構造付きコーパスからの統語的知識の自動獲得とその精密化

橋本泰一　白井清昭　徳永健伸　田中穂積
東京工業大学大学院 情報理工学研究科 計算工学専攻
{taiichi,kshirai,take,tanaka}@cs.titech.ac.jp

コーパスなどの言語資源から自動的に獲得された文脈自由文法 (CFG) は、多様な文を構文解析できる反面、数多くの不要な構文木を生成するといった問題がある。本論文では、一般化 LR 法を用いて文法から LR 表を作成し、訓練コーパスの例文を構文解析する際にあまり使われないアクションを LR 表から削除することにより、構文的曖昧性を抑制する手法を提案する。本手法は、規則を削除することで構文的曖昧性を抑制する手法に比べ、より精密に不要な構文木の生成を妨げることができると考えられる。予備実験の結果、文法の受理率を維持しながら LR 表の大きさを 90%縮小することができた。

The Automatic Extraction and Refinement of Syntactic Information from a Bracketted Corpus

Taiichi Hashimoto, Kiyoaki Shirai, Takenobu Tokunaga, Hozumi Tanaka
Department of Computer Science, Graduate School of Information Science and Engineering,
Tokyo Institute of Technology
{taiichi,kshirai,take,tanaka}@cs.titech.ac.jp

Using a context free grammar (CFG) automatically acquired from a large corpus allows us analyse a wide array sentence types. On the other hand, such a method may also produce many unnecessary parse trees. In this paper, we propose a new method of decreasing the number of superfluous parse trees, in which we generate the LR table from the CFG, then remove actions which are not frequently used for parsing sentences in the training corpus. This method enables us to suppress to prevent the production of producing unnecessary parse trees more subtly than would be the case for removal of rules in the CFG. Based on the results of a preliminary experiment, we can reduce the size of the LR table by 90 %,retaining the coverage of the original CFG.

1　はじめに

近年、言語データの電子化と計算機性能の向上により大量の言語データから言語知識を自動獲得する研究が行われている。多くの言語知識の中でも、文法、特に文脈自由文法 (Context Free Grammar 以下 CFG) は、効率的な解析アルゴリズムが多く存在し、構文解析などで用いられる場合が多い。この様に、CFG は利用価値の高い知識であるが、人手でこれを作成しようとした場合、コストが高く効率的でない。また様々な言語現象に人手で対応することは、非常に困難である。これに対し、大量のコーパスを用いて CFG を自動獲得することができれば、多様な言語現象に対応できる CFG が得られる。Charniak らは、構文木付きコーパスから単純に CFG 規則を抽出することで、CFG を自動獲得する手法を提案している [7]。この手法の問題点は、コーパスの例文数の増加に伴う規則数の増加である。コーパスにおける出現頻度の少ない規則を削除することで文法の大きさは縮小できるが、その文法を用いて構文解析を行った場合、正しい構文木を得られなくなる可能性がある。

一方、LR 法において、構文木の生成を制御する方法が提案されている。LR 法とは構文解析アルゴリズムの一つであり、LR 文法から LR 表を作成し、それを基に解析を行う [9, 5]。Aho らは、LR 表上のアクションを削除し、コンフリクトを解消することで決定的な解析を行う手法を示している [4]。また、CFG が取り扱えるように LR 法を拡張したアルゴリズムとして、一般化 LR 法がある [10]。田中は、一般化 LR 法において、LR 表のアクションを削除することで導出される構文木を意図的に操作できることを示している [3]。一般に、LR 表のアクションを削除することによって生成されなくなる構文木の種類の数は、規則そのものを削除した場合よりも少ない。したがって、規則を削除するよりも、LR 表のアクションを削除した方が、文法が生成する構文木の種類を柔軟に制御できると考えられる。(このことは 2 節で詳述する)

本論文では、構文構造付きコーパスから LR 表のアクションの頻度を計算し、その頻度を基に LR 表を洗練し、文法が生成する解析木の数を抑制する手法を提案する。

2　提案手法

2.1　構文的曖昧性の抑制

一般に、CFG の規則を増やせば、解析可能な文を増やすことができる。しかしながら、同時に構文的曖昧性も増加する。文法が生成する構文木の中には、実際にはあり得ない構文木が存在する恐れがある。Charniak らは、構文木付きコーパスから自動的に CFG を抽出し、頻度 1 の規則を削除しても文法の被覆率[1]はそれほど下がらないと報告している [7]。また、Krotov らは、ほかの複数の規則を組み合わせることで補うことが可能で、かつコーパスにおける出現頻度の低い規則を削除することで、CFG の被覆率を下げずにサイズを小さくすることができると報告している [6]。しかし、これらの手法では、文法の被覆率や正解率について評価しているが、文法が生成する構文木の数、すなわち構文的曖昧性については評価していない。

また、規則を削除するという方法では解決できない問題がある。“ I know Jack walked in the park. ” という文を例にとって考える。図 1 のような CFG でこの文を解析した場合、図 2 のような 2 つの構文木が導出される。

これは、PP attachment の例である。(1) の構文木は、PP(“in the park”) が VP (“walked”) に係ることを示している。一方、(2) の構文木は、PP が VP(“know”) に係ることを示している。この場合、(1) の構文構造は適切であるが、(2) の

[1]受理される文の割合

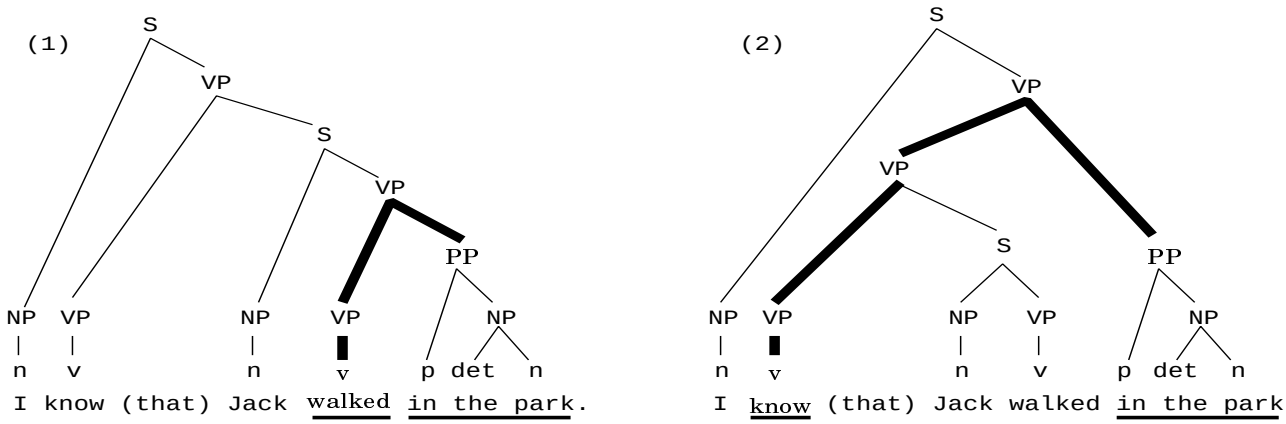

図 2: 構文木の例

1) S → NP VP
2) VP → VP NP
3) VP → VP S
4) VP → VP PP
5) VP → v
6) NP → NP PP
7) NP → n
8) NP → det n
9) PP → p NP

図 1: 文法の例

構文構造は適切でないことは明らかである。ところが、2 つの構文木の生成に使われた CFG の規則は完全に同一である。したがって、規則を削除するだけでは、(1) の構文木のみを生成するような文法を作ることはできない。

ここで、一般化 LR 法によって構文解析を行う場合、生成される構文木は使用された LR 表におけるアクションの列によって表現できることに着目する。図 1 の文法から作成される LR 表を図 3 に示す。図 3 において、“(1)” が付与されたアクションは図 2 の (1) の構文木の生成のみに使われたアクションを、“(2)” が付与されたアクションは、図 2 の (2) の構文木の生成のみに使われたアクションを、“*” が付与されたアクションは両方の構文木の生成に使われたアクションを表わしている。したがって、“ (2)” が付与されたアクションを削除すれば (2) の構文木の生成のみを妨げることができる。

文法が生成する不要な構文木の生成を抑制するには、規則を削除するよりもアクションを削除する方がより精密である。先ほどの例では、規則を削除することは、それに対応する reduce アクションをすべて削除することに相当する。例えば、規則 3) を削除することは、図 3 の LR 表のすべてのアクション “re3” が削除されることに相当する。この場合、(2) だけではなく (1) の構文木も生成できなくなる。この様な理由から、我々は LR 表のアクションを削除することにより、文法の構文的曖昧性を抑制する手法を提案する。

	det	n	v	p	$	S	NP	VP	PP
0	sh1	$sh2^*$				4	3		
1		sh5							
2	re7	re7	$re7^*$	re7	$re7^*$				
3			$sh6^*$	sh8				7	9
4					acc				
5	re8	re8	re8	re8	re8				
6	re5	$re5^*$		$re5^*$	re5				
7	re1/sh1	re1/$sh2^*$		$re1^{(2)}/sh8^*$	$re1^*$	11	10		12
8	sh1	$sh2^*$					13		
9	re6	re6	re6	re6	re6				
10	re2	re2	$sh6^*$	re2/sh8	re2			7	9
11	re3	re3		$re3^{(2)}$	$re3^{(1)}$				
12	re4	re4		re4	$re4^*$				
13	re9	re9	re9	re9/sh8	re9				9

表 1: LR 表の例

2.2 削除するアクションの決定方法

ここで問題となるのは、削除すべきアクションをどのように決定するかということである。本手法では、構造付きコーパスから LR 表上の各アクションの使用頻度を計算し、次の条件に当てはまるアクションを削除する。

- 使用頻度が閾値よりも低いアクション
- コンフリクトを起こすアクション中で、一番頻度の多いアクションに対する比が、閾値以下であるアクション

使用頻度の低いアクションは使われにくいアクションとみなすことができ、これらを削除することにより被覆率を落とすことなく LR 表の大きさを縮小することができると考えられる。また、コンフリクトを起こすアクションを削除することで、生成される構文木の数を削減することができる。

3 評価実験

3.1 評価実験

本手法の評価実験を行った。EDR コーパス [2] から白井のアルゴリズム [1] を用いて CFG を抽出した。本実験では、右辺の長さが 5 以上の規則が抽出される例文は対象外とし、それ以外の例文 66,573 文を用いて実験を行った。ランダムに選択した 63,260 文を文法の抽出と LR 表における各アクションの使用頻度の計算に、残りの 3,313 文をテストに用いた。

構文的曖昧性を抑制するために、以下の 4 つの手法を比較した。

(1) 元の文法と LR 表を使用

(2) LR 表から 0 頻度のアクションを削除

(3) 文法から頻度 19 以下の規則を削除し、LR 表を作成

- 受理率 $= \dfrac{\text{受理された文の数}}{\text{解析できた文の数}}$
- 正解木含有率 $= \dfrac{\text{統語圧縮森に正解木を含む文の数}}{\text{受理された文の数}}$
- 正解率 $= \dfrac{\text{50 位以内に正解木があった文の数}}{\text{受理された文の数}}$
- 平均構文木数 $= \dfrac{\text{受理された文の解析木数の和}}{\text{受理された文の数}}$
- $overflow =$ メモリオーバーフローにより解析に失敗した文の数

図 3: 評価尺度の定義

(4) PCFG で確率 0.00013 以下の規則を削除し、LR 表を作成

ここで、手法 (3) の頻度、手法 (4) の確率の閾値は、手法 (2) で生成された LR 表と手法 (3),(4) それぞれで生成される LR 表がほぼ同じアクション数を持つように調整した。アクション数とは、LR 表に存在するアクションの総数で、LR 表の大きさを表わす。それぞれの手法で作成された LR 表を用いて、品詞列を入力として、テスト文の構文解析を行った。結果を表 2 に示す。また、表 2 における評価尺度の定義を図 3 に示す。

図 3 において, 正解率を求める際には、PGLR モデルで各構文木の生成確率を計算し、その上位 50 位までの解析木の中で、コーパスに付与された構文木と一致するものがあるかどうかで調べた。PGLR モデルとは、LR 表の各アクションに確率を与え、その積によって構文木の生成確率を求める確率モデルである [8]。アクションの確率は、アクションの使用頻度をもとに計算される。今回の実験では、すべてのアクションに一定の頻度をあらかじめ与えることでスムージングを行う。

次に、手法 (3) における規則の出現頻度の閾値、および手法 (4) における規則の確率の閾値を変化させたときの受理率、平均解析木数、アクション数をそれぞれ表 5,6 に示す。また、手法 (2) において、使用頻度 0 のアクションを削除するだけでなく、コンフリクトを起こすアクションの使用頻度の比が閾値以上のとき、低頻度のアクションを削除した場合の実験結果をを表 7 に示す。

3.2 考察

表 2 の結果から、手法 (2),(3),(4) にそれほど大きな違いはみられなかった。各手法によって LR 表の大きさを 1/10 に縮小したが、受理率、正解率はほとんど下がらず、正解木含有率は約 7%減少した。また、平均解析木数にはほとんど変化がみられなかった。ただし、解析時にオーバーフローした文は減少し、解析可能となった文は増えている。文法を縮小させた 3 つの手法を比較した場合、規則の頻度を用いた手法が若干良い結果が得られているが、有意な差はない。

表 3 より、規則の使用頻度を基に規則を削除した場合、規則数を減少させても受理率、平均構文木数は減少していない。また、表 4 より、PCFG の確率を基に規則を削除する手法を用いた場合、

表 2: LR 表の縮小に関する実験結果

手法	受理率	overflow	正解木含有率	平均解析木数	正解率	action 数
ベース (1)	99.81%	138	98.61%	7.21×10^8	51.37%	252,136
本手法 (2)	97.51%	8	91.01%	5.25×10^8	49.67%	25,229
規則 (3)	97.10%	0	93.25%	4.20×10^8	50.23%	25,437
PCFG(4)	95.77%	0	91.93%	4.04×10^8	46.89%	25,050

表 3: 規則の使用頻度を用いた規則の削減と解析木数

閾値	0	20	50	100
アクション数	252,136	24,178	15,350	10,488
受理率	99.81%	96.83%	95.20%	91.15%
平均解析木数	6.38×10^8	4.13×10^8	3.56×10^8	3.12×10^8

表 4: PCFG を用いた規則の削減と解析木数

閾値	0	10^{-4}	10^{-3}	10^{-2}
アクション数	252,136	28,024	7,853	1,619
受理率	99.81%	96.26%	78.05%	34.44%
平均解析木数	6.38×10^8	3.64×10^8	1.84×10^8	1.69×10^7

表 5: コンフリクトの解消と解析木数

閾値	0	0.1	0.2	0.3	0.5
アクション数	25,229	24,658	22,225	21,629	20,310
受理率	97.51%	97.49%	96.86%	96.71%	96.20%
平均解析木数	5.25×10^8	2.86×10^8	1.07×10^6	6.92×10^5	4.95×10^3

規則数を減らせば受理率が低下している。しかしながら、平均解析木数は減少していないことがわかる。一方、表 5 より、コンフリクトを起こすアクションを削除することで、受理率を下げずに効果的に平均解析木を減少させることができた。

4 結論

LR 表のアクションを削除することにより、構文的曖昧性を抑制する方法を提案した。すなわち、構文構造付きコーパスからアクションの使用頻度を計算し、頻度の低いアクションを LR 表から削除することで LR 表の洗練を行った。実験の結果、受理率をほとんど変えることなく LR 表のサイズを約 90%縮小することができた。しかしながら、CFG の規則を削除する手法も同様の結果が得られ、本手法の有効性を示すには至らなかった。また、LR 表上のコンフリクトを起こすアクションの使用頻度の比を計算し、その比が十分に大きい場合に低頻度のアクションを削除することで、生成される解析木数を抑制することを試みた。コンフリクトを解消することで、効果的に解析木を抑制することができた。しかも、規則を削除する手法に比べ、受理率が約 96%と高い値を維持できた。LR 表上のアクショ

ンを削除するべきか否かの判断にPGLRモデルよって与えるアクションの確率を用いるなど工夫することで、更なる向上が望めると思われる。今後は、他の手法で自動獲得された文法や人手で作成された文法を用いた実験や他言語での実験も試みたい。

参考文献

[1] 白井 清昭 徳永 健伸 田中 穂積. 括弧付きコーパスからの日本語確率文脈自由文法の自動抽出. 自然言語処理, 4(1):125–146, 1997.

[2] 日本電子化辞書研究所. EDR 電子化辞書仕様説明書第２版. Technical Report TR–045, 1995.

[3] 田中穂積. GLR 法に基づく統計語解析過程の制御法 - LR 表工学の提案 -. 言語処理学会第 *4* 回年次大会発表論文集, pages 302–305, 1998.

[4] A. V. Aho, S. C. Johnson, and J. D. Ullman. Deterministic parsing of ambiguous grammars. *Comm. of ACM*, 18(8):441–452, 1975.

[5] A.V. Aho, S. Ravi, and J.D. Ullman. *Compilers, Principle, Techniques, and Tools*. Addision Wesely, 1986.

[6] Robert Gaizauskas Alexander Kronov, Mark Hepple and Yorick Wilks. Compacting the penn treebank grammar. *Proceeding of the COLING-ACL '98*, pages 699–703, 1998.

[7] Eugene Charniak. Tree-bank grammars. *Proccedings of the Thirteenth National Confrence on Artificial Intelligence(AAA-96)*, pages 1031–1036, 1996.

[8] K. Inui, V. Sornlertlamvanich, H. Tanaka, and T. Tokunaga. A new formalization of probabilistic glr parsing. In *International Workshop on Parsing Technologies*, pages 123–134, 1997.

[9] D.E. Knuth. On the translation of languages left to right. *Information and Control*, 8(6):607–639, 1965.

[10] Masaru Tomita. An efficient augmented-content-free parsing algorithm. *Computational Linguistics*, 13(1-2), 1987.