String Extension Learning

Jeffrey Heinz
University of Delaware
Newark, Delaware, USA
heinz@udel.edu

Abstract

This paper provides a unified, learning-theoretic analysis of several learnable classes of languages discussed previously in the literature. The analysis shows that for these classes an incremental, globally consistent, locally conservative, set-driven learner always exists. Additionally, the analysis provides a recipe for constructing new learnable classes. Potential applications include learnable models for aspects of natural language and cognition.

1 Introduction

The problem of generalizing from examples to patterns is an important one in linguistics and computer science. This paper shows that many disparate language classes, many previously discussed in the literature, have a simple, natural and interesting (because non-enumerative) learner which exactly identifies the class in the limit from distribution-free, positive evidence in the sense of Gold (Gold, 1967).[1] These learners are called String Extension Learners because each string in the language can be mapped (extended) to an element of the grammar, which in every case, is conceived as a finite set of elements. These learners have desirable properties: they are incremental, globally consistent, and locally conservative.

Classes previously discussed in the literature which are string extension learnable include the Locally Testable (LT) languages, the Locally Testable Languages in the Strict Sense (Strictly Local, SL) (McNaughton and Papert, 1971; Rogers and Pullum, to appear), the Piecewise Testable (PT) languages (Simon, 1975), the Piecewise Testable languages in the Strict Sense (Strictly Piecewise, SP) (Rogers et al., 2009), the Strongly Testable languages (Beauquier and Pin, 1991), the Definite languages (Brzozowski, 1962), and the Finite languages, among others. To our knowledge, this is the first analysis which identifies the common structural elements of these language classes which allows them to be identifiable in the limit from positive data: each language class induces a natural partition over all logically possible strings and each language in the class is the union of finitely many blocks of this partition.

One consequence of this analysis is a recipe for constructing new learnable classes. One notable case is the Strictly Piecewise (SP) languages, which was originally motivated for two reasons: the learnability properties discussed here and its ability to describe long-distance dependencies in natural language phonology (Heinz, 2007; Heinz, to appear). Later this class was discovered to have several independent characterizations and form the basis of another subregular hierarchy (Rogers et al., 2009).

It is expected string extension learning will have applications in linguistic and cognitive models. As mentioned, the SP languages already provide a novel hypothesis of how long-distance dependencies in sound patterns are learned. Another example is the Strictly Local (SL) languages which are the categorical, symbolic version of n-gram models, which are widely used in natural language processing (Jurafsky and Martin, 2008). Since the SP languages also admit a probabilistic variant which describe an efficiently estimable class of distributions (Heinz and Rogers, 2010), it is plausible to expect the other classes will as well, though this is left for future research.

String extension learners are also simple, mak-

[1] The allowance of negative evidence (Gold, 1967) or restricting the kinds of texts the learner is required to succeed on (i.e. non-distribution-free evidence) (Gold, 1967; Horning, 1969; Angluin, 1988) admits the learnability of the class of recursively enumerable languages. Classes of languages learnable in the harder, distribution-free, positive-evidence-only settings are due to structural properties of the language classes that permit generalization (Angluin, 1980b; Blumer et al., 1989). That is the central interest here.

Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pages 897–906,
Uppsala, Sweden, 11-16 July 2010.

ing them accessible to linguists without a rigorous mathematical background.

This paper is organized as follow. §2 goes over basic notation and definitions. §3 defines string extension grammars, languages, and language classes and proves some of their fundamental properties. §4 defines string extension learners and proves their behavior. §5 shows how important subregular classes are string extension language classes. §6 gives examples of nonregular and infinite language classes which are string extension learnable. §7 summarizes the results, and discusses lines of inquiry for future research.

2 Preliminaries

This section establishes notation and recalls basic definitions for formal languages, the paradigm of identification in the limit from positive data (Gold, 1967). Familiarity with the basic concepts of sets, functions, and sequences is assumed.

For some set A, $\mathcal{P}(A)$ denotes the set of all subsets of A and $\mathcal{P}_{fin}(A)$ denotes the set of all finite subsets of A. If f is a function such that $f : A \rightarrow B$ then let $f^{\diamond}(a) = \{f(a)\}$. Thus, $f^{\diamond} : A \rightarrow \mathcal{P}(B)$ (note $f^{\diamond}$ is not surjective). A set π of nonempty subsets of S is a *partition* of S iff the elements of π (called *blocks*) are pairwise disjoint and their union equals S.

Σ denotes a fixed finite set of symbols, the *alphabet*. Let Σ^n, $\Sigma^{\leq n}$, Σ^*, Σ^+ denote all strings formed over this alphabet of length n, of length less than or equal to n, of any finite length, and of any finite length strictly greater than zero, respectively. The term *word* is used interchangeably with *string*. The *range* of a string w is the set of symbols which are in w. The empty string is the unique string of length zero denoted λ. Thus $range(\lambda) = \emptyset$. The length of a string u is denoted by $|u|$, e.g. $|\lambda| = 0$. A language L is some subset of Σ^*. The reverse of a language $L^r = \{w^r : w \in L\}$.

Gold (1967) establishes a learning paradigm known as identification in the limit from positive data. A *text* is an infinite sequence whose elements are drawn from $\Sigma^* \cup \{\#\}$ where $\#$ represents a non-expression. The ith element of t is denoted $t(i)$, and $t[i]$ denotes the finite sequence $t(0), t(1), \ldots t(i)$. Following Jain et al. (1999), let SEQ denote the set of all possible finite sequences:

$$SEQ = \{t[i] : t \text{ is a text and } i \in \mathbb{N}\}$$

The *content* of a text is defined below.

$$content(t) = \{w \in \Sigma^* : \exists n \in \mathbb{N} \text{ such that } t(n) = w\}$$

A text t is a *positive text for a language* L iff $content(t) = L$. Thus there is only one text t for the empty language: for all i, $t(i) = \#$.

A *learner* is a function ϕ which maps initial finite sequences of texts to grammars, i.e. $\phi : SEQ \rightarrow \mathcal{G}$. The elements of $\mathcal{G}$ (the grammars) generate languages in some well-defined way. A learner *converges on a text* t iff there exists $i \in \mathbb{N}$ and a grammar G such that for all $j > i$, $\phi(t[j]) = G$.

For any grammar G, the language it generates is denoted $L(G)$. A learner ϕ *identifies a language* L *in the limit* iff for any positive text t for L, ϕ converges on t to grammar G and $L(G) = L$. Finally, a learner ϕ *identifies a class of languages* $\mathcal{L}$ *in the limit* iff for any $L \in \mathcal{L}$, ϕ identifies L in the limit. Angluin (1980b) provides necessary and sufficient properties of language classes which are identifiable in the limit from positive data.

A learner ϕ of language class $\mathcal{L}$ is *globally consistent* iff for each i and for all texts t for some $L \in \mathcal{L}$, $content(t[i]) \subseteq L(\phi(t[i]))$. A learner ϕ is *locally conservative* iff for each i and for all texts t for some $L \in \mathcal{L}$, whenever $\phi(t[i]) \neq \phi(t[i-1])$, it is the case that $t(i) \notin L(\phi([i-1]))$. These terms are from Jain et al. (2007). Also, learners which do not depend on the order of the text are called *set-driven* (Jain et al., 1999, p. 99).

3 Grammars and Languages

Consider some set A. A string extension function is a total function $f : \Sigma^* \rightarrow \mathcal{P}_{fin}(A)$. It is not required that f be onto. Denote the class of functions which have this general form $\mathcal{SEF}$.

Each string extension function is naturally associated with some formal class of grammars and languages. These functions, grammars, and languages are called *string extension functions*, *grammars*, and *languages*, respectively.

Definition 1 Let $f \in \mathcal{SEF}$.

1. A *grammar* is a finite subset of A.
2. The *language of grammar* G is

$$L_f(G) = \{w \in \Sigma^* : f(w) \subseteq G\}$$

3. The *class of languages* obtained by all possible grammars is

$$\mathcal{L}_f = \{L_f(G) : G \in \mathcal{P}_{fin}(A)\}$$

The subscript f is omitted when it is understood from context.

A function $f \in \mathcal{SEF}$ naturally induces a partition π_f over Σ^*. Strings u and v are equivalent ($u \sim_f v$) iff $f(u) = f(v)$.

Theorem 1 Every language $L \in \mathcal{L}_f$ is a finite union of blocks of π_f.

Proof: Follows directly from the definition of $\sim_f$ and the finiteness of string extension grammars. □

We return to this result in §6.

Theorem 2 $\mathcal{L}_f$ is closed under intersection.

Proof: We show $L_1 \cap L_2 = L(G_1 \cap G_2)$. Consider any word w belonging to L_1 and L_2. Then $f(w)$ is a subset of G_1 and of G_2. Thus $f(w) \subseteq G_1 \cap G_2$, and therefore $w \in L(G_1 \cap G_2)$. The other inclusion follows similarly. □

String extension language classes are not in general closed under union or reversal (counterexamples to union closure are given in §5.1 and to reversal closure in §6.)

It is useful to extend the domain of the function f from strings to languages.

$$f(L) = \bigcup_{w \in L} f(w) \tag{1}$$

An element g of grammar G for language $L = L_f(G)$ is *useful* iff $g \in f(L)$. An element is *useless* if it is not useful. A grammar with no useless elements is called *canonical*.

Remark 1 *Fix a function $f \in \mathcal{SEF}$. For every $L \in \mathcal{L}_f$, there is a canonical grammar, namely $f(L)$. In other words, $L = L(f(L))$.*

Lemma 1 Let $L, L' \in \mathcal{L}_f$. $L \subseteq L'$ iff $f(L) \subseteq f(L')$

Proof: ($\Rightarrow$) Suppose $L \subseteq L'$ and consider any $g \in f(L)$. Since g is useful, there is a $w \in L$ such that $g \in f(w)$. But $f(w) \subseteq f(L')$ since $w \in L'$.

($\Leftarrow$) Suppose $f(L) \subseteq f(L')$ and consider any $w \in L$. Then $f(w) \subseteq f(L)$ so by transitivity, $f(w) \subseteq f(L')$. Therefore $w \in L'$. □

The significance of this result is that as the grammar G monotonically increases, the language $L(G)$ monotonically increases too. The following result can now be proved, used in the next section on learning.[2]

Theorem 3 For any finite $L_0 \subseteq \Sigma^*$, $L = L(f(L_0))$ is the smallest language in $\mathcal{L}_f$ containing L_0.

Proof: Clearly $L_0 \subseteq L$. Suppose $L' \in \mathcal{L}_f$ and $L_0 \subseteq L'$. It follows directly from Lemma 1 that $L \subseteq L'$ (since $f(L) = f(L_0) \subseteq f(L')$). □

4 String Extension Learning

Learning string extension classes is simple. The initial hypothesis of the learner is the empty grammar. The learner's next hypothesis is obtained by applying function f to the current observation and taking the union of that set with the previous one.

Definition 2 For all $f \in \mathcal{SEF}$ and for all $t \in SEQ$, define ϕ_f as follows:

$$\phi_f(t[i]) = \begin{cases} \emptyset & \text{if } i = -1 \\ \phi_f(t[i-1]) & \text{if } t(i) = \# \\ \phi_f(t[i-1]) \cup f(t(i)) & \text{otherwise} \end{cases}$$

By convention, the initial state of the grammar is given by $\phi(t[-1]) = \emptyset$. The learner ϕ_f exemplifies *string extension learning*. Each individual string in the text reveals, by extension with f, aspects of the canonical grammar for $L \in \mathcal{L}_f$.

Theorem 4 ϕ_f is globally consistent, locally conservative, and set-driven.

Proof: Global consistness and local conservativeness follow immediately from Definition 2. For set-drivenness, witness (by Definition 2) it is the case that for any text t and any $i \in \mathbb{N}$, $\phi(t[i]) = f(content(t[i]))$. □

The key to the proof that ϕ_f identifies $\mathcal{L}_f$ in the limit from positive data is the finiteness of G for all $L(G) \in \mathcal{L}$. The idea is that there is a point in the text in which every element of the grammar has been seen because (1) there are only finitely many useful elements of G, and (2) the learner is guaranteed to see a word in L which yields (via f) each element of G at some point (since the learner receives a positive text for L). Thus at this point

[2]The requirement in Theorem 3 that L_0 be finite can be dropped if the qualifier "in $\mathcal{L}_f$" be dropped as well. This can be seen when one considers the identity function and the class of finite languages. (The identity function is a string extension function, see §6.) In this case, $id(\Sigma^*) = \Sigma^*$, but Σ^* is not a member of $\mathcal{L}_{fin}$. However since the interest here is learners which generalize on the basis of finite experience, Theorem 3 is sufficient as is.

the learner ϕ is guaranteed to have converged to the target G as no additional words will add any more elements to the learner's grammar.

Lemma 2 For all $L \in \mathcal{L}_f$, there is a finite sample S such that L is the smallest language in $\mathcal{L}_f$ containing S. S is called a *characteristic sample* of L in $\mathcal{L}_f$ (S is also called a *tell-tale*).

Proof: For $L \in \mathcal{L}_f$, construct the sample S as follows. For each $g \in f(L)$, choose some word $w \in L$ such that $g \in f(w)$. Since $f(L)$ is finite (Remark 1), S is finite. Clearly $f(S) = f(L)$ and thus $L = L(f(S))$. Therefore, by Theorem 3, L is the smallest language in $\mathcal{L}_f$ containing S. □

Theorem 5 Fix $f \in \mathcal{SEF}$. Then ϕ_f identifies $\mathcal{L}_f$ in the limit.

Proof: For any $L \in \mathcal{L}_f$, there is a characteristic finite sample S for L (Lemma 2). Thus for any text t for L, there is i such that $S \subseteq content(t[i])$. Thus for any $j > i$, $\phi(t(j))$ is the smallest language in $\mathcal{L}_f$ containing S by Theorem 3 and Lemma 2. Thus, $\phi(t(j)) = f(S) = f(L)$. □

An immediate corollary is the efficiency of ϕ_f in the length of the sample, provided f is efficient in the length of the string (de la Higuera, 1997).

Corollary 1 ϕ_f is efficient in the length of the sample iff f is efficiently computable in the length of a string.

To summarize: string extension grammars are finite subsets of some set A. The class of languages they generate are determined by a function f which maps strings to finite subsets of A (chunks of grammars). Since the size of the canonical grammars is finite, a learner which develops a grammar on the basis of the observed words and the function f identifies this class exactly in the limit from positive data. It also follows that if f is efficient in the length of the string then ϕ_f is efficient in the length of the sample and that ϕ_f is globally consistent, locally conservative, and set-driven. It is striking that such a natural and general framework for generalization exists and that, as will be shown, a variety of language classes can be expressed given the choice of f.

5 Subregular examples

This section shows how classes which make up the subregular hierarchies (McNaughton and Papert, 1971) are string extension language classes. Readers are referred to Rogers and Pullum (2007) and Rogers et al. (2009) for an introduction to the subregular hierarchies, as well as their relevance to linguistics and cognition.

5.1 K-factor languages

The k-factors of a word are the contiguous subsequences of length k in w. Consider the following string extension function.

Definition 3 For some $k \in \mathbb{N}$, let

$$fac_k(w) = \{x \in \Sigma^k : \exists u, v \in \Sigma^* \text{ such that } w = uxv\} \text{ when } k \leq |w| \text{ and } \{w\} \text{ otherwise}$$

Following the earlier definitions, for some k, a grammar G is a subset of $\Sigma^{\leq k}$ and a word w belongs to the language of G iff $fac_k(w) \subseteq G$.

Example 1 Let $\Sigma = \{a, b\}$ and consider grammars $G = \{\lambda, a, aa, ab, ba\}$. Then $L(G) = \{\lambda, a\} \cup \{w : |w| \geq 2 \text{ and } w \notin \Sigma^* bb \Sigma^*\}$. The 2-factor bb is a *prohibited* 2-factor for $L(G)$. Clearly, $L(G) \in \mathcal{L}_{fac_2}$.

Languages in $\mathcal{L}_{fac_k}$ make distinctions based on which k-factors are permitted or prohibited. Since $fac_k \in \mathcal{SEF}$, it follows immediately from the results in §§3-4 that the k-factor languages are closed under intersection, and each has a characteristic sample. For example, a characteristic sample for the 2-factor language in Example 1 is $\{\lambda, a, ab, ba, aa\}$; i.e. the canonical grammar itself. It follows from Theorem 5 that the class of k-factor languages is identifiable in the limit by ϕ_{fac_k}. The learner ϕ_{fac_2} with a text from the language in Example 1 is illustrated in Table 1.

The class $\mathcal{L}_{fac_k}$ is not closed under union. For example for $k = 2$, consider $L_1 = L(\{\lambda, a, b, aa, bb, ba\})$ and $L_2 = L(\{\lambda, a, b, aa, ab, bb\})$. Then $L_1 \cup L_2$ excludes string aba, but includes ab and ba, which is not possible for any $L \in \mathcal{L}_{fac_k}$.

K-factors are used to define other language classes, such as the Strictly Local and Locally Testable languages (McNaughton and Papert, 1971), discussed in §5.4 and §5.5.

5.2 Strictly k-Piecewise languages

The Strictly k-Piecewise (SP_k) languages (Rogers et al., 2009) can be defined with a function whose co-domain is $\mathcal{P}(\Sigma^{\leq k})$. However unlike the function fac_k, the function SP_k, does not require that the k-length subsequences be contiguous.

i	$t(i)$	$fac_2(t(i))$	Grammar G	$L(G)$
-1			$\emptyset$	$\emptyset$
0	$aaaa$	$\{aa\}$	$\{\mathbf{aa}\}$	aaa^*
1	aab	$\{aa,\ ab\}$	$\{aa,\ \mathbf{ab}\}$	$aaa^* \cup aaa^*b$
2	a	$\{a\}$	$\{\mathbf{a},\ aa,\ ab\}$	$aa^* \cup aa^*b$
...				

Table 1: The learner ϕ_{fac_2} with a text from the language in Example 1. Boldtype indicates newly added elements to the grammar.

A string $u = a_1 \dots a_k$ is a *subsequence* of string w iff $\exists\ v_0, v_1, \dots\ v_k \in \Sigma^*$ such that $w = v_0 a_1 v_1 \dots a_k v_k$. The empty string λ is a subsequence of every string. When u is a subsequence of w we write $u \sqsubseteq w$.

Definition 4 For some $k \in \mathbb{N}$,

$$SP_k(w) = \{u \in \Sigma^{\leq k} : u \sqsubseteq w\}$$

In other words, $SP_k(w)$ returns all subsequences, contiguous or not, in w up to length k. Thus, for some k, a grammar G is a subset of $\Sigma^{\leq k}$. Following Definition 1, a word w belongs to the language of G only if $SP_2(w) \subseteq G$.[3]

Example 2 Let $\Sigma = \{a, b\}$ and consider the grammar $G = \{\lambda, a, b, aa, ab, ba\}$. Then $L(G) = \Sigma^* \backslash (\Sigma^* b \Sigma^* b \Sigma^*)$.

As seen from Example 2, SP languages encode long-distance dependencies. In Example 2, L prohibits a b from following another b in a word, no matter how distant. Table 2 illustrates ϕ_{SP_2} learning the language in Example 2.

Heinz (2007,2009a) shows that consonantal harmony patterns in natural language are describable by such SP_2 languages and hypothesizes that humans learn them in the way suggested by ϕ_{SP_2}. Strictly 2-Piecewise languages have also been used in models of reading comprehension (Whitney, 2001; Grainger and Whitney, 2004; Whitney and Cornelissen, 2008) as well as text classification(Lodhi et al., 2002; Cancedda et al., 2003) (see also (Shawe-Taylor and Christianini, 2005, chap. 11)).

5.3 K-Piecewise Testable languages

A language L is k-Piecewise Testable iff whenever strings u and v have the same subsequences of length at most k and u is in L, then v is in L as well (Simon, 1975; Simon, 1993; Lothaire, 2005).

A language L is said to be Piecewise-Testable (PT) if it is k-Piecewise Testable for some $k \in \mathbb{N}$. If k is fixed, the k-Piecewise Testable languages are identifiable in the limit from positive data (García and Ruiz, 1996; García and Ruiz, 2004). More recently, the Piecewise Testable languages has been shown to be linearly separable with a subsequence kernel (Kontorovich et al., 2008).

The k-Piecewise Testable languages can also be described with the function $SP_k^\diamond$. Recall that $f^\diamond(a) = \{f(a)\}$. Thus functions $SP_k^\diamond$ define grammars as a finite list of *sets of subsequences* up to length k that may occur in words in the language. This reflects the fact that the k-Piecewise Testable languages are the boolean closure of the Strictly k-Piecewise languages.[4]

5.4 Strictly k-Local languages

To define the Strictly k-Local languages, it is necessary to make a pointwise extension to the definitions in §3.

Definition 5 For sets $A_1, \dots, A_n$, suppose for each i, $f_i : \Sigma^* \to \mathcal{P}_{fin}(A_i)$, and let $f = (f_1, \dots, f_n)$.

1. A *grammar* G is a tuple $(G_1, \dots, G_n)$ where $G_1 \in \mathcal{P}_{fin}(A_1), \dots, G_n \in \mathcal{P}_{fin}(A_n)$.
2. If for any $w \in \Sigma^*$, each $f_i(w) \subseteq G_i$ for all $1 \leq i \leq n$, then $f(w)$ is a *pointwise subset* of G, written $f(w) \subseteq G$.
3. The *language of grammar* G is
$$L_f(G) = \{w : f(w) \subseteq G\}$$
4. The *class of languages* obtained by all such possible grammars G is $\mathcal{L}_f$.

[3] In earlier work, the function SP_2 has been described as returning the set of precedence relations in w, and the language class $\mathcal{L}_{SP_2}$ was called the precedence languages (Heinz, 2007; Heinz, to appear).

[4] More generally, it is not hard to show that $\mathcal{L}_{f^\diamond}$ is the boolean closure of $\mathcal{L}_f$.

i	$t(i)$	$SP_2(t(i))$	Grammar G	Language of G
-1			$\emptyset$	$\emptyset$
0	$aaaa$	$\{\lambda, a, aa\}$	$\{\lambda$**, a, aa**$\}$	a^*
1	aab	$\{\lambda, a, b, aa, ab\}$	$\{\lambda$, a, aa, **b, ab**$\}$	$a^* \cup a^*b$
2	baa	$\{\lambda, a, b, aa, ba\}$	$\{\lambda$, a, b, aa, ab, **ba**$\}$	$\Sigma^* \backslash (\Sigma^* b \Sigma^* b \Sigma^*)$
3	aba	$\{\lambda, a, b, ab, ba\}$	$\{\lambda$, a, b, aa, ab, ba$\}$	$\Sigma^* \backslash (\Sigma^* b \Sigma^* b \Sigma^*)$
...				

Table 2: The learner ϕ_{SP_2} with a text from the language in Example 2. Boldtype indicates newly added elements to the grammar.

These definitions preserve the learning results of §4. Note that the characteristic sample of $L \in \mathcal{L}_f$ will be the union of the characteristic samples of each f_i and the language $L_f(G)$ is the intersection of $L_{f_i}(G_i)$.

Locally k-Testable Languages in the Strict Sense (Strictly k-Local) have been studied by several researchers (McNaughton and Papert, 1971; Garcia et al., 1990; Caron, 2000; Rogers and Pullum, to appear), among others. We follow the definitions from (McNaughton and Papert, 1971, p. 14), effectively encoded in the following functions.

Definition 6 Fix $k \in \mathbb{N}$. Then the (left-edge) prefix of length k, the (right-edge) suffix of length k, and the interior k-factors of a word w are

$$L_k(w) = \{u \in \Sigma^k : \exists v \in \Sigma^* \text{ such that } w = uv\}$$

$$R_k(w) = \{u \in \Sigma^k : \exists v \in \Sigma^* \text{ such that } w = vu\}$$

$$I_k(w) = fac_k(w) \backslash (L_k(w) \cup R_k(w))$$

Example 3 Suppose $w = abcba$. Then $L_2(w) = \{ab\}$, $R_2(w) = \{ba\}$ and $I_2(w) = \{bc,\ cb\}$.

Example 4 Suppose $|w| = k$. Then $L_k(w) = R_k(w) = \{w\}$ and $I_k(w) = \emptyset$.

Example 5 Suppose $|w|$ is less than k. Then $L_k(w) = R_k(w) = \emptyset$ and $I_k(w) = \{w\}$.

A language L is *k-Strictly Local* (k-SL) iff for all $w \in L$, there exist sets $L, R,$ and I such that $w \in L$ iff $L_k(w) \subseteq L$, $R_k(w) \subseteq R$, and $I_k(w) \subseteq I$. McNaughton and Papert note that if w is of length less than k than L may be perfectly arbitrary about w.

This can now be expressed as the string extension function:

$$LRI_k(w) = (L_k(w), R_k(w), I_k(w))$$

Thus for some k, a grammar G is triple formed by taking subsets of Σ^k, Σ^k, and $\Sigma^{\leq k}$, respectively. A word w belongs to the language of G only if $LRI_k(w) \subseteq G$. Clearly, $\mathcal{L}_{LRI_k} = k$-SL, and henceforth we refer to this class as k-SL. Since, for fixed k, $LRI_k \in \mathcal{SEF}$, all of the learning results in §4 apply.

5.5 Locally k-Testable languages

The Locally k-testable languages (k-LT) are originally defined in McNaughton and Papert (1971) and are the subject of several studies (Brzozowski and Simon, 1973; McNaughton, 1974; Kim et al., 1991; Caron, 2000; García and Ruiz, 2004; Rogers and Pullum, to appear).

A language L is k-testable iff for all $w_1, w_2 \in \Sigma^*$ such that $|w_1| \geq k$ and $|w_2| \geq k$, and $LRI_k(w_1) = LRI_k(w_2)$ then either both w_1, w_2 belong to L or neither do. Clearly, every language in k-SL belongs to k-LT. However k-LT properly include k-SL because a k-testable language only distinguishes words whenever $LRI_k(w_1) \neq LRI_k(w_2)$. It is known that the k-LT languages are the boolean closure of the k-SL (McNaughton and Papert, 1971).

The function $LRI_k^\diamond$ exactly expresses k-testable languages. Informally, each word w is mapped to a set containing a single element, this element is the triple $LRI_k(w)$. Thus a grammar G is a subset of the triples used to define k-SL. Clearly, $\mathcal{L}_{LRI_k^\diamond} = k$-LT since it is the boolean closure of $\mathcal{L}_{LRI_k}$. Henceforth we refer to $\mathcal{L}_{LRI_k^\diamond}$ as the k-Locally Testable (k-LT) languages.

5.6 Generalized subsequence languages

Here we introduce generalized subsequence functions, a general class of functions to which the SP_k and fac_k functions belong. Like those functions, generalized subsequence functions map words to a set of subsequences found within the words. These functions are instantiated by a vector whose number of coordinates determine how many times a subsequence may be discontiguous

and whose coordinate values determine the length of each contiguous part of the subsequence.

Definition 7 For some $n \in \mathbb{N}$, let $\vec{v} = \langle v_0, v_1, \ldots, v_n \rangle$, where each $v_i \in \mathbb{N}$. Let k be the length of the subsequences; i.e. $k = \sum_0^n v_i$.

$$f_{\vec{v}}(w) = \{u \in \Sigma^k : \exists x_0, \ldots, x_n, u_0, \ldots, u_{n+1} \in \Sigma^* \text{ such that } w = u_0 x_0 u_1 x_1, \ldots, u_n x_n u_{n+1} \text{ and } |x_i| = v_i \text{ for all } 0 \leq i \leq n\} \text{ when } k \leq |w|, \text{ and} \{w\} \text{ otherwise}$$

The following examples help make the generalized subsequence functions clear.

Example 6 Let $\vec{v} = \langle 2 \rangle$. Then $f_{\langle 2 \rangle} = fac_2$. Generally, $f_{\langle k \rangle} = fac_k$.

Example 7 Let $\vec{v} = \langle 1, 1 \rangle$. Then $f_{\langle 1,1 \rangle} = SP_2$. Generally, if $\vec{v} = \langle 1, \ldots 1 \rangle$ with $|\vec{v}| = k$. Then $f_{\vec{v}} = SP_k$.

Example 8 Let $\vec{v} = \langle 3, 2, 1 \rangle$ and *a, b, c, d, e, f*$\in \Sigma$. Then $\mathcal{L}_{f_{\langle 3,2,1 \rangle}}$ includes languages which prohibit strings w which contain subsequences $abcdef$ where abc and de must be contiguous in w and $abcdef$ is a subsequence of w.

Generalized subsequence languages make different kinds of distinctions to be made than PT and LT languages. For example, the language in Example 8 is neither k-LT nor k'-PT for any values k, k'. Generalized subsequence languages properly include the k-SP and k-SL classes (Examples 6 and 7), and the boolean closure of the subsequence languages ($f^{\diamond}_{\vec{v}}$) properly includes the LT and PT classes.

Since for any $\vec{v}$, $f_{\vec{v}}$ and $f^{\diamond}_{\vec{v}}$ are string extension functions the learning results in §4 apply. Note that $f_{\vec{v}}(w)$ is computable in time $O(|w|^k)$ where k is the length of the maximal subsequences determined by $\vec{v}$.

6 Other examples

This section provides examples of infinite and nonregular language classes that are string extension learnable. Recall from Theorem 1 that string extension languages are finite unions of blocks of the partition of Σ^* induced by f. Assuming the blocks of this partition can be enumerated, the range of f can be construed as $\mathcal{P}_{fin}(\mathbb{N})$.

grammar G	Language of G
$\emptyset$	$\emptyset$
$\{0\}$	$a^n b^n$
$\{1\}$	$\Sigma^* \backslash a^n b^n$
$\{0,\ 1\}$	Σ^*

Table 3: The language class $\mathcal{L}_f$ from Example 9

In the examples considered so far, the enumeration of the blocks is essentially encoded in particular substrings (or tuples of substrings). However, much less clever enumerations are available.

Example 9 Let A = {0,1} and consider the following function:

$$f(w) = \begin{cases} 0 & \text{iff } w \in a^n b^n \\ 1 & \text{otherwise} \end{cases}$$

The function f belongs to $\mathcal{SEF}$ because it is maps strings to a finite co-domain. $\mathcal{L}_f$ has four languages shown in Table 3.

The language class in Example 9 is not regular because it includes the well-known context-free language $a^n b^n$. This collection of languages is also not closed under reversal.

There are also infinite language classes that are string extension language classes. Arguably the simplest example is the class of finite languages, denoted $\mathcal{L}_{fin}$.

Example 10 Consider the function id which maps words in Σ^* to their singleton sets, i.e. $id(w) = \{w\}$.[5] A grammar G is then a finite subset of Σ^*, and so $L(G)$ is just a finite set of words in Σ^*; in fact, $L(G) = G$. It follows that $\mathcal{L}_{id} = \mathcal{L}_{fin}$.

It can be easily seen that the function id induces the trivial partition over Σ^*, and languages are just finite unions of these blocks. The learner ϕ_{id} makes no generalizations at all, and only remembers what it has observed.

There are other more interesting infinite string extension classes. Here is one relating to the Parikh map (Parikh, 1966). For all $a \in \Sigma$, let $f_a(w)$ be the set containing n where n is the number of times the letter a occurs in the string w. For

[5] Strictly speaking, this is not the identity function per se, but it is as close to the identity function as one can get since string extension functions are defined as mappings from strings to sets. However, once the domain of the function is extended (Equation 1), then it follows that id is the identity function when its argument is a set of strings.

example $f_a(babab) = \{2\}$. Thus f_a is a total function mapping strings to singleton sets of natural numbers, so it is a string extension function. This function induces an infinite partition of Σ^*, where the words in any particular block have the same number of letters a. It is convenient to enumerate the blocks according to how many occurrences of the letter a may occur in words within the block. Hence, B_0 is the block whose words have no occurrences of a, B_1 is the block whose words have one occurrence of a, and so on.

In this case, a grammar G is a finite subset of $\mathbb{N}$, e.g. $\{2, 3, 4\}$. $L(G)$ is simply those words which have either 2, 3, or 4, occurrences of the letter a. Thus $\mathcal{L}_{f_a}$ is an infinite class, which contains languages of infinite size, which is easily identified in the limit from positive data by ϕ_{f_a}.

This section gave examples of nonregular and nonfinite string extension classes by pursuing the implications of Theorem 1, which established that $f \in \mathcal{SEF}$ partition Σ^* into blocks of which languages are finite unions thereof. The string extension function f provides an effective way of encoding all languages L in $\mathcal{L}_f$ because $f(L)$ encodes a finite set, the grammar.

7 Conclusion and open questions

One contribution of this paper is a unified way of thinking about many formal language classes, all of which have been shown to be identifiable in the limit from positive data by a string extension learner. Another contribution is a recipe for defining classes of languages identifiable in the limit from positive data by this kind of learner.

As shown, these learners have many desirable properties. In particular, they are globally consistent, locally conservative, and set-driven. Additionally, the learner is guaranteed to be efficient in the size of the sample, provided the function f itself is efficient in the length of the string.

Several additional questions of interest remain open for theoretical linguistics, theoretical computer science, and computational linguistics.

For theoretical linguistics, it appears that the string extension function $f = (LRI_3, P_2)$, which defines a class of languages which obey restrictions on both contiguous subsequences of length 3 and on discontiguous subsequences of length 2, provides a good first approximation to the segmental phonotactic patterns in natural languages (Heinz, 2007). The string extension learner for this class is essentially two learners: ϕ_{LRI_3} and ϕ_{P_2}, operating simultaneously.[6] The learners make predictions about generalizations, which can be tested in artificial language learning experiments on adults and infants (Rogers and Pullum, to appear; Chambers et al., 2002; Onishi et al., 2003; Cristiá and Seidl, 2008).[7]

For theoretical computer science, it remains an open question what property holds of functions f in $\mathcal{SEF}$ to ensure that $\mathcal{L}_f$ is regular, context-free, or context-sensitive. For known subregular classes, there are constructions that provide deterministic automata that suggest the relevant properties. (See, for example, Garcia et al. (1990) and Garica and Ruiz (1996).)

Also, Timo Kötzing and Samuel Moelius (p.c.) suggest that the results here may be generalized along the following lines. Instead of defining the function f as a map from strings to finite subsets, let f be a function from strings to elements of a lattice. A grammar G is an element of the lattice and the language of the G are all strings w such that f maps w to a grammar less than G. Learners ϕ_f are defined as the least upper bound of its current hypothesis and the grammar to which f maps the current word.[8] Kasprzik and Kötzing (2010) develop this idea and demonstrate additional properties of string extension classes and learning, and show that the pattern languages (Angluin, 1980a) form a string extension class.[9]

Also, hyperplane learning (Clark et al., 2006a; Clark et al., 2006b) and function-distinguishable learning (Fernau, 2003) similarly associate language classes with functions. How those analyses relate to the current one remains open.

Finally, since the stochastic counterpart of k-SL class is the n-gram model, it is plausible that probabilistic string extension language classes can form the basis of new natural language processing techniques. (Heinz and Rogers, 2010) show

[6] This learner resembles what learning theorists call *parallel learning* (Case and Moelius, 2007) and what cognitive scientists call *modular learning* (Gallistel and King, 2009).

[7] I conjecture that morphological and syntactic patterns are generally not amenable to a string extension learning analysis because these patterns appear to require a paradigm, i.e. a set of data points, before any conclusion can be confidently drawn about the generating grammar. Stress patterns also do not appear to be amenable to a string extension learning (Heinz, 2007; Edlefsen et al., 2008; Heinz, 2009).

[8] See also Lange et al. (2008, Theorem 15) and Case et al. (1999, pp.101-103).

[9] The basic idea is to consider the lattice $\mathbb{L} = \langle \mathcal{L}_{fin}, \supseteq \rangle$. Each element of $\mathbb{L}$ is a finite set of strings representing the intersection of all pattern languages consistent with this set.

how to efficiently estimate k-SP distributions, and it is conjectured that the other string extension language classes can be recast as classes of distributions, which can also be successfully estimated from positive evidence.

Acknowledgments

This work was supported by a University of Delaware Research Fund grant during the 2008-2009 academic year. I would like to thank John Case, Alexander Clark, Timo Kötzing, Samuel Moelius, James Rogers, and Edward Stabler for valuable discussion. I would also like to thank Timo Kötzing for careful reading of an earlier draft and for catching some errors. Remaining errors are my responsibility.

References

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

Dana Angluin. 1980b. Inductive inference of formal languages from positive data. *Information Control*, 45:117–135.

Dana Angluin. 1988. Identifying languages from stochastic examples. Technical Report 614, Yale University, New Haven, CT.

D. Beauquier and J.E. Pin. 1991. Languages and scanners. *Theoretical Computer Science*, 84:3–21.

Anselm Blumer, Andrzej Ehrenfeucht, David Haussler, and Manfred K. Warmuth. 1989. Learnability and the Vapnik-Chervonenkis dimension. *J. ACM*, 36(4):929–965.

J.A. Brzozowski and I. Simon. 1973. Characterization of locally testable events. *Discrete Math*, 4:243–271.

J.A. Brzozowski. 1962. Canonical regular expressions and minimal state graphs for definite events. In *Mathematical Theory of Automata*, pages 529–561. New York.

Nicola Cancedda, Eric Gaussier, Cyril Goutte, and Jean-Michel Renders. 2003. Word-sequence kernels. *Journal of Machine Learning Research*, 3:1059–1082.

Pascal Caron. 2000. Families of locally testable languages. *Theoretical Computer Science*, 242:361–376.

John Case and Sam Moelius. 2007. Parallelism increases iterative learning power. In *18th Annual Conference on Algorithmic Learning Theory (ALT07)*, volume 4754 of *Lecture Notes in Artificial Intelligence*, pages 49–63. Springer-Verlag, Berlin.

John Case, Sanjay Jain, Steffen Lange, and Thomas Zeugmann. 1999. Incremental concept learning for bounded data mining. *Information and Computation*, 152:74–110.

Kyle E. Chambers, Kristine H. Onishi, and Cynthia Fisher. 2002. Learning phonotactic constraints from brief auditory experience. *Cognition*, 83:B13–B23.

Alexander Clark, Christophe Costa Florêncio, and Chris Watkins. 2006a. Languages as hyperplanes: grammatical inference with string kernels. In *Proceedings of the European Conference on Machine Learning (ECML)*, pages 90–101.

Alexander Clark, Christophe Costa Florêncio, Chris Watkins, and Mariette Serayet. 2006b. Planar languages and learnability. In *Proceedings of the 8th International Colloquium on Grammatical Inference (ICGI)*, pages 148–160.

Alejandrina Cristiá and Amanda Seidl. 2008. Phonological features in infants phonotactic learning: Evidence from artificial grammar learning. *Language, Learning, and Development*, 4(3):203–227.

Colin de la Higuera. 1997. Characteristic sets for polynomial grammatical inference. *Machine Learning*, 27:125–138.

Matt Edlefsen, Dylan Leeman, Nathan Myers, Nathaniel Smith, Molly Visscher, and David Wellcome. 2008. Deciding strictly local (SL) languages. In Jon Breitenbucher, editor, *Proceedings of the Midstates Conference for Undergraduate Research in Computer Science and Mathematics*, pages 66–73.

Henning Fernau. 2003. Identification of function distinguishable languages. *Theoretical Computer Science*, 290:1679–1711.

C.R. Gallistel and Adam Philip King. 2009. *Memory and the Computational Brain*. Wiley-Blackwell.

Pedro García and José Ruiz. 1996. Learning k-piecewise testable languages from positive data. In Laurent Miclet and Colin de la Higuera, editors, *Grammatical Interference: Learning Syntax from Sentences*, volume 1147 of *Lecture Notes in Computer Science*, pages 203–210. Springer.

Pedro García and José Ruiz. 2004. Learning k-testable and k-piecewise testable languages from positive data. *Grammars*, 7:125–140.

Pedro Garcia, Enrique Vidal, and José Oncina. 1990. Learning locally testable languages in the strict sense. In *Proceedings of the Workshop on Algorithmic Learning Theory*, pages 325–338.

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

J. Grainger and C. Whitney. 2004. Does the huamn mnid raed wrods as a wlohe? *Trends in Cognitive Science*, 8:58–59.

Jeffrey Heinz and James Rogers. 2010. Estimating strictly piecewise distributions. In *Proceedings of the ACL*.

Jeffrey Heinz. 2007. *The Inductive Learning of Phonotactic Patterns*. Ph.D. thesis, University of California, Los Angeles.

Jeffrey Heinz. 2009. On the role of locality in learning stress patterns. *Phonology*, 26(2):303–351.

Jeffrey Heinz. to appear. Learning long distance phonotactics. *Linguistic Inquiry*.

J. J. Horning. 1969. *A Study of Grammatical Inference*. Ph.D. thesis, Stanford University.

Sanjay Jain, Daniel Osherson, James S. Royer, and Arun Sharma. 1999. *Systems That Learn: An Introduction to Learning Theory (Learning, Development and Conceptual Change)*. The MIT Press, 2nd edition.

Sanjay Jain, Steffen Lange, and Sandra Zilles. 2007. Some natural conditions on incremental learning. *Information and Computation*, 205(11):1671–1684.

Daniel Jurafsky and James Martin. 2008. *Speech and Language Processing: An Introduction to Natural Language Processing, Speech Recognition, and Computational Linguistics*. Prentice-Hall, Upper Saddle River, NJ, 2nd edition.

Anna Kasprzik and Timo Kötzing. to appear. String extension learning using lattices. In *Proceedings of the 4th International Conference on Language and Automata Theory and Applications (LATA 2010)*, Trier, Germany.

S.M. Kim, R. McNaughton, and R. McCloskey. 1991. A polynomial time algorithm for the local testability problem of deterministic finite automata. *IEEE Trans. Comput.*, 40(10):1087–1093.

Leonid (Aryeh) Kontorovich, Corinna Cortes, and Mehryar Mohri. 2008. Kernel methods for learning languages. *Theoretical Computer Science*, 405(3):223 – 236. Algorithmic Learning Theory.

Steffen Lange, Thomas Zeugmann, and Sandra Zilles. 2008. Learning indexed families of recursive languages from positive data: A survey. *Theoretical Computer Science*, 397:194–232.

H. Lodhi, N. Cristianini, J. Shawe-Taylor, and C. Watkins. 2002. Text classification using string kernels. *Journal of Machine Language Research*, 2:419–444.

M. Lothaire, editor. 2005. *Applied Combinatorics on Words*. Cmbridge University Press, 2nd edition.

Robert McNaughton and Seymour Papert. 1971. *Counter-Free Automata*. MIT Press.

R. McNaughton. 1974. Algebraic decision procedures for local testability. *Math. Systems Theory*, 8:60–76.

Kristine H. Onishi, Kyle E. Chambers, and Cynthia Fisher. 2003. Infants learn phonotactic regularities from brief auditory experience. *Cognition*, 87:B69–B77.

R. J. Parikh. 1966. On context-free languages. *Journal of the ACM, 13, 570581.*, 13:570–581.

James Rogers and Geoffrey Pullum. to appear. Aural pattern recognition experiments and the subregular hierarchy. *Journal of Logic, Language and Information*.

James Rogers, Jeffrey Heinz, Gil Bailey, Matt Edlefsen, Molly Visscher, David Wellcome, and Sean Wibel. 2009. On languages piecewise testable in the strict sense. In *Proceedings of the 11th Meeting of the Assocation for Mathematics of Language*.

John Shawe-Taylor and Nello Christianini. 2005. *Kernel Methods for Pattern Analysis*. Cambridge University Press.

Imre Simon. 1975. Piecewise testable events. In *Automata Theory and Formal Languages*, pages 214–222.

Imre Simon. 1993. The product of rational languages. In *ICALP '93: Proceedings of the 20th International Colloquium on Automata, Languages and Programming*, pages 430–444, London, UK. Springer-Verlag.

Carol Whitney and Piers Cornelissen. 2008. SERIOL reading. *Language and Cognitive Processes*, 23:143–164.

Carol Whitney. 2001. How the brain encodes the order of letters in a printed word: the SERIOL model and selective literature review. *Psychonomic Bulletin Review*, 8:221–243.