数理解析研究所講究録
第790巻 1992年 50-56

根方向型語彙機能文法の正データからの学習

Learning Frontier-to-Root Lexical-Functional Grammars from Positive Data

清水直昭，西野哲朗，一松信
Naoaki Shimizu, Tetsuro Nishino, and Shin Hitotsumatsu

東京電機大学 理工学部 情報科学科
Department of Information Sciences, Tokyo Denki University

Abstract

It is known that the class of languages generated by frontier-to-root lexical-functional grammars properly includes the class of context-free languages and is included in the class of context-sensitive languages. In this paper, we show an algorithm which identifies a structurally equivalent frontier-to-root lexical-functional grammar in the limit and generates a conjecture in polynomial time in the sum of the sizes of the input data.

1 Introduction

N. Chomsky proposed transformational grammars in 1965 in order to specify the syntax of natural languages. It is shown, however, that transformational grammars can generate recursively enumerable sets, that is, the generative capability of transformational grammars is too powerful. Since then, various grammars whose generative capability is weeker than transformational grammars have been proposed. As one of such grammars, in 1982, R. M. Kaplan and J. Bresnan proposed Lexical-Functional Grammars (LFGs) by extending context-free grammars (CFGs). In 1982, R. C. Berwick showed, however, that the membership problem for LFGs is NP-hard. Furthermore, some other results showing computational intractabilities of LFGs have been proved [4, 7]. So in 1990, T. Nishino proposed Frontier-to-Root LFGs (FRLFGs) as a subclass of LFGs meeting the dual demands of parsability and learnability from computational theoretic points of view [5]. FRLFGs are defined by attaching functional assignments to each production of CFGs. The class of languages generated by FRLFGs properly includes the class of context-free languages and is included in the class of context-sensitive languages [6].

In 1988, Y. Sakakibara showed an algorithm which generates a conjecture from structural descriptions of trees in polynomial time and identifies the unknown CFGs in the limit [8, 9]. Here, a structural description of a tree is a tree whose internal nodes are labeled by a single symbol. In this paper, we show an efficient learning algorithm for FRLFGs by extending the Sakakibara's algorithm.

2 Basic Definitions

In this section, we describe basic concepts in formal language theory. For details, see [2, 3]. Let Σ be an alphabet. The set of all finite length strings over Σ is denoted by Σ^*. Each string w in Σ^* has a finite *length*. A string of the length 0 is called an *empty string*, which is

denoted by ε. Let N be the set of non-negative integers, and $N_+ = N - \{0\}$. For $x, y \in N_+^*$, $x \preceq y$ iff there exists $z \in N_+^*$ such that $y = xz$, and $x \prec y$ iff $x \preceq y$ and $x \neq y$.

A *ranked alphabet* Σ is an alphabet associated with a finite relation $r_\Sigma \subseteq \Sigma \times N$. If $(a, n) \in r_\Sigma$, a is called a symbol of the *rank* n. We will denote the set of symbols of rank n by Σ_n. A symbol in the set Σ_0 is called a *constant symbol*.

Definition 1 A *tree* over Σ is a mapping t from $Dom(t)$ into Σ, where the domain $Dom(t)$ is a finite subset of N_+^* satisfying the following conditions :

1. If $x \in Dom(t)$ and $y \prec x$, then $y \in Dom(t)$,
2. If $yi \in Dom(t)$ for $i \in N_+$, then $yj \in Dom(t)$ for $j \in N_+$, $1 \leq j \leq i$,
3. If $t(x) \in \Sigma_n$, then $xi \in Dom(t)$ for $i \in N_+$, $1 \leq i \leq n$.

An element of $Dom(t)$ is called a *node* in t. If $t(x) = a$, then a is said to be the *label* of the node x in t. The set of all trees over Σ is denoted by $\mathcal{T}_\Sigma$.

Let $t \in \mathcal{T}_\Sigma$. A node x in t is a *terminal node* or a *leaf* iff for all $y \in Dom(t)$, $x \not\prec y$. While a node x in t is an *internal node* iff x is not a terminal node. Especially, a node $\varepsilon \in Dom(t)$ is called a *root* of t.

Definition 2 A ranked alphabet Σ uniquely determines a set $Term(\Sigma)$ of *terms* over Σ defined to be the least subset of Σ^* satisfying the following conditions :

1. $\Sigma_0 \subseteq Term(\Sigma)$,
2. If $f \in \Sigma_n$ and $t_1, t_2, \ldots, t_n \in Term(\Sigma)$, then $f(t_1, t_2, \ldots, t_n) \in Term(\Sigma)$.

Since the finite trees over Σ can be identified with terms over Σ, we will represent trees as terms.

Let σ be a special symbol. A *skeletal alphabet* $\Sigma = \bigcup_{i=0}^{n} \Sigma_i$ with the maximal rank n is a ranked alphabet such that $\Sigma_i = \{\sigma\}$ for each i, $1 \leq i \leq n$. A tree over a skeletal alphabet is called a *skeleton*. The *structural description* of a tree t, which is denoted by $s(t)$, is a skeleton with $Dom(s(t)) = Dom(t)$ which satisfies the following condition : if x is a terminal node in t then $s(t)(x) = t(x)$ else $s(t)(x) = \sigma$. The *skeleton set corresponding* a set T of trees, which is denoted by $K(T)$, is $\{s(t) \mid t \in T\}$.

A *context-free grammar* (CFG for short) Gr is a 4-tuple (NA, TA, P, S), where NA, TA, P, and S are *nonterminals, terminals, productions*, and a *start symbol*, respectively. We assume that $NA \cap TA = \emptyset$. A production in P is of the form $A \rightarrow B_1 \cdots B_n$, where $n \geq 1$, $A \in NA$, and $B_i \in NA \cup TA$ $(1 \leq i \leq n)$. If $n = 1$, B_1 may be an empty string ε. If $A \rightarrow \beta \in P$, then for any $\alpha, \gamma \in (NA \cup TA)^*$, we write $\alpha A \gamma \Rightarrow \alpha\beta\gamma$, where $\beta \in (NA \cup TA)^*$. $\stackrel{*}{\Rightarrow}$ is the *reflexive and transitive closure* of $\Rightarrow$. The *language generated by* Gr, which is denoted by $L(Gr)$, is the set $\{ w \mid w \in TA^* \text{ and } S \stackrel{*}{\Rightarrow} w \}$. A language L is called a *context-free language* if $L = L(Gr)$ for some CFG Gr.

Definition 3 Let $Gr = (NA, TA, P, S)$ be a CFG. For $A \in NA \cup TA \cup \{\varepsilon\}$, the set $D_A(Gr)$ of *derivation trees* of Gr from A is recursively defined to be a set of trees over $NA \cup TA$ as follows :

$$D_A(Gr) = \begin{cases} \{A\} & \text{if } A \in TA \cup \{\varepsilon\}, \\ \{A(t_1, \ldots, t_k) \mid A \rightarrow B_1 \cdots B_k \in P,\ t_i \in D_{B_i}(Gr) \text{ for } 1 \leq i \leq k\} & \text{if } A \in NA. \end{cases}$$

For the set $D_S(Gr)$ of derivation trees of Gr from the start symbol S, the S-subscript will be abbreviated. A derivation tree is said to be *complete* iff its root is labeled by S and all its leaves are labeled by terminal symbols or ε's. A *structural description* of a CFG Gr is a skeleton in $K(D(Gr))$. Two CFGs Gr_1 and Gr_2 is said to be *structurally equivalent* iff $K(D(Gr_1)) = K(D(Gr_2))$.

A CFG $Gr = (NA, TA, P, S)$ is said to be *invertible* iff $A \rightarrow \alpha$ and $B \rightarrow \alpha$ in P implies $A = B$. A CFG $Gr = (NA, TA, P, S)$ is said to be *reset-free* iff, for $B, C \in NA$ and $\alpha, \beta \in (NA \cup TA)^*$, $B = C$ whenever $A \rightarrow \alpha B \beta$ and $A \rightarrow \alpha C \beta$ in P. A CFG Gr is said to be *reversible* iff Gr is invertible and reset-free. A CFG $Gr = (NA, TA, P, S)$ is said to be *extended reversible* iff, for $P' = P - \{S \rightarrow a \mid a \in TA \cup \{\varepsilon\}\}$, $Gr' = (NA, TA, P', S)$ is reversible. An extended reversible CFG is a normal form for CFGs [9].

3 Frontier-to-Root Lexical-Functional Grammars

In order to specify the syntax of natural languages, T. Nishino introduced frontier-to-root lexical-functional grammars (FRLFGs for short) in 1990 [5]. An FRLFG is defined by attaching functional assignments to each production of a CFG.

An FRLFG describes a set of grammatical sentences by using two types of trees, *constituent trees* (*c-trees*) and *functional trees* (*f-trees*). A c-tree is a derivation tree of a CFG. An f-tree is a rooted ordered tree whose nodes are labeled by a special symbol \$, function names, and function values. First, we give a formal definition of an FRLFG. For details, see [5, 6].

Definition 4 A *frontier-to-root lexical-functional grammar* (FRLFG for short) G is a 6-tuple (NA, TA, S, FN, FV, AR) consists of 1-6 as follows :

1. NA is a *nonterminal alphabet.*
2. TA is a *terminal alphabet.* We assume that $NA \cap TA = \emptyset$.
3. $S \in NA$ is a *start symbol.*
4. FN is a finite set of *function names.*
5. FV is a finite set of *function values.* We assume that $FN \cap FV = \emptyset$.
6. AR is a finit set of *annotated phrase structure rules.* An annotated phrase structure rule is of the form

$$A \rightarrow (B_1, E_1)(B_2, E_2) \cdots (B_n, E_n),$$

where $n \geq 1$, $A \in NA$, and $B_i \in NA \cup TA$ $(1 \leq i \leq n)$. We assume that at least one of $B_1, B_2, \ldots, B_n$ is a nonterminal symbol if $n \geq 2$. If $n = 1$, B_1 may be an empty string ε. E_i is a set of *functional assignments.* A functional assignment is a statement of one of the following forms :

i. (in the case when $B_i \in NA$)

$((\uparrow F_1)F_2) := \downarrow$, $\quad (\uparrow F_1) := \downarrow$, $\quad \uparrow := \downarrow$,

ii. (in the case when $B_i \in TA \cup \{\varepsilon\}$)

$((\uparrow F_1)F_2) := V$, $\quad (\uparrow F_1) := V$, $\quad \uparrow := V$,

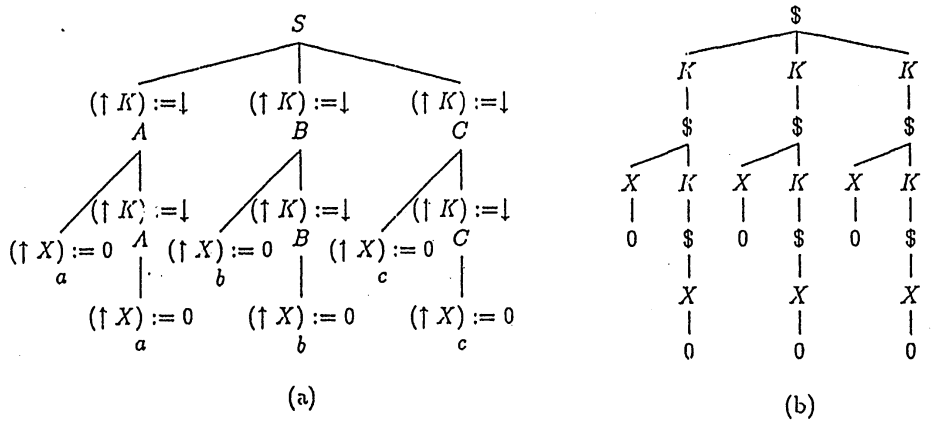

Figure 1: A derivation of a string $x = aabbcc$, (a) an annotated phrase structure tree for x, (b) a well-formed f-tree $f(x)$.

where $F_1, F_2 \in FN$ and $V \in FV$. The symbols $\uparrow$ and $\downarrow$ are called *metavariables*. Annotated phrase structure rules of the forms $A \rightarrow (b, E)$ and $A \rightarrow (\varepsilon, E)$ are especially called a *lexical insertion rule* and an *ε-rule* respectively, where $b \in TA$ and E is a nonempty finite set of functional assignments of the forms in ii. We assume that *each set of functional assignments is a singleton except the sets attached to the lexical insertion rules and the ε-rules.*

For an FRLFG $G = (NA, TA, S, FN, FV, AR)$, the CFG $Gr = (NA, TA, P, S)$ is called the *underlying context-free grammar* of G, where

$$P = \{ A \rightarrow B_1 B_2 \cdots B_n \mid A \rightarrow (B_1, E_1)(B_2, E_2) \cdots (B_n, E_n) \in AR \}.$$

For an annotated phrase structure rule $r : A \rightarrow (B_1, E_1)(B_2, E_2) \cdots (B_n, E_n)$, the production $A \rightarrow B_1 B_2 \cdots B_n$ is called the *underlying rule* of r. For any FRLFG, we assume the followings.

- *The underlying CFG is cycle-free and extended reversible.*
- *There are no two distinct annotated phrase structure rules having the same underlying rule.*

A *c-tree* is a derivation tree of an underlying CFG of an FRLFG. An *annotated phrase structure tree* is a tree obtained by attaching functional assignments to a c-tree. Fig.1 (a) illustrates an example of an annotated phrase structure tree. An f-tree f is a rooted ordered tree satisfying the following conditions : the root of f is labeled by a special symbol \$; each internal node in f, excluding the root, is labeled by \$ or a function name; and the leaves of f are labeled by function values. Fig.1 (b) illustrates an example of an f-tree. A *\$-free-tree* corresponding to an f-tree f is an f-tree obtained by removing all internal nodes of f, excepting the root of f, whose labeles are \$. We assume that each internal node in an annotated phrase structure tree is associated with an f-tree. An f-tree associated to the root of an annotated phrase structure tree for a string x, which is denoted by $f(x)$, is called the f-tree *assigned* to x.

In the FRLFG theory, three *well-formedness conditions*, called *uniqueness, completeness*, and *coherency* are defined on f-trees. For details of the well-formedness conditions,

see [6]. The f-tree shown in Fig.1 (b) satisfies the well-formedness conditions. An f-tree f is said to be *well-formed* iff f satisfies well-formedness conditions. A terminal string x is said to be *grammatical* only if it has a valid c-tree and is assigned a well-formed f-tree $f(x)$. A *language generated by an FRLFG* G, which is denoted by $L(G)$, is a set of the grammatical strings of G. If the underlying CFG of G is ambiguous, in order to decide whether $x \in L(G)$, we need to check the well-formedness on f-trees for all c-trees of x. If x is assigned at least one well-formed f-tree $f(x)$ then $x \in L(G)$, else $x \notin L(G)$. The class of languages generated by FRLFGs is denoted by $\mathcal{L}_{FRLFG}$. Let CFL be the class of context-free languages, and CSL be the class of context-sensitive languages. T. Nishino has shown the following theorem about the generative power of FRLFGs.

Theorem 1 [6] $CFL \subsetneq \mathcal{L}_{FRLFG} \subseteq CSL$ □

Let G and G' be FRLFGs. And let Gr and Gr' be the underlying CFGs of G and G', respectively. G is said to be *equivalent* to G' iff

(1) $L(Gr) = L(Gr')$, and
(2) G assignes f-trees whose \$-free-trees are the same as the ones of f-trees which G' assignes to every string.

And G is said to be *structurally equivalent* to G' iff

(1) $K(D(Gr)) = K(D(Gr'))$, and
(2) G assignes f-trees whose \$-free-trees are the same as the ones of f-trees which G' assignes to every skeleton.

4 The Learning Algorithm

In [1], E. M. Gold introduced a fundamental concept in inductive inference called *identification in the limit*. An inference machine M identifies a set of rules R in the limit iff, given a set of examples of R, an output sequence which M generates *converge* to some expression τ for a set of rules, and τ is a correct expression for R. In [8, 9], Y. Sakakibara proposed an algorithm identifying a CFG which is structurally equivalent to an unknown CFG Gr from a set of structural descriptions of Gr. This algorithm generates a conjecture in polynomial time in the sum of the sizes of the input skeletons. In this section, we show that an unknown FRLFG is learnable in the limit by extending the Sakakibara's algorithm. Before we show our learning algorithm LA for FRLFGs, we define some related terminologies.

Let Sa be a finite set of skeletons. We define the *primitive context-free grammar* $Gr(Sa) = (NA, TA, P, S)$ for Sa as follows :

$$\begin{aligned} NA &= (Sub(Sa) - (TA \cup \{\varepsilon\})) \cup \{S\}, \\ P &= \{\sigma(A_1, \ldots, A_k) \rightarrow A_1 \cdots A_k \mid \sigma(A_1, \ldots, A_k) \in NA\} \\ &\quad \cup \{S \rightarrow A_1 \cdots A_k \mid \sigma(A_1, \ldots A_k) \in Sa\}, \end{aligned}$$

where $Sub(Sa)$ is the set of subtrees of the elements in Sa. Then $Gr(Sa)$ is a CFG such that $K(D(Gr(Sa))) = Sa$.

Let $\Gamma^G = \bigcup_{j=0}^{m} \Gamma_j^G$ be a ranked alphabet such that $\Gamma_0^G = FV$, $\Gamma_1^G = FN \cup \{\$\}$ and $\Gamma_i^G = \{\$\}$ $(2 \leq i \leq m)$. And let $t \in T_{\Gamma^G}$. Now we define a new operation $GETF$ as follows :

$$GETF(t) = \begin{cases} t & \text{if } ROOT(t) \in FV \cup \{\$\}, \\ GETF(t') & \text{if } ROOT(t) \in FN, \end{cases}$$

where t' is a subtree of t whose root is the unique son of the root of t and $ROOT(t)$ denotes the label of the root of t.

We describe an outline of our learning algorithm LA for FRLFGs. Let $G_U = (NA, TA, S, FN, FV, AR)$ be an unknown FRLFG and Gr be the underlying CFG of G_U. It is assumed that the learner knows TA, FV, FN, and the maximum length m of the right hand side of the productions in Gr. The learner is presented pairs of a structural description t of a complete c-tree and an f-tree associated to the root of t, which are generated by G_U. That is, the learner is presented only positive data of G_U.

Our learning algorithm LA consists of the folowing two step.

Step 1. Learning of the underlying CFG of G_U.

Step 2. Learning of the set of functional assignments of G_U.

An outline of the learning process of the underlying CFG Gr of G_U is as follows (for details, see [8, 9]). Let Sa be a finite set of structural descriptions of c-trees. We first consider the case when skeletons of the form $\sigma(x)$ $(x \in TA \cup \{\varepsilon\})$ are not included in Sa. Given an input Sa, LA first constructs the primitive CFG $Gr_0 = Gr(Sa) = (NA_0, TA, P_0, S_0)$ for Sa. Then it merges two distinct nonterminal symbols A and B repeatedly if one of the following conditions is satisfied.

1. There exist two productions of the forms $A \rightarrow \alpha$ and $B \rightarrow \alpha$.

2. There exist two productions of the forms $C \rightarrow \alpha A \beta$ and $C \rightarrow \alpha B \beta$.

When there no longer remains a pair of nonterminal symbols satisfying the above 1 or 2, the resulting grammar is the underlyning CFG of an FRLFG which LA outputs as a conjecture.

Next we consider the case when skeletons of the form $\sigma(x)$ $(x \in TA \cup \{\varepsilon\})$ are included in Sa. Let Uni be the set of skeletons of the form $\sigma(x)$ included in Sa. And let $Sa' = Sa - Uni$. First, LA performs the above operation on Sa'. Let $Gr' = (NA', TA, P', S')$ be the resulting CFG. Then, let $P'' = \{ S' \rightarrow x \mid \sigma(x) \in Uni \}$. The resulting CFG is $(NA', TA, P' \cup P'', S')$.

Next, we describe an outline of Step 2. The sets of functional assignments of the unknown FRLFG G_U can be obtained in the following way.

Case 1. Let $t = \sigma(x)$ be a structural description of a c-tree suct that $x \in TA \cup \{\varepsilon\}$, and $\$(f_1, \ldots, f_k)$ be the f-tree associated to the root of t. And let E be the set of functional assignments associated to x whose initial value is an empty set $\emptyset$. For each i, $1 \leq i \leq k$, if f_i is $F_1(F_2(V))$ for some $F_1, F_2 \in FN$ and $V \in FV$, then add a functional assignment of the form $((\uparrow F_1)F_2) := V$ to E. The other types of f_i's are similarly processed as special cases when both F_1 and F_2 are null or only F_2 is null.

Case 2. Let $t = \sigma(t_1, \ldots, t_k)$ be a structural description of a c-tree, which was not processed in Case 1, and $\$(f_1, \ldots, f_k)$ be the f-tree associated to the root of t. And let E_i be the set of functional assignments associated to the root of t_i for each i, $1 \leq i \leq k$.

1. If $t_i \in TA$, then E_i is obtained in the same way as Case 1. In this case, E_i is a singleton.

2. If $t_i \notin TA$, LA learns E_i according to the following procedure. Note that the f-tree associated to the root of t_i is $GETF(f_i)$. If $f_i - GETF(f_i) = F_1(F_2)$ $(F_1, F_2 \in FN)$, then $E_i = \{ ((\uparrow F_1)F_2) := \downarrow \}$, where "$-$" represents a subtree pruning operation. The

cases when the result of $f_i - GETF(f_i)$ is λ (an empty tree) or F_1 are similarly processed.

Theorem 2 *The algorithm LA identifies in the limit an FRLFG which is structurally equivalent to an unknown FRLFG G_U from positive data of G_U. Further, algorithm LA may be implemented to generate a conjecture in polynomial time in the sum of the sizes of the input skeletons that have ever been read, where the size of a skeleton is the number of nodes in it.* □

5 Conclusion

In this paper, we described an outline of an efficient learning algorithm for FRLFGs whose underlying CFGs are extended reversible. In order to guarantee the learnability of any FRLFG, we have to show that, for any FRLFG G, there exists an FRLFG equivalent to G such that the underlying CFG is extended reversible. This is a subject for the further research.

References

[1] E. M. Gold, Language Identification in the Limit. *Information and Control*, 10, pp.447-474, 1967.

[2] J. E. Hopcroft and J. D. Ullman, *Introduction to Automata Theory, Languages, and Computation*, Addison-Wesley, 1979.

[3] 小林孝次郎，高橋正子,「オートマトンの理論」，共立出版，1983.

[4] 中西隆一，関浩之，嵩忠雄,「語彙機能文法の生成能力について」，'91 夏のＬＡシンポジウム資料，1991 年 7 月.

[5] T. Nishino, An Efficiently Parsable and Learnable Subclass of Lexical-Functional Grammars, *IEICE Technical Report*, 90:25, pp.55-64, 1990.

[6] T. Nishino, *Formal Methods in Natural Language Syntax*, Doctoral Dissertation, Waseda University, 1991.

[7] T. Nishino, N. Shimizu, S. Yamada, and T. Yaku, On Normal Forms and Decision Problems for Lexical-Functional Grammars, *IEICE Technical Report*, 90:93, pp.21-32, 1991.

[8] Y. Sakakibara, Learning Context-Free Grammars from Structural Data in Polynomial Time, In *Proceedings of 1st Workshop on Computational Learning Theory*, pp.330-344, 1988.

[9] Y. Sakakibara, *An Efficient Learning of Context-Free Grammars from Positive Structural Examples*, TR-93, IIAS-SIS, Fujitsu Limited, 1989.