BERTによる日本語構文解析の精度向上

柴田 知秀[†‡]　河原 大輔[†]　黒橋 禎夫[†‡]
[†]京都大学　[‡]科学技術振興機構 CREST
{shibata, dk, kuro}@i.kyoto-u.ac.jp

1 はじめに

深層学習による End-to-End 学習の出現により、構文解析や述語項構造解析などの基礎解析が本当に必要なのかという議論がある。例えば機械翻訳では大量の対訳データを用いて End-to-End 学習を行うことにより、構文木や述語項構造などの中間状態を必要とせずとも高精度な翻訳を実現している。

しかし、どのタスクでも大量のデータを用意できるわけではない。また、情報集約などのようにそもそも正解を定義しにくいタスクもあり、そのようなタスクでは基礎解析結果を用いることにより、精度を改善することができると考えられる。構文解析の精度は 90% を超えているが、1 文が完全に正解である精度は 50〜60%と低く、構文解析の精度を 1 ポイントでも向上させることは意義がある。また、述語項構造解析の精度は 60%程度であり、アプリケーションで利用するにはまだまだ向上させる必要がある。

構文解析や述語項構造解析、Semantic Role Labeling などの基礎解析において、ニューラルネットワークを用いることによって精度が向上してきている [4, 3, 8, 2, 9] 。精度をさらに向上させるには様々な要因が必要であるが、大きなものとして以下の 2 点が考えられる。

- 文または文章に対する強力な encoding が必要となる。一般に用いられる BiLSTM では長距離の関係を捉えることは難しい。
- 一般にはトレーニングコーパスのみからモデルを学習している。生コーパスから word2vec や ELMo[5] を用いて embedding を学習しそれをモデルトレーニングの際の初期値と用いているが、生コーパスから単語単位以上の情報を学習できることが望ましい。

これらの要件を満たすものとして、本研究では BERT [1] を利用し、日本語構文解析を対象とし精度を向上させる手法を提案する。BERT は Transformer [11] をベースとし、まず大規模な生テキストで言語モデルなどの目的関数のもとにモデルを pre-training する。そして各タスクで fine-tuning することにより、様々なタスクで SOTA を更新している。Transformer によって文または文章に対する強力な encoding を得ることができる。本研究では大規模な日本語テキストで pre-training し、構文解析タスクで fine-tuning する。

実験を行ったところ、提案手法は既存の構文解析器の精度を大幅に上回り、BiLSTM を用いた強力なベースライン手法に対して約 1.5 ポイントもの精度向上を達成した。

2 BERT

BERT (Bidirectional Encoder Representations from Transformers) [1] は大規模な生コーパスで pre-training し、各タスクで fine-tuning するという 2 ステップからなっている。

2.1 モデル

BERT は Transformer をベースにしている。Transformer は RNN や CNN を使わず、self-attention のみを使用したモデルであり、長距離の依存関係を捉えることができる。

2.2 入力表現

BERT への入力は 1 文、文ペアもしくは文書であり、いずれの場合もトークン列として表現する。各トークンは token embedding、segment embedding、position embedding の和で表現される。

各単語はサブワードに分割される。サブワードに分割された語のうち先頭ではないものには “##” を付与する[1]。segment embedding は入力が 2 文の場合に、1 文目のトークンには文 A embedding、2 文目には文 B embedding を入れる (2 文の間に [SEP] トークンをは

[1]例えば、“playing” は “play” と “##ing” に分割される。

さむ)。また、各トークンの位置は position embedding として学習される。文の先頭には [CLS] トークンを入れる。文分類問題または 2 文分類問題ではこのトークンに対応する最終層の embedding が文または 2 文の representation となる。

2.3 Pre-training

生コーパスを使って以下の 2 つのタスクで pre-training する。

2.3.1 Masked LM

近年、生コーパスを用いて言語モデルの目的関数を用いてモデルを学習する手法が多く提案されている。ELMo では双方向の浅い結合、OpenAI GPT [6] では単方向で Transformer であったが、BERT はこれらとは異なり、Masked LM をタスクと設定することにより、双方向 (文脈として対象の前も後ろも使える) で Transformer を利用する。

例えば以下の文を考える。

(1) the man went to the store

上記の文からランダムに選んだ語「went」を mask し、下記の文を作る。

(2) the man [MASK] to the store

そして、この文に Transformer を適用し、[MASK] に相当するトークンを正しく推測できるようにモデルを訓練する。

2.3.2 Next sentence prediction

質問応答やテキスト含意認識などのタスクでは 2 文間の関係を捉えることが重要となる。そこで、next sentence prediction タスクでモデルを pre-training する。50%のものは本当に存在する次の文をつなげて正例 (以下の (3)) とし[2]、残りの 50%はランダムにサンプルした文をつなげて負例 (以下の (4)) とし、これを識別する問題を解く。

(3) [CLS] the man went to the [MASK] [SEP] he bought a gallon of milk [SEP]

(4) [CLS] the man [MASK] to the store [SEP] penguin [MASK] are flight ##less birds [SEP]

[2] 2 文を [SEP] ではさむ。

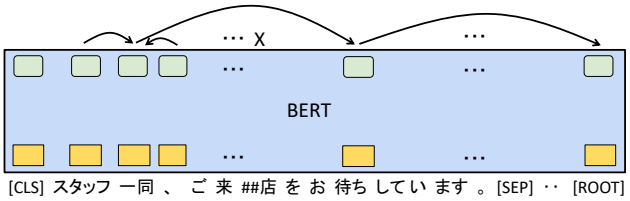

図 1: 構文解析における BERT の利用

2.4 Fine-tuning

Transformer の上に各タスクに応じた最終層を加えることによって、文ペア分類問題、1 文分類問題、質問応答 (SQuAD)、系列ラベリング問題を扱っている。例えば文ペア分類問題もしくは 1 文分類問題では、[CLS] に対応する最終層の embedding を C とすると、$W \in \mathbb{R}^{K \times H}$ (K はクラス数) が追加されるパラメータとなり、$P = \mathrm{softmax}(CW^{\mathrm{T}})$ で各クラスの確率を求める。

3 日本語構文解析における BERT の利用

3.1 日本語 Pre-training

日本語に適用するにあたり、BERT 論文で公開されている多言語モデルを利用することも考えられるが、解析単位がほぼ文字であり、単語単位のタスクに用いることには適さないと考えられる。そこで、日本語のみのテキストでモデルを pre-training する。入力文に形態素解析を適用し、さらに BPE[7] を適用することによりサブワードに分割し、それを基本単位とする[3]。

3.2 Fine-tuning

構文解析を head selection 問題 [12] と捉える。つまり、各入力トークンに対して主辞のトークンを推測する問題とする。このようなタスクは BERT 論文では扱われていないが、最終層に一層追加するだけで実現することができる。

図 1 に BERT による構文解析例を示す。入力に特別なトークンとして [ROOT] を末尾に挿入する[4]。一番上の層に head selection を入れる。入力文がトークン列 $(t_0, t_1, \cdots, t_N)$ (t_0 は [CLS] トークン) で表される時、各 $t_i (i \neq 0)$ に対して主辞 t_j を求める問題とする。

[3] 形態素解析を行わずに生文に対して sentencepiece などを用いることも考えられるが、構文解析時の解析単位が大きくずれてしまう恐れがある。

[4] この embedding は pre-training では現れないので、fine-tuning でのみ学習される。

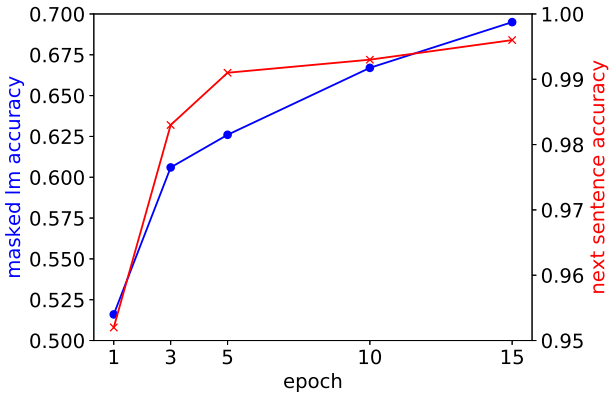

図 2: 日本語 pre-training における 2 タスクの精度

t_i の主辞が t_j である確率 $P_{head}(t_j|t_i, S)$ を以下の式で計算する。

$$P_{head}(t_j|t_i, S) = \frac{\exp(s(t_j, t_i))}{\sum_k \exp(s(t_k, t_i))} \quad (1)$$

t_i の主辞が t_j となるスコア $s(t_j, t_i)$ は以下の式で計算する。

$$s(t_j, t_i) = \boldsymbol{v}_h^{\mathrm{T}} \tanh(U_h \boldsymbol{t}_j + W_h \boldsymbol{t}_i) \quad (2)$$

ここで、$\boldsymbol{t}_i, \boldsymbol{t}_j$ はトークン t_i,t_j に対応する Transformer の最終層の embedding、$\boldsymbol{v}_h, U_h, W_h$ は fine-tuning で新たに導入されたパラメータである。サブワードに分割された単語については先頭のトークンのみ主辞を求め、それ以外のトークンは主辞を求めない。図 1 の例では単語 “来店” が “来” と “##店” に分割されており、“##店” に対しては主辞を求めない。ロス関数はクロスエントロピーロスとする。

4 実験

4.1 実験設定

pre-training に用いる生テキストとして日本語 Wikipedia 全部 (約 1,800 万文) を利用した。語彙数 (サブワードも含む) は 32,000 に設定した。pre-training には Google が公開している Tensorflow 実装[5]を用い、fine-tuning は PyTorch 実装[6]を改良した。

まず、図 2 に日本語 pre-training における 2 タスクの精度を示す。next sentence accuracy は 15 epoch でほぼ 1.0 で飽和しているのに対し、masked lm accuracy はまだ上がり続けている。なお、1GPU を用いて 1epoch に 1 日かかった。fine-tuning は元論文にならい 3 epoch 回した。

[5] https://github.com/google-research/bert

[6] https://github.com/huggingface/pytorch-pretrained-BERT

	新聞		ウェブ	
手法 ＼ 解析単位	形態素	基本句	形態素	基本句
KNP	-	90.22	-	88.44
Cabocha	-	91.12	-	89.32
J.depP	-	91.22	-	90.07
BiLSTM	95.10	91.84	93.53	90.20
BERT	**95.83**	**93.36**	**94.41**	**92.03**

表 1: 実験結果

訓練・評価には京都大学テキストコーパス (**新聞**) 約 4 万文と京都大学ウェブ文書リードコーパス (**ウェブ**) 約 1.5 万文の 2 種類のコーパスを用いた。基本句単位の係り受けを自立語 head の単語単位に変換して用いた。訓練は新聞とウェブを混合したものを用い、評価は新聞・ウェブ別々に対して行った。評価時は形態素解析 Juman++ [10] で自動解析した結果に対して構文解析を行った。また、単語単位係り受けに加えて、構文解析器 KNP を用いて基本句単位の係り受けに変換し、基本句単位でも評価を行った。

ベースライン手法として、既存の構文解析器 KNP、Cabocha (ver. 0.69)、J.DepP (2015-10-05)[7]、ならびに、encoder として 2 層 BiLSTM を用い提案手法と同様に head selection を行うものを採用した。

4.2 実験結果

表 1 に実験結果を示す。提案手法は新聞・ウェブともに既存の構文解析器ならびに強力な BiLSTM によるベースライン手法よりも高い精度を達成した。基本句単位で、BiLSTM よりも約 1.5 ポイントの精度向上を達成しており、精度 90%以上でこれだけの精度向上は大きな意義があるといえる。

表 2 にカテゴリごとの解析精度を示す。BiLSTM モデルに対して各カテゴリで全般的に精度向上を達成していることがわかる。特に、「無格」は格助詞の情報がないため BiLSTM モデルでは他のカテゴリに比べると精度が低くなっているが、BERT モデルでは大幅に精度が改善している。これは Transformer によって係り受け元と係り受け先の語の関係が捉えられていることを示唆している。

表 3 に様々な実験設定での解析精度を示す[8]。表より、pre-training の epoch 数を増やすほどよい、コーパスサイズは大きい方がよい、層数は多い方がよいことがわかる。したがって、これらの値を増やすとさらなる精度向上が達成される可能性が高いと思われる。

[7] Cabocha と J.DepP は公平な評価のために KNP により基本句単位した上で係り受け解析を行った。

[8] pre-training に長い時間を要するため、コーパスサイズならびに層数をかえた時の実験を epoch 数 10 で比較している。

	新聞				ウェブ			
	BiLSTM	BERT	Δ	数	BiLSTM	BERT	Δ	数
用言 → 用言	84.7	86.7	+2.0	2,050	86.0	89.3	+3.3	1,367
用言 → 体言	90.0	91.1	+1.1	1,367	92.2	95.5	+3.3	1,093
体言 → 体言	90.9	93.0	+2.1	3,653	90.0	91.7	+1.7	2,372
体言 → 用言	90.9	93.7	+2.8	6,300	91.0	93.4	+2.4	3,777
ガ格	91.6	94.4	+2.8	790	93.9	95.7	+1.8	393
ヲ格	97.7	98.7	+1.0	1,154	98.0	98.8	+0.8	605
ニ格	94.8	96.3	+1.5	1,114	94.9	95.8	+0.9	788
未格	88.1	90.0	+1.9	1,284	89.2	90.5	+1.3	861
無格	83.2	91.1	+7.9	328	75.5	89.4	+13.9	151

表 2: カテゴリごとの解析精度 (解析単位: 基本句)

			新聞		ウェブ	
コーパス	#e	#L	形態素	基本句	形態素	基本句
全部	15	12	**95.83**	**93.36**	**94.51**	**92.03**
全部	10	12	95.71	93.18	94.25	91.82
全部	5	12	95.46	92.83	94.22	91.58
3/4	10	12	95.63	93.10	94.22	91.56
1/2	10	12	95.53	92.81	94.07	91.17
全部	10	6	95.36	92.61	94.06	91.09

表 3: 様々な実験設定での解析精度 (#e: pre-training の epoch 数、#L: Transformer の層数)

5 おわりに

本論文では近年提案された BERT を用いて、日本語構文解析の精度を向上させる手法を提案した。Transformer により強力な encoding を得ることができ、また、日本語テキストで pre-training し構文解析タスクで fine-tuning することにより、大規模な生テキストを利用することができた。実験の結果、提案手法は既存の構文解析器の精度を大幅に上回り、BiLSTM を用いた強力なベースライン手法に対して基本句単位の係り受け解析で約 1.5 ポイントもの精度向上を達成した。

今後は本手法を述語項構造解析や共参照解析などに適用する予定である。

謝辞

本研究は科学技術振興機構 CREST「知識に基づく構造的言語処理の確立と知識インフラの構築」の支援のもとで行われた。

参考文献

[1] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[2] Timothy Dozat and Christopher D. Manning. Deep biaffine attention for neural dependency parsing. *CoRR*, Vol. abs/1611.01734, , 2016.

[3] Yuichiroh Matsubayashi and Kentaro Inui. Distance-free modeling of multi-predicate interactions in end-to-end Japanese predicate-argument structure analysis. In *COLING2018*, pp. 94–106, 2018.

[4] Hiroki Ouchi, Hiroyuki Shindo, and Yuji Matsumoto. Neural modeling of multi-predicate interactions for Japanese predicate argument structure analysis. In *ACL2017*, pp. 1591–1600, 2017.

[5] Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. In *EMNLP2018*, pp. 2227–2237, 2018.

[6] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding with unsupervised learning. Technical report, OpenAI, 2018.

[7] Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units. In *ACL2016*, pp. 1715–1725, 2016.

[8] Tomohide Shibata and Sadao Kurohashi. Entity-centric joint modeling of Japanese coreference resolution and predicate argument structure analysis. In *ACL2018*, pp. 579–589, 2018.

[9] Emma Strubell, Patrick Verga, Daniel Andor, David Weiss, and Andrew McCallum. Linguistically-informed self-attention for semantic role labeling. In *EMNLP2018*, pp. 5027–5038, 2018.

[10] Arseny Tolmachev, Daisuke Kawahara, and Sadao Kurohashi. Juman++: A morphological analysis toolkit for scriptio continua. In *EMNLP2018*, pp. 54–59, Brussels, Belgium, 2018.

[11] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NIPS2017*, pp. 5998–6008. 2017.

[12] Xingxing Zhang, Jianpeng Cheng, and Mirella Lapata. Dependency parsing as head selection. In *EACL 2017*, pp. 665–676, 2017.