トランスフォーマー文法

吉田遼　　大関洋平
東京大学
{yoshiryo0617, oseki}@g.ecc.u-tokyo.ac.jp

概要

本研究では、階層構造における部分木を一つのベクトルに動的に合成しつつ、階層構造を生成する Transformer である、**トランスフォーマー文法** (Transformer Grammar, TG) を提案する。さらに、 SyntaxGym を用いて TG を評価し、RNNG 及び PLM-mask の精度を上回ることを確認した。階層構造を明示的に扱う上で、アテンション機構と、部分木の合成関数の双方に利点があることを示唆したといえる。

1　はじめに

人間らしい文法能力を持った言語モデルとは、どのようなモデルだろうか。近年主流の Transformer [1] をベースとした大規模な事前学習モデルの一つである GPT-2 XL [2] は、言語モデルの文法能力を問うベンチマークである SyntaxGym [3] において最も高い精度を達成している [4]。しかし、これらのモデルの精度は人間が文法能力を獲得するまでに得るよりも遥かに大規模な学習データによって支えられており、アーキテクチャ自体が「人間らしい」かどうかは定かではないとの指摘もある [5]。Linzen は同論文 [5] で、「人間らしい」アーキテクチャの例としては、自然言語の階層構造を明示的に扱うモデルが考えられる、と述べているが、それらのモデルの代表が再帰的ニューラルネットワーク文法 (Recurrent Neural Network Grammars, RNNG) [6] である。RNNG は、少ないデータで統制した際に他のモデルよりも高い文法能力を獲得できることが示されている他 [7, 8]、人間の脳波を高い精度でモデル化できることから、人間の文処理モデルとしての妥当性も主張されているモデルである [9]。RNNG の最大の特徴は、階層構造における部分木を一つのベクトルに動的に合成する合成関数であり、先行研究では、この合成関数なしでは RNNG は構造をうまく扱えないことが明らかになっている [10, 8]。

一方で、近年 Qian ら [11] は、Transformer ベースの階層構造を明示的に扱うモデルを提案し、大規模な事前学習なしに高い文法能力を獲得できることを示している。しかし、彼らの提案したモデルは、RNNG においては必要不可欠であった合成関数を持っておらず、階層構造を明示的に扱うことによる利点を生かしきれていない可能性がある。そこで、本研究では、階層構造における部分木を一つのベクトルに動的に合成しつつ、階層構造を生成する Transformer である、**トランスフォーマー文法** (Transformer Grammar, TG) を提案する。

2　先行研究

2.1　再帰的ニューラルネットワーク文法

RNNG[1] は、RNN ベースの単語列 X と階層構造 Y の生成モデルであり、以下の同時確率をモデル化する：

$$p(X,Y) = p(a_1, \cdots, a_n) = \sum_{t=1}^{n} p(a_t | a_{<t}) \qquad (1)$$

ここで、a_t は各時点での階層構造または単語列の生成に関する動作である。RNNG は、スタックと呼ばれるデータ構造でベクトルを保持しており、各時点における動作の確率 $p(a_t|a_{<t})$ は、スタック LSTM [12] により算出されたスタックの状態を表すベクトルに基づき算出される。動作は以下の 3 つからなる：

- NT(X)：新たに開いた句 (X を生成する。開いた句の非終端記号 (X を表すベクトルが、スタックの先頭に追加される。
- GEN(x)：単語 x を生成する。生成した単語 x を表すベクトルが、スタックの先頭に追加される。

1）　本論文では、Dyer ら [6] が元論文で提案しているモデルではなく、Kuncoro ら [10] によって提案された Stack-only RNNG を扱う。

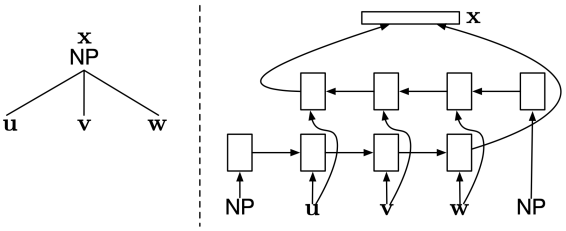

図 1　双方向 LSTM による合成関数。REDUCE の際の、閉じた句を表すベクトルの算出に用いられる。図は Dyer ら [6] より引用。

- REDUCE： 直近開かれた句を閉じる。最もスタックの先頭に近い、開いた句の非終端記号を表すベクトル及び、その句の娘にあたる全てのベクトルが、合成関数により閉じた句を表す一つのベクトルに合成される。合成関数には、双方向 LSTM が用いられる (図 1)。

前述したように、RNNG は、REDUCE の際の合成関数を取り除くと、構文解析精度 [10]、文法能力 [8]、及び脳波のモデル化精度 [9] が低下することが分かっており、合成関数が階層構造を正確に捉えるための中心的役割を担っているといえる。

2.2　Parsing as Language Modeling

Parsing as Language Modeling (PLM) は、Qian ら [11] によって提案された、Transformer ベースの単語列と階層構造の生成モデルであり、RNNG と同様に、式 1 の同時確率をモデル化する。PLM は、RNNG と異なり、NT(x)、GEN(x)、REDUCE の動作時に、それぞれ、開いた句の生成、単語の生成、REDUCE トークンの生成のみを行う。そして、通常の言語モデルが単語列 $w_1, \cdots, w_n$ を扱うのと同様に、動作列 $a_1, \cdots, a_n$ を扱い、Transformer により各時点における動作の確率 $p(a_t|a_{<t})$ を算出する。

また、Qian らは同論文で、PLM に部分木の情報を有効活用させるため、 全てのアテンションヘッドのうち 2 つに対して、1 つには直近の開かれた句の内部のみ、もう 1 つには直近開かれた句の外側のみ、に注意を注意を向けさせるという手法を提案している。Transformer 言語モデルが、訓練時にモデルが未来の情報を参照しないために用いられる、“マスク” を用いてこの手法を実現しているため、このモデルは PLM-mask と呼ばれる。

前述したように、Qian らは実験により、PLM 及び PLM-mask は、大規模な事前学習なしに高い文法能力を獲得できることを示している。また、彼らの結果からは PLM-mask が PLM の文法能力を僅かではあるが上回っていることが読み取れ、一見 PLM-mask は部分木の情報を有効活用できているようにも思われる。しかしながら、PLM-mask は、RNNG の合成関数のように部分木を表す一つのベクトルを生成しているわけではない。RNNG においては必要不可欠であった合成関数が、PLM のような Transformer ベースのモデルにとってもまた必要であるという可能性は捨て切れない。

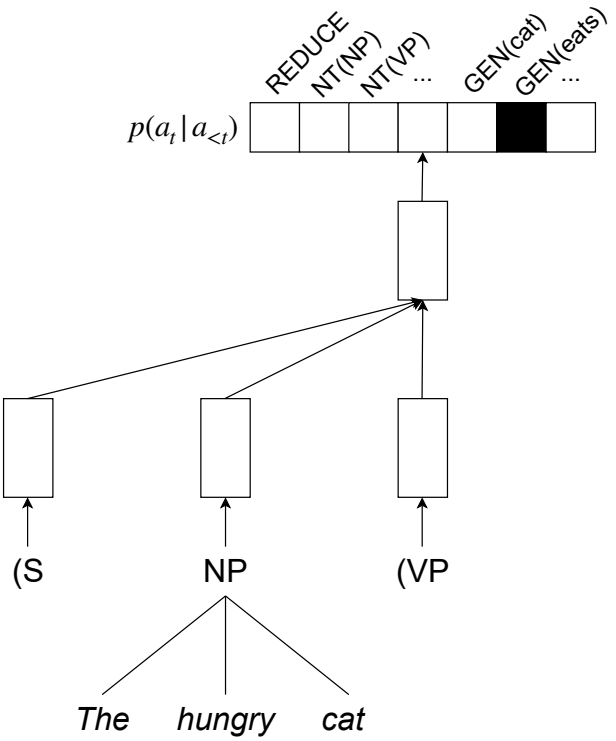

図 2　TG のアーキテクチャ。

3　トランスフォーマー文法

本研究では、階層構造における部分木を一つのベクトルに動的に合成しつつ、階層構造を生成する Transformer である、**トランスフォーマー文法** (Transformer Grammar, TG) を提案する。TG のアーキテクチャを、図 2 に示した。TG は、RNNG 及び PLM (-mask) と同様に、式 1 の同時確率をモデル化する。TG は、ベクトルを保持するデータ構造としてスタックを用い、各時点における動作の確率 $p(a_t|a_{<t})$ は、Transformer により算出された、スタックの状態を表すベクトルに基づき算出される。NT(X)、GEN(x)、REDUCE の動作時の挙動は RNNG と同じである。したがって、REDUCE の動作時には、合成関数 (図 1) により、部分木は一つのベクトルに合成される。

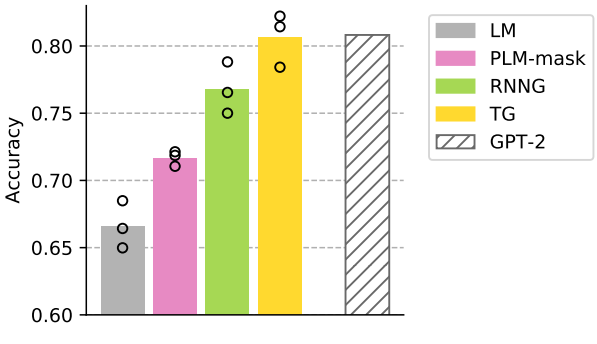

図 3 SyntaxGym 全体の結果。LM、PLM-mask、GPT-2 の結果は Qian ら [11] より引用。

TG の学習 同じく Transformer ベースのモデルである PLM (-mask) は、動作列 $a_1, \cdots, a_n$ を通常の言語モデルのように扱う。そのため、PLM (-mask) の学習時には、通常の Transformer 言語モデルの学習と同じように、全動作を同時に処理することが可能である。一方で、TG は各時点でスタック内部のベクトルが動的に変化する。そのため、通常の Transformer 言語モデルは文生成時のみに行う、各時点における逐次処理を、TG は学習時にも行う必要がある。また、TG は各時点での動作が文ごとに異なるため、RNNG と同様にミニバッチ化が難しい。そこで本研究では、固定長テンソルのスタックによる RNNG のミニバッチ化手法 [13] に数点の変更を施し、TG に適用した。具体的には、Transformer への入力には、バッチ内で最もスタックが深い文に合わせて、全ての文について、それよりも浅い位置にあるスタック内の全てのベクトルを用いるように変更し、[2] さらに、Transformer の出力からは、各文のスタックの深さを保持するポインタ[3] 位置のベクトルを取り出すように変更を施した。しかしながら、TG では、アテンション機構によるメモリの多量の消費により、この手法を用いても RNNG ほどの大きいバッチサイズでの並列化は実現できない。これらのように、階層構造における部分木を一つのベクトルに動的に合成しつつ、階層構造を生成する Transformer の実現には、計算効率の犠牲が伴っている。

2) 実際には、過去に計算された key と value をスタックの形で保持しておき、バッチ内で最もスタックが浅い文に合わせて、全ての文について、それよりも浅い位置の入力には、過去に計算された key と value を再利用している。
3) 詳細は能地ら [13] を参照。

4 実験

本節では、言語モデルの文法知識を問うベンチマークである SyntaxGym [3] を用いて、SG の文法能力を評価する.

4.1 実験設定

Qian ら [11] が報告している、PLM-mask などのモデルの評価結果と公平に比較をするために、本論文における実験設定の大部分は彼らの実験設定を踏襲している。

4.1.1 学習データ

BLLIP-MD [14] を用いた。600K 文、14M トークンからなるコーパスである。モデルの学習に用いる階層構造は、Qian ら [11] が PLM (-mask) を学習する際に、SoTA の句構造構文解析器 [15] によって付与したのと同じものを用いた。[4] 各文の単語は、Huggingface Transformer [16] 実装の BPE [17] トークナイザを用いてサブワード単位に分割した。

4.1.2 モデル及びパラメータ

TG 256 次元、3 層 4 アテンションヘッドのモデルを用いた。総パラメータ数は 16.6M。

RNNG 276 次元、2 層のモデルを用いた。総パラメータ数は 16.6M。Qian ら [11] は、Hu ら [4] により学習された RNNG と PLM (-mask) を比較しているが、Hu らは RNNG の学習に単語単位で分割したコーパスを用いている一方、Qian らは PLM (-mask) の学習にサブワード単位で分割されたコーパスを用いており、実験設定の乖離が見られた。そのため本論文では、サブワード単位で分割されたコーパスを用いて、TG と同じパラメータ数の RNNG を学習した。

PLM-mask Qian らにより報告されている、768 次元、12 層 12 アテンションヘッドのモデルの結果を引用する。総パラメータ数は 117M。総パラメータ数は TG と RNNG の約 7 倍であるが、学習コーパスなどの条件は TG と RNNG と同じである。

LM ベースラインの階層構造を明示的に扱わない言語モデル (LM) である。Qian らにより報告されている、768 次元、12 層 12 アテンションヘッドのモデルの結果を引用する。

4) BLLIP コーパスには元から階層構造が付与されているが、より正確な階層構造を得るために、その終端記号の単語列のみを取り出し、このような処理をしている。

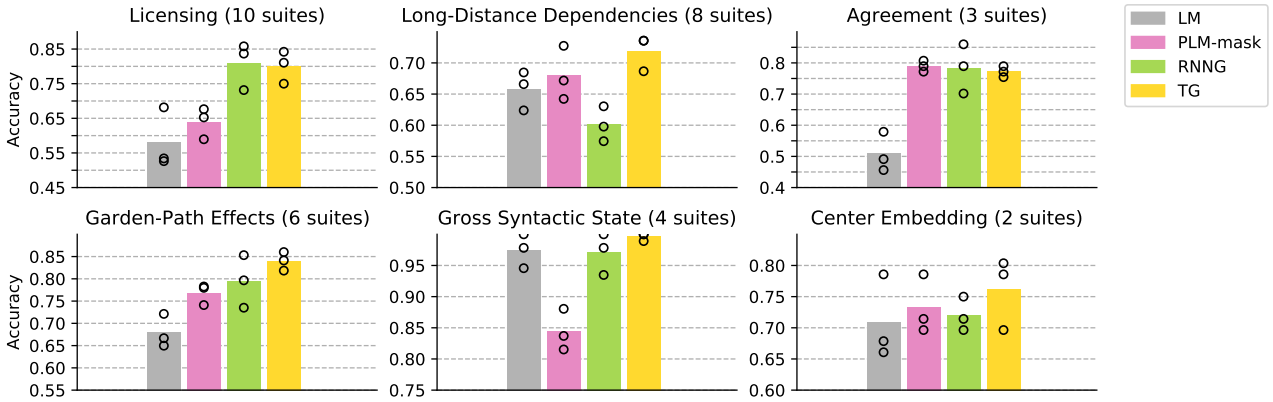

図 4　Circuit レベルの結果。LM、PLM-mask の結果は Qian ら [11] より引用。

4.1.3　評価データ

言語モデルの文法知識を問うベンチマークである、SyntaxGym [3] を用いた。幅広い文法知識をカバーしており、例えば、Agreement の circuit では、主語に動詞の数を一致させる能力が問われている。具体的には、言語モデルには以下の (1) のように動詞の数が異なる 2 文が与えられ、正しい数の動詞 ((1a) の下線部) に、誤った数の動詞 ((1b) の下線部) よりも高い確率を付与することが期待される：

(1)　a.　The author next to the senators is good.
　　b.*The author next to the senators are good.

階層構造を明示的に扱うモデルが単語に付与する確率は、先行研究 [18, 8] を踏襲して、word-synchronous beam search [19] を用いて求める。具体的には、単語列の背後に想定される階層構造のうち確率の高いものを複数保持しつつ、それらの確率を階層構造について周辺化することでモデルが単語に付与する確率を算出する。Qian ら [11] を踏襲し、動作ビーム幅 100、単語ビーム幅 10、ファストトラック幅 5 を採用した。

4.2　結果

4.2.1　SyntaxGym 全体

SyntaxGym 全体の結果を、表 3 に示した。棒グラフは seed の異なる 3 つのモデルの結果の平均値を表し、それぞれの点は各 seed の結果を表す。LM、PLM-mask、GPT-2 の結果は Qian らの論文 [11] より引用したものである。はじめに、本論文の提案モデルである TG を含む全てのモデルがベースラインの LM を上回っており、階層構造を明示的にモデル化することの利点があることが確認できる。次に、TG は、RNNG の精度を上回っている。このことから、合成関数により部分木を一つのベクトルに合成するモデルにおいても、アテンション機構により過去の全ての情報を直接参照できることの利点があることが分かる。次に、TG は、PLM-mask の精度を上回っている。このことから、Transformer ベースのモデルが階層構造を扱う際にも、合成関数によって部分木を一つのベクトルに動的に合成することの利点があることが分かる。最後に、Qian らの論文 [11] より引用した事前学習済み GPT-2 の結果 (80.8%) と比較すると、BLLIP-MD で学習したモデルのうち、唯一 TG のみが匹敵する精度 (80.7%) を達成している。

4.2.2　Circuit レベル

SyntaxGym の各 circuit レベルの結果を、表 4 に示した。TG は、Agreement を除くほぼ全ての circuit において、RNNG 及び PLM-mask の結果を上回っており、アテンション機構と、部分木の合成関数は、幅広い文法知識の獲得に有用であることが分かる。

5　おわりに

本研究では、階層構造における部分木を一つのベクトルに動的に合成しつつ、階層構造を生成する Transformer である、**トランスフォーマー文法** (Transformer Grammar, TG) を提案した。そして、SyntaxGym を用いて TG を評価し、RNNG 及び PLM-mask の精度を上回ることを確認した。階層構造を明示的に扱う上で、アテンション機構と、部分木の合成関数の双方に利点があることを示唆したといえる。

謝辞

本研究は、JST さきがけ JPMJPR21C2 の支援を受けたものです。

参考文献

[1] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention Is All You Need. In **Proceedings of NIPS**, pp. 5998–6008, 2017.

[2] Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners, 2019.

[3] Jon Gauthier, Jennifer Hu, Ethan Wilcox, Peng Qian, and Roger Levy. SyntaxGym: An online platform for targeted evaluation of language models. In **Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations**, pp. 70–76, Online, July 2020. Association for Computational Linguistics.

[4] Jennifer Hu, Jon Gauthier, Peng Qian, Ethan Wilcox, and Roger Levy. A systematic assessment of syntactic generalization in neural language models. In **Proceedings of the Association of Computational Linguistics**, 2020.

[5] Tal Linzen. How can we accelerate progress towards human-like linguistic generalization? In **Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics**, pp. 5210–5217, Online, July 2020. Association for Computational Linguistics.

[6] 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**, pp. 199–209, San Diego, California, June 2016. Association for Computational Linguistics.

[7] 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)**, pp. 1426–1436, Melbourne, Australia, July 2018. Association for Computational Linguistics.

[8] 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)**, pp. 3302–3312, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

[9] 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)**, pp. 2727–2736, Melbourne, Australia, July 2018. Association for Computational Linguistics.

[10] 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**, pp. 1249–1258, Valencia, Spain, April 2017. Association for Computational Linguistics.

[11] 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)**, pp. 3735–3745, Online, August 2021. Association for Computational Linguistics.

[12] Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A. Smith. Transition-based dependency parsing with stack long short-term memory. In **Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)**, pp. 334–343, Beijing, China, July 2015. Association for Computational Linguistics.

[13] Hiroshi Noji and Yohei Oseki. Effective batching for recurrent neural network grammars. In **Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021**, pp. 4340–4352, Online, August 2021. Association for Computational Linguistics.

[14] Eugene Charniak, Don Blaheta, Niyu Ge, Keith Hall, John Hale, and Mark Johnson. Bllip 1987-89 wsj corpus release 1. **Linguistic Data Consortium, Philadelphia**, Vol. 36, , 2000.

[15] Nikita Kitaev and Dan Klein. Constituency parsing with a self-attentive encoder. In **Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)**, pp. 2676–2686, Melbourne, Australia, July 2018. Association for Computational Linguistics.

[16] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. Transformers: State-of-the-art natural language processing. In **Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations**, pp. 38–45, Online, October 2020. Association for Computational Linguistics.

[17] Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units. In **Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)**, pp. 1715–1725, Berlin, Germany, August 2016. Association for Computational Linguistics.

[18] Richard Futrell, Ethan Wilcox, Takashi Morita, Peng Qian, Miguel Ballesteros, and Roger Levy. Neural language models as psycholinguistic subjects: Representations of syntactic state. 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)**, pp. 32–42, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

[19] Mitchell Stern, Daniel Fried, and Dan Klein. Effective inference for generative neural parsing. In **Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing**, pp. 1695–1700, Copenhagen, Denmark, September 2017. Association for Computational Linguistics.