コーパスから学習した確率文法を用いた k best parser のあいまい性抑制と正解率のトレードオフの検討

渥美 清隆　　増山 繁
atsumi@smlab.tutkie.tut.ac.jp,　masuyama@tutkie.tut.ac.jp
豊橋技術科学大学　知識情報工学系

1 はじめに

自然言語の分析のために, 文脈自由文法に基づいた構文解析を利用することが多い. しかし文脈自由文法はあいまい性を含むことが多く, この場合1つの入力文に対して, 複数の解析結果を出力することになる. 解析結果の数があまり多くなければ, 意味解析などに選択を任せることも可能であるが, 非常に多くの解析結果を出力することになった場合には, 意味解析で選択することも難しくなる.

実際, 誤り訂正を伴なう構文解析 [2] を行なった場合, 通常の構文解析に比べ大幅に解析木の数が増えることが予想される. これは, 誤り訂正の方法が通常1つ以上あるためである. 例えば, 1文字の置換は1文字の欠落と1文字の挿入によっても表現可能である.

もし, 構文解析の段階で解析木に何らかの順位付けがされれば, 上位の解析木のみを処理し, 下位の解析木はコンピュータの性能に合わせて処理すればよい. この順位付けを行なう1つの方法として確率文脈自由文法 [1] がある. ところが, 確率文脈自由文法を利用した構文解析に関する研究 [1, 3] を見ると最も大きな確率値を持つ解析木のみに注目し, 2番目以降については考慮されていない. このため, 構文解析の正解率が 50% 程度に留まり, 確率文脈自由文法による構文解析は能力的に不十分とされてきた. そこで, 確率値の大きい順に k 個の解析木を取り出し, その中に正解が存在すれば解析は成功したとするとき, 正解率はどの程度になるのか, k の値はどのくらいが適当かなどに興味が出てくる.

本論文では, 新聞の社説のコーパスから学習した確率文脈自由文法を用いた k-best parser のプログラムを作成し, 構文解析の出力として得られる k 個の解析木にはどのような性質があるのかを調べる. また, 我々は確率文脈自由文法が持っているあいまい性の抑制能力 [6] について議論してきたが, ここでは, 著者が手作業で作成した正解の解析木が k-best parser の出力の中の何番目であるのかを調べ, あいまい性抑制と正解率とのトレードオフについて議論する.

2 確率文脈自由文法 [1]

ここでは, 確率文脈自由文法 $SCFG$(Stochastic Context-Free Grammar) と, この文法を用いた構文解析法を定義し, その性質について述べる.

2.1 $SCFG$ の定義

確率文脈自由文法は $SCFG = (N, \Sigma, S, P, p)$ の5項組で表現し, それぞれ以下のように定義する.

- N は非終端記号の集合
- Σ は終端記号の集合
- S は出発記号
- P は導出規則の集合
- p は $P \to [0,1]$ の写像

導出規則は,

$$A \underset{p(A \to \beta_i | A)}{\to} \beta$$

$$A \in N, \beta \in (N \cup \Sigma)^*$$

と表現される. 各導出規則には確率値 p が割り当てられていることを除けば, 文脈自由文法と全く同じである. $p(A \to \beta|A)$ は導出規則 $A \to \beta$ を使うときの非終端記号 A に対する条件付き確率である. また, $p(A \to \beta)$ と書けば, 導出規則全体に対する出現確率を示す.

2.2 $SCFG$ を用いた構文解析法

$SCFG$ を用いた構文解析は文脈自由文法の構文解析と何等変わることはない. 構文解析を行なった結果, 文法のあいまい性から生じる複数の解析木を得ることができるが, $SCFG$ ではこれらの各解析木に確率値による順位付けをすることができる. 各解析木 T はいくつかの導出規則 $P_1, \cdots, P_i$ から成り立っており, それぞれの導出規則には確率値 $p(P_1), \cdots, p(P_i)$ が付与されて

いる．これらの確率値から，解析木に付与される確率値 $p(T)$ を以下のように定義する．

$$p(T) = \prod_{j=1}^{i} p(P_j).$$

$SCFG$ を用いた k best parser とは，確率値によって順位付けされた解析木のうち，確率値の大きい順に k 個の解析木を出力するものである．

2.3 $SCFG$ の性質 [6]

ある文を構文解析したときに出力される解析木が $T_1, \cdots, T_n$ であったとする．解析木は確率値の大きい順に整列されているものとする．ここで，入力された文の重みを

$$Sum = \sum_{i=1}^{n} p(T_i)$$

であると定義する．また，k 番目までの重みを

$$Sum_k = \sum_{i=1}^{k} p(T_i)$$

と定義する．このとき，$Rate_k = Sum_k/Sum$ は k 番目までの解析木の重みの割合となる．

もし，この重みの割合が，正しい解析木が含まれている可能性の割合であると言えるならば，少ない解析木で重みの割合をできるだけ大きくする方が良いことになる．

$SCFG$ を用いて k-best の構文解析を行なったとき，出力される解析木は単純に順位付けされているだけではなく，多くの場合確率値が小さい解析木の数に比べて，確率値の大きい解析木の数がかなり少ないことが経験的に分かっている．これは，導出規則に付与される確率値の組み合わせに起因する．

[6] では非常に少ない解析木で重みの割合をかなり大きくすることができることを示し，また文の長さが長くなったときに指数的に増える解析木も抑制する効果があることを述べている．

3 $SCFG$ の学習

現在，日本語を対象とした適当な $SCFG$ は公開されていないため，何らかの方法で作成する必要がある．今回の実験では，新聞の社説をコーパスとして学習することによって $SCFG$ を得ることにする．

3.1 $SCFG$ の学習方法の概略

文法の学習に関しては，すでに [1, 3, 4] などで議論されている．

ここでは，特別な情報 (例えば係り受けや解析木など) が付与されていないコーパスからの学習を考える．この学習問題を，コーパス中の各文に割り当てる解析木の最適化問題として定義すると，Simulated Annealing 法や Tabu Search 法あるいは GA 法などのいわゆる局所探索法 (local search)[5] が利用できるようになる．これらの基本的なアイディアを次に示す．

1. 全ての入力文に対して解析木をランダムに割り当てる．つまり解析木の形もランダムであり，解析木の各節点に割り当てる非終端記号もランダムである．この状態を最初の暫定解とする．
2. 暫定解から導出規則を抜き出し，評価関数により評価値を計算する．
3. ある入力文に対して別のランダムな解析木を割り当て，交換する．
4. 導出規則を抜き出し，評価関数により評価値を計算する．
5. 評価値が Step 3 で交換する前と比べてより最適な方向に進むのであれば，交換した解析木を採用し，そうでなければ解析木を元に戻して新しい暫定解とする．
6. Step 3,4,5 を規定回数繰り返す．

ここで重要となるのが，「最適」をどのように定義するかである．本研究上において確率文脈自由文法の役割はあいまいさの縮小であるので，それを反映したものであることが望ましい．あいまいさは解析木のある節点において，複数の規則が適用可能である場合を指す．このことから，各規則に付与される確率値は 0 または 1 に近ければ近いほど，あいまいさを抑えることができる [3]．そこで，評価関数として n-gram 統計で利用されるエントロピーの計算を用いることにする [3]．

$$E = -\sum_{A \to \beta \in P} p(A \to \beta) \log_2 (p(A \to \beta | A)).$$

この式では E の値が小さいほどあいまいさが少ない文法であると言うことができる．つまり，E の値が最小になるようにコーパス中の各文に解析木を付与することになる．

3.2 学習結果

学習手順は, 次の通りである. 最初にコーパス中の各文をJUMANで形態素解析し, 品詞列とする. この品詞列からTabu Searchによる学習を行い, 導出規則を得ることにした.

コーパスは日経新聞CD-ROM1990年度版から社説を選び, その中から品詞列の長さが15以下のものを1000文選択した. 長さ制限は学習や構文解析を行なう時間的制約による.

学習した結果, チョムスキー標準形の文法規則を152個得ることができた. また, この文法規則のエントロピーは $E = 2.364213$ であった.

4 $SCFG$ を用いた構文解析実験

この節で述べる実験の目的は, $SCFG$ を用いた k-best parserにおいて k の値をいくつにすればよいのかを明らかにすることである. そこで, 実際に構文解析を行なって正解の解析木が何番目にあったのか, またその解析木に付与されている確率値にはどの程度の意味があるのかを考察する.

4.1 構文解析プログラムの概要

$SCFG$ を扱う構文解析は通常最も確率値の高い解析木だけを出力するように作成する [1, 3] が, 本研究では k-bestまでの解析木とそれぞれに付与されている確率値を出力するようにした. また, 全ての解析木の確率値の和と解析木の数も合せて出力するようにした. これらのデータはどれだけあいまいさの縮小に役立ったかを調べるために使う.

4.2 実験手順

入力として与える文は学習用コーパスの中から10文を選択した. 各入力文には著者が手作業で正解の構文解析木を付与した.

入力文を構文解析すると k 個の解析木を得ることができる. このとき i 番目の解析木を T_i と表す. また解析木の総数は $|T|$ と表す. ただし, 解析木の数がそもそも k 個未満である場合には, 全ての解析木を出力する.

さて構文解析により出力された k 個の解析木のうち, 何番目が正解の解析木と一致するのかを調べる. 各解析木が持つ確率値を重みとしてとらえれば, この正解の解析木が i 番目であるとすると,

$$Rate_i = \frac{\sum_{j=1}^{i} p(T_j)}{\sum_{j=1}^{|T|} p(T_j)}$$

は, 正しい解析木を得るために必要な重みの割合であるといえる.

本実験では, $k = 50$ として構文解析を行なった.

4.3 構文解析結果

各文を構文解析したときの解析木の数 ($Total$), 解析木に付与された確率値の総和 (Sum), k 番目までの解析木に付与された確率値の和 (Sum_k) と k 番目までの解析木の重みの割合 ($Rate_k$) を表1に示す.

表 1: 解析木の数と確率値の総和

No	$Total$	Sum	Sum_k	$Rate_k$
1	1232	1.90×10^{-9}	1.52×10^{-9}	0.800
2	24920	1.99×10^{-9}	2.57×10^{-10}	0.428
3	448	9.15×10^{-11}	8.49×10^{-11}	0.928
4	9	2.43×10^{-8}	2.43×10^{-8}	1.000
5	436691	3.08×10^{-15}	5.87×10^{-16}	0.190
6	240	3.73×10^{-9}	3.36×10^{-9}	0.903
7	7939	2.17×10^{-8}	1.69×10^{-8}	0.781
8	61	2.61×10^{-9}	2.61×10^{-9}	0.998
9	232433	3.26×10^{-12}	1.17×10^{-12}	0.357
10	1682	1.21×10^{-9}	1.12×10^{-9}	0.924

解析木の数が少ないほどその文はあいまい性が小さいということができる. また解析木に付与された確率値の総和が大きいほどその文は出現しやすい文であるということができる. 今回, 入力として用意した文の数が少ないので一概には言えないが, 表1からあいまい性の小さい文は確率値の総和が大きくなる傾向にあるようだ.

次に正しい解析木と一致した解析木の順位が何番目であったのか ($Find$), 1番目の解析木から正しい解析木までの確率値の和 (Sum_{Find}) と全て解析木の確率値の総和 (Sum) に対する Sum_{Find} の割合 ($Rate_{Find}$) を表2に示す. nothingの表示は k 番目までの解析木に正しい解析木がなかったことを示す.

表 2: 正しい解析木と一致した解析木の順位と重み

No	$Find$	Sum_{Find}	$Rate_{Find}$
1	11	9.78×10^{-10}	0.516
2	nothing	0	0
3	37	8.14×10^{-11}	0.889
4	9	2.43×10^{-8}	1.000
5	39	5.29×10^{-16}	0.172
6	nothing	0	0
7	nothing	0	0
8	9	1.90×10^{-9}	0.727
9	nothing	0	0
10	nothing	0	0

4.4 考察

正しい解析木が出力できた文の場合を見てみると, $Rate_{Find}$ の値が大きくなる傾向にある. しかし, 正しい解析木の順位は解析木の総数に対してかなり上位であることが多い. このことから, [6] で議論したことが実際の場合についても正しいことが分かる.

k 個の解析木の中に正解が含まれていない, つまり解析に失敗した文も半分ある. これらの文について k の値を 50 から 500 に変更して構文解析をしなおしても, 出力される解析木の中に正しい解析木は含まれていなかった. このことは正しい解析木が含まれる場合はその解析木の順位がかなり上位の方であることを期待してもよいことを示している. 今回の実験では k の値が 40 から 50 程度が適当であるといえる.

正解率自体が低いことに関しては, 文法を学習するときに, あいまい性を除去するための目的関数は用意したが, 制約関数については考慮していなかったためだと考えられる. 制約関数としては, X バー理論のように解析木の構築の制限や, 解析木中の後置詞の位置と役割といったものが考えられる. これらを組み込むことにより, 人が自然であると考える解析木を出力することが可能となるだろう.

5 むすび

本論文では, コーパスから学習した $SCFG$ を用いた k-best parser から得られた解析木から次のことを明らかにした.

まず, $SCFG$ の学習については, 目的関数としたエントロピーだけでは十分な文法を学習することが困難であることが分った. 学習した導出規則の中には助詞や助動詞を前置詞のような扱いにしているものなどが多数存在した. また, 複合名詞に関する文法規則の学習も十分ではないため, 予期しない解析木を多数生成する結果になってしまった.

次に, k-best parser が出力する k 個の解析木については, k の値は実際の解析木の数に比べてかなり小さい数でも十分な性能が発揮できることが分った. これは k 個の解析木の確率値の和と全体の確率値の総和の比率が 1 に近ければ, 正しい解析木が k 個の解析木の中に含まれる可能性が高く, もし含まれていない場合には, 構文解析自体失敗してい場合が多いためである. 今回学習によって得られた $SCFG$ の場合は $k = 50$ 程度でよいことも分かった. しかし実用的な観点から考えれば, $SCFG$ が十分によい文法規則, つまり正しい解析木により大きな確率値が付与されるような文法規則を用いる必要がある.

今後の課題としては, いかによい $SCFG$ を得るかということである. 現在, 構文解析済みコーパスから $SCFG$ の学習をする研究が行なわれている [4] が, このようなコーパスが今のところ十分であるとは言えない. 逆に日経新聞の CDROM のように特別な加工がされていないコーパスは多数存在するので, 両方のコーパスから $SCFG$ の学習をさせる必要がある.

参考文献

[1] 長尾 他 : 自然言語処理, 岩波講座ソフトウェア科学 15, 岩波書店(1996).

[2] 渥美, 増山 : "構文解析上の自由度をもった非文訂正法の一提案", 信学論, Vol.J76-D-I, pp.686-688, Dec. (1993).

[3] 横田 他 : "コーパスに基づく日本語文法の自動獲得", 言語処理学会, 第 2 回年次大会論文集, pp.169-172, Mar. (1996).

[4] T. Theeramunkong, 奥村 : "Learning a Grammar from a Bracketed Corpus", 情処研報, 96-NL-116, pp.85-92, Nov. (1996).

[5] 茨木 : アルゴリズムとデータ構造, 昭晃堂(1989).

[6] 渥美, 増山 : "確率文脈自由文法が持つ解析木の生成数抑制能力に関する検証", 情処研報, 96-NL-114, pp93-99, Jul. (1996).