent work in transducer induction, we have chosen to use the transducer rather than synchronized-automata approach, representing rules as subsequential finite-state transducers (Berstel [1979]; subsequential transducers will be defined below). Since the focus of our research is on adding prior knowledge to help guide an induction algorithm, rather than the particular automaton approach chosen, we expect our results to inform future work on the induction of other types of automata.

Subsequential finite-state transducers are a subtype of finite-state transducers with the following properties:

1. The transducer is deterministic, that is, there is only one arc leaving a given state for each input symbol.

2. Each time a transition is made, exactly one symbol of the input string is consumed.

3. A unique end-of-string symbol is introduced. At the end of each input string, the transducer makes an additional transition on the end-of-string symbol.

4. All states are accepting.

The length of the output string associated with a transition of a subsequential transducer is unconstrained. For our purposes, the key property is the first, because determinism is essential to the state-merging of the OSTIA algorithm. Subsequential transducers are essentially the most general type of deterministic transducers. The second property is merely a convention; any transducer with multiple input symbols on an arc can easily be transformed into one with single arcs with one symbol each. The introduction of an end-of-string symbol serves to expand the range of functions that can be represented. Finally, in a deterministic transducer, there is no need to

Table 1
A slightly expanded ARPAbet phoneset
(including alveolar flap, syllabic nasals and
liquids, and reduced vowels), and the
corresponding IPA symbols. Vowels may be
annotated with the numbers 1 and 2 to
indicate primary and secondary stress,
respectively.

IPA	ARPAbet	IPA	ARPAbet
b	b	p	p
d	d	t	t
g	g	k	k
ɑ	aa	s	s
æ	ae	z	z
ʌ	ah	ʃ	sh
ɔ	ao	ʒ	zh
ɛ	eh	f	f
ɝ	er	v	v
ɩ	ih	θ	th
i	iy	ð	dh
o	ow	tʃ	ch
ʊ	uh	ʒ	jh
u	uw	h	hh
ɑʷ	aw		
ɑʸ	ay	y	y
e	ey	r	r
ɔʸ	oy	w	w
l̩	el	l	l
m̩	em	m	m
n̩	en	n	n
ə	ax	ŋ	ng
ɨ	ix	ɾ	dx
ɚ	axr		

distinguish between accepting and non-accepting states, as there can be no ambiguity about which path is taken through the states.

A **subsequential relation** is any relation between strings that can represented by the input to output relation of a subsequential finite-state transducer. While subsequential relations are formally a subset of regular relations, any relation over a finite input language is subsequential if each input has only one possible output.

A sample phonological rule, the flapping rule for English shown in (1), is repeated in (2a). (2b) shows a positive application of the rule; (2c) shows a case where the conditions for the rule are not met. The rule realizes an underlying t as a flap after a stressed vowel and zero or more r's, and before an unstressed vowel. The subsequential transducer for (2a) is shown in Figure 2.

(2)   a. t → dx / V́ r* __ V

       b. latter: l ae1 t er → l ae1 dx er

       c. laughter: l ae1 f t er → l ae1 f t er

The most significant difference between our subsequential transducers and two-level models is that the two-level transducers described by Karttunen (1993) are non-
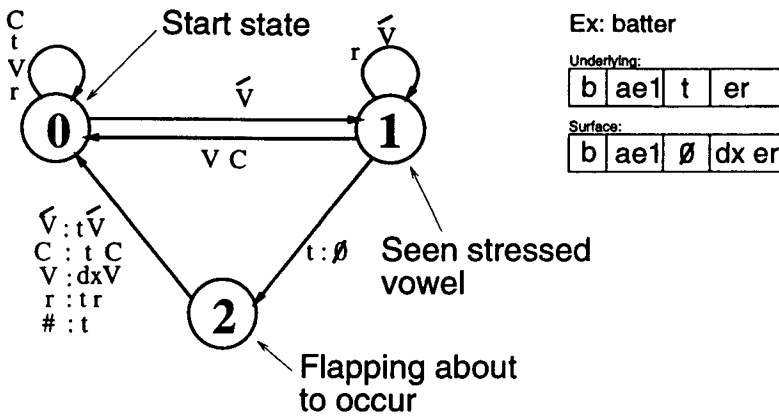
Figure 2
Subsequential transducer for English flapping; "#" is the end-of-string symbol.

deterministic. In addition, Karttunen's transducers may have only zero or one symbol as either the input or output of an arc, and they have no special end-of-string symbol. Finally, his transducers explicitly include both accepting and non-accepting states. All states of a subsequential transducer are valid final states. It is possible for a transduction to fail by finding no next transition to make, but this occurs only on bad input, for which no output string is possible.

These representational differences between the two formalisms lead to different ways of handling certain classes of phonological rules, particularly those that depend on the context to the right of the affected symbol. The subsequential transducer does not emit any output until enough of the right-hand context has been seen to determine how the input symbol is to be realized. Figure 2 shows the subsequential equivalent of Figure 1. This transducer emits no output upon seeing a t when the machine is at state 1. Rather, the machine goes to state 2 and waits to see if the next input symbol is the requisite unstressed vowel; depending on this next input symbol, the machine will emit the t or a dx along with the next input symbol when it makes the transition from state 2 to state 0.

In contrast, the nondeterministic two-level-style transducer shown in Figure 1 has two possible arcs leaving state 1 upon seeing a t, one with t as output and one with dx. If the machine takes the wrong transition, the subsequent transitions will leave the transducer in a non-accepting state, or a state will be reached with no transition on the current input symbol. Either way, the transduction will fail.

Generating a surface form from an underlying form is more efficient with a subsequential transducer than with a nondeterministic transducer, as no search is necessary in a deterministic machine. Running the transducer backwards to parse a surface form into possible underlying forms, however, remains nondeterministic in subsequential transducers. In addition, a subsequential transducer may require many more states than a nondeterministic transducer to represent the same rule. Our reason for choosing subsequential transducers, then, is solely that efficient techniques exist for learning them, as we will see in the next section. In particular, the algorithm we chose is able to learn from only positive evidence. Other algorithms make use of negative evidence in the form of transductions marked as invalid, or questions directed at an informant.
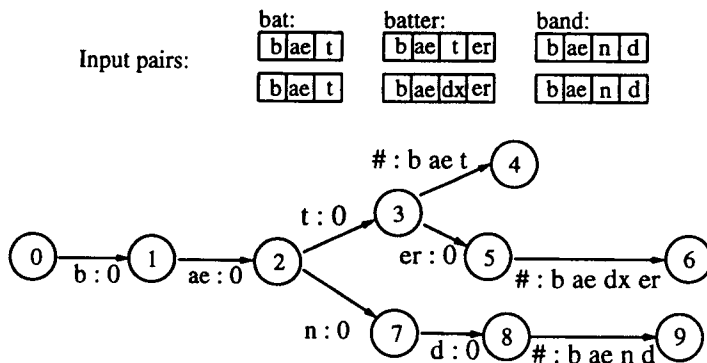
Figure 3
Initial tree transducer for *bat*, *batter*, and *band* with flapping applied.

This use of positive-only evidence is significant for both cognitive reasons (children have been shown to make little use of negative evidence) and practical ones (positive examples, but not negative examples, are easily derived automatically from corpora).

3. The OSTIA Algorithm

Our phonological-rule induction algorithm is based on augmenting the Onward Subsequential Transducer Inference Algorithm (OSTIA) of Oncina, García, and Vidal (1993). This section outlines the OSTIA algorithm to provide background for the modifications that follow; see their original paper for further details.

OSTIA takes as input a training set of valid input-output pairs for the transduction to be learned. The algorithm begins by constructing a tree transducer that covers all the training samples according to the following procedure: for each input pair, the algorithm walks from the initial state taking one transition on each input symbol, as if doing a transduction. When there is no move on the next input symbol from the present state, a new branch is grown on the tree. The entire output string of each transduction is initially stored as the output on the last arc of the transduction, that is, the arc corresponding to the end-of-string symbol. An example of an initial tree transducer constructed by this process is shown in Figure 3.

As the next step, the output symbols are "pushed forward" as far as possible towards the root of the tree. This process begins at the leaves of the tree and works its way to the root. At each step, the longest common prefix of the outputs on all the arcs leaving one state is removed from the output strings of all the arcs leaving the state and suffixed to the (single) arc entering the state. This process continues until the longest common prefix of the outputs of all arcs leaving each state is the null string—the definition of an **onward** transducer. The result of making the transducer of Figure 3 onward is shown in Figure 4.

At this point, the transducer covers all and only the strings of the training set. OSTIA now attempts to generalize the transducer, by merging some of its states together. For each pair of states (s, t) in the transducer, the algorithm will attempt to merge s with t, building a new state with all of the incoming and outgoing transitions of s and t. The result of the first merging operation on the transducer of Figure 4 is shown in Figure 5.

A conflict arises whenever two states are merged that have outgoing arcs with the same input symbol. When this occurs, an attempt is made to merge the destination
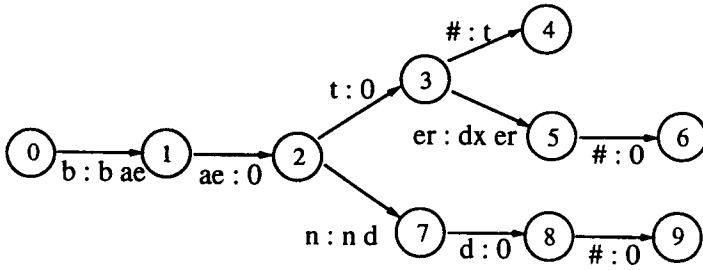
Figure 4
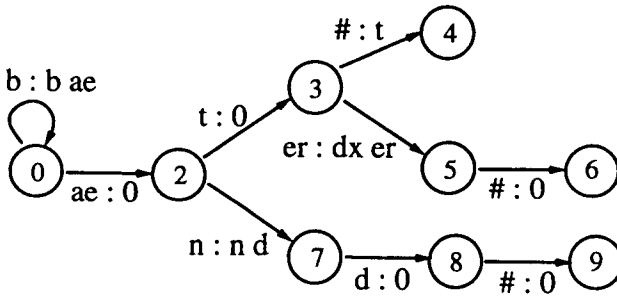Onward tree transducer for *bat, batter,* and *band* with flapping applied.

Figure 5
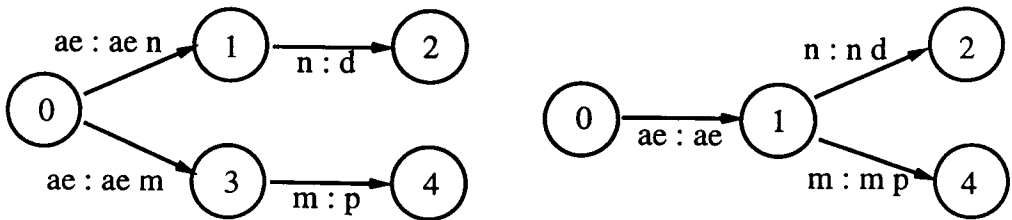Result of merging states 0 and 1 of Figure 4.

Figure 6
Example push-back operation and state merger. Input words *and* and *amp.*

states of the two conflicting arcs. First, all output symbols beyond the longest common prefix of the outputs of the two arcs are "pushed back" to arcs further down the tree. This operation is only allowed under certain conditions that guarantee that the transductions accepted by the machine are preserved. The push-back operation allows the two arcs to be combined into one and their destination states to be merged. An example of a push-back operation and subsequent merger on a transducer for the words *and* and *amp* is shown in Figure 6. This method of resolving conflicts repeats until no conflicts remain, or until resolution is impossible. In the latter case, the transducer is restored to its configuration before the merger causing the original conflict, and the algorithm proceeds by attempting to merge the next pair of states.

Table 2
Unmodified OSTIA learning flapping on 49,280-word test set. Error rates are the percentage of incorrect transductions.

Samples	States	% Error
6,250	19	2.32
12,500	257	16.40
25,000	141	4.46
50,000	192	3.14

4. Problems Using OSTIA to Learn Phonological Rules

The OSTIA algorithm can be proven to learn any subsequential relation in the limit. That is, given an infinite sequence of valid input/output pairs, it will at some point derive the target transducer from the samples seen so far. When trying to learn phonological rules from finite linguistic data, however, we found that the algorithm was unable to learn a correct, minimal transducer.

We tested the algorithm using a synthetic corpus of 99,279 input/output pairs. Each pair consisted of an underlying pronunciation of an individual word of English and a machine generated "surface pronunciation." The underlying string of each pair was taken from the phoneme-based CMU pronunciation dictionary (CMU 1993). The surface string was generated from each underlying form by mechanically applying the one or more rules we were attempting to induce in each experiment.

In our first experiment, we applied the flapping rule (repeated again in (3)) to training corpora of between 6,250 and 50,000 words. Figure 7 shows the transducer induced from 25,000 training samples, and Table 2 shows some performance results. For obvious reasons we have left off the labels on the arcs in Figure 7. The only difference between underlying and surface forms in both the training and test sets in this experiment is the substitution of dx for a t in words where flapping applies. Therefore, inaccuracies in predicting output strings represent real errors in the transducer, rather than manifestations of other phonological phenomena.

(3) $t \rightarrow dx \; / \; \acute{V} \; r^* \; \_\_ \; V$

Figure 7 and Table 2 show OSTIA's failure to learn the simple flapping rule. Recall that the optimal transducer, shown in Figure 2, has only 3 states, and would have no error on the test set of synthetic data. OSTIA's induced transducer not only is much more complex (between 19 and 257 states) but has a high percentage of error. In addition, giving the model more training data does not seem to help it induce a smaller or better model; the best transducer was the one with the smallest number of training samples.

Since OSTIA can learn any subsequential relation in the limit, why these difficulties with the phonological-rule induction task? The key provision here, of course, is "the limit"; we are clearly not giving OSTIA sufficient training data. There are two reasons this data may not be present in any reasonable training set. First, the necessary number of sample transductions may be several times the size of any natural language's vocabulary. Thus even the entire vocabulary of a language may be insuffi-
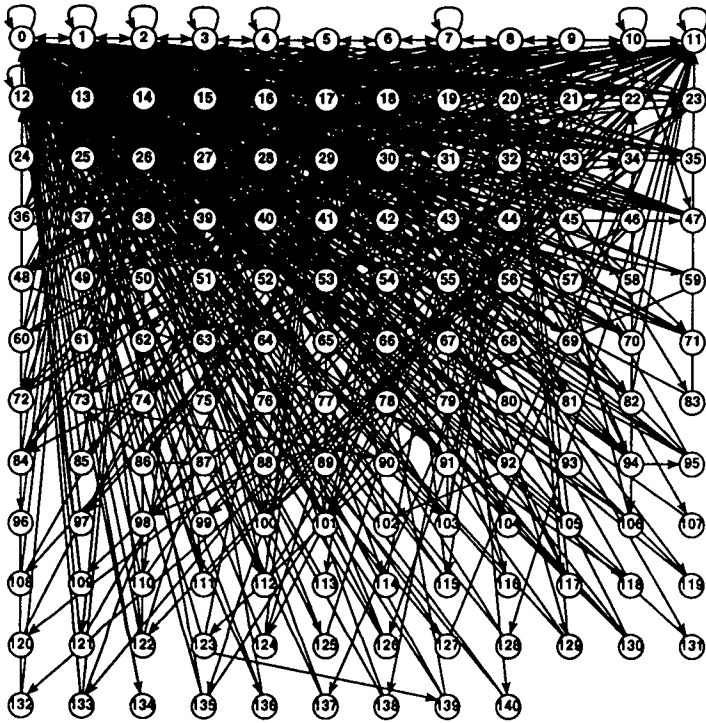
Figure 7
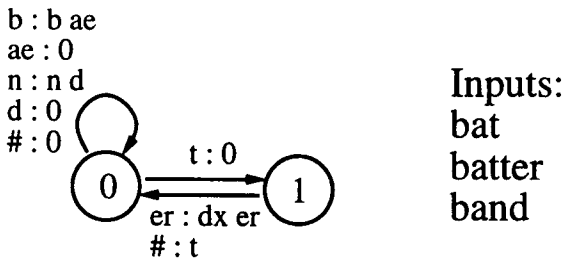First attempt of OSTIA to learn flapping. Transducer induced on 25,000 samples.

Figure 8
Final result of merging process on transducer from Figure 4.

cient in size to learn an efficient or correct transducer. Second, even if the vocabulary were larger, the necessary sample may require types of strings that are not found in the language for phonotactic or other reasons. Systematic phonological constraints such as syllable structure may make it impossible to obtain the set of examples that would be necessary for OSTIA to learn the target rule. For example, given one training set of examples of English flapping, the algorithm induced a transducer that realizes an underlying t as dx either in the environment $\acute{V}r^*\_V$ or after a sequence of six consonants. This is possible since such a transducer will accurately cover the training set, as no English words contain six consonants followed by a t. The lack of natural language bias causes the transducer to miss correct generalizations and learn incorrect transductions.

One example of an unnatural induction is shown in Figure 8, the final transducer induced by OSTIA on the three-word training set of Figure 4. OSTIA has a tendency to produce overly "clumped" transducers, as illustrated by the arcs with output b ae and n d in Figure 8, or even Figure 4. The transducer of Figure 8 will insert an ae after any b, and delete any ae from the input. OSTIA's default behavior is to emit the remainder of the output string for a transduction as soon as enough input symbols have been seen to uniquely identify the input string in the training set. This results in machines that may, seemingly at random, insert or delete sequences of four or five segments. This causes the machines to generalize in linguistically implausible ways, i.e., producing output strings incorrectly bearing little relation to their input. In addition, the incorrect distribution of output symbols prevents the optimal merging of states during the learning process, resulting in large and inaccurate transducers. The higher number of states reduces the number of training examples that pass through each state, making incorrect state mergers possible and introducing errors on test data.

A second problem is OSTIA's lack of generalization. The vocabulary of a language is full of accidental phonological gaps. Without an ability to use knowledge about phonological features to generalize across phones, OSTIA's transducers have missing transitions for certain phones from certain states. For example, the transducer of Figure 8 will fail completely upon seeing any symbol other than er or end-of-string after a t. Of course this transducer is only trained on three samples, but the same problem occurs with transducers trained on large corpora.

As a final example, if the OSTIA algorithm is trained on cases of flapping in which the preceding environment is every stressed vowel but one, the algorithm has no way of knowing that it can generalize the environment to all stressed vowels. Again, the algorithm needs knowledge about classes of segments to fill in these accidental gaps in training data coverage.

5. Augmenting the Learner with Phonological Knowledge

In order to give OSTIA the prior knowledge about phonology to deal with the problems in Section 4, we augmented it with three biases, each of which is assumed explicitly or implicitly by most if not all theories of phonology. These biases are intended to express universal constraints about the domain of natural language phonology.

Faithfulness: Underlying segments tend to be realized similarly on the surface.

Community: Phonologically similar segments behave similarly.

Context: Phonological rules need access to variables in their context.

As discussed above, our algorithm is not intended as a direct model of human learning of phonology. Rather, since only by adding these biases was a general-purpose algorithm able to learn phonological rules, and since most theories of phonology assume these biases as part of their model, we suggest that these biases may be part of the prior knowledge or state of the learner.

5.1 Faithfulness
As we saw above, the unaugmented OSTIA algorithm often outputs long clumps of segments when seeing a single input phone. Although each particular clump may be correct for the exact input example that contained it, it is rarely the case in general that a certain segment is invariably followed by a string of six other specific segments. Thus the model will tend to produce errors when it sees this input phone in a similar
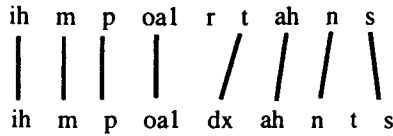
```
ih   m   p   oal   r   t   ah   n   s
 |   |   |    |    /   /   /       \
ih   m   p   oal  dx  ah   n   t   s
```

Figure 9
Alignment of *importance* with flapping, r-deletion and t-insertion.

left context. This behavior is caused by a paucity of training data, but even with a reasonably large training set, we found it was often the case that some particular strings of segments happened to only occur once.

In order to resolve this problem, and the related cases of arbitrary phone-deletion we saw above, we need to appeal to the fact that theories of generative phonology have always assumed that, all things being equal, surface forms tend to resemble underlying forms. This assumption was implicit, for example, in Chomsky and Halle's (1968) MDL-based evaluation procedure for phonological rule systems. They ranked the "value" of a grammar by the inverse of the number of symbols in the system. According to this metric, clearly, a grammar that does not contain "trivial" rules mapping an underlying phonology unit to an identical unit on the surface is preferable to an otherwise identical grammar that has such rules. Later work in Autosegmental Phonology and Feature Geometry extended this assumption by restricting the domain of individual phonological rules to changes in an individual node in a feature-geometric representation.

Recent two-level theories of Optimality Theory (e.g., McCarthy and Prince 1995) make the assumption of faithfulness (which is similar to Chomsky and Halle's) more explicit. These theories propose a constraint called FAITHFULNESS, which requires that the phonological output string match its input. Such a constraint is ranked below all other constraints in the optimality constraint ranking (since otherwise no surface form could be distinct from its underlying form), and is used to rule out the infinite set of candidates produced by GEN that bear no relation to the underlying form. Computational models of morphology have made use of a similar faithfulness bias. Ling (1994), for example, applied a faithfulness heuristic (called passthrough) as a default in a ID3-based decision-tree induction system for learning the past tense of English verbs. Orgun (1996) extends the two-level optimality-theoretic concept of faithfulness to require a kind of monotonicity from the underlying to the surface form: his MATCH constraint requires that every element of an output string contain all the information in the corresponding element of an input string.

Our model of faithfulness preserves the insight that, barring a specific phonological constraint to the contrary, an underlying element will be identical to its surface correspondent. But like Orgun's version, our model extends this bias to suggest that, all things being equal, a changed surface form will also be close to its underlying form in phonological feature space. In order to implement such a faithfulness bias in OSTIA, our algorithm guesses the most probable segment-to-segment alignment between the input and output strings, and uses this information to distribute the output symbols among the arcs of the initial tree transducer. This is demonstrated for the word *importance* in Figures 9 and 10.

This new distribution of output symbols along the arcs of the initial tree transducer no longer guarantees the onwardness of the transducer. (Although in fact, the final transducers induced by our new method do tend to be onward.) Onwardness happens

Figure 10
Resulting initial transducer for *importance*.

Table 3
Phonological features used in alignment.

vocalic	*consonant*	*sonorant*	*rhotic*
advanced	*front*	*high*	*low*
back	*rounded*	*tense*	*voiced*
w-offglide	*y-offglide*	*coronal*	*anterior*
distributed	*nasal*	*lateral*	*continuant*
strident	*syllabic*	*silent*	*flap*
stress	*primary-stress*		

to be an invariant of the unmodified OSTIA algorithm, but it is not essential to the working of the algorithm.[2]

Our modification proceeds in two stages: first, a dynamic programming method is used to compute a correspondence between input and output segments, and second, the alignment is used to distribute output symbols on the inital tree transducer.

The alignment is calculated using the algorithm of Wagner and Fischer (1974), which calculates the insertions, deletions, and substitutions that make up the minimum edit distance between the underlying and surface strings. The costs of edit operations are based on phonological features; we used the 26 binary articulatory features in Table 3.

This feature set was chosen merely because it was commonly used in other speech recognition experiments in our laboratory; none of our experiments or results depended in any way on this particular choice of features, or on their binary rather than privative or multivalued nature. For example, the decision-tree pruning algorithm discussed in Section 5.2.2, which successfully generalized about the importance of stressed vowels to the flapping rule, would have functioned identically with any feature set capable of distinguishing stressed from unstressed vowels.

The cost function for substitutions was equal to the number of features changed between the two segments. The cost of insertions and deletions was arbitrarily set at 6 (roughly one quarter the maximum possible substitution cost). From the sequence of edit operations, an alignment between input and output segments is calculated. Due to the shallow nature of the rules in question, the exact parameters used to calculate alignment are not very significant.

When building the initial tree transducer, the alignment is used to ensure that no output symbol appears on an arc further up the tree than the corresponding input symbol. To resolve conflicts between the output symbols for a given arc, symbols may

---

2 No matter what alignment is used, we are guaranteed that at least the correspondence learned will be some generalization that preserves the behavior of the training set. For the theoretical property of language identification in the limit, we must be guaranteed that the alignments used are correct: that is, the alignment must not show an output symbol to correspond to an input symbol that comes after the input symbol that, in the target transducer, generates the output symbol. This is because, while output symbols can be pushed back, the state-merging process cannot push the symbols forward if the alignment has caused them to be placed too far down the tree. For the shallow rules examined in this paper, finding the correct alignment is trivial.
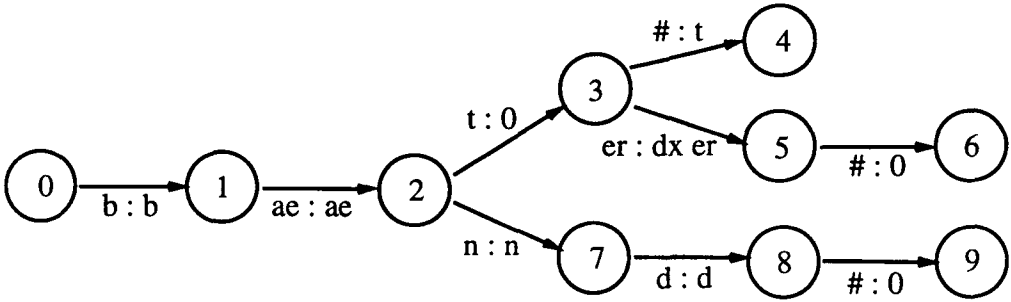
Figure 11
Initial tree transducer constructed with alignment information. Note that output symbols have been pushed back across state 3 during the construction.
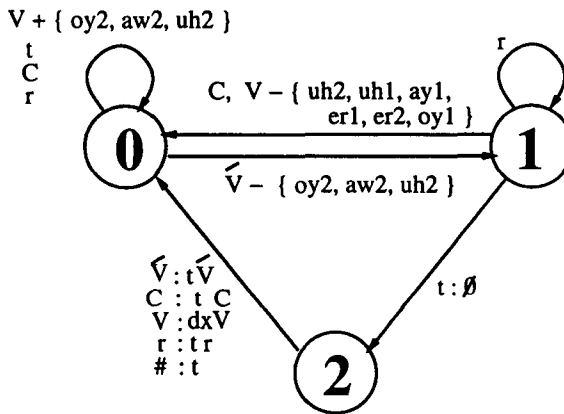

Figure 12
Flapping transducer induced with alignment, trained on 25,000 samples.

be pushed back down the tree as is done when merging states. The exact process used to build the initial tree transducer is described below.

When adding a new arc to the tree, all the unused output segments up to and including those that map to the arc's input segment become the new arc's output, and are now marked as having been used. When walking down branches of the tree to add a new input/output sample, we calculate the longest common prefix, n, of the sample's unused output and the output of each arc along the path. The next n symbols of the transduction's output are now marked as having been used. If the length, l, of the arc's output string is greater than n, it is necessary to push back the last $l - n$ symbols onto arcs further down the tree. A tree transducer constructed by this process is shown in Figure 11, for comparison with the unaligned version in Figure 4.

The final transducer produced with the alignment algorithm is shown in Figure 12. Purely to make the diagram easier to read we have used C and V to represent the set of consonants and of vowels on the arcs' labels. It is important to note that the learning algorithm did not have any knowledge of the concepts of vowel and consonant, other than through the features used to calculate alignment.

The size and accuracy of the transducers produced by the alignment algorithm are summarized in Table 4. Note that the use of alignment information in creating the initial tree transducer dramatically decreases the number of states in the learned

Table 4
Results using alignment information on English flapping.

Samples	OSTIA without Alignment		OSTIA with Alignment	
	States	% Error	States	% Error
6,250	19	2.32	3	0.34
12,500	257	16.40	3	0.14
25,000	141	4.46	3	0.06
50,000	192	3.14	3	0.01

Table 5
Results on r-deletion using
alignment information.

Samples	r-deletion	
	States	% Error
6,250	4	0.48
12,500	3	0.21
25,000	6	0.18
50,000	35	0.30

transducer as well as the error performance on test data. The improved algorithm induced a flapping transducer with the minimum number of states (3) with as few as 6,250 samples.

The use of alignment information also reduced the learning time; the additional cost of calculating alignments is more than compensated for by quicker merging of states. There was still a small amount of error in the final transducer, and in the next section we show how this remaining error was reduced still further.

The algorithm also successfully induced transducers with the minimum number of states for the t-insertion and t-deletion rules in (5) and (6), given only 6,250 samples. For the r-deletion rule in (4), the algorithm induced a machine that was not the theoretical minimal machine (3 states), as Table 5 shows. We discuss these results below.

(4) $r \rightarrow \emptyset / [+vocalic] \_ [+consonantal]$

(5) $\emptyset \rightarrow t / n \_ s$

(6) $t \rightarrow \emptyset / n \_ \begin{bmatrix} +vocalic \\ -stress \end{bmatrix}$

In our second experiment, we applied our learning algorithm to a more difficult problem: inducing multiple rules at once. One of the important properties of finite-state phonology is that transducers for two rules can be automatically combined to produce a transducer for the two rules run in series. With our deterministic transducers, the transducers are joined via **composition**. Any ordering relationships are preserved in this composed transducer—the order of the rules corresponds to the order in which

Table 6
Results on three rules composed.

Samples	OSTIA with Alignment	
	States	% Error
6,250	6	0.93
12,500	5	0.20
25,000	5	0.09
50,000	5	0.04

the transducers were composed.[3]

Our goal was to learn such a composed transducer directly from the original underlying and ultimate surface forms. The simple rules we used in our experiment contain no feeding (the output of one rule creating the necessary environment for another rule) or bleeding (a rule deleting the necessary environment, causing another rule not to apply) relationships among rules. Thus the order of their application is not significant. However the learning problem remains unchanged if the rules are required to apply in some particular order.

Setting r-deletion aside for the present, a data set was constructed by applying the t-insertion rule in (5), the t-deletion rule in (6), and the flapping rule already seen in (3) one after another. The minimum number of states for a subsequential transducer performing the composition of the three rules is five. As is seen in Table 6, our algorithm successfully induces a transducer of minimum size given 12,500 or more sample transductions.

5.2 Community
5.2.1 Decision-Tree Induction.
A second class of problems with our baseline OSTIA resulted from a lack of generalization across segments. Any training set of words from a language is likely to be full of accidental phonological gaps. Without an ability to use knowledge about phonological features to generalize across phones, OSTIA's transducers have missing transitions for certain phones from certain states. This causes errors when transducing previously unseen words after training is complete. Consider the transducer in Figure 12, reproduced below as Figure 13.

One class of errors in this transducer is caused by the input "falling off" the model. That is, a transduction may fail because the model has no transition specified from a given state for some phone. This is the case with (7), where there is no transition from state 1 on phone uh2.

(7)   showroom: sh ow1 r uh2 m → sh ow1 r

A second class of errors is caused by an incorrect transition; with (8), for example, the transducer incorrectly fails to flap after oy2 because, upon seeing oy2 in state 0, the machine stays in state 0, rather than making the transition to state 1.

---

3 When using nondeterministic transducers, for example, those of Karttunen described in Section 2, multiple rules are represented by intersecting, rather than composing, transducers. In such a system, for two rules to apply correctly, the output must lie in the intersection of the outputs accepted by the transducers for each rule on the input in question. We have not attempted to create an OSTIA-like induction algorithm for nondeterministic transducers.
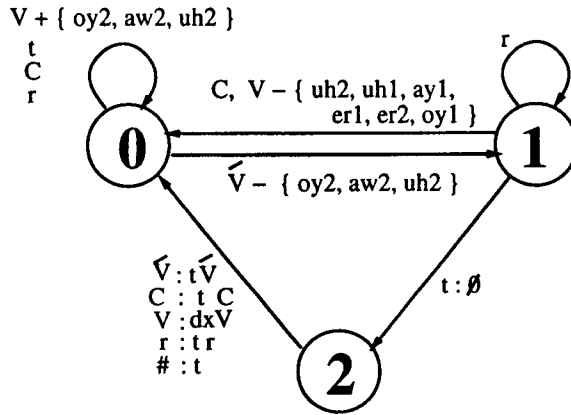
Figure 13
Flapping transducer induced with alignment. For simplicity, some of the phones missing from the transitions from state 2 to 0 and from 1 to 0 have been omitted. For clarity of explication, set-subtraction notation is used to show which vowels do not cause transitions between states 0 and 1.

(8)   exploiting: eh1 k s p l oy2 t ih ng → eh1 k s p l oy2 t ih ng

Both of these problems are caused by insufficiently general labels on the transition arcs in Figure 13. Compare Figure 13 with the correct transducer in Figure 2. We have used set-subtraction notation in Figure 13 to highlight the differences. Notice that in the correct transducer, the arc from state 1 to state 0 is labeled with C and V, while in the incorrect transducer the transition is missing six of the vowels. These vowels were simply never seen at this position in the input.

The intuition that OSTIA is missing, then, is the idea that phonological constraints are sensitive to phonological features that pick out certain equivalence classes of segments. Since the beginning of generative grammar, and based on Jakobson's early insistence on the importance of binary oppositions (Jakobson 1968; Jakobson, Fant, and Halle 1952), phonological features, and not the segment, have generally formed the vocabulary over which linguistic rules are formed. Giving such knowledge to OSTIA would allow it to hypothesize that if every vowel it has seen has acted a certain way, that the rest of them might act similarly.

This phonological feature knowledge may be innate or may merely be learned extremely early. There is a significant body of psychological results, for example, indicating that infants one to four months of age are already sensitive to the phonological oppositions which characterize phonemic contrasts; Eimas et al. (1971), for example, showed that infants were able to distinguish the syllables /ba/ and /pa/, but were unable to distinguish acoustic differences that were of a similar magnitude but that do not form phonemic contrast in any language. Similar studies have shown that this sensitivity appears to be cross-linguistic. But it is by no means necessary to assume that this knowledge is innate. Ellison (1992) showed that a purely empiricist induction algorithm, based on the information-theoretic metric of choosing a minimum-length representation, was able to induce the concepts "V" and "C" in a number of different languages. Promising results from another field of linguistic learning, syntactic part-of-speech induction, suggest that an empiricist approach may be feasible. Brown et al. (1992) used a purely data-driven greedy, incremental clustering algorithm to derive word-classes for *n*-gram grammars; their algorithm successfully induced classes like
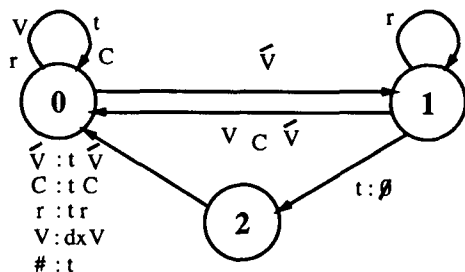
Figure 14
Flapping transducer induced from 50,000 samples.

"days of the week," "male personal name," "body-part noun," and "auxiliary." Only future research will determine whether phonological constraints are innate, or merely learned extremely early, and whether empiricist algorithms like Ellison's will be able to induce a full phonological ontology without them.

Whether phonological features may be innately guided or derived from earlier induction, then, the community bias suggests adding knowledge of them to OSTIA. We did this by augmenting OSTIA to use phonological feature knowledge to generalize the arcs of the transducer, producing transducers that are slightly more general than the ones OSTIA produced in our previous experiments. Our intuition was that these more general transducers would correctly classify stressed vowels together as environments for flapping, and similarly solve other problems caused by gaps in training data.

In the rest of this section we will describe how these generalized transducers are produced and tested. To peek ahead at the results of the algorithm, however, consider Figure 14. The algorithm produced the arcs of Figure 14 by generalizing the arcs from Figure 13 above. The difference is that the arcs in Figure 13 have more general labels.

The mechanism works by applying the standard data-driven decision-tree induction algorithm (based on Quinlan's [1986] ID3 algorithm) to learn a decision tree over the arcs of the transducer. We add prior knowledge to the induction by adding **language bias**; that is, the induction language will use phonological features as a language for making decisions. The resulting decision trees describe the behavior of the machine at a given state in terms of the next input symbol by generalizing from the arcs leaving the state. Since we are generalizing over arcs at a given state of an induced transducer, rather than directly from the original training set of transductions, the input to the ID3 algorithm is limited to the number of phonemes, and is not proportional to the size of the original training set.

We begin by briefly summarizing the decision-tree induction algorithm. A decision tree takes a set of properties that describe an object and outputs a decision about that object. It represents the process of making a decision as a rooted tree, in which each internal node represents a test of the value of a given property, and each leaf node represents a decision. A decision about an object is reached by descending the tree, at each node taking the branch indicated by the object's value for the property at that node. The decision is then read off from the leaf node reached. We will use decision trees to decide what actions and outputs a transducer should produce given certain phonological inputs. Thus the internal nodes of the tree will correspond to tests of the values of phonological features, while the leaf nodes will correspond to state transitions and outputs from the transducer.

The ID3 algorithm is given a set of objects, each labeled with feature values and a decision, and builds a decision tree for a problem given. It does this by iteratively

515

choosing the single feature that best splits the data, i.e., that is the information-theoretically best single predictor of the decision for the samples. A node is built for this feature, and examples are divided into subsets based on their values for it. These values are attached to the new node's children, and the algorithm is run again on the children's subsets, until each leaf node has a set of samples that are all of the same category. Thus for each state in a transducer, we gave the algorithm the set of arcs leaving the state (the samples), the phonological features of the next input symbol (the features), and the output/transition behaviors of the automaton (the decisions). Because we used binary phonological features, we obtained binary decision trees (although we could just as easily have used multivalued features). The alignment information previously calculated between input and output strings is used again in determining which arcs have the same behavior. Two arcs are considered to have the same behavior if the same phonological features have changed between the input segment and the output segment that corresponds to it, and if the preceding and following output segments of the two arcs are identical. The same 26 binary phonological features used in calculating edit distance were used to classify segments in the decision trees. It is worth noting that conflicts in the input to the ID3 algorithm (where the same path to a leaf covers examples that behave differently) are impossible: no two phonemes agree in every feature, and because our transducers are deterministic, there is at most one arc leaving a state labeled with a given input phoneme.

Figure 15 shows a resulting decision tree that generalized the transducer in Figure 13 to avoid the problem of certain inputs "falling off" the transducer. We automatically induced this decision tree from the arcs leaving state 1 in the machine of Figure 13. The outcomes at the leaves of the decision tree specify the output of the next transition to be taken in terms of the input segment, as well as as the transition's destination state. We use square brackets to indicate which phonological features of the input segment are changed in the output; the empty brackets in Figure 15 simply indicate that the output segment is identical to the input segment. Note that if the underlying phone is a t ([-rhotic,-voice,-continuant,-high,+coronal]), the machine jumps to state 2. If the underlying phone is an r, the machine outputs r and goes to state 1. Otherwise, the machine outputs its input and moves to state 0.

Because the decision tree specifies a state transition and an output string for every possible combination of phonological features, one can no longer "fall off" the machine, no matter what the next input segment is. Thus in a transducer built using the newly induced decision tree for state 1, such as the machine in Figure 18, the arc from state 1 to state 0 is taken on seeing any vowel, including the six vowels missing from the arc of the machine in Figure 13.

Our decision trees superficially resemble the organization of phonological features into functionally related classes proposed in the Feature Geometry paradigm (see McCarthy [1988] for a review). Feature-geometric theories traditionally proposed a unique, language-universal grouping of distinctive features to explain the fact that phonological processes often operate on coherent subclasses of the phonological features. For example, facts such as the common cross-linguistic occurrence of rules of nasal assimilation, which assimilate the place of articulation of nasals to the place of the following consonant, suggest a natural class *place* that groups together (at least) the labial and coronal features. The main difference between decision trees and feature geometry trees is the scope of the proposed generalizations; where a decision tree is derived empirically from the environment of a single state of a transducer, feature geometry is often assumed to be unique and universal (although recent work has questioned this assumption; see, for example, Padgett [1995a, b]). Information-theoretic distance metrics similar to those in the ID3 algorithm were used by McCarthy (1988,
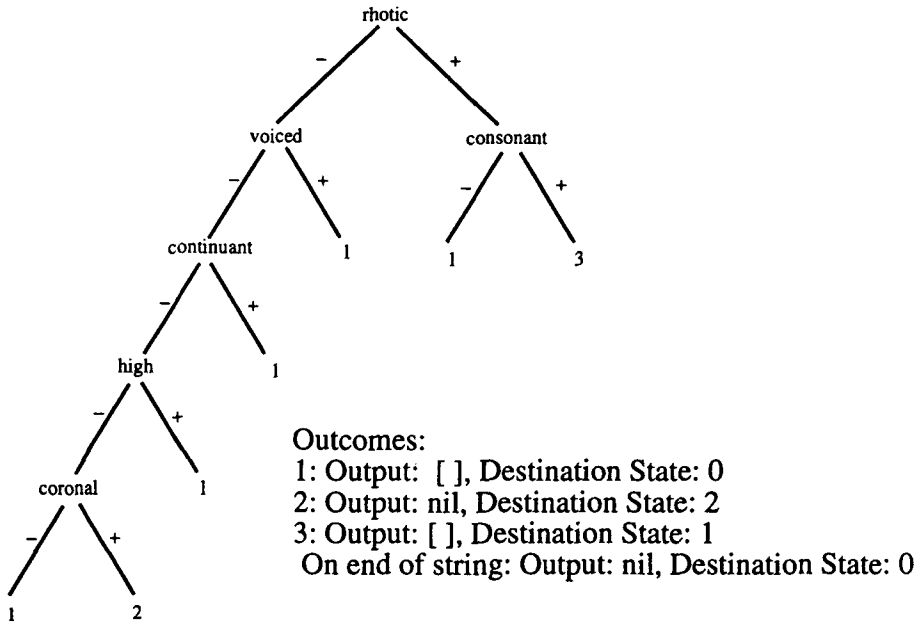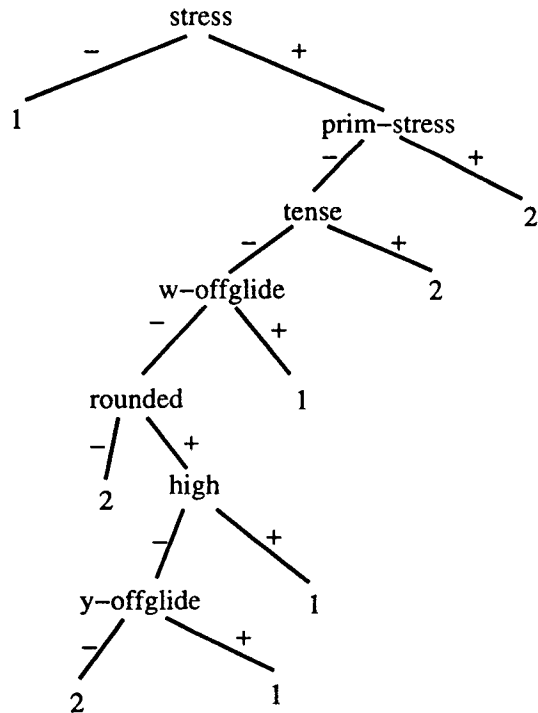
Outcomes:
1: Output: [ ], Destination State: 0
2: Output: nil, Destination State: 2
3: Output: [ ], Destination State: 1
   On end of string: Output: nil, Destination State: 0

Figure 15
Example decision tree. This tree describes the behavior of state 1 of the transducer in Figure 2.
[ ] in the output string indicates the arc's input symbol (with no features changed).

101), who used a cluster analysis on a dictionary of Arabic to argue for a particular feature-geometric grouping; the relationship between feature geometries and empirical classification algorithms like decision trees clearly bears further investigation.

To recapitulate, the transducers induced by OSTIA suffered from undergeneralization in a number of ways. Because OSTIA had no knowledge of similarities among phones, the induced transducer often had no transition specified for a given phone, or had an incorrect one specified. We took the arcs leaving each state of our transducers and used a decision-tree induction algorithm to replace them by a smoother and more general set of arcs. In the next section we show how these arcs were further generalized.

5.2.2 Further Generalization: Decision Tree Pruning. Although inducing decision trees on the arcs of the transducer improved the generalization behavior of our transducers, we found that some transducers needed to be generalized even further. Consider again the English flapping rule, which applies in the context of a preceding stressed vowel. Our algorithm first learned an incorrect transducer whose decision tree for state 0 is shown in Figure 16. In this transducer all arcs leaving state 0 correctly lead to the flapping state on stressed vowels, except for those stressed vowels that happen not to have occurred before an instance of flapping in the training set. For these unseen vowels (which consisted of the vowel uh and the diphthongs oy and ow all with secondary stress), the transducer incorrectly returns to state 0. In this case, we wish the algorithm to make the generalization that the rule applies after *all* stressed vowels.

Again, this correct generalization (all stressed vowels) is expressible as a (single node) decision tree over the phonological features of the input phones. But the key insight is that the current transducer is incorrect because the absence of particular

Outcomes:
1: Output: [ ], Destination State: 0
2: Output: [ ], Destination State: 1
On end of string: Output: nil, Destination State: 0

Figure 16
Decision tree before pruning. The initial state of the flapping transducer.

training patterns (the three particular stressed vowels) caused the decision tree to make a number of complex unnecessary decisions. This problem can be solved by **pruning** the decision trees at each state of the machine. Pruning is done by stepping through each state of the machine and pruning as many branches as possible from the fringe of the current state's decision tree. Each time a branch is pruned, one of the children's outcomes is picked arbitrarily for the new leaf, and the entire training set of transductions is tested to see if the new transducer still produces the right output. As discussed in Section 6, this is computationally quite expensive. If any errors are found, testing is repeated using the outcome of the pruned node's other child (e.g., the leaf with the positive rather than negative value for the feature being tested at the pruned node). If errors are still found, the pruning operation is undone. This process continues at the fringe of the decision tree until no more pruning is possible. Figure 17 shows the correct decision tree for flapping, obtained by pruning the tree in Figure 16.

The process of pruning the decision trees is complicated by the fact that the pruning operations allowed at one state depend on the status of the trees at each other state. Thus it is necessary to make several passes through the states, attempting additional pruning at each pass, until no more improvement is possible. Testing each pruning operation against the entire training set is expensive, but in the case of synthetic data it gives the best results. For other applications it may be desirable to keep a cross-validation set for this purpose.
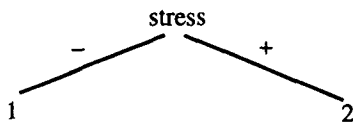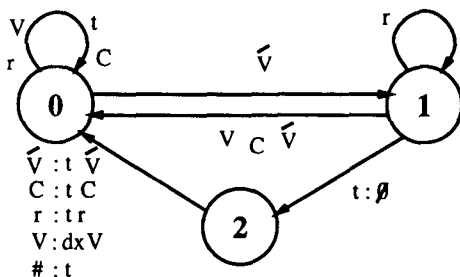
Figure 17
The same decision tree after pruning.

Figure 18
Flapping transducer induced from 50,000 samples (same as Figure 14).

Table 7
Results on three rules composed;
12,500 training size, 49,280 test size.

Method	States	% Error
OSTIA	329	22.09
Alignment	5	0.20
Add D-trees	5	0.04
Prune D-trees	5	0.01

The transducer obtained for the flapping rule after pruning decision trees is shown in Figure 18. In contrast to Figure 13, the arcs now correspond to the natural classes of consonants, stressed vowels, and unstressed vowels. The only difference between our result and the hand-drawn transducer in Figure 2 is the transition from state 1 upon seeing a stressed vowel—this will be discussed in Section 7.

The effects of adding decision trees at each state of the machine for the composition of t-insertion, t-deletion, and flapping are shown in Table 7.

Figure 19 shows the final transducer induced from this corpus of 12,500 words with pruned decision trees. We will discuss the remaining 0.01% error in Section 7 below.

We conclude our discussion of the community bias by seeing how a more on-line implementation of the bias might have helped our algorithm induce a transducer for r-deletion. Recall that the failure of the algorithm on r-deletion shown in Table 5 was not due to the difficulty of deletion per se, since our algorithm successfully learns the t-deletion rule. Rather, we believe that the difficulty with r-deletion is the broad context in which the rule applies: after any vowel and before any consonant. Since our segment set distinguishes three degrees of stress for each vowel, the alphabet size is 72; we believe this was simply too large for the algorithm without some prior concept of "vowel" and "consonant." While our decision tree augmentation adds these concepts to the algorithm, it only does so only after the initial transducer has been induced, and so cannot help in building the initial transducer. We need some method of interleaving
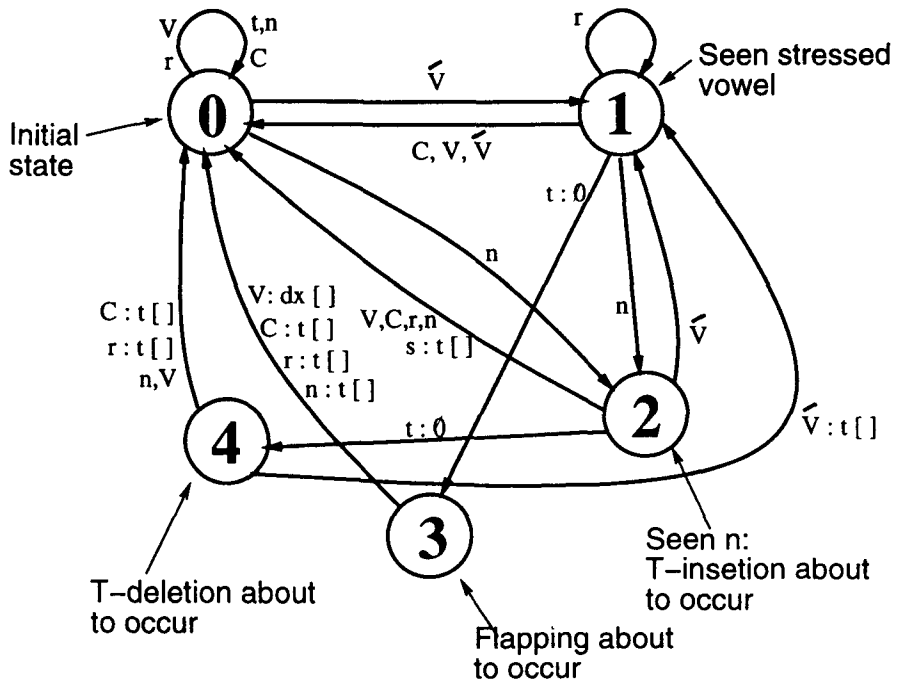
Figure 19
Three-rule transducer induced from 12,500 samples. [] indicates that the input symbol is
emitted with no features changed.

the generalization of segments into classes, performed by the decision trees, and the
induction of the structure of the transducer by merging states. Making generalizations
about input segments would in effect reduce the alphabet size on the fly, making the
learning of structure easier.

5.3 The Context Principle
Our final problem with the unaugmented OSTIA algorithm concerns phonological
rules that are both very general and also contain rightward context effects. In these
rules, the transducer must wait to see the right-hand context of a rule before emitting
the rule's output, and the rule applies to a general enough set of phones that additional
states are necessary to store information about the pending output. In such cases, a
separate state is necessary for each phone to which the rule applies. Thus, because
subsequential transducers are an inefficient model of these sorts of rules, representing
them leads to an explosion in the number of states of the machine, and an inability to
represent certain generalizations. One example of such state explosion is the German
rule to devoice word-final stops:

$$(9) \qquad \begin{bmatrix} -sonorant \\ -continuant \end{bmatrix} \rightarrow \begin{bmatrix} -voiced \end{bmatrix} / \underline{\quad} \#$$

In this case, a separate state must be created for each stop subject to devoicing, as
in Figure 20. Upon seeing a voiced stop, the transducer jumps to the appropriate state,
without emitting any output. If the end-of-word symbol follows, the corresponding
unvoiced stop will be emitted. If any other symbol follows, however, the original
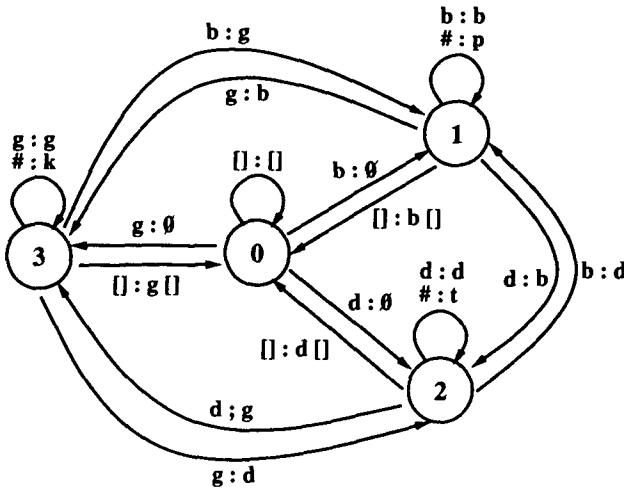
Figure 20
Transducer for word-final stop devoicing. [] indicates that the input symbol is emitted with no features changed.

voiced stop will be emitted, along with the current input symbol. In essence, the algorithm has learned three distinct rules:

(10)   b → p / __ #

(11)   d → t / __ #

(12)   g → k / __ #

Because of the inability to refer to previous input symbols, it is impossible to make a subsequential transducer that captures the generalization of the rule in (9). While the larger transducer of Figure 20 is accurate, the smaller transducer is desirable for a number of reasons. First, rules applying to larger classes of phones will lead to an even greater explosion in the number of states. Second, depending on the particular training data, this lack of generalization can cause the transducer to make mistakes on learning such rules. As mentioned in Section 4, smaller transducers significantly improve the general accuracy of the learning algorithm.

We turn to the context principle for an intuition about how to solve this problem. The context principle suggests that phonological rules refer to variables in their context. We found that subsequential transducers tend to handle leftward context much better than rightward context. This is because a separate state is only necessary for each distinct context in which segments behave differently. The behavior of different phones within each context is represented by the different arcs, without making separate states necessary. Thus our transducers only needed to be modified to deal with rightward context.[4] Our solution is to add a simple kind of memory to the model of transduction. The transducer keeps track of the input symbols seen so far. Just as the generalized arcs can now specify one of their output symbols as being the current input symbol with certain phonological features changed, they are now able to reference previous

---

4 The rules previously discussed in this paper avoid this problem because they apply to only one phone.
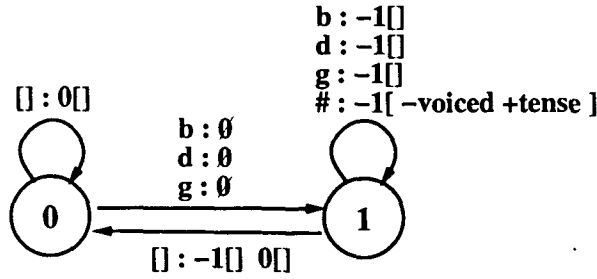
Figure 21
Word-final stop devoicing with variables. Variables are denoted by a number indicating the position of the input segment being referred to and a set of phonological features to change. Thus 0[] simply denotes the current input segment, while −1[−voiced +tense] means the unvoiced, tense version of the previous input segment. −1[] −0[] indicates that the machine outputs a string consisting of the previous input segment followed by the current segment.

input symbols. The transducer for word-final stop devoicing using variables is shown in Figure 21.

It is important to note that while we are changing the model of transduction, we are not increasing its formal power. As long as the alphabet is of finite size, any machine using variables can be translated into a potentially much larger machine with separate states for each possible value the variables can take.

When constructing the algorithm's original tree transducer, variables can be included in the output strings of the transducer's arcs. When performing a transduction, variables are interpreted as referring to a certain symbol in the input string with specific phonological features changed. The variables contain two pieces of information: an index of the input segment referenced by the variable relative to the current position in the index string, and a (possibly empty) list of phonological feature values to change in the input segment.

After calculating alignment information for each input/output pair, all output symbols determined to have arisen from substitutions (that is, all output segments other than those arising from insertions) are rewritten in variable notation. The variable's index is the relative index of the corresponding input segment as calculated by the alignment; the features specified by the variable are only those that have changed from the input segment. Thus rewriting each output symbol in variable notation is done in constant time and adds nothing to the algorithm's computational complexity.

When performing the state mergers of the OSTIA algorithm, two variables are considered to be the same symbol if they agree in both components: the index and list of phonological features. This allows arcs that previously had different output strings to merge, as for example in the arc from state 1 to state 0 of Figure 21, which is a generalization over the arcs into state 0 in Figure 20.

We applied the modified algorithm with variables in the output strings to the problem of the German rule that devoices word-final stops. Our data set was constructed from the CELEX lexical database (Celex 1993), which contains pronunciations for 359,611 word forms—including various inflected forms of the same lexeme. For our experiments we used the CELEX pronunciations as the surface forms, and generated underlying forms by revoicing the (devoiced) final stop for the appropriate forms (those for which the word's orthography ends in a voiced stop). Although the segment set used was slightly different from that of the English data, the same set of 26 binary articulatory features was used. Results are shown in Table 8.

Table 8
Results on German word-final stop devoicing;
50,000-word test set.

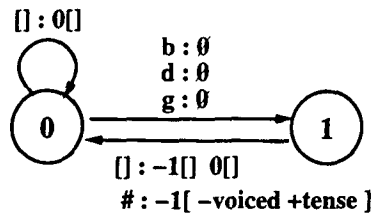|	No variables		Using variables	
Samples	States	% Error	States	% Error
700	8	0.218	8	7.996
10,000	11	0.240	11	0.568
20,000	24	0.392	2	0.000
50,000	19	0.098	2	0.000

Figure 22
Transducer induced for word-final stop devoicing. [] indicates that the input symbol is emitted with no features changed.

Using the model of transduction augmented with variables, a machine with the minimum two states and perfect performance on test data was induced with 20,000 samples and greater. This machine is shown in Figure 22. The only difference between this transducer and the hand-drawn transducer of Figure 21 is that the arcs leaving state 1 go to state 0 rather than looping back to state 1. Thus the transducer will fail to perform devoicing when two voiced stops occur at the end of a word. As the corpus contains no such cases, no errors were produced. As we will discuss in Section 7, this is similar to what occurred in the machine induced for flapping.

5.3.1 Search Over Sequences of State Mergers. The results quoted in the previous section were achieved with a slightly different method than those for the English data. The difference lies in the order in which state mergers are attempted, and can have significant effects in the results.

We performed experiments using two versions of the algorithm, varying the order in which the algorithm tries to merge pairs of states. The mergers are performed in a nested loop over the states of the initial tree transducer. The ordering of states for this loop in the original OSTIA algorithm as described in Oncina, García, and Vidal (1993) is the lexicographic ordering of the string of input symbols as one walks from the root of the tree to the state in question. This is the method used in the first column of results in Table 9. In the second column of results, the ordering of the states was simply the order of their creation as the sample transductions were read as input. This is also the method used in the results previously described for the various English rules.

The correctness of the algorithm requires that the states be ordered such that state numbers always increase as one walks outward from the root of the tree. This still leaves a large space of permissible orderings, and, as can be seen from our results, the ordering chosen can have a significant effect on the algorithm's outcome. While

Table 9
Results on German word-final stop devoicing; 50,000-word test set.

	Lexicographic ordering of states		Input-based ordering of states	
Samples	States	% Error	States	% Error
700	8	7.996	6	0.004
10,000	11	0.568	8	0.288
20,000	2	0.000	12	0.296
50,000	2	0.000	9	0.034

neither method is consistently better in the German experiments, we found that lexico-graphic orderings performed more poorly than the input-based ordering of the input samples for the English experiments.[5] The lexicographic ordering of the original algorithm is not always optimal. Furthermore, results with lexicographic orderings vary with the ordering of segments used. The segment ordering used for the results in Table 9 grouped similar segments together, and performed better than a randomized segment ordering. Presumably this is because the ordering grouping similar segments together causes states reached on similar input symbols to be merged, which is both linguistically reasonable and necessary in order to generate the correct transducer.

The underlying principle of the algorithm is to generalize by reducing the number of states in the transducer. Because the OSTIA algorithm tends to settle in local minima when merging states, the problem becomes one of searching the space of permissible orderings of state mergers. Some linguistically based heuristic for ordering states might produce more consistent results on different types of phonological rules, perhaps by reordering the remaining states as the initial states are merged.

6. Complexity

The OSTIA algorithm as described by Oncina, García, and Vidal (1993) had a worst-case complexity of $O(n^3(m + k) + nmk)$, where n is the sum of all the input strings' lengths, m is the length of the longest output string, and k is the size of the input alphabet; Oncina, García, and Vidal's (1993) experiments showed the average case time to grow more slowly. We will discuss the complexity implication of each of our enhancements to the algorithm.

The calculation of alignment information adds a preprocessing step to the algorithm that requires $O(nm)$ time for the dynamic programming string-alignment algorithm. After the initial tree is constructed using the alignment information, the above-mentioned worst-case bound still applies for the process of merging states; it does not require that the initial tree be onward. Since this modification only alters the initial tree transducer, the behavior of the main state-merging loop of the OSTIA algorithm is essentially unchanged. In practice, we found the use of alignment information significantly sped up the algorithm by allowing states to collapse more quickly. In any case, the $O(nm)$ complexity of the preprocessing step is subsumed by the $O(nmk)$ term of OSTIA's complexity.

---

5 The behavior of the input-based ordering depends on the ordering of the training set. We used a random ordering of our training set, but a corpus-based ordering would not be significantly different. While more frequent words tend to be seen earlier in a corpus, there is no reason to think that more frequent words provide better chances of successful state mergers.

The induction of decision trees adds a new stage after the OSTIA algorithm completes. The number of nodes in each decision tree is bounded by $O(k)$, since there are at most k arcs out of a given state. Calculating information content of a given feature can be done in $O(k)$ time because k is an upper bound on the number of possible outcomes of the decision tree. Therefore, choosing the feature with the maximum information content can be done in $O(fk)$ time, where f is the number of features, and the entire decision tree can be learned in $O(fk^2)$ time. Since there are at most n states, this stage of the algorithm is $O(nfk^2)$. However, because k is relatively small and because decision trees are induced only after merging states down to a small number, decision-tree induction in fact takes only a fraction the time of any other step of computation. The process of pruning the trees, however, is very expensive, as the entire training set is verified after each pruning operation. Since each verification of the input is $O(nk)$, and there are $O(k)$ nodes at each of $O(n)$ states to attempt to prune, one iteration through the set of states attempting pruning at each state is therefore $O(n^2k^2)$. There are at most $O(nk)$ iterations through the states, since at least one node of one state's decision tree must be pruned in each iteration. Therefore, the entire pruning process is $O(n^3k^3)$. This is a rather pessimistic bound since pruning occurs after state merger, and there are generally far less than nk states left. In fact, adding input pairs makes finding the smallest possible automaton more likely, and reduces the number of states at which pruning is necessary. Nevertheless the verification of pruning operations dominates all other steps of computation.

Once alignment information for each input/output pair has been computed, an output symbol can be rewritten in variable notation in constant time. Using variables can increase the size of the output alphabet, but none of the complexity calculations depend on this size. Therefore using variables is essentially free and contributes nothing to overall complexity. After adding all the steps together, we get $O(n^3(m + k) + nmk + nfk^2 + n^3k^3)$ time. Thus, even using the expensive method of verifying the entire training set after each pruning operation, the entire algorithm is still polynomial. Furthermore, our additions have not worsened the complexity of the algorithm with respect to n, the total number of input string symbols.

On a typical run on 10,000 German words with final stop devoicing applied using a SPARC 10, calculating alignment information, rewriting each output string in variable notation and building the initial tree transducer took 19 seconds, the state merging took 5 seconds, inducing the decision trees took under 1 second, and the pruning took 16 minutes and 1 second. When running on 50,000 words from the same data set, alignment, variable notation, and building the initial tree took 1 minute 37 seconds, the state merging took 4 minutes 44 seconds, inducing decision trees took 2 seconds and pruning decision trees took 2 hours, 9 minutes and 9 seconds.

7. Another Implicit Bias

An examination of the final few errors (three samples) in the induced flapping and three-rule transducers in Section 5.2.2 turned out to demonstrate a significant problem in the assumption that an SPE-style rule is isomorphic to a regular relation.

While the learned transducer correctly makes the generalization that flapping occurs after any stressed vowel, it does not flap after two stressed vowels in a row:

      sky-writing: s k ay1 r ay2 t ih ng → s k ay1 r ay2 t ih ng

      sky-writers: s k ay1 r ay2 t er z → s k ay1 r ay2 t er z

      gyrating: jh ay1 r ey2 t ih ng → jh ay1 r ey2 t ih ng

This is possible because no samples containing two stressed vowels in a row (or separated by an r as here) immediately followed by a flap were in the training data. This transducer will flap a t after any odd number of stressed vowels, rather than simply after any stressed vowel. Such a rule seems quite unnatural phonologically, and makes for an odd SPE-style context-sensitive rewrite rule. The SPE framework assumed (Chomsky and Halle 1968, 330) that the well-known Minimum Description Length (MDL) criterion be applied as an evaluation metric for phonological systems. Any sort of MDL criterion applied to a system of rewrite rules would prefer a rule such as

(13) $t \rightarrow dx$ / V __ V

to a rule such as

(14) $t \rightarrow dx$ / V́ (V́ V́)* __ V

which is the equivalent of the transducer learned from the training data. Similarly, the transducer learned for word-final stop devoicing would fail to perform devoicing when a word ends in two voiced stops, as it too returns to its state 0 upon seeing a second voiced stop, rather than staying in state 1.

These kinds of errors suggest that while a phonological rewrite rule can be expressed as a regular relation, the evaluation procedures for the two mechanisms (rewrite rules and transducers) must be different; the correct flapping transducer is in no way smaller than the incorrect one. In other words, the traditional formalism of context-sensitive rewrite rules contains implicit biases about how phonological rules usually work that are not present in the transducer system.

8. Related Work

Recent work in the machine learning of phonology includes algorithms for learning both segmental and nonsegmental information. Nonsegmental approaches include those of Daelemans, Gillis, and Durieux (1994) for learning stress systems, as well as approaches to learning morphology such as Gasser's (1993) system for inducing Semitic morphology, and Ellison's (1992) extensive work on syllabicity, sonority, and harmony. Since our approach learns only segmental structure, a more relevant comparison is with other algorithms for inducing segmental structure.

Johnson (1984) gives one of the first computational algorithms for phonological rule induction. His algorithm works for rules of the form

(15) $a \rightarrow b/C$

where C is the feature matrix of the segments around a. Johnson's algorithm sets up a system of constraint equations that C must satisfy, by considering both the positive contexts, i.e., all the contexts C_i in which a b occurs on the surface, as well as all the negative contexts C_j in which an a occurs on the surface. The set of all positive and negative contexts will not generally determine a unique rule, but will determine a set of possible rules. Johnson then proposes that principles from Universal Grammar might be used to choose between candidate rules, although he does not suggest any particular principles.

Johnson's system, while embodying an important insight about the use of positive and negative contexts for learning, did not generalize to insertion and deletion rules,

and it is not clear how to extend his system to modern autosegmental phonological systems. Touretzky, Elvgren, and Wheeler (1990) extended Johnson's insight by using the **version spaces** algorithm of Mitchell (1981) to induce phonological rules in their Many Maps architecture. Like Johnson's, their system looks at the underlying and surface realizations of single segments. For each segment, the system uses the version space algorithm to search for the proper statement of the context. The model also has a separate algorithm that handles harmonic effects by looking for multiple segmental changes in the same word, and has separate processes to deal with epenthesis and deletion rules. Touretzky, Elvgren, and Wheeler's approach seems quite promising; our use of decision trees to generalize each state is a similar use of phonological feature information to form generalizations.

Riley (1991) and Withgott and Chen (1993) first proposed a decision-tree approach to segmental mapping. A decision tree is induced for each segment, classifying possible realizations of the segment in terms of contextual factors such as stress and the surrounding segments. One problem with these particular approaches is that since the decision tree for each segment is learned separately, the technique has difficulty forming generalizations about the behavior of similar segments. In addition, no generalizations are made about segments in similar contexts, or about long-distance dependencies. In a transducer-based formalism, generalizations about segments in similar contexts follow naturally from generalizations about the behavior of individual segments. The context is represented by the current state of the machine, which in turn depends on the behavior of the machine on the previous segments. A possible adjustment to the decision-tree approach to capture some of these generalizations would be to augment the decision tree with information about the features of the output segment, or about features of more distant phones, perhaps about nearby syllables.

9. Conclusion

Our goal in this paper has been to explore the role of prior knowledge in phonological learning. We showed that a domain-independent, empiricist induction algorithm, OSTIA, failed to induce minimal transducers even for very simple rules like flapping. But adding three domain-specific learning biases to OSTIA allowed it to successfully learn transducers implementing simple phonological rules of English and German: faithfulness (underlying segments tend to be realized similarly on the surface), community (similar segments behave similarly), and context (phonological rules need access to variables in their context). These biases are so fundamental to generative phonology that, although they are present in some respect in every phonological theory, they are left implicit in most. Furthermore, we have shown that some of the remaining errors in our augmented model are due to implicit biases in the traditional SPE-style rewrite system that are not similarly represented in the transducer formalism, suggesting that while transducers may be formally equivalent to rewrite rules, they may not have identical evaluation procedures.

Because our biases were applied to the learning of very simple SPE-style rules, and to a nonprobabilistic theory of purely deterministic transducers, we do not expect that our model as implemented has any practical use as a phonological learning device. Indeed, because of the noise and nondeterminism inherent to linguistic data, we feel strongly that stochastic algorithms for language induction are much more likely to be a fruitful research direction (e.g., Kupiec 1992; Lucke 1993; Stolcke and Omohundro 1993, 1994; Ron, Singer, and Tishby 1994). But we believe that the biases we have relied on to improve the OSTIA algorithm may also prove useful when applied to such stochastic linguistic-rule induction algorithms. For example Wooters and Stolcke

(1994) used the Stolcke and Omohundro model-merging algorithm to induce word-pronunciation HMMs for a speech recognition system. This algorithm has no domain knowledge about phonology, and so is unable to classify together similar phones, or generalize across phones that were missing in the input data. Adding phonological feature biases to such a model could improve its generalization performance just as it improved OSTIA.

In summary, we believe that augmenting an empirical learning element with relatively abstract learning biases is a very fruitful ground for research between the often restated strict nativist and strict empiricist language learning paradigms.

References
Aha, David W., Dennis Kibler, and Marc K. Albert. 1991. Instance-based learning algorithms. *Machine Learning*, 6:37–66.

Aronoff, Mark. 1976. *Word-Formation in Generative Grammar*. Linguistic Inquiry Monograph no. 1. MIT Press, Cambridge, MA.

Berstel, Jean. 1979. *Transductions and Context-free Languages*. Teubner, Stuttgart.

Bird, Steven. 1995. *Computational Phonology: A Constraint-based Approach*. Cambridge University Press, Cambridge.

Bird, Steven and T. Mark Ellison. 1994. One-level phonology: Autosegmental representations and rules as finite automata. *Computational Linguistics*, 20(1).

Brown, Peter F., Vincent J. Della Pietra, Peter V. deSouza, Jennifer C. Lai, and Robert L. Mercer. 1992. Class-based *n*-gram models of natural language. *Computational Linguistics*, 18(4):467–479.

Celex. 1993. The CELEX lexical database. Centre for Lexical Information, Max Planck Institute for Psycholinguistics.

Chomsky, Noam. 1981. *Lectures on Government and Binding*. Foris, Dordrecht.

Chomsky, Noam and Morris Halle. 1968. *The Sound Pattern of English*. Harper and Row, New York.

CMU. 1993. The Carnegie Mellon Pronouncing Dictionary v0.1. Carnegie Mellon University.

Daelemans, Walter, Steven Gillis, and Gert Durieux. 1994. The acquisition of stress: A data-oriented approach. *Computational Linguistics*, 20(3):421–451.

Dresher, Elan and Jonathan Kaye. 1990. A computational learning model for metrical phonology. *Cognition*, 34:137–195.

Eimas, P. D., E. R. Siqueland, P. Jusczyk, and J. Vigorito. 1971. Speech perception in infants. *Science*, 171:303–306.

Ellison, T. Mark. 1992. *The Machine Learning of Phonological Structure*. Ph.D. thesis, University of Western Australia.

Ellison, T. Mark. 1994. Phonological derivation in optimality theory. In *COLING-94*.

Freund, Y., M. Kearns, D. Ron, R. Rubinfeld, R. Schapire, and L. Sellie. 1993. Efficient learning of typical finite automata from random walks. In *Proceedings of the 25th ACM Symposium on Theory of Computing*, pages 315–324.

Gasser, Michael. 1993. Learning words in time: Towards a modular connectionist account of the acquisition of receptive morphology. Unpublished manuscript.

Goldsmith, John. 1993. Harmonic phonology. In John Goldsmith, editor, *The Last Phonological Rule*. University of Chicago Press, Chicago, pages 21–60.

Gupta, Prahlad and David S. Touretzky. 1994. Connectionist models and linguistic theory: Investigations of stress systems in language. *Cognitive Science*, 18:1–50.

Jakobson, Roman. 1968. *Child Language, Aphasia, and Phonological Universals*. Mouton, The Hague.

Jakobson, Roman, Gunnar Fant, and Morris Halle. 1952. *Preliminaries to Speech Analysis*. MIT Press, Cambridge, MA.

Johnson, C. Douglas. 1972. *Formal Aspects of Phonological Description*. Mouton, The Hague.

Johnson, Mark. 1984. A discovery procedure for certain phonological rules. In *Proceedings of the Tenth International Conference on Computational Linguistics*, pages 344–347, Stanford.

Kaplan, Ronald M. and Martin Kay. 1994. Regular models of phonological rule systems. *Computational Linguistics*,

20(3):331–378.

Karttunen, Lauri. 1993. Finite-state constraints. In John Goldsmith, editor, *The Last Phonological Rule*. University of Chicago Press, Chicago.

Koskenniemi, Kimmo. 1983. Two-level morphology: A general computational model of word-form recognition and production. Publication No. 11, Department of General Linguistics, University of Helsinki.

Kupiec, Julian. 1992. Hidden Markov estimation for unrestricted stochastic context-free grammars. In *Proceedings of ICASSP-92*, pages 177–180, San Francisco.

Lakoff, George. 1993. Cognitive phonology. In John Goldsmith, editor, *The Last Phonological Rule*. University of Chicago Press, Chicago.

Ling, Charles X. 1994. Learning the past tense of English verbs: The symbolic patter associator vs. connectionist models. *Journal of Artificial Intelligence Research*, 1:209–229.

Lucke, Helmut. 1993. Inference of stochastic context-free grammar rules from example data using the theory of Bayesian belief propagation. In *Eurospeech 93*, pages 1195–1198, Berlin.

McCarthy, John J. 1988. Feature geometry and dependency: A review. *Phonetica*, 45:84–108.

McCarthy, John J. and Alan Prince. 1995. Prosodic morphology. In J. Goldsmith, editor, *Handbook of Phonological Theory*. Basil Blackwell Ltd., pages 318–366.

Mitchell, Tom M. 1981. Generalization as search. In Bonnie Lynn Webber and Nils J. Nilsson, editors, *Readings in Artificial Intelligence*. Morgan Kaufmann, Los Altos, pages 517–542.

Oncina, José, Pedro García, and Enrique Vidal. 1993. Learning subsequential transducers for pattern recognition tasks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15:448–458, May.

Orgun, Orhan. 1995. A declaritive theory of phonology-morphology interleaving. Unpublished manuscript, University of California-Berkeley, Department of Linguistics, October.

Orgun, Orhan. 1996. Correspondence and identity constraints in two-level optimality theory. In *Proceedings of the 14th West Coast Conference on Formal Linguistics (WCCFL-95)*.

Padgett, Jaye. 1995a. Feature classes. In *Papers in Optimality Theory*. GLSA, UMass, Amherst. University of Massachusetts Occasional Papers (UMOP) 18.

Padgett, Jaye. 1995b. Partial class behavior

and nasal place assimilation. In *Proceedings of the Arizona Phonology Conference: Workshop on Features in Optimality Theory*, Coyote Working Papers, University of Arizona, Tucson. To appear.

Prince, Alan and Paul Smolensky. 1993. Optimality theory: Constraint interaction in generative grammar. Unpublished manuscript, Rutgers University.

Pulman, Stephen G. and Mark R. Hepple. 1993. A feature-based formalism for two-level phonology: A description and implementation. *Computer Speech and Language*, 7:333–358.

Quinlan, J. R. 1986. Induction of decision trees. *Machine Learning*, 1:81–106.

Riley, Michael D. 1991. A statistical model for generating pronunciation networks. In *IEEE ICASSP-91*, pages 737–740.

Ron, Dana, Yoram Singer, and Naftali Tishby. 1994. The power of amnesia. In Jack Cowan, Gerald Tesauro, and Joshua Alspector, editors, *Advances in Neural Information Processing Systems 6*. Morgan Kaufmann, San Mateo, CA.

Stolcke, Andreas and Stephen Omohundro. 1993. Hidden Markov model induction by Bayesian model merging. In *Advances in Neural Information Processing Systems 5*. Morgan Kaufman, San Mateo, CA.

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

Tesar, Bruce. 1995. *Computational Optimality Theory*. Ph.D. thesis, University of Colorado, Boulder.

Tesar, Bruce and Paul Smolensky. 1993. The learnability of optimality theory: An algorithm and some basic complexity results. Technical Report CU-CS-678-93, University of Colorado at Boulder, Department of Computer Science.

Touretzky, David S., Gillette Elvgren, III, and Deirdre W. Wheeler. 1990. Phonological rule induction: An architectural solution. In *Proceedings of the 12th Annual Conference of the Cognitive Science Society (COGSCI-90)*, pages 348–355.

Touretzky, David S. and Deirdre W. Wheeler. 1990. A computational basis for phonology. In *Advances in Neural Information Processing Systems 2*, pages 372–379.

Wagner, R. A. and M. J. Fischer. 1974. The string-to-string correction problem. *Journal of the Association for Computation Machinery*, 21:168–173.

Withgott, M. M. and F. R. Chen. 1993.

Computation Models of American Speech.
Center for the Study of Language and
Information.
Wooters, Chuck and Andreas Stolcke. 1994.

Multiple-pronunciation lexical modeling
in a speaker-independent speech
understanding system. In *ICSLP-94.*