解析失敗の発生しにくい PCFG-LA 句構造構文解析

小田 悠介　Graham Neubig　波多腰 優斗　Sakriani Sakti　戸田 智基　中村 哲
奈良先端科学技術大学院大学 情報科学研究科
{oda.yusuke.on9, neubig, hatakoshi.yuto.hq8, ssakti, tomoki, s-nakamura}@is.naist.jp

1 はじめに

確率的文脈自由文法 (Probabilistic Context Free Grammar: PCFG) に潜在変数の推定を導入した句構造解析法である PCFG-LA(PCFG-Latent Annotation)[6, 14] は，通常の PCFG と比較して構文構造の推定を高精度に行うことができる．しかし解析アルゴリズムの性質上，文によっては解析に失敗してしまい，構文木を生成することができない場合がある．

本稿ではこの欠点を克服するために，通常の PCFG-LA 解析法に対して 3 つの改良を加える．具体的には，段階的な解析法が失敗した際の途中結果の出力，未知の構造に対する頑健性向上に向けた単語生成確率の補間，アンダーフローを予防する確率のスケーリングを提案する．

内部評価実験として，既存の PCFG-LA 構文解析器に対して構文解析精度と失敗した事例数の比較を行った．この結果，提案法を加えたアルゴリズムでも既存の解析器と同等の構文解析精度を実現でき，さらに解析失敗がほとんど発生しないことが分かった．また各構文解析器の出力を機械翻訳に用いた外部評価実験では，既存の構文解析器よりも高精度な翻訳結果を生成できることが分かった．[1]

2 PCFG-LA 構文解析

2.1 構文解析法

英語の単語「a」と「the」は文法上は限定詞「DT」として分類される．これらの単語は使用される文脈が異なるため，構文解析においても異なる確率分布を持つようなモデルを使用するのが適切であると考えられる．しかし通常の PCFG では，これらを同一の文法タグ「DT」上の単語として扱うため，単語の使用される文脈に関する情報はモデルから失われてしまう．PCFG-LA ではこの現象を取り扱うために，各文法タグを文脈の情報を使用してより細かい潜在クラスに分割し，生成されたクラスが独立した確率分布を持つような構文解析モデルを構築する [6]．潜在クラスを推定する方法は様々であるが，現在最も普及している Petrov らの手法 [14] では，各文法タグの潜在クラスの推定を EM アルゴリズムで段階的に行う．これにより，元の文法タグを根とする木構造で表現されるような潜在クラスの階層関係が得られる．この手法では階層関係が深くなるごとに潜在クラス数が指数的に増加する．PCFG の探索アルゴリズムは潜在クラス数の多項式に比例する計算量であるため，最も細かい単位の潜在クラス集合を直接使用して構文解析を行うのは現実的ではない．このため，計算量の小さい荒い単位の潜在クラス集合から解析を始め，この結果を用いて枝刈りを行うことで全体的な計算量の削減を行う．このような解析手法は Coarse-To-Fine 解析と呼ばれる．

2.2 構文解析失敗の原因

PCFG-LA は通常の PCFG に比べて高い精度で文の構文構造を推定できるが，アルゴリズムの性質上，正常に構文解析を実行できる保証はなく，一定の割合で解析に失敗してしまう可能性がある．これには次のような理由が存在する．

Coarse-To-Fine 解析の枝刈り失敗 Coarse-To-Fine 解析において，特定の閾値 ϵ を用いて，構文木に特定のパスが生成される確率が ϵ を下回る場合に次回の探索候補から除外する枝刈りを行う．この際に解析に最低限必要なパスを全て除外してしまう可能性があり，次回以降の構文解析で構文木を生成できなくなってしまうことがある．

モデルと解析対象の不整合 構文の破綻した文を解析しようとする場合，モデルの元となった学習データに全く含まれないようなルールが必要となる場合があり，このような場合に構文解析が失敗してしまうことがある．この問題に対しては学習時に確率分布の平滑化を行う [14] ことでモデル自体がある程度対応することができるが，記録するべきルールの数が膨大となる．このため，学習時ではなく解析時に平滑化を行うことで，モデルに記録されていないルールを補完することが考えられる．

結合確率のアンダーフロー 構文解析結果の結合確率は，解析対象の単語数が増加するに従って指数的に減少する．これが浮動小数点数の扱える指数部の限界を超えると確率が 0 となってしまい，正しい解析結果が得られなくなる．この問題の解決に

[1] 本稿のアルゴリズムはオープンソースの句構造解析器 Ckylark として公開している．
http://odaemon.com/?page=tools_ckylark

Algorithm 1 Coarse-To-Fine 解析の中断

```
T_{-1} ← nil
S_0 ← {}                                  ▷ 探索を無視する条件の集合
for l ← 0 .. L do                         ▷ L 段の Coarse-To-Fine
    T_l, S_{l+1} ← parse_and_prune(w, l, S_l)
    if T_l = nil then                     ▷ 解析に失敗
        return T_{l-1}                    ▷ 前回の解析結果を返す
    end if
end for
return T_L                                ▷ 最終段の解析に成功
```

は計算に対数確率を使用し，確率の積を和に，和を logsumexp に置き換える方法がある．しかし logsumexp は積や和に比べて計算コストの大きい処理であり，頻繁に使用すると実行速度の低下を招く．

枝刈りの失敗は PCFG-LA に特有の問題であり，これ以外の 2 点は PCFG に基づく手法に普遍的な問題である．これらの問題を解決するために，基本となる PCFG-LA 解析法に加えて，次節に示す修正を導入する．

3 構文解析モデルの修正

3.1 Coarse-To-Fine 解析の中断

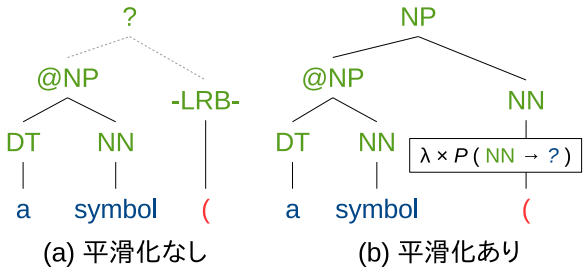
Coarse-To-Fine 解析は最終段の構文解析結果を最終的な出力とするが，途中の段階でも構文木を生成することが可能である．このため，ある段階で解析に失敗し，これ以降構文木を生成できないことが判明した場合，それ以前の解析結果を代わりに出力することが考えられる．具体的なアルゴリズムをアルゴリズム 1 に示す．これにより，モデルの問題で最初の段階で解析失敗する場合を除けば正しい解析結果を得ることができるようになる．

3.2 単語生成確率の補間

記号「(」は通常「)」と一組で句を形成するが，図 1(a) のように単体で出現すると適用可能なルールが存在せず，解析に失敗してしまう可能性がある．ここで (b) のように「(」をある一定の割合で未知語として扱うことにより，想定外の文脈で単語が使用された場合にも解析を行うことができるようになる．

これを実現するために，全ての単語の生成確率に対して式 (1) に示す補間を用いて僅かに未知語の確率を導入する．λ は補間係数であり，本来の単語生成確率への影響を抑えるために非常に小さい値を設定する．本稿の実験では $\lambda = 10^{-10}$ を使用した．

$$P'(X \to w) \equiv (1-\lambda)P(X \to w) + \lambda P(X \to w_{unk}) \quad (1)$$



(a) 平滑化なし　(b) 平滑化あり

図 1: 未知語確率による平滑化

3.3 確率のスケーリング

確率の計算を対数で置き換えずに長い文の解析を可能とするためには，モデルの確率を適切に操作することで見かけ上の値を 1 に近付け，アンダーフローを発生させにくくすることが考えられる．つまり，モデルの確率をそのまま使用するのではなく，式 (2)〜(4) に示すようなスケーリングを行った関数 Q を計算に使用する．Q は確率分布ではないが P の順序性を維持しており，最終的な解析結果が同等となることが保証される．

$$Q(X \to w) \equiv P'(X \to w)/s_l(w) \quad (2)$$
$$Q(X \to Y) \equiv P(X \to Y) \quad (3)$$
$$Q(X \to Y\ Z) \equiv P(X \to Y\ Z)/s_g \quad (4)$$

単語生成確率 $P(X \to w)$ に対応するスケーリング係数 $s_l(w)$ は生成単語 w により決まる定数である．本稿では式 (5) に示す単語生成確率の親文法要素に関する幾何平均を使用した．一方，二分ルール生成確率 $P(X \to Y\ Z)$ に対応するスケーリング係数 s_g はモデル全体で一意な値であり，式 (6) に示す全てのルール生成確率の親文法要素に関する幾何平均とした．これらの式に現れる確率 $P(X)$ はの計算は自明ではないが，モデル上のグラフ伝播に基づくアルゴリズムにより近似的に求めることができる [12].

$$s_l(w) \equiv \exp \sum_X P(X) \log P'(X \to w) \quad (5)$$
$$s_g \equiv \exp \sum_X P(X) H(X) \quad (6)$$
$$H(X) \equiv \sum_{Y,Z} P(X \to Y\ Z) \log P(X \to Y\ Z) \quad (7)$$

図 2 に示す単語生成確率の例では，w に対するスケーリング係数 $s_l(w)$ は 1/4 となる．二分ルール生成確率の場合，スケーリング係数は図 2 の $P(X_i \to w)$ を $\exp H(X_i)$ で置き換えたものと考えることができる．

4 評価実験

4.1 実験の概要

提案した解析法を評価するために，既存の PCFG-LA を用いた構文解析器と解析精度の比較を行った．ま

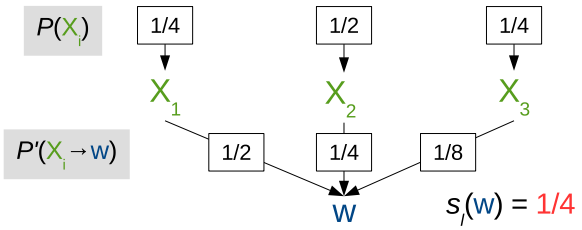


図 2: 単語生成確率の例とスケーリング係数

表 1: データセットの概要

種別	文数	単語数	
		En	Ja
WSJ-train/dev	41.5 k	990 k	—
WSJ-test	2.42 k	56.7 k	—
NTCIR-train	3.08 M	99.0 M	117 M
NTCIR-dev	822	28.6 k	33.5 k
NTCIR-test	1.38 k	44.3 k	52.5 k

た，各解析結果を Tree-To-String 翻訳 [2] への適用した場合の翻訳精度の比較を行った．Tree-To-String は原言語の構文木を用いて目的言語の文を生成する機械翻訳の手法であり，翻訳結果の精度は構文解析に依存する．

4.2 比較対象

本稿で示す手法は Ckylark の一部として実装したため，以降は提案法の名称として Ckylark を用いる．

比較対象とした解析器は Berkeley Parser[14] と Egret[2] である．Berkeley Parser は基本的な PCFG-LA にいくつかの高速化や最適化を追加した解析器である．Egret は PCFG-LA による解析に加えて構文解析森を出力可能であるため，森を用いた機械翻訳 [7] の前処理として使用されることがある．

4.3 データセット

構文解析モデルの学習には Penn Treebank[5] の WSJ データセットのうち，セクション 2 から 22 を用いた．また構文解析精度の測定には同データセットのセクション 23 を使用した．機械翻訳のデータセットとしては，NTCIR[1, 16] の特許英日翻訳タスクのデータから 80 単語以上の文を除いたものを使用した．表 1 に各データの文数及び単語数を示す．

また，各コーパスは構文解析と機械翻訳に入力する前に単語分割が行われている必要がある．英語の単語分割は Stanford Tokenizer [3] に準拠した手法で行った．日本語の単語分割は KyTea[9] で行った．

4.4 実験結果

4.4.1 構文解析精度・品詞推定精度

WSJ の学習データを使用し，Berkeley Parser の学習器により PCFG-LA の構文解析モデルを構築した．

[2]https://sites.google.com/site/zhangh1982/egret
[3]http://nlp.stanford.edu/software/tokenizer.shtml

表 2: 各ツールの構文解析精度

ツール	F1 (all)	F1 ($\|\boldsymbol{w}\| \leq 40$)
Berkeley Parser	**89.98**	**90.54**
Egret	89.05	89.70
Ckylark (10^{-5})	89.44	90.07
Ckylark (10^{-7})	89.85	90.39

表 3: 各ツールの品詞推定の正答率

ツール	Acc (all)	Acc($\|\boldsymbol{w}\| \leq 40$)
Berkeley Parser	**97.39**	97.37
Egret	97.33	97.28
Ckylark (10^{-5})	97.37	97.35
Ckylark (10^{-7})	**97.39**	**97.38**

また，Egret 及び Ckylark では Berkeley Parser の学習結果から生成されたモデルを使用した．これにより同じモデルによる解析精度を測定する．

構文解析は Intel Core i7-3770(3.40GHz, 4 コア, キャッシュ8MB)，及び 4GB の RAM を搭載した Debian 7.1 マシン上で実行した．いずれのツールも実行スレッド数は 1 である．

表 2 に，WSJ-test データに対して生成された構文木の参照木に対する Bracketing F1 値 [4] を示す．また表 3 に品詞推定の正解率を示す．テストデータの単語数によって解析結果の特徴が変化するため，全データを用いた結果と 40 語以下のデータのみを用いた結果の両方を示す．Ckylark による実験結果は，枝刈りの閾値を 10^{-5} 及び 10^{-7} の 2 種類について示す．これらの結果より，Ckylark の閾値を 10^{-7} とした結果では，構文解析と品詞推定において Berkeley Parser とほぼ同等の性能となっている．

4.4.2 実行速度

表 4 に各ツールによる WSJ-test データの解析時間を示す．枝刈りの閾値を小さくすると探索範囲が大きくなるため構文解析にかかる時間が増加するが，これは Ckylark の閾値ごとの実行時間の比較からも明らかである．またいずれの設定でも Ckylark は Egret よりも高速，かつ高精度に解析できていることが分かる．Berkeley Parser は全ての設定の中で最も高速であるが，これは Ckylark に実装されていない最適化手法が多数適用されているためであり，これらを Ckylark にも導入すれば更なる高速化と高精度化が可能であると考えられる．

4.4.3 解析失敗の頻度

表 5 に，WSJ-test 及び NTCIR-train の各データに対して構文解析の失敗が発生した数を示す．WSJ-test ではいずれの構文解析器も失敗はなかった．一方 NTCIR-train では Berkeley Parser で 0.01%，Egret で 0.5%の頻度で構文解析に失敗し，構文木を出力することができなかった．Ckylark は NTCIR-train にお

[4]http://nlp.cs.nyu.edu/evalb/

表 4: 各ツールの実行速度

ツール	実行速度
Berkeley Parser	278
Egret	3378
Ckylark (10^{-5})	923
Ckylark (10^{-7})	2157

表 5: 各ツールの解析失敗数とその割合

ツール	失敗文数			
	WSJ-test		NTCIR-train	
	(#)	(%)	(#)	(%)
Berkeley Parser	**0**	**0**	419	0.0136
Egret	**0**	**0**	17287	0.561
Ckylark (10^{-5})	**0**	**0**	**0**	**0**

いても解析失敗が発生しなかった．このことから，本稿で導入した修正が解析失敗の抑制に効果的であることが分かる．

失敗要因としては Coarse-To-Fine の枝刈り失敗によるものが最も多く，逆にモデルの不整合によるものはほとんど発生しなかった．これはコーパス自体が比較的整合性の高い文で構成されているためと考えられ，よりノイズの多い文を解析しようとする場合はモデルの不整合を抑える平滑化が有効に働くと考えられる．

4.4.4 機械翻訳への適用

NTCIR の各データに対する構文解析結果を使用して Tree-To-String 翻訳器の構築と性能測定を行った．ここで，各ツールが解析に失敗した文に関しては Stanford Parser[4] [5] による解析結果で補完し，全ての文に対する構文木が揃った状態で学習を行った．Tree-To-String 翻訳器として Travatar[8] を使用した．前処理として必要な単語アライメントには Nile[13]，目的言語の言語モデルには SRILM[15] による 5-gram モデルを使用した．NTCIR-train を翻訳ルールの抽出に使用し，生成されたルール集合は NTCIR-dev を参照訳として MERT[10] により最適化した．

表 6 に，各翻訳器の NTCIR-test に対する翻訳結果の自動評価尺度による評価値を示す．尺度としては標準的に使用される BLEU[11] と，英日翻訳のような語順の大きく異なる言語対に対して有効な RIBES[3] を使用した．この結果から，Ckylark による解析結果が BLEU に関して既存のツールよりも高い翻訳精度を達成しており，また RIBES においては Egret と同等程度の性能であることが分かる．

5 おわりに

本稿で提案した PCFG-LA 句構造解析の探索法に対する 3 種類の改良により，既存のツールで問題であった解析失敗をほとんど発生させずに句構造解析を行うことが可能となった．今後の課題としては，既存のツールで用いられる最適化法の適用やプログラムの効率化により，より速く精度の高い句構造解析器を開発することが挙げられる．

[5]Stanford Parser は PCFG-LA と異なる手法を用いており解析精度はやや劣るが，A*探索を導入することにより構文木を必ず生成することが保証される．ただし，この探索法を PCFG-LA に適用するのは容易ではない．

表 6: 各ツールによる解析結果を用いた翻訳器の評価

ツール	BLEU %	RIBES %
Berkeley Parser	39.38	78.93
Egret	39.49	**79.17**
Ckylark (10^{-5})	**39.73**	79.16

謝辞

本研究の一部は，JSPS 科研費 25730136 の助成を受け実施した．

参考文献

[1] Atsushi Fujii, Masao Utiyama, Mikio Yamamoto, Takehito Utsuro, Terumasa Ehara, Hiroshi Echizen-ya, and Sayori Shimohata. Overview of the patent translation task at the NTCIR-7 workshop. In *Proc. NTCIR-7*, 2008.

[2] Liang Huang, Kevin Knight, and Aravind Joshi. Statistical syntax-directed translation with extended domain of locality. In *Proc. AMTA*, Vol. 2006, pp. 223–226, 2006.

[3] Hideki Isozaki, Tsutomu Hirao, Kevin Duh, Katsuhito Sudoh, and Hajime Tsukada. Automatic evaluation of translation quality for distant language pairs. In *Proc. EMNLP*, pp. 944–952, 2010.

[4] Dan Klein and Christopher D Manning. Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1*, 2003.

[5] Mitchell P Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. Building a large annotated corpus of english: The Penn Treebank. *Computational linguistics*, Vol. 19, No. 2, 1993.

[6] Takuya Matsuzaki, Yusuke Miyao, and Jun'ichi Tsujii. Probabilistic CFG with latent annotations. In *Proc. ACL*, pp. 75–82, 2005.

[7] Haitao Mi, Liang Huang, and Qun Liu. Forest-based translation. In *Proc. ACL-HLT*, pp. 192–199, Columbus, Ohio, June 2008.

[8] Graham Neubig. Travatar: A forest-to-string machine translation engine based on tree transducers. In *Proc. ACL*, pp. 91–96, Sofia, Bulgaria, August 2013.

[9] Graham Neubig, Yosuke Nakata, and Shinsuke Mori. Pointwise prediction for robust, adaptable japanese morphological analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, Portland, Oregon, USA, June 2011.

[10] Franz Josef Och. Minimum error rate training in statistical machine translation. In *Proc. ACL*, pp. 160–167, Sapporo, Japan, July 2003.

[11] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: A method for automatic evaluation of machine translation. In *Proc. ACL*, pp. 311–318, 2002.

[12] Slav Petrov and Dan Klein. Improved inference for unlexicalized parsing. In *Proc. HLT-NAACL*, 2007.

[13] Jason Riesa, Ann Irvine, and Daniel Marcu. Feature-rich language-independent syntax-based alignment for statistical machine translation. In *Proc. EMNLP*, July 2011.

[14] Romain Thibaux Slav Petrov, Leon Barrett and Dan Klein. Learning accurate, compact, and interpretable tree annotation. In *Proc. COLING-ACL*, Sydney, Australia, July 2006.

[15] Andreas Stolcke, et al. SRILM-an extensible language modeling toolkit. In *Proc. InterSpeech*, 2002.

[16] Atsushi Fujii Masao Utiyama Mikio Yamamoto and Sayori Shimohata. Overview of the patent translation task at the NTCIR-8 workshop. In *Proc. NTCIR-8*, 2010.