Recurrent neural network grammar の並列化

能地宏
産業技術総合研究所
人工知能研究センター
hiroshi.noji@aist.go.jp

大関洋平
東京大学
大学院総合文化研究科
oseki@g.ecc.u-tokyo.ac.jp

1 はじめに

自然言語処理のモデルは言語の統語構造を明示的に扱うべきだろうか。その必要はないだろうか。近年の大規模コーパス上での事前学習モデルの成功を見ると (Devlin et al., 2019)、特定タスクで高い精度を得るためには明示的な統語構造は不要に思える。一方、これらのモデルの脆弱性 (McCoy et al., 2019; Yanaka et al., 2020) や人間の汎化能力との解離 (Linzen, 2020) も指摘されており、人の持つ頑健な言語能力を計算機上で実現するためには、シンボリックな統語構造の操作など強力な帰納バイアスがやはり必要であるという可能性も捨てきれない。

現在 Transformer (Vaswani et al., 2017) を始めとする単純で弱い帰納バイアスのモデルがもてはやされている大きな理由は、それらの GPU 上での並列計算との親和性による高いスケーラビリティである。明示的に構造を扱う深層学習モデルも提案されているものの、計算グラフが文毎に変化するという特性から並列化が難しく、大規模なデータ上で効率的な学習が行えないことが実用上の大きな妨げとなっている。例えば Kuncoro et al. (2020) は、 Recurrent neural network grammar (RNNG; Dyer et al. (2016)) に対して BERT の訓練データの約 3% (340 万文) を使った訓練に 3 週間を要したと報告している。

本研究では、構造を扱うモデルとしてこの RNNG に着目し、このモデルのミニバッチ化がモデル構造を注意深く観察し、データ構造を工夫することで実現可能となることを示す。RNNG はこれまで DyNet (Neubig et al., 2017) による実装しか存在しなかった。DyNet は計算グラフからミニバッチ単位を自動で見つける Autobatch と呼ばれる機能を備えるが、この機能自体のオーバーヘッドが大きいために、大きいバッチサイズによる並列化の恩恵が得られにくいという問題があった。我々は提案アルゴリズムを新しく PyTorch で実装したところ、 C++ DyNet の実装と比較して大幅な速度向上を達成した。本研究で提案するアイデアは他のモデルにも適用可能なものであり、様々な構造を考慮に入れたモデルを設計する際の新しい方法論となり得る。

2 Recurrent neural network grammar

RNNG (Dyer et al., 2016) は文及び対応する句構造を同時に生成する生成モデルであり、言語モデルとして LSTM 言語モデルなど他のモデルよりも統語能力、例えば英語での離れた主語と動詞の数を一致させる能力 (Linzen et al., 2016) などに優れることが示されている他 (Hu et al., 2020)、人の文処理のモデルとしての妥当性も主張されている (Hale et al., 2018)。本研究では元論文のモデルではなく、Kuncoro et al. (2017) で提案された Stack-only RNNG を取り扱う。このモデルは各時点で LSTM の先頭の状態が次の動作の確率を定めるという点で、LSTM 言語モデルとの類似性が認められる。

RNNG は各時点で (S (NP A dog) (VP barks)) のような句構造を左から右に生成するモデルと理解される。[1] この際、データ構造として各要素がベクトルであるスタックを用い、その上での LSTM (スタック LSTM; Dyer et al. (2015)) によって状態を更新していく点に特徴がある。次の三つの動作からなる。

- OPEN(X): スタック上に非終端記号 X に対応するベクトル $\mathbf{e}_X$ を置く。新しく開いた (X の生成を意味する。
- GEN(w): スタック上に単語ベクトル $\mathbf{e}_w$ を置く。単語 w の生成を意味する。
- REDUCE: スタックが左から $[\cdots \mathbf{e}_X\ \mathbf{e}_{w_1}\ \cdots\ \mathbf{e}_{w_n}]$ であるとき、$\mathbf{e}_X\ \mathbf{e}_{w_1}\ \cdots\ \mathbf{e}_{w_n}$ を取り除き、句 X に対応する新しいベクトル $\mathbf{e}_{X'}$ で置き換える。

$$\mathbf{e}_{X'} = \textsc{Composition}(\mathbf{e}_X, \mathbf{e}_{w_1}, \cdots, \mathbf{e}_{w_n})$$

1) 生成過程に品詞を含めると (NP (DT A) (NN dog)) などとなるが、元論文のモデルに従い、本研究では品詞の生成はモデル化せず、単語を直接生成するモデルを扱う。

この関数がトップダウンの木構造生成過程において句構造の階層性を捉えるための中心的役割を果たす。本研究では元論文に従い、$\mathbf{e}_X, \mathbf{e}_{w_1}, \cdots, \mathbf{e}_{w_n}$ を入力とする双方向 LSTM の両端の出力を結合することで $\mathbf{e}_{X'}$ を得る。本動作が REDUCE であるのは、OPEN(X) で開いた句を閉じる操作に対応するためである。例えば上の簡単な文で GEN(dog) の後のスタックは $[\mathbf{e}_S\ \mathbf{e}_{NP}\ \mathbf{e}_A\ \mathbf{e}_{dog}]$ となり、ここから REDUCE により (NP A dog) に相当する句ベクトル $\mathbf{e}_{NP'}$ を計算した後、スタックは $[\mathbf{e}_S\ \mathbf{e}_{NP'}]$ となる。

これらスタック要素 $\mathbf{e}_X, \mathbf{e}_w$ をもとに、各時点で次の動作 a_i に対する確率が、単方向 LSTM によって計算される。スタック要素が $\mathbf{e}_1, \cdots, \mathbf{e}_t$ であるとき、$\mathbf{u}_i = \mathrm{MLP}(\mathrm{LSTM}(\mathbf{e}_1, \cdots, \mathbf{e}_t))$ として、

$$p(a_i | a_1, \cdots, a_{i-1}) \propto \exp(\mathbf{W} \cdot \mathbf{u}_i) \tag{1}$$

が次の動作の分布となる。LSTM は逐次先頭から実行する必要はなく、各ステップの隠れ層をスタックの形で保持しておけば、REDUCE 時には適宜 pop しつつ 1 ステップの実行を行うのみで良い。本機構がスタック LSTM と呼ばれる所以である。学習は、正解の木構造が与えられたときにそれを復元する動作確率を最大化する教師あり学習により行われる。

並列化の難しさ　単純な LSTM の学習が本モデルより効率的なのは、複数文をまとめたミニバッチの中で i 番目の単語をまとめて処理できるためである。Transformer は更に再帰性を取り除くことで、全単語を同時に処理することを可能としている。RNNG は一方、各時点 i で選択される動作が文 (木構造) 毎に異なるため、単語単位の並列化を行うことが困難となる。DyNet (Neubig et al., 2017) の Autobatch はこれを軽減するための仕組みだが、仕組み自体のオーバーヘッドが無視できず、特に大きいバッチサイズでは効果が限定的となる (第 5 節)。

3　並列化

本研究では複数文からなるミニバッチに対して、バッチ内の全文の i 番目の動作を並列に行うアルゴリズムを提案する。鍵となるのは、上で導入した OPEN、GEN、REDUCE の内部動作が、次の二つの基本動作の組み合わせで表現可能という点である。

- PUSH($\mathbf{e}_x$): $\mathbf{e}_x$ をスタック先頭に追加する。
- CLOSE(): スタック上で開いている句 (X を閉じた後 Composition() で得られる $\mathbf{e}_{X'}$ を返す。

これらを用いると、元の RNNG の動作は

- OPEN(X) ⇒ PUSH($\mathbf{e}_X$)
- GEN(w) ⇒ PUSH($\mathbf{e}_w$)
- REDUCE ⇒ PUSH(CLOSE())

と表現できる。いずれも PUSH($\mathbf{e}_x$) を含んでいる点に注目されたい。これらをバッチ内の文に渡って並列化するには、まず各文について PUSH すべき要素 $\mathbf{e}_x$ を得た後、まとめてスタックに追加すれば良い。

スタックのテンソル化　これを実現するには、スタックを表すデータ構造を工夫する必要がある。我々は B をバッチサイズとして $\mathbf{E}_i = [\mathbf{e}^1 \cdots \mathbf{e}^B]$ の要素をまとめてスタックに追加したい。これを一つのテンソル計算で実現するためには、挿入先のスタック自体がテンソルである必要がある。式で表すと、

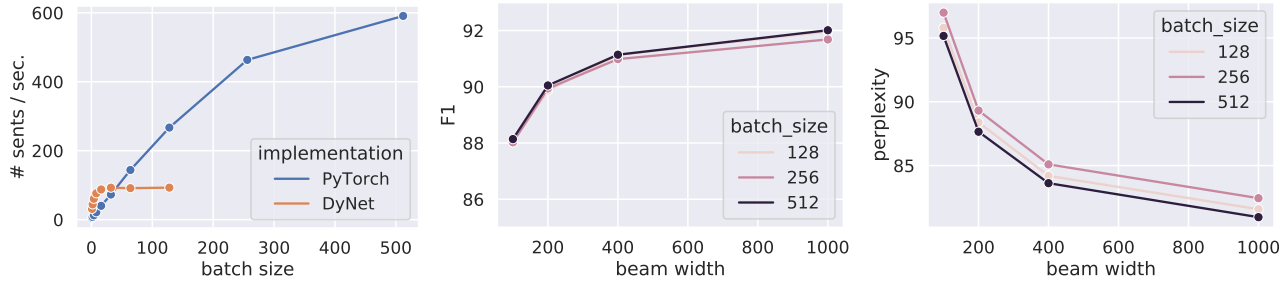
$$\mathbf{j} = \mathbf{j} + 1; \quad \mathbf{S}[(1, \mathbf{j}[1]), \cdots, (B, \mathbf{j}[B])] = \mathbf{E}_i$$

と書ける。ここで $\mathbf{j}$ は B 次元のベクトルであり、各文に対するスタックの深さのポインタを保持する。$\mathbf{S}$ がスタックを保持するテンソルであり、D をあらかじめ定まる最大のスタックの深さとして $(B, D, |\mathbf{e}_x|)$ の次元を持つ。このようにスタックを固定長のテンソルで表現することで PUSH 操作のバッチ内に渡る並列化が可能となる。[2)]

CLOSE のためのデータ構造　上記操作で PUSH を実現する場合、$\mathbf{E}_i$ を具体的にどう得るかが問題となる。次の動作を表すベクトル $\mathbf{a}_i = [a_i^1 \cdots a_i^B]$ が与えられたとき、OPEN、 GEN に関しては、 $\mathbf{a}_i$ からそれぞれの動作のみ選択するためのマスク行列を得て、単語及び非終端記号の埋め込み行列から値を取り出せば良い。REDUCE に伴う CLOSE 処理は若干複雑である。CLOSE を `for` や `if` を伴わず効率的に実現するには、更にいくつかのデータ構造を用意し、それらを操作する必要がある。

- `nt_positions`: `nt_positions[b,d]=k` で文 b のスタック中で d 番目の非終端記号が深さ k にあることを表す、(B, D) 次元の行列。
- `nt_ids`: `nt_positions[b,d]=x` で文 b のスタック中で d 番目の非終端記号が x であることを表す、(B, D) 次元の行列。

2)　D はバッチ毎にあらかじめ定めておく必要があるが、これを大きくすると GPU メモリを多量に消費するという問題も生じる。訓練時であれば、各文に対して正解の動作列をあらかじめシミュレートすることで最小値を得ておき、バッチ内の文で最大の値を用いれば良い。推論の際は深さが理論上無限に増えるが、英語の標準的な文であれば、150 単語を超える長い文であっても、深さ 70 程度で抑えられることが分かった。そのため、推論時には $D = 100$ に設定した。



(a) 訓練時の文処理速度の比較　**(b)** PTB 開発セットでの構文解析精度　**(c)** PTB 開発セットでの perplexity

図 1: Penn Treebank (PTB) での実験結果。いずれのポイントも seed の異なる 3 モデルの平均である。

- `num_nts`: 文 b のスタック中の非終端記号の数を保持する B 次元ベクトル。

例えば `nt_positions[(1,num_nts[1]),···,(B,num_nts[B])]` でバッチに渡り先頭の非終端記号のスタック深さを得ることができ、$\mathbf{a}_i$ からのマスク行列と組み合わせることで CLOSE の対象となるスタック要素をテンソル計算で取り出すことが可能となる。

既存研究との関連　スタック LSTM を固定長テンソルによって並列化する研究は本研究が初ではなく、Ding and Koehn (2019) がより単純なラベルなし依存構造解析に対してアイデアを提示している。ただし彼らのアルゴリズムは Composition が扱えず、更に二分木の場合にしか適用できない。本研究の貢献は、より複雑なスタック上の操作を要する RNNG に対しても、効率的なバッチ処理が実現可能であることを示した点にある。

4　その他の改善

Beam search の並列化　RNNG を言語モデル、もしくは逐次的な構文解析器として用いる場合、各単語毎にビーム内の足切りを行う word-synchronous beam search (Stern et al., 2017) の有効性が示されている。ただし既存の DyNet の実装は非常に遅く、小規模な実験のためであってもかなりの計算リソースを要する。我々はこの beam search も、ビーム幅の次元をスタックのテンソルに組み込むことで並列化することに成功し、大幅な速度の向上を実現した。

サブワード　未知語を含まない言語モデルとしてサブワード単位の言語モデルが標準的となっている。本研究は RNNG の予測をサブワード単位に拡張した。具体的には、予測する木構造の終端記号を分割し、 (S (NP The do## g) (VP bar## ks)) のような木構造をモデル化する。なお Kuncoro et al. (2020) ではより複雑なサブワードへの対応法が議論されているが、我々の実装ではこの単純な方法でも高い性能が維持できることが分かった。

5　DyNet 実装との比較

まず、Penn Treebank (PTB) を用いて C++ DyNet の実装との訓練時の処理時間の比較を行う。モデルサイズ、ハイパーパラメータ及びデータ前処理は Dyer et al. (2016) に従い、モデルの全ての次元を 256 に設定した。[3] 訓練データのうち 1 回しか出現しない単語は未知語に置き換え、語彙サイズは 23,794 となった。バッチサイズを $2^0, \cdots, 2^9 = 512$ まで変化させた。実験環境は 16GB の Nvidia V100 GPU × 1 である。各設定 3 回の試行を行い、平均値を報告する。

訓練を 1 エポック行った際の一秒あたりの処理文数の比較を図 1a に示す。DyNet 実装はメモリ不足のためバッチサイズ 128 までしか実行できなかったが、32 程度で速度の上限に達してしまった。一方我々の PyTorch 実装は GPU メモリの許すバッチサイズ 512 まで処理速度の向上を達成し、両者の最大の速度を比較すると約 6 倍と大幅な改善を達成した。

更に重要な点として、大きいバッチサイズのモデルは実時間でより早く収束し、また推論時により高い性能を示すことが分かった。図 1b, 1c に、これらモデルでビーム幅を変えてビームサーチを行った際の EVALB での F1 スコア (構文解析精度) と perplexity の値をまとめた。[4] F1 スコアの差は小さいが、 perplexity はバッチサイズ 512 の場合が一貫して最も低い。これより、以下の大規模実験ではバッチサイズは 512 に固定した。

3)　DyNet 実装では最適化に SGD が使われているが、我々の実装では Adam (Kingma and Ba, 2015) の方が安定して高い性能を示した。学習率は 0.001 に設定した。

4)　言語モデルとしての perplexity の計算方法などは Hale et al. (2018) に譲る。これらはアクションビーム幅 (Stern et al., 2017) であり、word-synchronous search の単語ビーム幅はその 1/10、fast track の大きさを 1/00 とした。これは Hale et al. (2018) と同じ設定であるが、彼らの数値より一貫して 0.8 ポイント程度高い F1 値を得ており、我々の実装がより高品質なモデルの学習を可能にすることを示している。

6 大規模データでの検証と議論

本研究によって大規模データ上で RNNG の学習を行う道が開けたといえる。過去の研究で RNNG と LSTM は対比されてきたが (Wilcox et al., 2019; Hu et al., 2020)、RNNG のスケーラビリティが低いために PTB など小規模のデータ上での実験に限定されていた。より大量のデータで同じ条件で訓練した際、RNNG は LSTM に対する優位性を保持するだろうか。大量のデータを与えられた LSTM は統語能力をある程度獲得し、明示的に構造を追跡する利点が相対的に薄れるという可能性も考えられる。

本節ではこの点を検証するために、英語 Wikipedia から新たにサンプルした約一億語 (PTB の約 100 倍) の訓練データで学習した LSTM と RNNG を、言語モデルとしての統語能力の観点で評価する。様々な評価方法が提案されているが、本研究では、その中で現時点で最も広い言語現象をカバーする SyntaxGym (Gauthier et al., 2020) を用いる。[5] これは様々な文法項目について、言語モデルがそれらを把握する能力がどれほどあるかを調べるためのテストセットである。例えば Agreement のテストでは “The farmer near the clerks knows/*know many people.” のように動詞の数が違う二文が与えられ、下線部の単語に対してモデルが付与する尤度を比較したとき、適格な単語 (knows) に高い尤度を付与していれば正解となる。

本実験はサブワードを用いる。Sentencepiece (Kudo and Richardson, 2018) 実装の byte-pair encoding (Sennrich et al., 2016) を利用し、語彙サイズ 30,720 とした。LSTM 言語モデルは Noji and Takamura (2020) でチューニングした文単位のモデルを用いる。LSTM、RNNG 共に、パラメータ数がこの語彙数で約 35M となるように調整した。[6] RNNG の学習データはアノテートされた句構造からなる。Wikipedia 文書に対するなるべく高精度の句構造を得る目的で、BERT に基づく Berkeley neural parser (Kitaev and Klein, 2018) を利用し、正解の構造とみなした。

全体の結果を図 2 にまとめた。特に LSTM 言語モデルで性能が 70%程度までに留まる項目 (Agreement、Center Embedding、Licensing) で 10 ポイント以上の性能向上を果たしていることから、一億語ほどの学習データを用いても、RNNG で明示的に構造を捉える利点は確実に存在することが分かる。

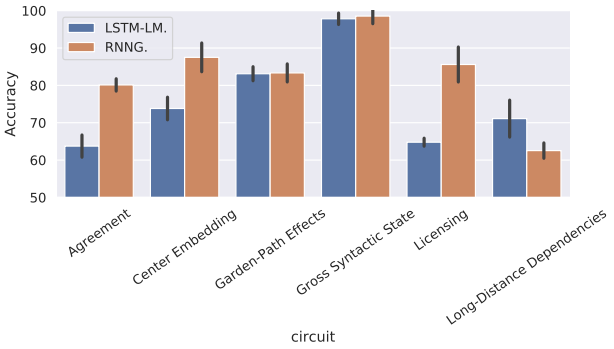


図 2: SyntaxGym での各項目 (circuit) 毎の評価結果。エラーバーは seed の異なる 3 モデルでの標準偏差を表す。

一方、Long-Distance Dependencies では精度が低下している。この項目は分裂文 (cleft) 及びフィラー・ギャップ構造の扱いをテストするものである。中身を分析したところ、特に単純な擬似分裂文に対して、LSTM は精度がほぼ 100%であるのに RNNG は 60%程度しか解けていないことが判明した。例えば次の二文で、下線部の尤度が (1a) > (1b) であることを判定する問題である。

(1) a. What he did was prepare the meal .
b. * What he ate was prepare the meal .

RNNG は構造を追跡するが故に、この現象を扱う能力が低下した可能性が考えられる。(1b) が不適であるのは、was に続く句が動詞句であることが許されるのは主節に現れる動詞が do の場合のみだからである。一方、PTB アノテーションでは両者の主節は (SBAR (WHNP What) (S (NP he) (VP did/ate))) と同じ構造で、実際モデルはこの構造を正しく認識している。これは RNNG は構造を追跡するが故に、その構造により依拠した予測をしがちであり、そのために LSTM が表層的に獲得できるヒューリスティクスの獲得に失敗する可能性があることを示唆している。

図 2 より RNNG が LSTM より統語処理能力全般に優れるのは明らかであり、このような現在苦手である現象に対処するためのモデル面での改善は今後の課題である。本研究で提案したバッチ化により、RNNG は研究のためだけの道具ではなく LSTM と比較できるモデル候補となった。今後のモデルの更なる精緻化や付与すべきアノテーションの検討をはじめ、様々な研究に繋がる礎を築いたといえる。

謝辞

本研究は JSPS 科研費 JP20K19877 及び JP19H04990 の助成を受けたものです。

5) 各項目の例は Hu et al. (2020) にまとめられている。
6) LSTM は 3 層、単語ベクトル 450 次元で、隠れ層 1150 次元である。RNNG は 2 層で、単語ベクトル、隠れ層の次元ともに 608 とした。

参考文献

J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova (2019) "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," 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. 4171–4186, Minneapolis, Minnesota: Association for Computational Linguistics, June, DOI: 10.18653/v1/N19-1423.

S. Ding and P. Koehn (2019) "Parallelizable Stack Long Short-Term Memory," in *Proceedings of the Third Workshop on Structured Prediction for NLP*, pp. 1–6, Minneapolis, Minnesota: Association for Computational Linguistics, June, DOI: 10.18653/v1/W19-1501.

C. Dyer, M. Ballesteros, W. Ling, A. Matthews, and N. A. Smith (2015) "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: Association for Computational Linguistics, July, DOI: 10.3115/v1/P15-1033.

C. Dyer, A. Kuncoro, M. Ballesteros, and N. A. Smith (2016) "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: Association for Computational Linguistics, June, DOI: 10.18653/v1/N16-1024.

J. Gauthier, J. Hu, E. Wilcox, P. Qian, and R. Levy (2020) "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: Association for Computational Linguistics, July, DOI: 10.18653/v1/2020.acl-demos.10.

J. Hale, C. Dyer, A. Kuncoro, and J. Brennan (2018) "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: Association for Computational Linguistics, July, DOI: 10.18653/v1/P18-1254.

J. Hu, J. Gauthier, P. Qian, E. Wilcox, and R. Levy (2020) "A Systematic Assessment of Syntactic Generalization in Neural Language Models," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 1725–1744, Online: Association for Computational Linguistics, July, DOI: 10.18653/v1/2020.acl-main.158.

D. P. Kingma and J. Ba (2015) "Adam: A Method for Stochastic Optimization," in Y. Bengio and Y. LeCun eds. *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, URL: http://arxiv.org/abs/1412.6980.

N. Kitaev and D. Klein (2018) "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: Association for Computational Linguistics, July, URL: http://www.aclweb.org/anthology/P18-1249.

T. Kudo and J. Richardson (2018) "SentencePiece: A simple and language independent subword tokenizer and detokenizer for Neural Text Processing," in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pp. 66–71, Brussels, Belgium: Association for Computational Linguistics, November, DOI: 10.18653/v1/D18-2012.

A. Kuncoro, M. Ballesteros, L. Kong, C. Dyer, G. Neubig, and N. A. Smith (2017) "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: Association for Computational Linguistics, April, URL: https://www.aclweb.org/anthology/E17-1117.

A. Kuncoro, L. Kong, D. Fried, D. Yogatama, L. Rimell, C. Dyer, and P. Blunsom (2020) "Syntactic Structure Distillation Pretraining for Bidirectional Encoders," *Transactions of the Association for Computational Linguistics*, Vol. 8, pp. 776-794, DOI: 10.1162/tacl_a_00345.

T. Linzen (2020) "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: Association for Computational Linguistics, July, DOI: 10.18653/v1/2020.acl-main.465.

T. Linzen, E. Dupoux, and Y. Goldberg (2016) "Assessing the Ability of LSTMs to Learn Syntax-Sensitive Dependencies," *Transactions of the Association for Computational Linguistics*, Vol. 4, pp. 521–535, DOI: 10.1162/tacl_a_00115.

T. McCoy, E. Pavlick, and T. Linzen (2019) "Right for the Wrong Reasons: Diagnosing Syntactic Heuristics in Natural Language Inference," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 3428–3448, Florence, Italy: Association for Computational Linguistics, July, DOI: 10.18653/v1/P19-1334.

G. Neubig, Y. Goldberg, and C. Dyer (2017) "On-the-fly Operation Batching in Dynamic Computation Graphs," in I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett eds. *Advances in Neural Information Processing Systems*, Vol. 30, pp. 3971–3981: Curran Associates, Inc. URL: https://proceedings.neurips.cc/paper/2017/file/c902b497eb972281fb5b4e206db38ee6-Paper.pdf.

H. Noji and H. Takamura (2020) "An Analysis of the Utility of Explicit Negative Examples to Improve the Syntactic Abilities of Neural Language Models," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 3375–3385, Online: Association for Computational Linguistics, July, DOI: 10.18653/v1/2020.acl-main.309.

R. Sennrich, B. Haddow, and A. Birch (2016) "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: Association for Computational Linguistics, August, DOI: 10.18653/v1/P16-1162.

M. Stern, D. Fried, and D. Klein (2017) "Effective Inference for Generative Neural Parsing," in *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pp. 1695–1700, Copenhagen, Denmark: Association for Computational Linguistics, September, DOI: 10.18653/v1/D17-1178.

A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin (2017) "Attention is All you Need," in I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett eds. *Advances in Neural Information Processing Systems*, Vol. 30, pp. 5998–6008: Curran Associates, Inc. URL: https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

E. Wilcox, P. Qian, R. Futrell, M. Ballesteros, and R. Levy (2019) "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: Association for Computational Linguistics, June, DOI: 10.18653/v1/N19-1334.

H. Yanaka, K. Mineshima, D. Bekki, and K. Inui (2020) "Do Neural Models Learn Systematicity of Monotonicity Inference in Natural Language?" in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 6105–6117, Online: Association for Computational Linguistics, July, DOI: 10.18653/v1/2020.acl-main.543.