LFGにおける機能スキーマの帰納的獲得の試み*

4B－7

小島丈幸　山口昌也　乾伸雄　小谷善行　西村恕彦
(東京農工大学　工学部　電子情報工学科)

1　はじめに

これまでの文法学習の研究では、CFGが用いられることが多い。これらの研究では、文法的曖昧性を確率CFGで解消する[1]。一方で、HPSGやLFGを用いて、確率に基づかない制約による曖昧性解消を行なうこともできるが、学習方法はまだ明らかではない。

本稿では、木構造で与えられる表層構造と機能構造で与えられる深層構造からLFGの構成素構造規則(以降、C規則と略記)を帰納的に獲得するアルゴリズムを提案する。この際に、語彙に関する辞書は既知の情報としてシステムに持たせることとした。

2　機能構造と機能スキーマ

LFGのC規則は、CFG規則に機能スキーマと呼ばれる制約の記述を加えたものである[2]。本稿で提案する獲得アルゴリズムでは、木構造を入力にとる。つまりCFG規則を与えることになるので、実際に獲得するのは、機能スキーマだけである。

2.1　機能構造

機能構造 F は、LFGにおいて深層構造を表し、属性 a とその属性値 v の対 $\langle a, v\rangle$ を要素として持つ集合である。機能構造 F が属性名 a に v という値を持つこと、すなわち、$\langle a, v\rangle \in F$ を $(a\ F) = v$ と書く。

また、$(a\ F)$ が機能構造であることを $fst[(a\ F)]$、機能因子(機能構造以外)であることを $\neg fst[(a\ F)]$ と書くことにする。

2.2　機能スキーマの種類

n 個の子ノードを持つC規則

$$P \rightarrow C_1\ C_2\ C_3\ \cdots\ C_n \qquad (1)$$

を考える。本稿で扱うC規則には、一つの子ノード $C_i\ (i = 1, \cdots, n)$ につき継承性スキーマ θ_i が一つ、制限性スキーマ $\lambda_{ij}\ (j = 1, \cdots, r_i)$ が $r_i\ (\geq 0)$ 個つくものとする。

継承性スキーマ θ_i は、主に親子ノードの機能構造の対応関係を規定する。子ノードの機能構造をそのまま親ノードに持ち上げることを示す $\uparrow{=}\downarrow$ や、子ノードの機能構造を親ノードの属性名 a の値にすることを示す $(a, \uparrow) = \downarrow$ が継承性スキーマである。

制限性スキーマ $\lambda_{ij} = \langle a_{ij}, v_{ij}\rangle$ は、そのC規則が適用可能な子ノードを規定する。$(a_{ij}, \downarrow) = v_{ij}$ の形式を持ち、$(a_{ij}\ F_i) = v_{ij}$ を要求する。

2.3　機能構造の含有因子

ある機能構造 F に対して、F に含まれているすべての機能因子を取り出したものを含有因子 F^* とする。

$$F^* = \bigcup_{fst[(F\ a)]} \{\langle a, (F\ a)\rangle\} \cup \bigcup_{\neg fst[(F\ a)]} (F\ a)^*.$$

含有因子の性質上、式(1)の親ノードの機能構造 F_0 の含有因子 F_0^* は、子ノード $C_i (i = 1 \cdots n)$ の機能構造を F_i とすると、$F^* = \bigcup_i F_i^*$ となる。語彙に関する辞書を与えた場合には、木構造の葉ノードの機能構造の含有因子が分かり、木構造を与えていればトップノードの機能構造の含有因子を得ることができる。

3　機能スキーマの生成

式(1)の親ノード P の機能構造 F_0 の属性値 $(a\ F_0) = v$ を順に取り出し、それがどの子ノードから由来しているのかを調べて子ノードにつく機能スキーマを生成する。この段階では、制限性スキーマを最もきつい制約になるように付ける。

3.1　機能因子に対する機能スキーマ

$(a\ F_0) = v, \neg fst[v]$ のとき、$\langle a, v\rangle \in F_i^*$ なら $\langle a, v\rangle$ がその子ノード C_i から由来したものとする。

---

*Inductive Acquisition of Functional Schemata on LFG,
Takeyuki KOJIMA,Masaya YAMAGUCHI, Nobuo INUI, Yoshiyuki KOTANI, Hirohiko NISIMURA
Tokyo University of Agric. and Tech., Dept. of Computer Science

$\langle a, v\rangle \in F_i^*$ となる可能性は二つ考えられる。一方は $(a\ F_i) = v$ の場合で、$F_i \subseteq F_0$ である。このとき、継承性スキーマは $\uparrow=\downarrow$ がつき、制限性スキーマは $(a, \downarrow) = v$ がつく。

他方は $(a\ F_i) \neq v$ でも、$\langle a, v\rangle \in (a\ F_i)^*$ が成り立つ場合である。この場合には継承性スキーマ $\uparrow=\downarrow$ はつかないこともある。

しかし、F_i^* が既知でも F_i は未知なので、$(a\ F_i) = v$ が成り立つかどうかを判断することはできない。本研究では、後者があり得ないものと仮定した。

3.2 機能構造に対する機能スキーマ

$(a\ F_0) = v, fst[v]$ のとき、$v^* \subseteq F_i^*$ になら $\langle a, v\rangle$ が C_i から由来したものとする。

$v^* \subseteq F_i^*$ となる可能性も二つに分けて考えられる。一方は $F_i = v$ のときで、継承性スキーマとしては $(a, \downarrow) = \downarrow$ がつく。制限性スキーマは $\langle a', v'\rangle \in v$かつ $\neg fst[a']$ を満たすすべての a' について、$(a', \downarrow) = v'$ がつく。

他方は、$(a\ F_i) = v$ のときである。このときは継承性スキーマとして、$\uparrow=\downarrow$ をつける必要がある。しかし、ここでも F_i が未知なために、$(a\ F_i) = v$ が成り立つかどうかを調べることができない。本研究では、機能構造に関する機能スキーマの生成にさきだって機能因子に関する機能スキーマの生成を行い、すでに $\uparrow=\downarrow$ がついている場合は $(a\ F_i) = v$ を、ついていない場合は $F_i = v$ を仮定した。

3.3 規則の一般化

少ない例から効率良く規則を獲得するために、機能スキーマの生成によって得られた規則の一般化を行う。ここで一般化の対象にするのは、過剰に付加された制限性スキーマである。

親ノード P とすべての子ノード C_i 、そして各子ノードに付加された継承性スキーマ θ_i が等しい二つの C 規則 R, R' の間に距離 $d(R, R')$ を定義する。子ノード C_i の制限性スキーマを $\Lambda_i = (\lambda_{i1}, \lambda_{i2}, \cdots, \lambda_{ir_i})$ として、距離 $d(R, R')$ は、

$$d(R, R') = \sum_{i=1}^{n} d(\Lambda_i, \Lambda_i'), \quad (2)$$

$$d(\Lambda_i, \Lambda_i') = d_0(\Lambda_i - X, \Lambda_i' - X), \quad (3)$$

ただし、$X = \Lambda_i \cap \Lambda_i'$

$$d_0(A, B) = \left|\{a|\langle a, v\rangle \in A\} \cup \{a|\langle a, v\rangle \in B\}\right| \quad (4)$$

になる。

一般化は $d(R, R') = 1$ のときに行うが、R の $\langle a_{ij}, v_{ij}\rangle$ と R' の $\langle a_{ij}, v_{ij}'\rangle$ 以外の制限性スキーマがすべて等しい場合と、一方の $\langle a_{ij}, v_{ij}\rangle$ が他方の Λ_i に含まれていない場合がある。

前者の場合は、二つの C 規則から属性名 a_{ij} に関する制限性機能スキーマを消去することで C 規則の一般化ができる。後者の場合は、属性名 a_{ij} のない C 規則に吸収される形で一般化がなされる。

4 獲得例

本稿で提案した獲得アルゴリズムへの入力の木構造、機能構造、そして、生成された規則、ほかの規則を元に一般化した規則の例を示す。

木構造: 文 (助詞句 (名詞句 (名詞 (犬)), 助詞 (が)),
助詞句 (名詞句 (名詞 (猫)), 助詞 (を)),
動詞 (噛む))

機能構造:

動作主	pred	犬
	格マーカ	ガ
対象	pred	猫
	格マーカ	ヲ
pred	噛む	
時制	現在	

生成規則:助詞句→

名詞句	助詞
$\uparrow=\downarrow$	$\uparrow=\downarrow$
$(\mathbf{pred}, \downarrow) =$ 犬	$(格マーカ, \downarrow) =$ ガ

一般化規則:助詞句 →

名詞句	助詞
$\uparrow=\downarrow$	$\uparrow=\downarrow$
	$(格マーカ, \downarrow) =$ ガ

5 おわりに

本稿では、木構造と機能構造から帰納的に LFG の機能スキーマを獲得するアルゴリズムを提案した。より表現力の強い機能スキーマに対する獲得や過剰一般化を防ぐ機構が今後の課題として挙げられる。

参考文献

[1] 白井 清昭, 徳永 健伸, 田中 穂積 : コーパスからの文法の自動抽出, 情報処理学会自然言語処理研究会, 94-NL-101-11, 1994.

[2] 淵 一博 監修, 古川 康一, 溝口 文雄 共編 : 自然言語の基礎理論 , 共立出版, 1986.