Recurrent Neural Network CCG 構文解析器の実装

田上青空　戸次大介
お茶の水女子大学
{tagami.sora,bekki}@is.ocha.ac.jp

概要

自然言語処理において広く利用されている深層学習モデルは、文をトークン列とみなして処理を行う。しかし文には統語構造が存在しており、意味合成のような言語学的な計算を行うためには、表層的な情報を越えた統語構造を明示的にモデル化する必要があると考えられる。文の統語構造を考慮する言語モデルとして、Recurrent Neural Network Grammars (RNNGs)[1] がある。RNNGs は言語モデルと構文解析器の二つの側面を持ち、内部の文法としては CFG を用いている。本研究では RNNGs の内部文法として、CFG に代えて、より自然言語の統語構造を適切に捉える文法である CCG を用いた実装を行い、CFG によるモデルとの比較・検証を実施した。

1　はじめに

近年、BERT[2] や GPT-3[3] に代表される深層学習を用いた言語モデルの発展はめざましく、あらゆる自然言語処理タスクにおいて高精度を誇っている。これらのモデルは、膨大なデータと巨大なパラメータによって、データからある種の「傾向」を抽出することに優れている。しかし、文を文字またはサブワードからなるトークン列とみなして解析を行なっているため、文がもつ統語構造や依存関係を考慮しているわけではない。したがって、これらのモデルからは文が持つ言語学的な情報を得られず、統語解析やそれに基づく意味解析を行うことは難しい。

一方で、C&C[4]、EasyCCG[5]、depccg[6] など、組合せ範疇文法 (Combinatory Categorial Grammar; 以後 CCG) を用いた構文解析器の開発が進んでいる。組合せ範疇文法は形式統語論の理論であり、これらの構文解析器の出力として得られる CCG の統語構造は、ccg2lambda[7] にみられるように意味合成に用いることができる。しかしながら、構文解析器の性能を向上するには、人手でアノテーションした専門的でコストの高いデータが必要となるため、このアプローチには大量のデータを用意するのが難しいという欠点が存在する。また言語モデルではないため、単語の埋め込みという形での出力や単語の出現確率などの確率的な情報は得ることができない。

そのような潮流のなかで、Recurrent Neural Network Grammars (以後 RNNGs)[1] は、文の統語構造を考慮した深層学習モデルであり、言語モデルと構文解析器の 2 つの側面を併せ持つ。このモデルは、両方の側面で文脈自由文法 (Context-Free Grammar; 以後 CFG) を用いて統語情報を獲得する。RNNGs は LSTMs に比べ長期的な依存関係が捉えられることが知られている [8, 9]。また、RNNGs が学習する内容は脳波に現れる人間の言語処理の困難さの指標とも対応し、人の文処理モデルとしての妥当性も示唆されている [10]。

本研究では、RNNGs のモデルで使用する文法を CFG から CCG に変更したモデル (RNN-CCG) を構成・実装することを試みる。この変更には、CCG の統語構造を獲得し、そのまま意味合成につなげることができるという利点がある。実験を通して、文法として CFG を使用した場合とどのような違いが生じるかを検証を行い、CCG 特有の特徴を考慮するモデル構成や手法を検討する。

2　Recurrent Neural Network Grammars

RNNGs は、単語間やフレーズ間のネストした階層関係を明示的にモデル化する言語モデルおよび CFG 構文解析器である。ここでは構文解析モデルとしての振る舞いを例に挙げる。内部では Stack、Buffer の 2 つのデータ構造が用いられており、初期状態では Buffer に全単語ベクトルが格納されている。それらに対する操作は Action として以下のように定義される。

- SHIFT: 単語ベクトルを Buffer の先頭から除いて、Stack に追加。
- NT X: 非終端記号 X に対応するベクトルを

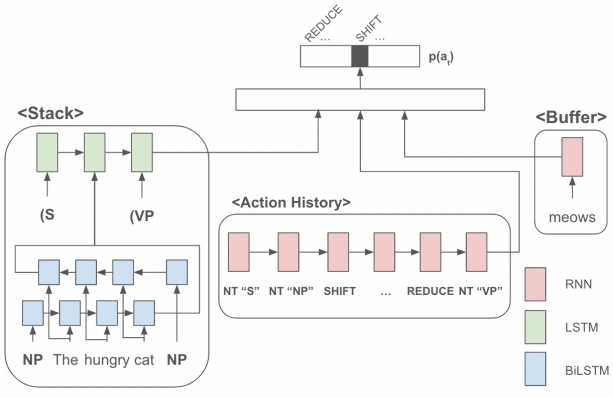

図 1　RNNGs のアーキテクチャ

Stack に追加。この時追加される非終端記号 X は、「開いた」状態とされる。

- REDUCE:　Stack から、最初に見つけた開いた非終端記号 X までを除く。それらをエンコードした新しいベクトルを生成し、新たな要素として Stack に再追加する。

各時点での Stack、Buffer、これまでの Action 履歴はそれぞれ LSTMs と RNNs を用いてエンコードされ、それらをもとに次の Action が決定される形で構文解析が行われる。Stack に対して操作が行われるたび先頭から計算し直すことは非効率であるため、stack LSTMs[11] という機構を用いて最適化されている。図 1 に示すのは、*The hungry cat meows.* という例文がすべて正しく解析されている場合において、*meows* に対応するベクトルを SHIFT することを予測する際の状態を表したものである。

RNNGs には後続する研究が行われており、いくつかの類似するモデルが存在する。Buffer を廃止し Stack のみを用いた stack-only RNNGs[12]、並列実行を実現しより大きなデータを学習することに対応した Pytorch 実装モデル [13]、RNNs ではなく Transformer を用いたモデル [14, 15] などがあげられるが、本稿では最もシンプルな形である元論文のモデル [1] を参照し、構文解析器の側面に着目して実験を行った。

3　RNN-CCG

本研究では、RNNGs の再実装である RNN-CFG と、RNNGs で使用する文法を CFG ではなく CCG に替えたモデルである RNN-CCG を実装した。CCG の統語範疇を CFG の終端記号として扱えば、CCG も句構造文法とみなすことができる。したがって、CCG を用いた場合でも、Action の判定は多値分類問題として解くことができ、CFG を使用した場合と同様に実装することができる。

3.1　組合せ範疇文法（CCG）

組合せ範疇文法（CCG）[16, 17, 18] は語彙化文法の一種である。文脈自由文法のような句構造文法では統語構造に関わる情報の大部分が書き換え規則によって記述され、辞書は比較的単純であるのに対し、語彙化文法では、それらの情報の大部分が辞書に記述され、組合せ規則は比較的単純であるという特徴がある。また、CCG による統語構造は、そのまま意味合成の計算経路を決定するという大きな利点をもつ。

文脈自由文法において、*The hungry cat meows.* を生成するには以下の規則が必要である。[1] 非終端記号がこれらの書き換え規則によって、再帰的に書き換えられることにより文が解析される。

$$\begin{aligned} S &\rightarrow NP\ VP \\ VP &\rightarrow meows \\ NP &\rightarrow The\ hungry\ cat \end{aligned}$$

一方で CCG においては、各語彙は以下のように定義される。この例の場合には、CCG の組合せ規則である逆関数適用規則を用いて二つの語彙が組み上げられ、一つの文となる。

$$\begin{aligned} The\ hungry\ cat &\vdash NP \\ meows &\vdash (S\backslash NP) \end{aligned}$$

CCG の組合せ規則の一部を以下に挙げる。

関数適用規則：$\dfrac{X/Y \quad Y}{X}\,{>}$　　$\dfrac{Y \quad X\backslash Y}{X}\,{<}$

関数合成規則：$\dfrac{X/Y \quad Y/Z}{X/Z}\,{>\mathbf{B}}$　　$\dfrac{Y\backslash Z \quad X\backslash Y}{X\backslash Z}\,{<\mathbf{B}}$

3.2　RNN-CCG vs. Transformer

RNNGs が登場した当初は BERT[2]、GPT-3[3] のような大規模言語モデルがまだ存在していなかった。しかし、その後の大規模言語モデルの成功により、言語モデルとしての RNNGs の存在意義は薄れつつあるように見受けられる。後続する研究 [14, 15] は、RNNGs の後継モデルというよりは、Transformer に構成素が持つ情報を与えてファインチューニングす

1）　RNNGs 元論文では各単語の品詞を考慮していないため、本稿でもそれに倣って表記する。

る方法論を採っている。しかし RNNGs には、以下の 2 つの側面において再評価すべき余地があると考える。

1. RNNGs も Transformer も、ある箇所に現れる言語表現の情報を、周囲の文脈を使って予測する点では共通している。しかし、周囲の文脈として参与するのは Transformer ではサブワードであるのに対し、RNNGs では構成素である。それによって、RNNGs では「構成素の埋め込み」が「周囲を予測する情報」を担うことになる。
2. また RNNGs ではある箇所に現れる言語表現というのは、サブワードだけではなく構成素である。したがって RNNGs ではサブワードの出現と構成素の出現を区別することなく予測しなければならない。それによって、RNNGs では「構成素の埋め込み」が「周囲によって予測される情報」を担うことになる。

結果的に、RNNGs における構成素は、同様の環境に現れうる構成素と近い埋め込みを持つことが期待される。これによって、RNNGs では語と句を区別することなく捉えることができる。これは Transformer にはない特徴である。

また、構文解析にせよ文生成にせよ、RNNGs は（各構成素の分散表現だけではなく）文全体の統語構造を出力する。これもまた Transformer にはない特徴である。

RNNGs のこれら二つの特徴を考慮すると、RNNGs において CFG ではなく CCG を採用することには 2 つの利点があると考えられる。以下順に述べる。

まず 1 つ目の特徴に対して、CCG は CFG と比較して（等位接続構文や長距離移動構文等を含む）より多彩な統語環境において、何が構成素であり何が構成素ではないのかを適切に切り分けることができる。もし RNNGs が採用する文法が CFG ではなく CCG であれば、RNNGs はより妥当な構成素概念を持つことになり、それに対する分散表現の割り当てもより適切なものになることが期待される。

次に 2 つ目の特徴に対して、CCG は CFG と異なり、意味合成の仕組みを誘導する。すなわち、CCG では統語構造と意味合成が準同型の関係にあるため、CCG では統語解析に成功すれば意味合成は必ず成功する、という保障が得られる。これは CFG に基づく文法にはみられない利点である。

このように、RNNGs には Transformer ベースの言語モデルにはない二つの特徴があるが、採用する文法理論を CFG から CCG に置き換えることによって、この二つの特徴を明確な利点に置き換えることにつながると考えられる。

4 実験

4.1 実験設定

RNN-CFG と RNN-CCG の実装には、Torch の Haskell インターフェースである hasktorch[2] を用いた。学習には、CFG データとして Penn Treebank[3]、CCG データとして CCGbank[4] を用いた。どちらも Wall Street Jarnal コーパスの §2-21 を学習データ、§24 を検証データ、§23 を評価データとして使用した。詳細は表 1 の通りである。

表 1 コーパス情報

	PTB		CCGbank	
	train	test	train	test
文数	39,832	2,416	39,604	2,407
トークン数	44,987	8,461	44,211	8,393
Action 数	1,182	236	810	258

実験環境には、産総研 AI 橋渡しクラウド (ABCI)[5] の rt_G.large (NVIDIA V100 for NVLink 16GiB HBM2) を 1 ノード使用した。

4.2 実験結果

それぞれのデータでの、いくつかの Action の F1 スコアと、全体の microF1 スコアを表 2,3 に示した。

表 2 CFG での実験結果

	NT S	NT NP-PRD	SHIFT	REDUCE	micro F1
F1	83.4	22.7	85.8	89.2	80.8

表 3 CCG での実験結果

	NT S[dcl]	NT S[to]\NP	SHIFT	REDUCE	micro F1
F1	91.4	25.5	81.4	99.5	82.6

一般的に構文解析器は間違った予測をするとそれ以降の予測にも影響がでるが、ここでの F1 スコアの値においては、各タイムステップ以前の予測結果は全て正解データを用いた予測の上で算出している。

4.3 考察

UNK ラベルの位置づけ 学習データに含まれてい

2） http://hasktorch.org/
3） https://catalog.ldc.upenn.edu/LDC99T42
4） https://catalog.ldc.upenn.edu/LDC2005T13
5） https://abci.ai/

ないラベルが評価時に出現した場合は誤った出力結果とした。一般に、出現頻度の低いラベルについては unknown タグ (以下 UNK) としてまとめてしまうことが多い。

この選択肢は、RNN-CFG では問題とならない。たとえば、本来 $A \leftarrow B, C, D$ という書き換え規則があるところで、C の頻度が低く UNK となった場合は、Stack には B, UNK, D が並ぶことになる。これを、A として REDUCE したところで、解析としては問題なく、得られる統語構造も正解の近似といった位置づけとなる。

しかし RNN-CCG では、たとえば本来 Stack が $NP, S\backslash NP$ となって REDUCE して S となるべきところを、仮に NP の頻度が低く UNK となった場合などでは、Stack には UNK, $S\backslash NP$ が並ぶことになる。そのような場合には組合せ規則の適用ができず、自動的に不整合な予測結果となってしまうことを考慮し、本研究では UNK を用いない設定で評価を行った。

RNN-CFG vs. RNN-CCG　各ラベルごとの F1 スコアをみると、RNN-CFG、RNN-CCG のどちらにおいても、SHIFT、REDUCE、NT S 等、出現頻度が高い Action に関しては正しく予測されていることが多い。一方で、CFG ではハイフンで区切られた小分類を含む非終端記号、CCG では /, \ を含む統語範疇を生成する Action などは、学習データにおいて出現頻度が低い傾向にあり精度の低下に繋がっていると考えられる。

また RNN-CFG と比較して RNN-CCG の microF1 が高い原因として、以下の 2 つが挙げられる。まず、RNN-CFG の方が学習データにおいて出現頻度の低い Action の割合が高いことである。前述した通り、出現頻度が低い Action は十分に学習されておらず精度に影響するが、出現頻度が 10 回以下の Action は、RNN-CFG では全 Action の 83.9%、RNN-CCG では 67.9%を占める。次に、RNN-CFG の方が Action 次元数が多いことである。Action 次元数は表 1 の通りであり、これはそのまま多値分類問題のクラス数となるため、多い方が難易度の高いタスクとなる。

POS タグについて　RNN-CFG では POS タグは考慮されていない。そのため同じ枠組みで RNN-CCG を実装した場合には、出力された Action 系列から図 1 に示すような構文木を得ることになる。この例において、*disclosed* の統語範疇は $S[pss]\backslash NP$ であることが推測されるが、CCG 統語構造の利点である意味合成の経路を得るにはこのような語の統語範疇を補完し、もう一段階解像度の高い統語情報を復元する必要がある。さらに、*were* や *'nt* に関しては統語範疇の補完自体が自明ではなく、意味合成に用いる文法としては現状の RNN-CCG は不十分であると考えられる。

```
(S[dcl]
  (S[dcl]
    (NP Terms)
    (S[dcl]\NP
      ((S[dcl]\NP) / (S[pss]\NP) were 'nt)
      disclosed
    )
  )
  .
)
```

図 1　RNN-CCG で扱われる構文木

5　おわりに

本研究では、RNNGs モデルの内部で使用する言語を CFG から CCG に置き換えたモデルである RNN-CCG を実装し、従来の RNNGs モデルである RNN-CFG と比較実験を行った。その結果、RNNGs を CCG を用いて実装した場合も、CFG を用いた実装と同様の恩恵を得られることが示された。ただし、4.3 節で述べたように、文法として CFG ではなく CCG を用いることの利点を生かすためには、更に解像度の高い出力を得る必要も生じている。

今後の課題としては 2 つ挙げられる。第一に、不整合な出力結果の除去である。RNN-CCG では RNN-CFG と異なり、出力された Action 系列が不整合である場合がある。RNN-CFG では“NT X”で生成されるどのような非終端記号列も、そのような書き換え規則が存在するとみなせば適切である。しかし RNN-CCG では、組合せ規則を適用できない出力は不整合な結果であり、意味合成にも用いることができない。また 4.3 節に述べた通り、単語の統語範疇が出力されないことも構文解析器の出力結果としては不十分であり、後続タスクに影響を及ぼす。これらの問題への対処は今後の課題としたい。

第二に、出現頻度が低い統語範疇への対応である。実験結果が示す通り、出現頻度が低く十分に学習できていない統語範疇は精度が低下する原因であるため、改善が望まれる。

3.2 節で述べた通り、RNN-CCG は RNN-CFG とは異なった利点を持つものと考えられる。そのため、構文解析器としての精度を向上させることに加え、CCG の特徴をより活かしたモデル構成を探っていきたい。

謝辞　本研究の一部は、JST CREST JPMJCR20D2の支援を受けたものである。

参考文献

[1] Chris Dyer, Adhiguna Kuncoro, Miguel Ballesteros, and Noah A. Smith. Recurrent neural network grammars. In **Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies**, June 2016.

[2] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. **CoRR**, Vol. abs/1810.04805, , 2018.

[3] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. **CoRR**, Vol. abs/2005.14165, , 2020.

[4] Stephen Clark and James R. Curran. Wide-coverage efficient statistical parsing with CCG and log-linear models. **Computational Linguistics**, Vol. 33, No. 4, pp. 493–552, 2007.

[5] Mike Lewis and Mark Steedman. A* CCG parsing with a supertag-factored model. In **Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)**, October 2014.

[6] Masashi Yoshikawa, Hiroshi Noji, and Yuji Matsumoto. A* CCG parsing with a supertag and dependency factored model. **CoRR**, Vol. abs/1704.06936, , 2017.

[7] Pascual Martínez-Gómez, Koji Mineshima, Yusuke Miyao, and Daisuke Bekki. On-demand injection of lexical knowledge for recognising textual entailment. In **Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers**, April 2017.

[8] Adhiguna Kuncoro, Chris Dyer, John Hale, Dani Yogatama, Stephen Clark, and Phil Blunsom. LSTMs can learn syntax-sensitive dependencies well, but modeling structure makes them better. In **Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)**, July 2018.

[9] Ethan Wilcox, Peng Qian, Richard Futrell, Miguel Ballesteros, and Roger Levy. Structural supervision improves learning of non-local grammatical dependencies. In **Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)**, June 2019.

[10] John Hale, Chris Dyer, Adhiguna Kuncoro, and Jonathan Brennan. Finding syntax in human encephalography with beam search. In **Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)**, July 2018.

[11] Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A. Smith. Transition-based dependency parsing with stack long short-term memory. **CoRR**, Vol. abs/1505.08075, , 2015.

[12] Adhiguna Kuncoro, Miguel Ballesteros, Lingpeng Kong, Chris Dyer, Graham Neubig, and Noah A. Smith. What do recurrent neural network grammars learn about syntax? In **Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers**, April 2017.

[13] Hiroshi Noji and Yohei Oseki. Effective batching for recurrent neural network grammars. **CoRR**, Vol. abs/2105.14822, , 2021.

[14] Laurent Sartran, Samuel Barrett, Adhiguna Kuncoro, Miloš Stanojević, Phil Blunsom, and Chris Dyer. Transformer grammars: Augmenting transformer language models with syntactic inductive biases at scale. **Transactions of the Association for Computational Linguistics**, Vol. 10, pp. 1423–1439, 2022.

[15] Peng Qian, Tahira Naseem, Roger Levy, and Ramón Fernandez Astudillo. Structural guidance for transformer language models. In **Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)**, August 2021.

[16] Mark J. Steedman. **Surface Structure and Interpretation**. The MIT Press, Cambridge, 1996.

[17] Mark J. Steedman. **The Syntactic Process (Language, Speech, and Communication)**. The MIT Press, Cambridge, 2000.

[18] 戸次大介. 日本語文法の形式理論 — 活用体系・統語範疇・意味合成 —. くろしお出版, 2010.