辞書情報と単語分散表現を組み込んだ リカレントニューラルネットワークによる日本語単語分割

池田 大志　　進藤 裕之　　松本 裕治
奈良先端科学技術大学院大学 情報科学研究科
{ikeda.taishi.io7, shindo, matsu}@is.naist.jp

1 はじめに

近年，ニューラルネットワークを用いた中国語や日本語の単語分割手法がいくつも提案されているが，従来の条件付確率場 (Conditional Random Fields; CRF) や点推定を用いた手法と比較して性能が低い場合があることが報告されている [10]．その一つの原因として，あらかじめ与えられる単語辞書の情報を十分に利用することができていない可能性が考えられる．日本語や中国語など多くの言語では，単語分割の学習データに加えて，単語辞書が入手可能な場合が多いため，単語辞書を上手く利用することによって，学習データに出現しない単語や低頻度語の分割精度を向上させることが期待できる．

しかしながら，ニューラルネットワークによって文字ごとに BIO タギングを行う単語分割モデルにおいて，辞書の情報をどのようにモデルに組み込むかは自明でない．これは，辞書が単語ベースであるのに対して，モデルは文字ベースであることに起因する．そこで，本研究では，リカレントニューラルネットワークによる文字単位の単語分割モデルにおいて，辞書情報を効率的に利用する手法を提案する．

具体的には，入力テキストの各文字において，その文字で始まる辞書エントリーが一つ以上存在するかというバイナリ情報や，実際に辞書エントリーの開始文字にマッチした場合には，マッチした辞書エントリーの単語分散表現を素性として導入することによって，辞書情報をニューラルネットワークベースのモデルに組み込むことが可能となる．

実験の結果，現代日本語書き言葉均衡コーパスとマイクロブログコーパスにおいて，上記の提案手法に基づいて辞書情報を組み込むことによって，単語分割の精度が大きく向上することを確認した．また，本研究で作成したプログラムコードは，著者のページにて公開する [1]．

2 関連研究

日本語の形態素解析は，文を単語 (形態素) に区切り，品詞を同定する処理である．伝統的な日本語の形態素解析の手法として，入力文に対する可能な系列ラベリングを表現した形態素ラティスを，辞書を用いて構築し，CRF などの機械学習の手法を用いて，ラティス中の全候補系列から最適なパスを求めることで，単語境界とその品詞を同時に求める手法が存在する [6, 5, 4, 8]．また，周囲の文字列の情報のみを利用した点推定の手法を用いて単語分割し，その後，品詞を付与することで形態素解析を行う手法が存在する [9]．どちらの手法も新聞などの整った文に対しては，非常に高い精度を実現している．しかし，依然として，口語表現や表記ゆれを多く含むマイクロブログ上の文に対しては，解析精度の低下が問題となっている．

近年，中国語の単語分割に対して，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 [1]．Chen ら [1] は，リカレントニューラルネットワークモデル (Recurrent Neural Network Model; RNNM) を用いて，中国語の単語分割に取り組み，複雑な素性設計を用いることなく，高い単語分割の精度を実現している．本研究では，Chen ら [1] の RNNM をベースとして，彼らの手法に辞書情報と単語の分散表現を組み込むことで，日本語の単語分割の精度向上を実現する．

3 リカレントニューラルネットワークによる単語分割

本節では，提案手法のベースとなる Chen ら [1] の RNNM の詳細について説明する．

Chen ら [1] は，Long Short-Term Memory(LSTM) を用いて，入力文 $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$ に対するタグの系列 $\boldsymbol{y} = (y_1, y_2, \ldots, y_n)$ を予測するタスクとして中国語の単語分割を定式化している．各文字に対し，B, M, E, S (Begin，Middle，End，Single) の 4 つのタグを付与することで，入力文に対する単語分割を実現している．ここで，n，x_t，y_t はそれぞれ，入力文の文字数，時刻 t の文字，その文字に対するタグを表す．

まず，入力文から時刻 $t = 1, \ldots, n$ に対して素性を抽出し，素性ベクトルを作る．ここで，入力文 $\boldsymbol{x}$ は，素性ベクトルの系列 $\boldsymbol{x'} = (x'_1, x'_2, \ldots, x'_n)$ に変換される．具体的な素性ベクトルの抽出方法については，4 節で説明する．

次に，抽出した素性ベクトルをもとに，LSTM を用いて，再帰的に隠れ層を計算し，系列の情報を学習する．時刻 t の隠れ層のベクトル h_t は，次の式に従って計算される．ここで，本研究で用いる LSTM については，Lample ら [7] の論文を参考にされたい．

$$h_t = \mathrm{LSTM}(x'_t, h_{t-1}) \tag{1}$$

タグ系列の予測を行う出力層では，CRF を用いる．そのため，時刻 t における，各タグに対応する確率値を以下の式により計算する．

$$P_{t,y} = W_{tag} h_t + b_{tag} \tag{2}$$

[1] https://github.com/taishi-i/nws-for-japanese

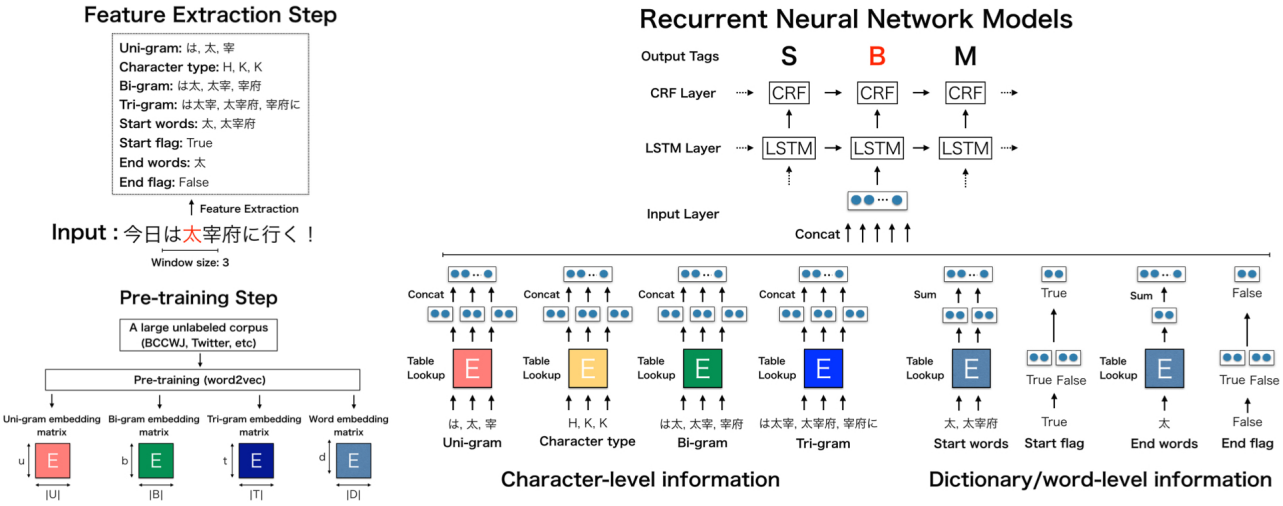

図 1: 文字単位の処理を行うリカレントニューラルネットワークに辞書情報と単語の分散表現を組み込む方法

ここで，W_{tag} と b_{tag} はそれぞれ，重み行列，バイアスベクトルを表す．また，$P_{t,y}$ の各要素は，各タグに対応する確率値である．入力文 $\boldsymbol{x}$ に対する，タグの系列 $\boldsymbol{y}$ のスコアは，以下の式により計算される．

$$s(\boldsymbol{x}, \boldsymbol{y}) = \sum_{i=0}^{n} A_{y_i, y_{i+1}} + \sum_{i=1}^{n} P_{i, y_i} \tag{3}$$

ここで，A はタグ間の遷移確率の行列を表す．また，遷移確率は時刻 $t-1$ のタグのみを考慮する．

$$-\log\left(p(\boldsymbol{y}|\boldsymbol{x};\theta)\right) = -s(\boldsymbol{x},\boldsymbol{y}) + \log\Big(\sum_{\tilde{y}\in Y_x} e^{s(\boldsymbol{x},\tilde{y})}\Big) \tag{4}$$

訓練を行う際，式 4 の負の対数尤度を確率的勾配降下法を用いて最小化することにより，最適化なパラメータ集合 θ を求める．ここで，$\tilde{y} \in Y_x$ は入力文 $\boldsymbol{x}$ の全ての取りうるタグの候補の系列を表す．

$$\boldsymbol{y}^* = \underset{\tilde{y}\in Y_x}{\operatorname{argmax}}\, s(\boldsymbol{x}, \tilde{y}; \theta) \tag{5}$$

デコードの際，入力文に対する式 5 のスコアを最大とするタグの系列 $\boldsymbol{y}^*$ を動的計画法で求めることで，単語分割を実現する．

4 提案手法

本節では，各時刻に対して，入力文から素性を抽出し，素性ベクトルをつくる処理について説明する．素性は，文字単位の情報と単語単位の情報を組み合わせてニューラルネットワークに入れる．このとき，文字ベースのモデルに対して，単語辞書の情報を二種類の単語分散表現として導入することが提案手法のポイントである．図 1 では，入力文の時刻 $t=4$ の「太」に対して，素性ベクトルを作る過程を示している．左上に時刻 $t=4$ で図の例文から抽出される素性の一覧，左下に大規模なラベルなしコーパスから得られる分散表現行列の一覧，右に時刻 $t=4$ で LSTM の入力となる素性ベクトルの一覧を示している．

例えば，図 1 の左下の Uni-gram embedding matrix は，文字数 U の u 次元ベクトルからなる文字の分散表現行列 $C \in \mathbb{R}^{U\times u}$ を示す．各文字は，この行列 C の各列ベクトルに紐付いている．したがって，それぞれの文字に紐付いたベクトルを行列 C から抽出し，素性ベクトルに用いる．ここで，窓枠 k を設定し，周囲の文字 $(c_{t-k-1}, \ldots, c_t, \ldots, c_{t+k-1})$ を結合した素性を時刻 t の文字素性ベクトルとする．また，文字と同様に，文字種，文字 Bi-gram，文字 Tri-gram も窓枠 k を設定し，時刻 t の素性ベクトルを作る．

次に，辞書の情報を単語分散表現として導入する過程について説明する．図 1 の例文の文脈で，「太」の文字を開始とする辞書に存在する単語は「太」と「太宰府」の 2 つの単語である．ここで，「太」と「太宰府」に紐付いた単語のベクトルを単語分散表現行列から抽出し，それぞれのベクトルの和を取ることにより，辞書とマッチしたエントリーの単語分散表現ベクトル s_t を作る．

$$s_t = \sum_{word\in StartWords_t} \mathrm{Embed}_{\mathrm{start}}[word] \tag{6}$$

この素性は Neubig ら [9] が用いた素性と基本的に同様であるが，本研究では，離散的な素性ではなく，単語を実数値のベクトルとして表現しているところが異なる．この単語分散表現は，大規模なラベルなしコーパスを用いて事前学習することができる．また，「太」の文字を開始とする場合の素性抽出と同様に，「太」の文字を終了とする辞書に存在する単語も素性ベクトルとして抽出し，単語分散表現ベクトル e_t を作る．本研究では，開始・終了ごとに異なる単語分散表現行列を準備し，それぞれパラメーターの更新を行う．

ここで，時刻 t で文字 x_t を開始とする単語が 2 単語以上存在する場合に，True となるバイナリの辞書素性を新たに導入する．これは，時刻 t に文字だけではなく，単語が含まれていることを意味し，時刻 t で単語の開始となることの情報を表す．同様に，時刻 t で文字 x_t を終了とする単語が 2 単語以上存在する場合に，True となるバイナリの辞書素性と組み合わせることで，どちらの辞書素性も False になる場合，単語の開始・終了の情報だけではなく，時刻 t の文字 x_t が単語の間の文字であること，または 1 文字だけであることを表現することができる．そして，この辞書素性も分散表現のベクトルとして用いる．

本研究の提案手法では，単語の分散表現と辞書素性を用いて，単語境界のない言語において，文字単位の処理を行う RNNM に対しても，明示的に単語の情報と単語の始まりと終わりの情報を与えることができる．

最終的に，それぞれ素性を組み合わせ，抽出した素性ベクトルを 1 つに結合したものを時刻 t の素性ベクトル x'_t として用いる．

5 実験

本研究では，現代日本語書き言葉均衡コーパス (以下，BCCWJ) とマイクロブログコーパス (以下，Microblog) を用いた単語分割の評価を行う．評価は，単語分割の適合率と再現率の調和平均である F 値によって評価する．

5.1 実験で用いる言語資源

本節では，実験で用いる言語資源について説明する．BCCWJ を用いて評価を行う場合，まず，コアデータを訓練・開発・評価データに分割する．本研究では，次のデータ分割法を採用した．Project Next NLP の形態素解析タスクで ClassA-1[2] として公開されている ID のリストを用いて評価データを抽出し，それ以外を訓練・開発データとして利用した．短単位の原文文字列を抽出し，訓練・開発・評価データの文数はそれぞれ，55,948，500，2,983 である．KyTea を再学習する場合，コアデータより原文文字列，品詞大分類，発音形出現形を抽出し，フルアノテーションコーパスを作成した (例：コーパス/名詞/こーぱす)．また，UniDic(2.1.2) より抽出した 756,460 エントリーを辞書として利用する．本研究では，活用語尾の分割の処理は行っていない．

Microblog を用いて評価を行う場合，訓練・開発として，京都大学テキストコーパス (Version 4.0) と京都大学ウェブ文書リードコーパスを組み合わせたデータを用いた．両コーパスから抽出した訓練・開発の文数はそれぞれ，47,642, 500 である．また，評価データとして用いる Microblog[4] は，Twitter から収集した 1,000 ツイートに対して，JUMAN 辞書の基準に従ってアノテーションされたコーパスである．アノテーション情報は，単語境界・品詞だけではなく，崩れ表記に対しては，その正規形も付与されている (例: すげー → すごい)．この Microblog から原テキストと正規テキストの文をそれぞれ，1,831 文，545 文抽出し，評価データとして用いた．Microblog のデータは，訓練には一切使用していない．

また，大規模なラベルなしコーパスとして，BCCWJ のコアデータのジャンルである OW(白書)，PB(書籍)，PN(新聞)，PM(雑誌)，OC(Yahoo!知恵袋)，OY(Yahoo!ブログ) を用いた．ClassA-1 に含まれていない，3,512,522 文を利用する．

5.2 実装詳細

本研究の提案手法の実装は，深層学習ライブラリー Theano[3] を利用した．Uni-gram ベクトル，文字種，Bi-gram ベクトル，Tri-gram ベクトルの次元数をそれぞれ，100, 10, 50, 50 に設定した．単語ベクトルと開始・終了を示す辞書素性のベクトルの次元数をそれぞれ，100 に設定した．また Uni-gram，Bi-gram，Tri-gram，単語のベクトルは，大規模なラベルなしコーパスを用いて事前学習を行い初期値として設定した．本研究では，Gensim[4] を用いて，分散表現の事前学習を行った．ハイパーパラメーターの設定は，Gensim のデフォルト設定を使用したが，学習アルゴリズムのみ Skip-gram(sg=1) を採用した．その他の重み・分散表現（LSTM，タグの遷移確率，文字種）は，[-0.08, 0.08] から一様分布に従ってサンプリングした値を初期値として設定する．

タグの種類は，B, M, E, S の 4 種類，文字種は，ひらがな，カタカナ，漢字，数字，アルファベット，その他の 6 種類を採用した．

エポック数は 50 に設定し，開発データの F 値が最も良いエポックでの評価データの結果を報告する．パラメーターの最適化は，確率的勾配降下法 (SGD) で行う．なお，学習率は 0.005 に設定し，各エポックで開発データの F 値が 5 回改善しない場合，学習率を半減する．窓枠を 3 に設定した．勾配クリッピングを 5 に設定した．入力層に対して，ドロップアウト (Dropout rate = 0.2) を適応し，訓練を行った．

5.3 結果・考察

本研究では，文字，文字種，Bi-gram を素性として用いた RNNM をベースライン（LSTM-CRF）とす

	適合率	再現率	F 値
LSTM-CRF	98.93	99.01	98.97
+Trigram (Pre-trained)	99.11	99.11	99.11
+DictFlags	99.27	99.24	99.25
+Trigram (Pre-trained), DictFlags	**99.33**	**99.34**	**99.33**
+AutoSegWords (Random)	98.94	99.07	99.01
+AutoSegWords (Pre-trained)	99.14	99.14	99.14
KyTea	98.42	98.37	98.39
+Dictionary	99.18	99.10	99.14
KyTea (0.4.7) [9]	99.46	99.71	99.58
MeCab (0.996) [5]	99.23	99.82	99.52

表 1: BCCWJ ClassA-1 に対する単語分割精度

	適合率	再現率	F 値
LSTM-CRF	**82.02**/82.03	86.89/86.73	84.38/84.32
	(12,445/15,174)	(12,445/14,324)	
+Trigram (Pre-trained), DictFlags*	81.71/81.78	87.49/87.65	**84.50**/84.61
	(12,531/15.336)	(12,531/14,324)	
Juman (0.7)	79.44/80.76	88.74/89.41	83.83/**84.86**
	(12,711/16,001)	(12,711/14,324)	
JUMAN++ (1.01) [8]	77.73/78.18	**89.51/89.80**	83.21/83.59
	(12,822/16,495)	(12,822/14,324)	
MeCab (0.996) [5]	75.20/77.08	86.81/88.71	80.59/82.49
	(12,435/16,535)	(12,435/14,324)	

表 2: マイクロブログコーパスの原テキストと正規テキストに対する単語分割の適合率，再現率と F 値を示す．また，原テキストにおける各解析器の出力形態素数も示す．

[2] http:/plata.ar.media.kyoto-u.ac.jp/mori/research/topics/PST/NextNLP.html

[3] http://deeplearning.net/software/theano/

[4] https://radimrehurek.com/gensim/models/word2vec.html

GOLD	よろしく/（＊＾ω＾＊）
LSTM-CRF+Trigram, DictFlags*	よろしく/（＊＾ω＾＊）
MeCab	よろしく/（/＊＾ω＾＊）
Juman	よろしく/（/＊/＾/ω/＾/＊/）
JUMAN++	よろしく/（/＊/＾/ω/＾/＊/）
GOLD	ふぉろわー/集め/の/ため
LSTM-CRF+Trigram, DictFlags*	ふぉろわー/集め/の/ため
MeCab	ふ/ぉろ/わ/ー/集/め/の/ため
JUMAN	ふ/ぉろ/わー/集め/の/ため
JUMAN++	ふ/ぉろ/わー/集め/の/ため

表 3: マイクロブログ文に対する単語分割の比較結果．「/」は単語境界を表す．

る．このベースラインに，辞書素性と単語の分散表現を追加することで，提案手法の有効性を検証する．

まず，BCCWJ を用いた単語分割の結果を表 1 に示す．ここでは，本研究の提案手法とベースライン，また形態素解析器 MeCab，KyTea の F 値を示す．ここで，Web 上で配布されている形態素解析器には，訓練データとして，ClassA-1 が含まれている可能性があるため，厳密な比較はできないことを注意されたい．

まず，ベースラインに辞書素性を追加すること (+DictFlags) で，ベースラインに比べて，F 値が 0.28 ポイント向上した．これは，文字や文字種だけの情報ではなく，明示的に単語の始まりと終わりを与えることで，単語分割の精度向上に貢献したと考えられる．

次に，単語の分散表現をベースラインに追加する提案手法 (+AutoSegWords) の有効性を検証する．ここで，単語の分散表現は，ベースライン (LSTM-CRF) を用いて，大規模なラベルなしコーパスを単語分割し，事前学習を行い，分散表現の初期値とした (Pre-trained)．その結果，単語の分散表現を追加することで，ベースラインに比べて，F 値が 0.17 ポイント向上した．今後の課題として，提案手法 (+Trigram (Pre-trained), DictFlags) を用いて，より高い精度で単語分割を行い得られた大規模な単語分割済みコーパスを用いて，事前学習した分散表現をベースラインに追加する実験も行いたいと考えている．最後に，点推定の手法を用いた手法である KyTea を辞書素性を追加し再学習した場合と比較する．提案手法 (+Trigram (Pre-trained), DictFlags) と比較した結果，F 値で 0.19 ポイントの差があった．これは，点推定の手法に比べて，入力の系列を考慮したことで，F 値の差があったと考えられる．

次に，Microblog を用いた単語分割の結果を表 2 に示す．ここでは，本研究の提案手法とベースライン，また形態素解析器 MeCab(JUMAN 辞書)，JUMAN JUMAN++ の F 値を示す．ここで，MeCab は，京都大学テキストコーパスのみを訓練データとしているため，厳密な比較はできないことを注意されたい．

これまで多くの研究で議論されているが，口語表現や表記ゆれなどを多く含むツイートなど文に対して，形態素解析の精度を大きく低下させる原因の 1 つとして，従来のラティスのベースの形態素解析手法では，顔文字や表記ゆれ（ひらがな化）に対して，過剰な分割をしてしまう傾向があると考えられる．その具体的な例を表 3 に示す．

ラティスのベースの手法では，原テキストの全形態数が 14,324 に対して，各解析器とも，出力結果が 16,001 から 16,535 と多くなっている．この過剰な分割が適合率を下げ，単語分割の精度を下げる原因となっている．しかし，提案手法 (+Trigram (Pre-trained), DictFlags*) では，入力文の各文字に対する系列ラベリングの手法を採用しているため，出力結果は 15,336 となり，ラティスのベースの手法に比べて，出力の形態素数が少ない．これは，RNNM を用いることで，入力の系列の情報を考慮することができ，顔文字や未知語となるようなひらがなの連続した単語は，１つの纏まった単語として捉えることができているためだと考えれれる．他にも「にゅーす, らーぶっ，うぜー，かっけぇ」などの表記ゆれや口語表現に対しても，適切に単語分割できている例を多く確認できた．その結果，適合率が上がり，既存の解析器に比べて，原テキストの F 値が約 0.7 ポイント上回る結果を得た．

6 おわりに

本研究では，文字単位の処理を行うリカレントニューラルネットワークに辞書情報と単語の分散表現を組み込む手法を提案した．今後は，単語分割と同時に，品詞タグ付け [2] や崩れ表記の正規化 [3] を行う手法に取り組む予定である．

参考文献

[1] Xinchi Chen, Xipeng Qiu, Chenxi Zhu, Pengfei Liu, and Xuanjing Huang. Long short-term memory neural networks for chinese word segmentation. In *EMNLP*, 2015.

[2] Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, Vol. 12, No. Aug, pp. 2493–2537, 2011.

[3] Taishi Ikeda, Hiroyuki Shindo, and Yuji Matsumoto. Japanese text normalization with encoder-decoder model. In *WNUT*, pp. 129–137, 2016.

[4] Nobuhiro Kaji and Masaru Kitsuregawa. Accurate word segmentation and pos tagging for japanese microblogs: Corpus annotation and joint modeling with lexical normalization. In *EMNLP*, pp. 99–109, 2014.

[5] Taku Kudo, Kaoru Yamamoto, and Yuji Matsumoto. Applying conditional random fields to japanese morphological analysis. In *EMNLP*, Vol. 4, pp. 230–237, 2004.

[6] Sadao Kurohashi, Toshihisa Nakamura, Yuji Matsumoto, and Makoto Nagao. Improvements of japanese morphological analyzer juman. In *Proceedings of The International Workshop on Sharable Natural Language*, pp. 22–28, 1994.

[7] Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. Neural architectures for named entity recognition. *arXiv preprint arXiv:1603.01360*, 2016.

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

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

[10] 北川善彬, 小町守. 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. 言語処理学会 第 22 回年次大会発表論文集, 2016.