深層ニューラルネットワークを利用した日本語単語分割

北川 善彬　　小町 守
首都大学東京 システムデザイン研究科
{kitagawa-yoshiaki@ed., komachi@}tmu.ac.jp

1 はじめに

日本語の処理において単語分割は機械翻訳，対話などの後段の処理のために必要となる基本的なタスクである．特に，日本語，中国語のようなスペースなどの区切り文字のない言語においては，単語分割のエラーによる後段のタスクへの影響は無視できない．単語分割のタスクは教師データを用いた系列ラベリングによる手法が主流であるが，素性を人手で作成する必要がありコストがかかる上に，素性の数が非常に大きくなることから，できたモデルは学習コーパスにオーバフィットしてしまう傾向がある．

最近の研究では，自然言語処理のタスクに対して，ニューラルネットワークのモデルの適用が盛んに研究されている．ニューラルネットワークのモデルは，構造によるハイパーパラメータのチューニングの問題を伴うが，以前のような素性エンジニアリングによる手間を軽減し，高次元でスパースな素性ではなく，低次元で密な素性による学習を実現している．中国語の単語分割においては，ニューラルネットワークを利用した単語分割が state-of-the-art を記録した [2]．Chen らの手法は系列ラベルの遷移を考慮した構造のニューラルネットワークを適用し，Long Short Term Memory により系列全体の情報を捉える構造になっている．

日本語の単語分割では，リカレントニューラルネットワーク言語モデル（以下，RNNLM）を利用した単語分割の研究 [7] があるものの，深いニューラルネットワークの構造を用いた系列ラベリングによる単語分割の研究はなされていない．

このような背景から，深層ニューラルネットワークを利用した日本語単語分割に関しての分析を行った．本研究の貢献は以下である．

- 日本語単語分割における深層ニューラルネットワークの構造による特性を調査し性能を評価した．
- ウェブ文書に対して，深層ニューラルネットワークの単語分割を適用し，問題点を考察した．

2 関連研究

日本語の形態素解析は，教師あり学習による手法が広く使われており，辞書を使ってラティスを作り，条件付き確率場（CRF） でコストを計算することにより形態素解析を行う手法 [5] と，周辺の単語や文字の情報から SVM 等の線形分類器により，単語分割をし，その後，品詞等の情報を推定することで形態素解析を行う手法 [9, 8] が存在する．

一方，中国語の単語分割において，深層ニューラルネットワークを用いた手法が盛んに研究されている．[2, 1, 10]．深層ニューラルネットワークによる手法は学習に時間がかかるが，文字 embedding を利用した線形時間で学習できる高性能な単語分割器も存在する [6]．日本語形態素解析においても，深層学習を利用した研究は RNNLM を利用した形態素解析 [7] があるが，中国語であるような，深層ニューラルネットワークを利用した単語分割の研究は存在しない．そこで，本研究では，日本語形態素解析のための深層ニューラルネットワークを利用した日本語単語分割について分析を行った．

3 日本語単語分割におけるニューラルモデル

日本語単語分割は，系列ラベリング問題として扱うのが一般的である．それぞれの文字は {B, I}，{B, I, S} ，{B, M, E, S} などのラベルがつけられる．ここで，B は Begin，I は Inside，M は Middle，E は End，S は Single を表す．本研究では上に挙げた 3 通りのラベル付けを採用する．また，本節では，日本語単語分割におけるニューラルモデルの基本的な構造に関しての説明を行う．ネットワークの全体像を図 1 に示す．

Copyright(C) 2016 The Association for Natural Language Processing.
All Rights Reserved.

3.1 フィードフォワードニューラルネットワーク

本節では，フィードフォワードニューラルネットワーク（以下，**FFNN**）について説明する．

文字数が n の文 $c_{1:n}$ が与えられたとき，ウィンドウサイズを k（k: 奇数）とすると，ある文字 c_t $(1 \leq t \leq n)$ に対する入力は，$(c_{t-\frac{k-1}{2}},...,c_t,...,c_{t+\frac{k-1}{2}})$ となる．また，文頭と文末に文頭記号，文末記号を付加する．

ニューラルネットワークを用いるための最初のステップとして，文字を実数値のベクトルにする必要がある．これを embedding と呼ぶ [3]．各文字の embedding は look up table から取り出され，それらを連結することで入力ベクトル $\boldsymbol{x} \in \mathbb{R}^{H_1}$ を得る．ここで，H_1 は (a) の入力文字列の embedding 層のサイズであり，その値は，$k \times d$ である．また，d は各文字の embedding の次元数である．

次に入力ベクトル $\boldsymbol{x_t}$ は以下の線形変換に渡され，成分ごとに *sigmoid*, *tanh* などの活性化関数 g にかけられ，$\boldsymbol{h_t}$ を得る．

$$\boldsymbol{h_t} = g(\boldsymbol{W_1 x_t} + \boldsymbol{b_1}) \tag{1}$$

ここで，$\boldsymbol{W_1} \in \mathbb{R}^{H_2 \times H_1}$，$\boldsymbol{b_1} \in \mathbb{R}^{H_2}$，$\boldsymbol{h_t} \in \mathbb{R}^{H_2}$ であり，H_2 は（b）の隠れ層の次元である．

さらに，{B, M, E, S} などの出力ラベル集合 T に対して，隠れ層のベクトル $\boldsymbol{h_t}$ は以下の線形変換に渡され，出力 $\boldsymbol{y_t}$ を得る．

$$\boldsymbol{y_t} = softmax(\boldsymbol{W_2 h_t} + \boldsymbol{b_2}) \tag{2}$$

ここで，$|T|$ を出力ラベルの次元として，$\boldsymbol{W_2} \in \mathbb{R}^{|T| \times H_2}$，$\boldsymbol{b_1} \in \mathbb{R}^{|T|}$，$\boldsymbol{h_t} \in \mathbb{R}^{|T|}$ である．

3.2 リカレントニューラルネットワーク

本節では，リカレントニューラルネットワーク（以下，**RNN**）について説明する．RNN は現在の隠れ層の入力に 1 つ前の隠れ層の値を入れることで出力層を得る，定式化すると，(1) 式に 1 つ前の入力を足して以下のようになる．

$$\boldsymbol{h_t} = g(\boldsymbol{U h_{t-1}} + \boldsymbol{W_1 x_t} + \boldsymbol{b_1}) \tag{3}$$

ここで，$\boldsymbol{U} \in \mathbb{R}^{H_2 \times H_2}$ である．RNN は自然言語処理の様々なタスクで成功を収めているが，誤差伝搬時に系列の初めに行くに連れて勾配が伝わらなくなるという問題を抱えている．

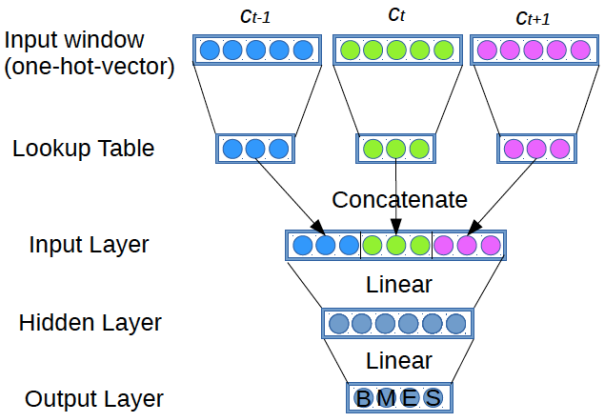

図 1: 日本語に対してのニューラル単語分割の図

3.3 Long Short Term Memory（長短期記憶ユニット）

本節では，Long Short Term Memory（以下，**LSTM**）について説明する [4]．LSTM は上で述べた RNN の問題を解決するための拡張である．LSTM の中心となるのは，各ステップにおける入力に対してのメモリセル c である．このセル c は入力ゲート，忘却ゲート，出力ゲートの 3 つのゲートにより制御されている．以下に LSTM の定式化を示す．ここで，σ，ϕ はそれぞれ *sigmoid*，*tanh* であり，$\odot$ は成分ごとの掛け算を表すアダマール積である．

$$\begin{aligned}
\boldsymbol{i_t} &= \sigma(\boldsymbol{W_{ix} x_t} + \boldsymbol{W_{ih} h_{t-1}}) && (4)\\
\boldsymbol{f_t} &= \sigma(\boldsymbol{W_{fx} x_t} + \boldsymbol{W_{fh} h_{t-1}}) && (5)\\
\boldsymbol{c_t} &= \boldsymbol{f_t} \odot \boldsymbol{c_t} + \boldsymbol{i_t} \odot \phi(\boldsymbol{W_{cx} x_t} + \boldsymbol{W_{ch} h_{t-1}}) && (6)\\
\boldsymbol{o_t} &= \sigma(\boldsymbol{W_{ox} x_t} + \boldsymbol{W_{oh} h_{t-1}}) && (7)\\
\boldsymbol{h_t} &= \boldsymbol{o_t} \odot \phi(\boldsymbol{c_t}) && (8)
\end{aligned}$$

4 提案手法

4.1 学習

本研究では，KyTea [9] のように系列中のある文字に対して，周辺の文字等の情報からラベルを推定する点推定による学習を検討した．図 1 における出力 $\boldsymbol{y_t}$ に対し，文字 c_t に対する正解ラベル分布 $\boldsymbol{l_t}$ を用意し，次式の交差エントロピー誤差によって目的関数を設定する．

$$\boldsymbol{loss} = \sum_t -\boldsymbol{l_t} \log \boldsymbol{y_t} + \frac{1}{2}\lambda \|\theta\|_2^2 \tag{9}$$

ここで，第 2 項は正則化項であり，λ はハイパーパラメータである．この関数を最小化するように，誤差逆

Copyright(C) 2016 The Association for Natural Language Processing.
All Rights Reserved.

伝搬を行い学習する．注意したいのは，点推定での学習としているが，RNN や LSTM を用いた点推定は前の隠れ層 h_{t-1} を入力していることから，以前のラベルを出力するための情報を保持していることである．また，KyTea は離散的な素性を利用しているが，本研究では連続的な素性を利用しているため，未知語などに頑健な解析が可能になると考えられる．

4.2 入力ベクトルの拡張

日本語は，ひらがな，カタカナ，漢字といった様々な文字種をもち，これらの情報が日本語単語分割において重要であることが知られている [9]．このことから，ひらがな，カタカナ，漢字，数字，アルフォベット，その他，文頭，文末の計 8 次元の one-hot ベクトルを作成し，これを文字 embedding と同様にして，文字種 embedding を作成した．これを入力ベクトル $\boldsymbol{x}$ に連結することにより，新たな入力ベクトル $\boldsymbol{x}$ を作成した．また，KyTea の入力とは異なり，本手法は，入力を embedding として表現しており，正解データから embedding を誤差逆伝搬により学習している点が異なっている．

5 実験

日本語の単語分割において，さまざまな構造の深層ニューラルネットワークを適用し，BCCWJ，Twitter コーパスの 2 つのコーパスにおける単語分割の性能の比較を行った．

評価は単語分割の適合率と再現率の調和平均である F 値によって評価した．実装は，Chainer (ver1.4.0) [12] を用いた．

5.1 データ

本研究では，日本語書き言葉均衡コーパス（以下，BCCWJ）と Takahashi ら [11] が使用したコーパスを Twitter コーパスとして利用する．BCCWJ は短単位で分割されたコアデータを用い，Project Next NLP の形態素解析タスク [13] で ClassA-1[1] として公開している ID list を用いてテストデータを抽出し，それ以外をトレーニングデータとして利用した．トレーニングデータ，テストデータの文数はそれぞれ，56,448，2,984 である．また，Twitter コーパスは，TWI（Twitter）という新しいジャンルとして BCCWJ を拡張するように作成されており，トレーニングデータ，テストデータの文数はそれぞれ，2,442，500 である．

表 1: BCCWJ と Twitter コーパスでのニューラルモデルと KyTea との比較（F 値）

Method	BCCWJ	Twitter
KyTea	98.34	92.38
FFNN （文字）	96.53	86.28
RNN （文字）	96.46	86.24
LSTM （文字）	97.00	86.34
FFNN（文字 + 文字種）	96.77	88.78
RNN （文字 + 文字種）	96.90	88.73
LSTM（文字 + 文字種）	97.25	89.27

5.2 ハイパーパラメータ

文字 embedding の次元，隠れ層の次元，正則化項の係数は Chen ら [2] の設定を用い，それぞれ，100，150，0.0001 とした．文字種 embedding の次元については 1，5，10，20，50 から最も F 値が高かった 10 を採用した．ウィンドウサイズについては，$k = 3, 5$ で，出力ラベルについては，3 節で説明した，3 通りについて事前に精度を評価し，$k = 5$，{B,E,M,S} によるラベル付けをそれぞれ採用した．最適化には Adam を利用しているがパラメータは Chainer のデフォルト設定を使用した．

5.3 実験結果

FFNN, RNN, LSTM とこれらそれぞれに文字種 embedding を加えた 6 通りの手法について BCCWJ, Twitter コーパスを用いて KyTea と比較した結果を表 1 に示す [2]．その結果，LSTM（文字＋文字種）がニューラルモデルの中では最高精度となった．

6 考察と分析

6.1 手法の比較

本研究では，ニューラルネットワークの構造として，文字，文字種 embedding のベクトルを concat した入力ベクトルを用い，FFNN，RNN，LSTM の構造がどの程度貢献するかを評価した．表 1 を見ると．文字

[1] データの詳細は http://plata.ar.media.kyoto-u.ac.jp/mori/research/topics/PST/NextNLP.html

[2] Twitter コーパスでは事前のテストによりラベル次元を 2（{B, I} ラベル）としている．

Copyright(C) 2016 The Association for Natural Language Processing.
All Rights Reserved.

表 2: 単語分割の誤り事例

LSTM のみ不正解	何 者 な ん でしょう ねぇ
正解	何者 な ん でしょう ねぇ
LSTM のみ不正解	うち がまんまその 環境 です 。
正解	うち が まんま その 環境 です 。
KyTea のみ不正解	神経 の ズ 太い 人 は 、 おそらく
正解	神経 の ズ太い 人 は 、 おそらく
KyTea のみ不正解	思う とうんざり です 。
正解	思う と うんざり です 。

embedding のみを入力とした場合は，FFNN，RNN にあまり違いはないが，LSTM は 0.5 ポイント程度 F 値の向上に貢献した．このことから，前の系列の文字情報は重要であるが，必要な情報を取捨選択する必要があることを示している．また，文字種 embedding を加えた場合は，FFNN，RNN，LSTM の順に F 値が上昇しており，前の系列の文字種情報の有用性を示している．

また，KyTea との違いとして，入力が離散的ではなく，連続値であることから，未知語などに頑健な学習が可能であると考え，未知語などが多く存在する Twitter コーパスを用いて性能を評価したが，改善が見られなかった．この原因として，パラメータ数が多く，学習データが不足していること，6.2 節で言及する入力情報の問題などが考えられる．

6.2 入力情報の比較

本研究では，日本語単語分割への拡張として，入力情報に文字種情報を追加する手法を検討し有効性を確認した．KyTea では，文字，文字種の n-gram の素性が使われていることから，本研究でも同様の情報を入れることを試みた．しかし，表 1 を見ると，KyTea よりも精度が 1 ポイント程度低くなる結果となった．この原因として，文字，文字種の n-gram と同等の素性に対応するのが，文字，文字種の uni-gram embedding の concat でないことが考えられ，Convolutional Neural Network (CNN) のような手法で n-gram を表現するか，bi-gram embedding を入力として加えるなどの調整が必要であると考えられる．

6.3 解析結果の比較

KyTea では正解できるが本手法では解析を誤る例，本手法では正解できるが，KyTea では解析を誤る例を表 2 に示す．本手法の特徴として，表 2 の 3 つ目の例のように，カタカナや漢字が混在する単語の分割に成功しているが，その他の例のように，漢字のみで構成された熟語の分割をミスしたり，ひらがなでできた名詞，形容詞などが機能表現と繋がってしまう例が KyTea よりも多く見られた．

7 おわりに

本研究では，日本語単語分割分割に対して，深層ニューラルネットワークによる手法を適用し性能を評価した．今後の課題としては，品詞情報の付加，対話などの応用に向けた読み情報の付加（PE），深層学習を利用した正規化，深層ニューラルネットワークに対する辞書情報の追加手法の検討などが挙げられる．

参考文献

[1] Xinchi Chen, Xipeng Qiu, Chenxi Zhu, and Xuanjing Huang. Gated recursive neural network for Chinese word segmentation. In *ACL-IJCNLP*, pp. 1744–1753, 2015.

[2] Xinchi Chen, Xipeng Qiu, Chenxi Zhu, Pengfei Liu, and Xuanjing Huang. Long short-term memory neural networks for Chinese word segmentation. In *EMNLP*, pp. 1197–1206, 2015.

[3] Ronan Collobert and Jason Weston. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *ICML*, pp. 160–167, 2008.

[4] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, Vol. 9, No. 8, pp. 1735–1780, 1997.

[5] Taku Kudo, Kaoru Yamamoto, and Yuji Matsumoto. Applying conditional random fields to Japanese morphological analysis. In *EMNLP*, pp. 230–237, 2004.

[6] Jianqiang Ma and Erhard Hinrichs. Accurate linear-time Chinese word segmentation via embedding matching. In *ACL-IJCNLP*, pp. 1733–1743, 2015.

[7] Hajime Morita, Daisuke Kawahara, and Sadao Kurohashi. Morphological analysis for unsegmented languages using recurrent neural network language model. In *EMNLP*, pp. 2292–2297, 2015.

[8] Graham Neubig and Shinsuke Mori. Word-based partial annotation for efficient corpus construction. In *LREC*, pp. 2723–2727, 2010.

[9] Graham Neubig, Yosuke Nakata, and Shinsuke Mori. Pointwise prediction for robust, adaptable Japanese morphological analysis. In *ACL-HLT*, pp. 529–533, 2011.

[10] Wenzhe Pei, Tao Ge, and Baobao Chang. Max-margin tensor neural network for Chinese word segmentation. In *ACL*, pp. 293–303, 2014.

[11] Fumihiko Takahasi and Shinsuke Mori. Keyboard logs as natural annotations for word segmentation. In *EMNLP*, pp. 1186–1196, 2015.

[12] Seiya Tokui, Kenta Oono, Shohei Hido, and Justin Clayton. Chainer: a Next-Generation open source framework for deep learning. In *NIPS Workshop*, 2015.

[13] 鍛治伸裕, 森信介, 高橋文彦, 笹田鉄朗, 斉藤いつみ, 服部圭悟, 村脇有吾, 内海慶. 形態素解析のエラー分析. ProjectNext エラー分析ワークショップ, 2015.

Copyright(C) 2016 The Association for Natural Language Processing.
All Rights Reserved.