Learning Quantity Insensitive Stress Systems via Local Inference

Jeffrey Heinz
Linguistics Department
University of California, Los Angeles
Los Angeles, California 90095
jheinz@humnet.ucla.edu

Abstract

This paper presents an unsupervised batch learner for the quantity-insensitive stress systems described in Gordon (2002). Unlike previous stress learning models, the learner presented here is neither cue based (Dresher and Kaye, 1990), nor reliant on a priori Optimality-theoretic constraints (Tesar, 1998). Instead our learner exploits a property called neighborhood-distinctness, which is shared by all of the target patterns. Some consequences of this approach include a natural explanation for the occurrence of binary and ternary rhythmic patterns, the lack of higher n-ary rhythms, and the fact that, in these systems, stress always falls within a certain window of word edges.

1 Introduction

The central premise of this research is that phonotactic patterns are have properties which reflect properties of the learner. This paper illustrates this approach for quantity-insensitive (QI) stress systems (see below).

I present an unsupervised batch learner that correctly learns every one of these languages. The learner succeeds because there is a universal property of QI stress systems which I refer to as *neighborhood-distinctness* (to be defined below). This property, which is a structural notion of locality, is used by the learning algorithm to successfully infer the target pattern from samples.

A learner is a function from a set of observations to a grammar. An observation is some linguistic sign, in this case a word-sized sequence of stress values. A grammar is some device that must at least respond Yes or No when asked if a linguistic sign is a possible sign for this language (Chomsky and Halle, 1968; Halle, 1978).[1]

The remainder of the introduction outlines the typology of the QI stress systems, motivates representing phonotactics with regular languages, and examines properties of the attested patterns. In §2, I define the class of neighborhood-distinct languages. The learning algorithm is presented in two stages. §3 introduces a basic version of the learner the learner, which successfully acquires just under 90% of the target patterns. In §4, one modification is made to this learner which consequently succeeds on all target patterns. §5 discusses predictions made by these learning algorithms. The appendix summarizes the target patterns and results.

1.1 Quantity-Insensitive Stress Systems

Stress assignment in QI languages is indifferent to the weight of a syllable. For example, Latin is quantity-sensitive (QS) because stress assignment depends on the syllable type: if the penultimate syllable is heavy (i.e. has a long vowel or coda) then it receives stress, but otherwise the antepenult does. The stress systems under consideration here, unlike Latin, do not distinguish syllable types.

[1] In this respect, this work departs from (or is a special case of) gradient phonotactic models (Coleman and Pierrehumbert, 1997; Frisch et al., 2000; Albright, 2006; Hayes and Wilson, 2006)

Proceedings of the Eighth Meeting of the ACL Special Interest Group on Computational Phonology at HLT-NAACL 2006, pages 21–30,
New York City, USA, June 2006.

There are 27 types of QI stress systems found in Gordon's (2002) typology. Gordon adds six plausibly attestable QI systems by considering the behavior of all-light-syllabled words from QS systems.

These 33 patterns are divided into four kinds: single, dual, binary and ternary. Single systems have one stressed syllable per word, and dual systems up to two. Binary and ternary systems stress every second (binary) or third (ternary) syllable.

The choice to study QI stress systems was made for three reasons. First, they are well studied and the typology is well established (Hayes, 1995; Gordon, 2002). Secondly, learning of stress systems has been approached before (Dresher and Kaye, 1990; Gupta and Touretzky, 1991; Goldsmith, 1994; Tesar, 1998) making it possible to compare learners and results. Third, these patterns have been analyzed with adjacency restrictions (e.g. no clash), as disharmony (e.g. a primary stress may not be followed by another), and with recurrence requirements (e.g. build trochaic feet iteratively from the left). Thus the patterns found in the QI stress systems are representative of other phonotactic domains that the learner should eventually be extended to.

The 33 types are shown in Table 1. See Gordon (2002) and Hayes (1995) for details, examples, and original sources. Note that some patterns have a minimal word condition (Prince, 1980; McCarthy and Prince, 1990; Hayes, 1995), banning either monosyllables or light monosyllables. For example, Cayuvava bans all monosyllables, whereas Hopi bans only light monosyllables. Because this paper addresses QI stress patterns I abstract away from the internal structure of the syllable. For convenience, when stress patterns are explicated in this paper I assume (stressed) monosyllables are permitted. The learning study, however, includes each stress pattern both with and without stressed monosyllables. Predictions our learner makes with respect to the minimal word condition are given in §5.2.

[2]We use the (first) language name to exemplify the stress pattern. The number in parentheses is an index to the language Gordon's 2003 appendix. All stress representations follow Gordon's notation, who uses the metrical grid (Liberman and Prince, 1977; Prince, 1983). Thus, primary stress is indicated by 2, secondary stress by 1, and no stress by 0.

1.2 Phonotactics as Regular Languages

I represent phonotactic descriptions as regular sets, accepted by finite-state machines. A finite state machine is a 5-tuple $(\Sigma, Q, q_0, F, \delta)$ where Σ is a finite alphabet, Q is a set of states, $q_0 \in Q$ is the start state, $F \subseteq Q$ is a set of final states, and δ is a set of transitions. Each transition has an origin and a terminus and is labeled with a symbol of the alphabet; i.e. a transition is a 3-tuple (o, a, t) where $o, t \in Q$ and $a \in \Sigma$.

Empirically, it has been observed that most phonological phenomena are regular (Johnson, 1972; Kaplan and Kay, 1981; Kaplan and Kay, 1994; Ellison, 1994; Eisner, 1997; Karttunen, 1998). This is especially true of phonotactics: reduplication and metathesis, which have higher complexity, are not phonotactic patterns as they involve alternations.[3]

Formally, regular languages are widely studied in computer science, and their basic properties are well understood (Hopcroft et al., 2001). Also, a learning literature exists. E.g. the class of regular languages is not exactly identifiable in the limit (Gold, 1967), but certain subsets of it are (Angluin, 1980; Angluin, 1982). Thus it is becomes possible to ask: What subset of the regular languages delimits the class of possible human phonotactics and can properties of this class be exploited by a learner?

This perspective also connects to finite state models of Optimality Theory (OT) (Prince and Smolensky, 1993). Riggle (2004) shows that if OT constraints are made finite-state, it is possible to build a transducer that takes any input to a grammatical output. Removing from this transducer the input labels and hidden structural symbols (such as foot boundaries) in the output labels yields a phonotactic acceptor for the language, a target for our learner.

Consider Pintupi, #26 in Table 1, which exemplifies a binary stress pattern. Its phonotactic grammar is given in Figure 1. The hexagon indicates the start state, and final states are marked by the double perimeter.

This machine accepts the Pintupi words, but not other words of the same length. Also, the Pintupi grammar accepts an infinite number of words– just like the grammars in Hayes (1995) and Gordon

[3]See Albro (1998; 2005) for restricted extensions to regular languages.

Table 1: The Quantity-Insensitive Stress Systems.[2]

Single Systems									
1.	(1) Chitimacha	20000000	2000000	200000	20000	2000	200	20	2
2.	(2) Lakota	02000000	0200000	020000	02000	0200	020	02	2
3.	(3) Hopi (qs)	02000000	0200000	020000	02000	0200	020	20	2
4.	(4) Macedonian	00000200	0000200	000200	00200	0200	200	20	2
5.	(5) Nahuatl / Mohawk†	00000020	0000020	000020	00020	0020	020	20	2
6.	(6) Atayal / Diegueño‡	00000002	0000002	000002	00002	0002	002	02	2
Dual Systems									
7.	(7f) Quebec French	10000002	1000002	100002	10002	1002	102	12	2
8.	(9f) Udihe	10000002	1000002	100002	10002	1002	102	02	2
9.	(10i) Lower Sorbian	20000010	2000010	200010	20010	2010	200	20	2
10.	(11f) Sanuma	10000020	1000020	100020	10020	1020	020	20	2
11.	(15f) Georgian	10000200	1000200	100200	10200	0200	200	20	2
12.	(16i) Walmatjari	20000100	2000100	200100	20100	2010	200	20	2
	(optional variants)	20000010	2000010	200010	20010				
Binary Systems									
13.	(24i) Araucanian	02010101	0201010	020101	02010	0201	020	02	2
14.	(24f) Creek‡ (qs)	01010102	0101020	010102	01020	0102	020	02	2
15.	(25f) Urubu Kaapor	01010102	1010102	010102	10102	0102	102	02	2
16.	(26i) Malakmalak	20101010	0201010	201010	02010	2010	020	20	2
17.	(26f) Cavineña†	10101020	0101020	101020	01020	1020	020	20	2
18.	(27i) Maranungku	20101010	2010101	201010	20101	2010	201	20	2
19.	(27f) Palestinean Arabic‡ (qs)	10101020	1010102	101020	10102	1020	102	20	2
Binary Systems with Clash									
20.	(28i) Central Alaskan Yupik‡	01010102	0101012	010102	01012	0102	012	02	2
21.	(29i) Southern Paiute‡	02010110	0201010	020110	02010	0210	020	20	2
22.	(30i) Gosiute Shoshone	20101011	2010101	201011	20101	2011	201	21	2
23.	(32f) Biangai	10101020	1101020	101020	11020	1020	120	20	2
24.	(33f) Tauya	11010102	1010102	110102	10102	1102	102	12	2
Binary Systems with Lapse									
25.	(34f) Piro	10101020	1010020	101020	10020	1020	020	20	2
26.	(36i) Pintupi / Diyari†	20101010	2010100	201010	20100	2010	200	20	2
27.	(40f) Indonesian	10101020	1001020	101020	10020	1020	020	20	2
28.	(42i) Garawa	20101010	2001010	201010	20010	2010	200	20	2
Ternary Systems									
29.	(48i) Ioway-Oto	02001001	0200100	020010	02001	0200	020	02	2
30.	(49f) Cayuvava†	00100200	0100200	100200	00200	0200	200	20	2
31.	(67i) Estonian (qs)	20010010	2001010	201010	20010	2010	200	20	2
	(optional variants)	20101010	2010100	200100	20100				
		20100100	2010010						
		20010100							
32.	(71f) Pacific Yupik (qs)	01001002	0100102	010020	01002	0102	020	02	2
33.	(72i) Winnebago‡ (qs)	00200101	0020010	002001	00201	0020	002	02	2

† Bans monosyllables.

‡ Bans light monosyllables.

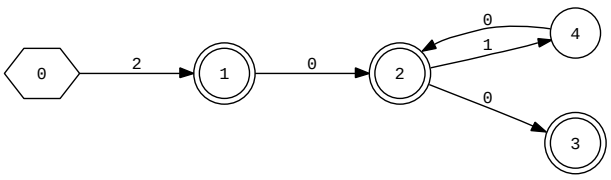

Figure 1: Stress in Pintupi as a finite state machine

(2002), who take the observed forms as instances of a pattern that extends to longer words. The learner's task is to take the Pintupi words in Table 1 and return the pattern represented by Figure 1.

1.3 Properties of QI Stress Patterns

The deterministic acceptor with the fewest states for a language is called the language's canonical acceptor. Therefore, let us ask what properties the canonical acceptors for the 33 stress types have in common that might be exploited by a learner.

One property shared by all grammars except Estonian is that they have exactly one loop (Estonian has two). Though this restriction is nontrivial, it is insufficient for learning to be guaranteed.[4] A second shared property is *slenderness*. A machine is slender iff it accepts only one word of length n. The only exceptions to this are Walmatjari and Estonian, which have free variation in longer words (see Table 1).

I focus in this paper on another property which are shared by all machines without exception. In 29 of the canonical acceptors, each state can be uniquely identified by its incoming symbol set, its outgoing symbol set, and whether it is final or non-final. These items make up the *neighborhood* of a state, which will be formally defined in the next section. The other four stress systems have non-canonical acceptors wherein each state can also be uniquely identified by its neighborhood. This property I call *neighborhood-distinctness*. Thus, neighborhood-distinctness is a universal property of QI stress systems, and it is this property that the learner will exploit.

[4]The proof is similar to the one used to show the cofinite languages are not learnable (Osherson et al., 1986).

2 Neighborhood-Distinctness

2.1 Neighborhood-Distinct Acceptors

The neighborhood of a state in an acceptor $(\Sigma, Q, q_0, F, \delta)$ is defined in (1).

(1) The **neighborhood** of a state q is triple (f, I, O) where $f = 1$ iff $q \in F$ and $f = 0$ otherwise, $I = \{a \mid \exists o \in Q,\ (o, a, q) \in \delta\}$, and $O = \{a \mid \exists t \in Q,\ (q, a, t) \in \delta\}$

Thus the neighborhood of state can be determined by looking solely at whether or not it is final, the set of symbols labeling the transitions which reach that state, and the set of symbols labeling the transitions which depart that state. For example in Figure 2, states p and q have the same neighborhood because they are both nonfinal, can both be reached by some element of $\{a, b\}$, and because each state can only be exited by observing a member of $\{c, d\}$.[5]

Figure 2: Two states with the same neighborhood.

Neighborhood-distinct acceptors are defined in (2).

(2) An acceptor is said to be **neighborhood-distinct** iff no two states have the same neighborhood.

This class of acceptors is finite: there are $2^{2|\Sigma|+1}$ neighborhoods, i.e. types of states. Since each state in a neighborhood-distinct machine has a unique neighborhood, this becomes an upper bound on machine size.[6]

[5]The notion of neighborhood can be generalized to neighborhoods of size k, where sets I and O are defined as the incoming and outgoing paths of length k. However, this paper is only concerned with neighborhoods of size 1.

[6]For some acceptor, the notion of neighborhood lends itself to an equivalence relation R_N over Q: pR_Nq iff p and q have the same neighborhood. Therefore, R_N partitions Q into blocks, and neighborhood-distinct machines are those where this partition equals the trivial partition.

2.2 Neighborhood-Distinct Languages

The class of neighborhood-distinct languages is defined in (3).

(3) **The neighborhood-distinct languages** are those for which there is an acceptor which is neighborhood-distinct.

The neighborhood-distinct languages are a (finite) proper subset of the regular languages over an alphabet Σ: all regular languages whose smallest acceptors have more than $2^{2|\Sigma|+1}$ states cannot be neighborhood-distinct (since at least two states would have the same neighborhood).

The canonically neighborhood-distinct languages are defined in (4).

(4) **The canonically neighborhood-distinct languages** are those for which the canonical acceptor is neighborhood-distinct.

The canonically neighborhood-distinct languages form a proper subset of the neighborhood-distinct languages. For example, the canonical acceptor shown in Figure 3 of Lower Sorbian (#9 in Table 1) is not neighborhood-distinct (states 2 and 3 have the same neighborhood). However, there is a non-canonical (because non-deterministic) neighborhood-distinct acceptor for this language, as shown in Figure 4.

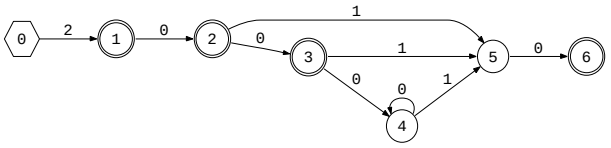

Figure 3: The canonical acceptor for Lower Sorbian.

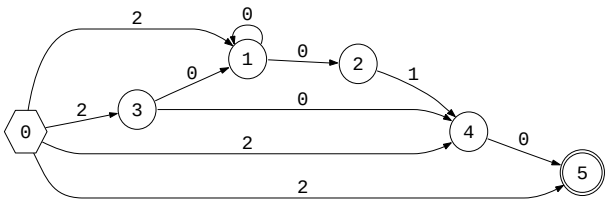

Figure 4: A neighborhood-distinct acceptor for Lower Sorbian.

Neighborhood-distinctness is a universal property of the patterns under consideration. Additionally, it is a property which a learner can use to induce a grammar from surface forms.

3 The Neighborhood Learner

In this section, I present the basic unsupervised batch learner, called the Neighborhood Learner, which learns 29 of the 33 patterns. In the next section, I introduce one modification to this learner which results in perfect accuracy.

The basic version of the learner operates in two stages: prefix tree construction and state-merging, cf. Angluin (1982). These two stages find smaller descriptions of the observed data; in particular state-merging may lead to generalization (see below).

A prefix tree is constructed as follows. Set the initial machine $M = (\Sigma, \{q_0\}, q_0, \emptyset, \emptyset)$ and the current state $c = q_0$. With each word, each symbol a is considered in order. If $\forall t \in Q$, $(c, a, t) \in \delta$ then set $c = t$. Otherwise, add a new state n to Q and a new arc (c, a, n) to δ. A new arc is therefore created on every symbol in the first word. The last state for a word is added to F. The process is repeated for each word. The prefix tree for Pintupi words from Table 1 is shown in Figure 5.

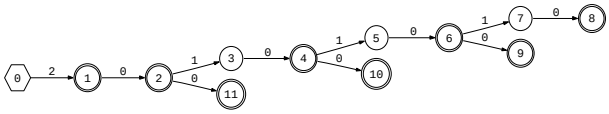

Figure 5: The prefix tree of Pintupi words.

The second stage of the learner is state-merging, a process which reduces the number of states in the machine. A key concept in state merging is that when two states are merged into a single state, their transitions are preserved. Specifically, if states p and q merge, then a merged state pq is added to the machine, and p and q are removed. For every arc that left p (or q) to a state r, there is now an arc from pq going to r. Likewise, for every arc from a state r to p (or q), there is now an arc from r to pq.

The post-merged machine accepts every word that the pre-merged machine accepts, and possibly more. For example, if there is a path between two states which become merged, a loop is formed.

What remains to be explained is the criteria the learner uses to determine whether two states in the prefix tree merge. The Neighborhood Learner merges two states iff they have the same neighborhood, guaranteeing that the resulting grammar is neighborhood-distinct.

The intuition is that the prefix tree provides a structured representation of the input and has recorded information about different environments, which are represented in the tree as states. Learning is a process which identifies actually different environments as 'the same'— here states are 'the same' iff their local features, i.e their neighborhoods, are the same. For example, suppose states p and q in the prefix tree are both final or both nonfinal, and they share the same incoming symbol set and outgoing symbol set. In the learner's eyes they are then 'the same', and will be merged.

The merging criteria partitions the states of the Pintupi prefix tree into five groups. States 3,5 and 7 are merged; states 2,4,6 are merged, and states 8,9,10,12 are merged. Merging of states halts when no two nodes have the same neighborhood– thus, the resulting machine is neighborhood-distinct. The result for Pintupi is shown in Figure 6.

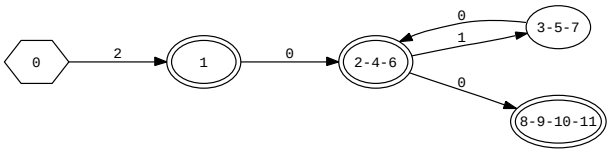

Figure 6: The grammar learned for Pintupi.

The machine in Figure 6 is equivalent to the one in Figure 1– they accept exactly the same language.[7] I.e. neighborhood merging of the prefix tree in Figure 5 generalizes from the data exactly as desired.

3.1 Results of Neighborhood Learning

The Neighborhood Learner successfully learns 29 of the 33 language types (see appendix). These are exactly the 29 canonically neighborhood-distinct languages. This suggests the following claim, which has not been proven.[8]

(5) **Conjecture:** The Neighborhood Learner identifies the class of canonically neighborhood-distinct languages.

In §4, I discuss why the learner fails where it does, and introduce a modification which results in perfect accuracy.

4 Reversing the Prefix Tree

This section examines the four cases where neighborhood learning failed and modifies the learning algorithm, resulting in perfect accuracy. The goal is to restrict generalization because in every case where learning failed, the learner overgeneralized by merging more states than it should have. Thus, the resulting grammars recognize multiple words with n syllables.

The dual stress pattern of Lower Sorbian places stress initially and, in words of four or more syllables, on the penult (see #9 Table 1). The prefix tree built from these words is shown in Figure 7.

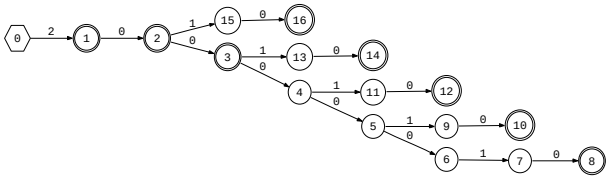

Figure 7: The prefix tree for Lower Sorbian.

Here the Neighborhood Learner fails because it merges states 2 and 3. The resulting grammar incorrectly accepts words of the form 20^*.

The proposed solution follows from the observation that if the prefix tree were constructed in reverse (reading each word from right to left) then the corresponding states in this structure would not have the same neighborhoods, and thus not be merged. A reverse prefix tree is constructed like a forward prefix tree, the only difference being that the order of symbols in each word is reversed. When neighborhood learning is applied to this structure and the resulting machine reversed again, the correct grammar is obtained, shown in Figure 4.

How is the learner to know whether to construct the prefix tree normally or in reverse? It simply does both and intersects the results. Intersection of two

[7] This can be verified by checking to see if the minimized versions of the two machines are isomorphic.

[8] The proof is made difficult by the fact that the acceptor returned by the Neighborhood Learner is not necessarily the canonical acceptor.

languages is an operation which returns a language consisting of the words common to both. Similarly, machine intersection returns an acceptor which recognizes just those words that both machines recognize. This strategy is thus conservative: the learner keeps only the most robust generalizations, which are the ones it 'finds' in both the forward and reverse prefix trees.

This new learner is called the Forward Backward Neighborhood (FBN) Learner and it succeeds with all the patterns (see appendix).

Interestingly, the additional languages the FBN Learner can acquire are ones that, under foot-based analyses like those in Hayes (1995), require feet to be built from the right word edge. For example, Lower Sorbian has a binary trochee aligned to the right word edge; Indonesian iteratively builds binary trochaic feet from the right word edge; Cayuvava iteratively builds anapests from the right word edge. Thus structuring the input in reverse appears akin to a footing procedure which proceeds from the right word boundary.

5 Predictions of Neighborhood Learning

In this section, let us examine some of the predictions that are made by neighborhood learning. In particular, let us consider the kinds of languages that the Neighborhood Learner can and cannot learn and compare them with the attested typology.

5.1 Binary and Ternary Stress Patterns

Neighborhood learning suggests an explanation of the fact that the stress rhythms found in natural language are binary or ternary and not higher n-ary, and of the fact that stress falls within a three-syllable window of the word edge: perhaps only systems with these properties are learnable. This is because the neighborhood learner cannot distinguish between sequences of the same symbol with length greater than two.

As an example, consider the quaternary (and higher n-ary) stress pattern $2(0001)^*(0|00|000)$.[9] If the learner is exposed to samples from this pattern, it incorrectly generalizes to $2(000^*1)^*(0|00|000)$.

[9] I follow Hopcroft et al (2001) in our notation of regular expressions with one substitution– we use | instead of + to indicate disjunction.

Similarly, neighborhood learning cannot distinguish a form like 02000 from 020000, so a system that places stress on the pre-antepenult (e.g. 02000, 002000, 0002000) is not learnable. With samples from the pre-antepenultimate language $(0^*2000|200|20|2)$, the learner incorrectly generalizes to 0^*20^*.

5.2 Minimal Word Conditions

A subtle prediction made by neighborhood-learning is that a QI stress language with a pattern like the one exemplified by Hopi (shown in Figure 8) cannot have a minimal word condition banning monosyllables. This is because if there were no monosyllables in this language, then state 4 in Figure 8 would have the same neighborhood as state 2 (as in Figure 9).

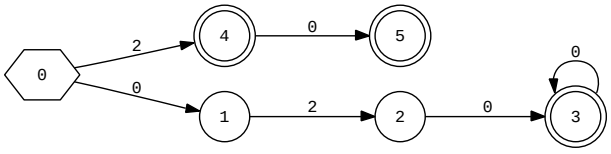

Figure 8: The stress pattern exemplified by Hopi, allowing monosyllables.

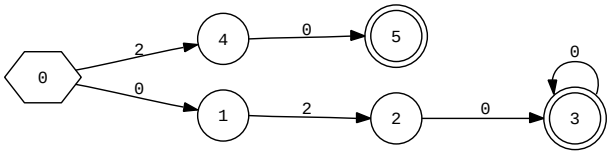

Figure 9: The stress pattern exemplified by Hopi, not allowing monosyllables.

Since such a grammar recognizes a non-neighborhood-distinct language it cannot be learned by the Neighborhood Learner.

As it happens, Hopi is a QS language which prohibits light, but permits heavy, monosyllables. Since I have abstracted away from the internal structure of the syllable in this paper, this prediction is not disconfirmed by the known typology: there are in fact no QI Hopi-like stress patterns in Gordon's (2002) typology which ban all monosyllables; i.e there are no QI patterns like the one in Figure 9.

Some QI languages do have a minimal word condition banning all monosyllables. To our knowledge these are Cavineña and Cayuvava (see Table 1), Mohawk (which places stress on the penult

like Nahuatl), and Diyari, Mohwak, Pitta Pitta and Wangkumara (all which assign stress like Pintupi) (Hayes, 1995). The Forward Backward Neighborhood Learner learns all of these patterns successfully irrespective of whether the patterns (and corresponding input samples) permit monosyllables, predicting that such patterns do not correlate with a prohibition on monosyllables (see appendix).

Other QI languages prohibit light monosyllables. Diegueño, for example, places stress finally like Atayal (see Table 1), but only allows heavy monosyllables. This is an issue to attend to in future research when trying to extend the learning algorithm to QS patterns, when the syllable type (light/heavy) is included in the representational scheme.

5.3 Restrictiveness and Other Approaches

There are languages that can be learned by neighborhood learning that phonologists do not consider to be natural. For example, the Neighborhood Learner learns a pattern in which words with an odd number of syllables bear initial stress but words with an even number of syllables bear stress on all odd syllables. However, the grammar for this language differs from all of the attested systems in that it has two loops but is slender (cf. Estonian which has two loops but is not slender). Thus this case suggests a further formal restriction to the class of possible stress systems.

More serious challenges of unattestable, but Neighborhood Learner-able, patterns exist; e.g. 21*. In other words, it does not follow from neighborhood-distinctness that languages with stress must have stressless syllables. Nor does the notion that every word must bear some stress somewhere (i.e. Culminativity– see Hayes (1995)).

However, despite the existence of learnable pathological languages, this approach is not unrestricted. The class of languages to be learned is finite—as in the Optimality-theoretic and Principles and Parameters frameworks—and is a proper subset of the regular languages. Future research will seek additional properties to better approximate the class of QI stress systems that can be exploited by inductive learning.

This approach offers more insight into QI stress systems than earlier learning models. Optimality-theoretic learning models (e.g. (Tesar, 1998)) and models set in the Principles and Parameters framework (e.g. (Dresher and Kaye, 1990)) make no use of any property of the class of patterns to be learned beyond its finiteness. Also, our learner is much simpler than these other models, which require a large set of a priori switches and cues or constraints.

6 Conclusions

This paper presented a batch learner which correctly infers the attested QI stress patterns from surface forms. The key to the success of this learner is that it takes advantage of a universal property of QI stress systems, neighborhood-distinctness. This property provides a natural explanation for why stress falls within a particular window of the word edge and why rhythms are binary and ternary. It is striking that all of the attested patterns are learned by this simple approach, suggesting that it will be fruitful and revealing when applied to other phonotactic learning problems.

Acknowledgements

I especially thank Bruce Hayes, Ed Stabler, Colin Wilson, Kie Zuraw and the anonymous reviewers for insightful comments and suggestions. I also thank Sarah Churng, Greg Kobele, Katya Pertsova, and Sarah VanWagenen for helpful discussion.

References

Adam Albright. 2006. Gradient phonotactic effects: lexical? grammatical? both? neither? Talk handout from the 80th Annual LSA Meeting, Albuquerque, NM.

Dan Albro. 1998. Evaluation, implementation, and extension of primitive optimality theory. Master's thesis, University of California, Los Angeles.

Dan Albro. 2005. *A Large-Scale, LPM-OT Analysis of Malagasy*. Ph.D. thesis, University of California, Los Angeles.

Dana Angluin. 1980. Finding patterns common to a set of strings. *Journal of Computer and System Sciences*, 21:46–62.

Dana Angluin. 1982. Inference of reversible languages. *Journal for the Association of Computing Machinery*, 29(3):741–765.

Noam Chomsky and Morris Halle. 1968. *The Sound Pattern of English*. Harper & Row.

John Coleman and Janet Pierrehumbert. 1997. Stochastic phonological grammars and acceptability. In *Compuational Phonolgy*, pages 49–56. Somerset, NJ: Association for Computational Linguistics. Third Meeting of the ACL Special Interest Group in Computational Phonology.

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

Jason Eisner. 1997. What constraints should ot allow? Talk handout, Linguistic Society of America, Chicago, January. Available on the Rutgers Optimality Archive, ROA#204-0797, http://roa.rutgers.edu/.

T.M. Ellison. 1994. The iterative learning of phonological constraints. *Computational Linguistics*, 20(3).

S. Frisch, N.R. Large, and D.B. Pisoni. 2000. Perception of wordlikeness: Effects of segment probability and length on the processing of nonwords. *Journal of Memory and Language*, 42:481–496.

E.M. Gold. 1967. Language identification in the limit. *Information and Control*, 10:447–474.

John Goldsmith. 1994. A dynamic computational theory of accent systems. In Jennifer Cole and Charles Kisseberth, editors, *Perspectives in Phonology*, pages 1–28. Stanford: Center for the Study of Language and Information.

Matthew Gordon. 2002. A factorial typology of quantity-insensitive stress. *Natural Language and Linguistic Theory*, 20(3):491–552. Appendices available at http://www.linguistics.ucsb.edu/faculty/gordon/pubs.html.

Prahlad Gupta and David Touretzky. 1991. What a perceptron reveals about metrical phonology. In *Proceedings of the Thirteenth Annual Conference of the Cognitive Science Society*, pages 334–339.

Morris Halle. 1978. Knowledge unlearned and untaught: What speakers know about the sounds of their language. In *Linguistic Theory and Psychological Reality*. The MIT Prss.

Bruce Hayes and Colin Wilson. 2006. The ucla phonotactic learner. Talk handout from UCLA Phonology Seminar.

Bruce Hayes. 1995. *Metrical Stress Theory*. Chicago University Press.

John Hopcroft, Rajeev Motwani, and Jeffrey Ullman. 2001. *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley.

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

Ronald Kaplan and Martin Kay. 1981. Phonological rules and finite state transducers. Paper presented at ACL/LSA Conference, New York.

Ronald Kaplan and Martin Kay. 1994. Regular models of phonological rule systems. *Computational Linguistics*, 20(3):331–378.

Lauri Karttunen. 1998. The proper treatment of optimality theory in computational phonology. *Finite-state methods in natural language processing*, pages 1–12.

Mark Liberman and Alan Prince. 1977. On stress and linguistic rhythm. *Linguistic Inquiry*, 8:249–336.

John McCarthy and Alan Prince. 1990. Foot and word in prosodic morphology. *Natural Language and Linguistic Theory*, 8:209–283.

Daniel Osherson, Scott Weinstein, and Michael Stob. 1986. *Systems that Learn*. MIT Press, Cambridge, Massachusetts.

Alan Prince and Paul Smolensky. 1993. Optimality theory: Constraint interaction in generative grammar. Technical Report 2, Rutgers University Center for Cognitive Science.

Alan Prince. 1980. A metrical theory for estonian quantity. *Linguistic Inquiry*, 11:511–562.

Alan Prince. 1983. Relating to the grid. *Linguistic Inquiry*, 14(1).

Jason Riggle. 2004. *Generation, Recognition, and Learning in Finite State Optimality Theory*. Ph.D. thesis, University of California, Los Angeles.

Bruce Tesar. 1998. An interative strategy for language learning. *Lingua*, 104:131–145.

Appendix. Target Grammars and Results

See Table 2. Circled numbers mean the learner identified the pattern. The $\times$ mark means the learner failed to identify the pattern. The number inside the circle indicates which forms were necessary for convergence. Specifically, in the "With Monosyllables" column, ⓝ means the learner succeeded learning the "With Monosyllables" pattern with words with one to n syllables. Likewise, in the "Without Monosyllables" column, ⓝ means the learner succeeded learning the "Without Monosyllables" pattern with words with two to n syllables. For example, in the "With Monosyllables" column, ⑤ means that the learner succeeded only with words with one to five syllables. The learners still succeed when given longer words. The number n may be thought of as the smallest word needed for generalization.

Table 2: Learning Results

		With Monosyllables			Without Monosyllables		
	Language	RegExp	NHL	FBNL	RegExp	NHL	FBNL
		Single					
1.	Chitimacha	20^*	④	④	20^+	④	④
2.	Lakota	$(2\|020^*)$	⑤	⑤	020^*	⑤	⑤
3.	Hopi (qs)	$(2\|20\|020^+)$	⑤	⑤	$(020^*\|2)0$	×	×
4.	Macedonian	$(2\|20\|0^*200)$	⑥	⑥	$(2\|0^*20)0$	×	⑥
5.	Nahuatl	$(2\|0^*20)$	⑤	⑤	0^*20	⑤	⑤
6.	Atayal	0^*2	④	④	0^+2	④	④
		Dual					
7.	Quebec French	$(2\|10^*2)$	⑤	⑤	10^*2	⑤	⑤
8.	Udihe	$(2\|(10^*)^?02)$	⑤	⑥	$(10^*)^?02$	⑤	⑥
9.	Lower Sorbian	$(2\|2(0\|0^+1)0)$	×	⑥	$2(0\|0^+1)0$	×	⑥
10.	Sanuma	$(2\|20\|020\|10^+20)$	⑥	⑦	$(2\|02\|10^+2)0$	⑥	⑦
11.	Georgian	$(2\|20\|0^?200\|10^+200)$	⑦	⑧	$(2\|0^?20\|10^+20)0$	×	⑧
12.	Walmatjari	$(2\|20(0^*10)^?0^?)$	×	⑥	$20(0^*10)^?0^?$	×	⑥
		Binary					
13.	Araucanian	$(2\|02(01)^*0^?)$	⑥	⑥	$02(01)^*0^?$	⑥	⑥
14.	Creek (qs)	$(2\|(01)^*020^?)$	⑥	⑥	$(01)^*020^?$	⑥	⑥
15.	Urubu Kappor	$0^?(10)^*2$	⑤	⑤	$(0\|10)(10)^*2$	⑤	⑤
16.	Malakmalak	$(2\|0^?2(01)^*0)$	⑥	⑥	$0^?2(01)^*0$	⑥	⑥
17.	Cavineña	$(2\|0^?(10)^*20)$	⑥	⑥	$0^?(10)^*20$	⑥	⑥
18.	Maranungku	$2(01)^*0^?$	⑤	⑤	$20(10)^*1^?$	⑤	⑤
19.	Palestinean Arabic (qs)	$(10)^*20^?$	⑤	⑤	$(20\|(10)^+20^?)$	⑤	⑤
		Binary w/clash					
20.	Central Alaskan Yupik	$(0(10)^*1^?)^?2$	⑤	⑤	$0(10)^*1^?2$	⑤	⑤
21.	Southern Paiute	$(2\|(2\|02(01)*1^?)0)$	⑦	⑧	$(2\|02(01)*1^?)0$	⑦	⑧
22.	Gosiute Shoshone	$2((01)^*0^?1)^?$	⑤	⑥	$2(01)^*0^?1$	⑤	⑥
23.	Biangai	$(2\|1^?(10)^*20)$	⑦	⑦	$1^?(10)^*20$	⑦	⑦
24.	Tauya	$(2\|1^?(10)^*2)$	⑥	⑥	$1^?(10)^*2$	⑥	⑥
		Binary w/lapse					
25.	Piro	$(2\|(10)^*0^?20)$	⑥	⑦	$(10)^*0^?20$	⑥	⑦
26.	Pintupi	$2(0(10)^*0^?)^?$	⑥	⑥	$20(10)^*0^?$	⑥	⑥
27.	Indonesian	$(2\|(10)^?0^?(10)^*20)$	×	⑧	$(10)^?0^?(10)^*20$	×	⑧
28.	Garawa	$2(00^?(10)^*)^?$	⑥	⑥	$200^?(10)^*$	⑥	⑥
		Ternary					
29.	Ioway Oto	$(2\|02(001)^*0^?0^?)$	⑦	⑧	$02(001)^*0^?0^?$	⑦	⑧
30.	Cayuvava	$(0^?0^?(100)^*200\|20\|2)$	×	⑨	$(0^?0^?(100)^*20\|2)0$	×	⑨
31.	Estonian (qs)	$20^?0^?(100\|10)^*$	⑥	⑥	$200^?(100\|10)^*$	⑥	⑥
32.	Pacific Yupik (qs)	$(2\|0(100)^*(20^?\|102))$	⑦	⑦	$0(100)^*(20^?\|102)$	⑦	⑦
33.	Winnebago (qs)	$(2\|02\|002(001)^*0^?1^?)$	⑨	⑨	$(02\|002(001)^*0^?1^?)$	⑨	⑨

NHL : Neighborhood Learner FBNL : Forward Backward Neighborhood Learner