大規模な自動解析データが形態素解析器をどこまで小さくできるか

Tolmachev Arseny　　河原 大輔　　黒橋 禎夫
京都大学 大学院情報学研究科
arseny@kotonoha.ws, {dk, kuro}@i.kyoto-u.ac.jp

1 はじめに

日本語のような自然な単語境界のない言語の処理には文字列を単語単位に分割する必要がある。機械翻訳などでは完全教師なしの分割方法が用いることはある[8]が、その分割結果は人間にとって不自然な分割判断を含むことが多い。従って、そのようなアプローチは全文検索のような、人間が単語単位で関わるインターフェイスでは扱えない。それらの応用には人間が定めた単語単位が必要である。日本語では単語分割と品詞推定は通常教師ありで同時に行われ、形態素解析と呼ばれる。

現代の形態素解析器は高精度である。例えば、Juman++[12]は新聞ドメインでの単語分割は99.5%、Webドメインでは98.6%程度のトークンF1値を達成している。ほとんどの形態素解析器は辞書の情報もしくは文字n-gramを素性として用いている。素性を用いる古典的なモデルのサイズは通常数百MBであり小さくない。単語や文字n-gramの埋め込みを必要とする深層モデルはそれ以上に大きくなる。このような大きなモデルはモバイルデバイスなどでの展開が困難である。

比較的シンプルなモデルであっても大規模な学習データを用いれば、高い精度を達成できることを示した研究がある[2, 3]。この考えに基づき、我々は大規模な自動タグ付きデータで簡単な深層モデルを学習し、高精度かつコンパクトな形態素解析器を実現する。従来の深層単語分割や形態素解析のモデルと異なり、我々のモデルは文字ユニグラムの埋め込みのみを用いている。辞書を明示的にモデルに埋め込まなくても、ニューラルネット(NN)は単語的な知識をコーパスから学習して

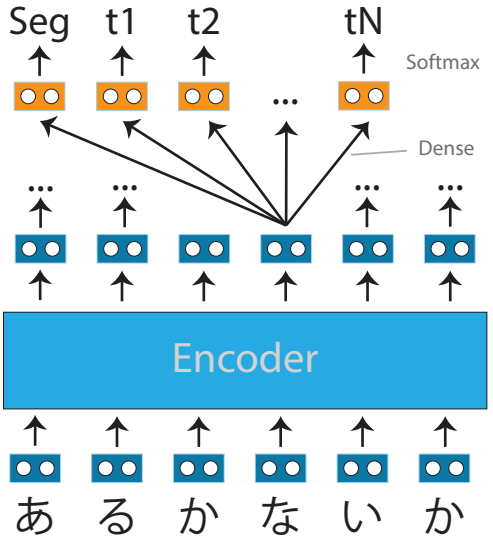

図 1: 提案する形態素解析。文字ユニグラムをエンコードしたあとに、単語分割タグ(Seg)および品詞タグ(t1,...,tN)を独立に推測する。

いることを実験で実証する。Juman++なみの精度を誇るにもかかわらず、深層モデルのサイズは20MB以下であり、Juman++より20倍以上小さい。さらに、極小のモデルでさえもJUMANなみの精度が得られる。

2 形態素解析の既存アプローチ

形態素解析は主に二つのアプローチがある。一つ目は探索型で、文字列に対して単語ラティスなどの構造を構築し、その構造で一番スコアの高いパスを探索する。二つ目は点推定型で、それぞれの文字の前もしくは後ろに対して境界があるか否かを推定する。ほとんどの日本語形態素解析器はどちらのアプローチを採るにしろ、辞書を用いて

いる。

探索型形態素解析器は実用的な解析器としてもっともよく用いられている。構造としては主にラティスが採用されている[5]。辞書の単語定義とルールベースの未定義語処理に基づき、文字列から単語候補を生成することよって、ラティスを構築し、そのラティスを通るもっともスコアの高いパスを探す。MeCab[9]はラティスを用いたもっとも有名な形態素解析器である。MeCabは辞書の情報のみを素性として用い、モデルの重みをCRFで学習している。Juman++[12]はSOTAの精度を出しており、素性ベースのモデルに加えて、NN言語モデルも利用している探索型形態素解析器である。

点推定型形態素解析器は日本語より中国語の方がよく用いられている。日本語だと、KyTea[11]は分割を二値分類問題としてとらえ、文字n-gramや辞書エントリの有無情報に基づき文字ごとに境界があるか否かの推定を行っている。このアプローチのNN拡張も存在しており[6]、やはり文字n-gram埋め込みと辞書を明示的に素性として用いている。

3 提案手法

形態素解析器の実用性は精度だけでなく、解析速度やモデルのサイズに大きく左右される。探索型手法の速度は主に重み付け関数に依存する。NNのような計算コストの高い重み付け関数を用いると、実用的な探索型形態素解析器を開発するのは困難と考えられる。さらに、辞書などの、文字を単語にまとめる情報はモデルのサイズを増大させてしまう。我々は素性として文字ユニグラムのみを使いた点推定型手法を採用し、大規模コーパスから辞書の知識をモデルに学習させる。

提案する形態素解析器の基本構造を図1に示す。まず入力文の文字をそれぞれ埋め込んで、エンコーダー部分を通して、文字の共有表現を得る。共有表現を独立にタグ表現に変換して、タグ別の埋込の行列をかけて、Softmax非線形関数の適用によってタグの正規化確率を得る。共有表現からタグ表現の変換には全結合とSeLU[7]非線形関数層を用いる。ここにおけるタグ表現とは単語分割タグと品詞タグを指す。単語分割はBIEタグ

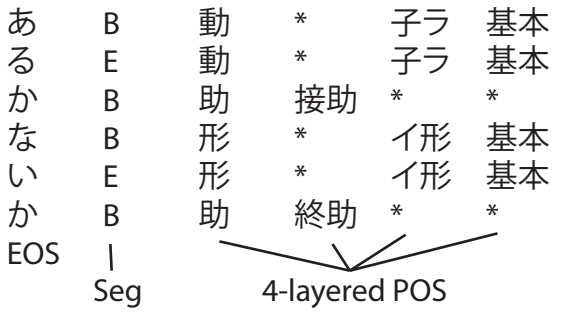

図 2: 学習データの例

セットで行い、品詞付与はJUMANの4層の品詞タグを採用する。学習データの例を図2に示す。

エンコーダーには二種類のアーキテクチャを使用する。一つ目は多層両方向LSTM(**bi-LSTM**)である。両方向のconcatした出力を次の層の入力とする。それに加えて、concatした出力にはlayer normalization [1]を適用する。二つ目はTransformer [13]に基づくself-attention network (**SAN**)である。後述するシルバーデータを学習に用いるときにはDropoutは適用しない。

学習データとして、専門家によってタグ付けされた**ゴールド**のデータだけではなく、Juman++によって自動解析された**シルバー**データも使用する。

4 実験と結果

日本語の形態素解析の実験を行った。ゴールドデータとして京都大学テキストコーパス[10](**KU**)と京都大学ウェブ文書リードコーパス[4](**Leads**)を用いた。シルバーデータとして日本語Webコーパス5億文をJuman++で解析し、使用した。

ベースラインとしてJUMAN, MeCab, KyTeaとJuman++の4つを用いる。ベースラインはKUとLeadsをマージした訓練セットで一つのモデルを学習し、評価セットでコーパス別のスコアを計算する。

採用したモデルのパラメータを表1に示す。入力の語彙サイズはWebコーパス中の全文字(18,581)とした。学習の速度はNVIDIA GeForce 1080 TiにおいてLSTMモデルとSANモデルはそれぞれ1秒間に4,500文と6,500文程度であった。シルバーデータを用いるNNモデルは1エポックのみで学習する。シルバーとゴールドデータを同時に

パラメータ	bi-LSTM	SAN
文字埋め込み次元	128	128
タグ埋め込み次元	32	32
層の数	4	6
隠れ層の次元	128×2	32
ヘッドの数	-	4
SANの内部描写次元	-	512
埋め込みパラメータ数	2.38M	2.38M
合計パラメータ数	3.88M	3.59M

表 1: 深層モデルのパラメータ

	KU (新聞)			Leads (Web)		
解析器	分割	+P1	+P2	分割	+P1	+P2
Baselines						
JUMAN	98.41	97.18	95.45	98.09	96.96	95.71
MeCab	99.10	98.56	97.59	98.25	97.60	96.22
KyTea	99.13	98.25	97.01	97.98	96.85	95.11
Juman++	**99.52**	**99.10**	97.86	98.61	98.07	96.70
bi-**L**STM						
L:G	97.46	96.56	94.78	96.33	95.43	93.46
L:S	99.33	98.90	97.59	98.68	98.16	96.71
L:SG	99.43	99.05	**98.01**	**98.71**	**98.19**	96.80
Self-**A**ttention						
A:G	98.28	97.67	95.66	97.23	96.36	93.91
A:S	99.23	98.78	97.36	98.66	98.15	96.75
A:SG	99.37	98.97	97.93	**98.70**	98.15	**96.83**

表 2: テストF1値の比較。凡例 : bi-**L**STM, S**A**N, **G**:ゴールド使用, **S**:シルバー使用

用いるときは20シルバー文に対して1ゴールド文を使用する。

実験結果を表2に示す。各解析器・コーパスに対して次の3つの値を表示している。**分割**はトークン一致で計算された単語分割のF1値である。**+P1**は、さらに品詞の大分類の一致も考慮する。**+P2**はそれに加えて品詞の細分類も考慮する。

ゴールドデータのみで学習したモデル(L:G, A:G)は過学習しがちである。すべてのF1値はJUMANよりも低い。シルバーデータのみのモデル(L:S, A:S)はエンコーダーにかかわらず、LeadsでJuman++と同等の精度を達成したが、KUでは両モデルはJuman++の精度を超えることはできなかった。シルバーデータにゴールドデータを加えると(L:SG, A:SG)、精度はさらに向上した。この設定の場合、Leadsの分割のエラーレートは、Juman++に比べてどのモデルも8%程度低下した。品詞の精度も上回っている。KUではLSTMモデルが+P2のスコアを上回ったが、分割はわずかながらJuman++の方が精度が高かった。

	サイズ, MB		
解析器	辞書	モデル	合計
JUMAN	288	1	289
MeCab	312	8	320
KyTea	-	569	569
Juman++	157	288	434
bi-LSTM	1	14	15
SAN	1	13	14

表 3: 形態素解析器のモデルサイズ

モデルのサイズも実用的な形態素解析器の重要な要素の一つである。表3に形態素解析器のモデルのサイズを示す。提案手法は高い精度を誇っているのみならず、そのモデルサイズも以前の解析器より圧倒的に小さい。例えば、Juman++より20倍以上小さくなっている。現在はモデルの重みはfloat32の形式で保存されており、重みあたり4バイトの容量を使っている。もし量子化などをすれば、さらモデルサイズを削減することが可能であろう。

提案モデルは辞書の知識を明示的に入れておらず、モデルがその知識をコーパスから学習することを期待している。しかし、辞書の知識が具体的にどこで蓄積されるか明らかではない。モデルの各部分の次元を減らしながら、どの部分が精度に関係するかを調べた。この実験はSANモデルのみで行った。学習にはシルバーデータの半分（2億5千万文）を使用した。

はじめに文字埋め込み次元、文字共有表現次元とSANの隠れ次元の変更を試みた。その結果を図3に示す。赤い横線はJUMANの精度である。埋め込み次元が大きくなればなるほど精度が向上するが、小さい埋め込み次元でもSANの隠れ次元が大きければ、高い精度を達成している。従って、埋め込みよりもエンコーダー自体が辞書の情報を学習していると思われる。

隠れ次元の影響の詳細を図4に示す。上と下の

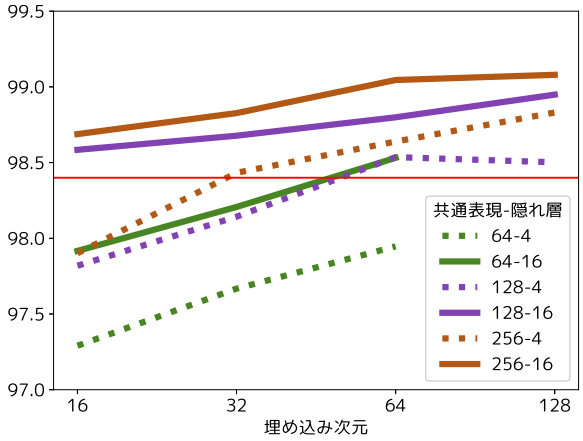

図 3: 埋め込み次元による分割のF1値の変動

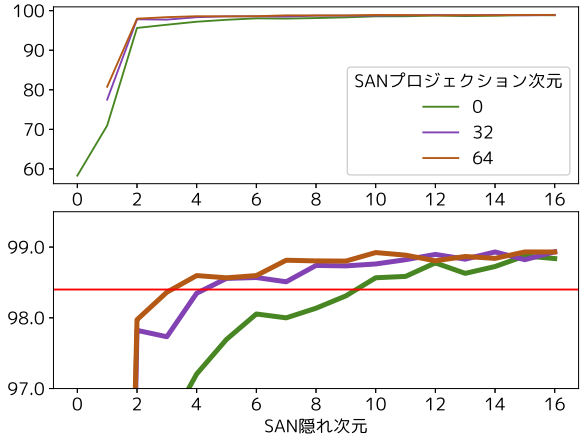

図 4: SAN隠れ次元による分割のF1値の変動

図は縮尺が違うが、同じグラフである。SANの内部プロジェクションなしのモデルは一律にプロジェクションありのモデルより精度が低い。隠れ次元の減少に従い精度が下がるにもかかわらず、非常に小さい隠れ次元(4)でもJUMANより精度が高い。従って、多少低い分割精度が許容できれば、非常にコンパクトな形態素解析器の実現が可能である。なお、非常に小さい隠れ次元の設定ではLSTMモデルの学習が収束しなかった。

5 おわりに

ニューラルネットワークによる文字ユニグラムのみを入力とし、辞書を完全に使わない形態素解析器を提案した。学習データとしてゴールドデータのみで学習しているときは精度が古典的な解析器より低いが、他の解析器で自動解析した大規模シルバーデータを学習に追加すれば、新たなSOTAを達成できた。モデルのサイズも非常にコンパクトで、辞書を使用するラティスに基づく解析器より20倍以上小さい。さらに、JUMAN程度の精度が許容されれば、1M以下の非常に小さなモデルも可能である。

参考文献

[1] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer Normalization. *arXiv:1607.06450 [cs]*, July 2016.

[2] Thorsten Brants, Ashok C. Popat, Peng Xu, Franz J. Och, and Jeffrey Dean. Large Language Models in Machine Translation. In *EMNLP*, pages 858–867, Prague, Czech Republic, June 2007.

[3] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *arXiv:1810.04805 [cs]*, 2018.

[4] Masatsugu Hangyo, Daisuke Kawahara, and Sadao Kurohashi. Building a Diverse Document Leads Corpus Annotated with Semantic Relations. In *PACLIC*, pages 535–544, Bali, Indonesia, 2012.

[5] Nobuhiro Kaji and Masaru Kitsuregawa. Efficient Word Lattice Generation for Joint Word Segmentation and POS Tagging in Japanese. In *IJCNLP*, pages 153–161, Nagoya, Japan, 2013.

[6] Yoshiaki Kitagawa and Mamoru Komachi. Long Short-Term Memory for Japanese Word Segmentation. In *PACLIC*. 2018.

[7] Günter Klambauer, Thomas Unterthiner, Andreas Mayr, and Sepp Hochreiter. Self-Normalizing Neural Networks. In *NIPS*, pages 971–980, 2017.

[8] Taku Kudo and John Richardson. SentencePiece: A simple and language independent subword tokenizer and detokenizer for Neural Text Processing. In *EMNLP*, pages 66–71, Brussels, Belgium, 2018.

[9] Taku Kudo, Kaoru Yamamoto, and Yuji Matsumoto. Applying Conditional Random Fields to Japanese Morphological Analysis. In *EMNLP*, 2004.

[10] Sadao Kurohashi and Makoto Nagao. Building A Japanese Parsed Corpus. In *Treebanks: Building and Using Parsed Corpora*, Text, Speech and Language Technology, pages 249–260. Springer Netherlands, Dordrecht, 2003.

[11] Graham Neubig, Yosuke Nakata, and Shinsuke Mori. Pointwise Prediction for Robust, Adaptable Japanese Morphological Analysis. In *ACL:HLT*, pages 529–533, Portland, Oregon, USA, 2011.

[12] Arseny Tolmachev, Daisuke Kawahara, and Sadao Kurohashi. Juman++: A Morphological Analysis Toolkit for Scriptio Continua. In *EMNLP*, pages 54–59, 2018.

[13] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is All you Need. In *NIPS*, pages 5998–6008. 2017.