Unicode符号位置に基づいた マルチバイト埋め込みによる形態素解析

松野 智紀[1]　林 克彦[2]
[1]LAPRAS 株式会社
[2] 東京大学　大学院情報理工学系研究科
[1]matsuno@lapras.com, [2]khayashi0201@gmail.com

1 はじめに

日本語や中国語のように文字の語彙数が大きな言語を対象としたニューラル形態素解析では，モデル全体のパラメタ数のうち文字埋め込みが大部分を占める．また，未知語や低頻度語に頑健とするため，教師なしコーパスを使って言語モデル学習で文字埋め込みを得る方法があるが，これは語彙数が増加し，パラメタ数をさらに増やすことになる．本稿では Unicode の符号位置に基づいてバイト埋め込みから文字表現を作ることで，モデル全体のパラメタ数を削減しつつ文字種などの情報に意識的な解析を行う手法を新たに提案する．実験では文字埋め込みを使ったモデルとの比較を行った．結果から，提案法は京大コーパスでの実験においてパラメタ数を 32.12% 削減しつつ文字埋め込みを使った従来法とほぼ同等の性能を示した．また，モデルの一部を RNN 言語モデルとして訓練することでパラメタ数を増やすことなくモデルの性能を向上させられることも確認した．

2 Unicode 符号位置

計算機で自然言語の文字を扱うために文字集合を定義し各文字をビット組み合わせにより一意に符号化したものを符号化文字集合と呼ぶ． 符号化文字集合のなかでビット組み合わせが取り得る値を符号位置と呼び，これは文字符号と文字の読み替え表の中で文字が割り当てられ得る位置をさす．

Unicode は符号化文字集合の国際規格である．世界中の文字を単一の規格で扱っているため，多言語の文字が同一の文書に現れる場合であっても切り替えや組み合わせといった処理を必要としない．

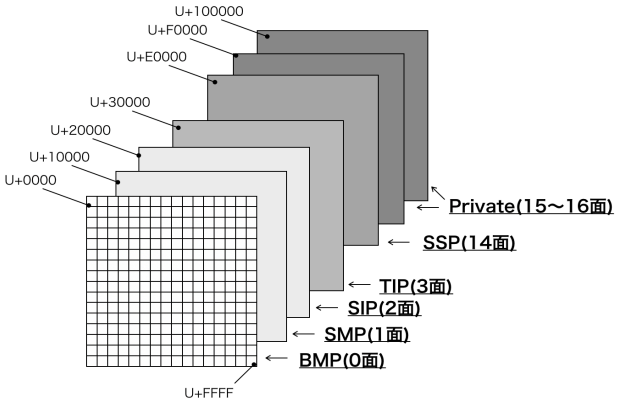

図 1: Unicode の符号空間

2.1 面・区・点

Unicode の符号位置は 3 つの単位で表される．それぞれ面・区・点と呼ばれ，図 1 中の平面・行・列に対応する．Unicode の符号空間で各単位が取り得る値を表 1 に示す．ただし，現時点では第 4~13 面は何に使われるかが決められておらず，文字が割り当てられていない．現在使われている 6 つの面には名称があり，それぞれ基本多言語面（BMP），追加多言語面（SMP），追加漢字面（SIP），第三漢字面（TIP），追加特殊用途面（SSP），私用面（Private）と呼ばれている．基本多言語面に通常使われる文字のほとんどが含まれている．

面	区	点
0~16	0~256	0~256

表 1: Unicode の符号空間

2.2 コードブロック

Unicode において各符号位置は符号位置の連続する範囲を意味するブロックに所属する．表 2 にいくつかの例を示す．各ブロックには一意に名前がつけられており，漢字やハングル，ギリシア文字など同じ文字種に属する文字の符号位置は同じブロックに所属する．そのため，符号位置に基づいたバイト埋め込みを用いて文字を表現することにより，ある程度文字種に意識的な表現を得ることが可能となる．

符号位置	名前
0370..03FF	Greek and Coptic
0590..05FF	Hebrew
0600..06FF	Arabic
0700..074F	Syriac
1100..11FF	Hangul Jamo
3040..309F	Hiragana
30A0..30FF	Katakana
4E00..9FFF	CJK Unified Ideographs
AC00..D7AF	Hangul Syllabl
20000..2A6DF 2A700..2B73F 2B740..2B81F 2B820..2CEAF 2CEB0..2EBEF 2F800..2FA1F	CJK Unified Ideographs Extension B ～ F

表 2: 符号位置と文字種

2.3 区の内部における文字の並び

Unicode では区の内部においても一定の規則性を持って文字の並びが決められている．例えば，コードブロックの CJK Unified Ideographs の中では同じ部首を持つ日本語，中国語，韓国語の漢字の符号位置が連続している．図 2 に第 0 面の第 103 区に含まれる文字を示す．この中には日編，月編，木編の漢字が含まれている．このようにある部首を持つ漢字が属している区には同じ部首を持つ他の漢字も属していることが多いため，未知の漢字が入力された場合でもある程度は部首の情報に意識的に解析を行えることが期待される．

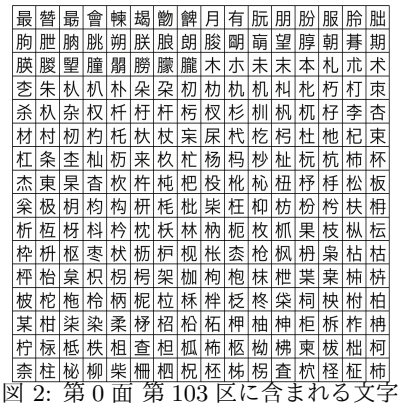

図 2: 第 0 面 第 103 区に含まれる文字

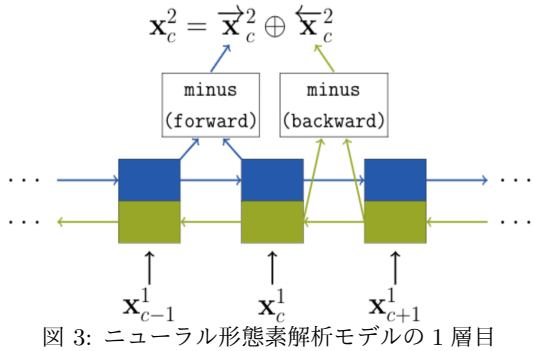

図 3: ニューラル形態素解析モデルの 1 層目

3 ニューラル形態素解析

3.1 モデルの構成

モデルは大きく分けて 2 層から成る．図 3 には 1 層目の概略を示す．以下では，$\mathbf{x}^l$ のように上付き添字で l 層目の構成要素であることを明記する．また，l 層目の前向き LSTM を $\mathbf{LSTM}_f^l$，時刻 t の前向き出力を $\overrightarrow{\mathbf{y}}_t^l$，後向き LSTM を $\mathbf{LSTM}_b^l$，時刻 t の後向き出力を $\overleftarrow{\mathbf{y}}_t^l$ と表す．

モデルの構成は以下のステップから成る．

1. まず，入力文字列をベクトルに変換する．c 番目の文字をベクトル $\mathbf{x}_c$ ($c \in \{1, 2, ..., n\}$, n は入力文の文字数) で表す．

2. 得られたベクトルの系列を単方向の前向き LSTM と後向き LSTM を用いてエンコードする．

$$\overrightarrow{\mathbf{y}}_c^1 = \mathbf{LSTM}_f^1(\mathbf{x}_c^1) \quad (1)$$

$$\overleftarrow{\mathbf{y}}_c^1 = \mathbf{LSTM}_b^1(\mathbf{x}_c^1). \quad (2)$$

3. 前向き LSTM と後向き LSTM の出力に LSTM-Minus を用いて c 番目の文字の表現にあたる 2 層目 LSTM への入力を以下のように求める

$$\overrightarrow{\mathbf{x}}_c^2 = \overrightarrow{\mathbf{y}}_c^1 - \overrightarrow{\mathbf{y}}_{c-1}^1 \quad (3)$$

$$\overleftarrow{\mathbf{x}}_c^2 = \overleftarrow{\mathbf{y}}_{c+1}^1 - \overleftarrow{\mathbf{y}}_c^1. \quad (4)$$

4. その結果を各文字について結合する．

$$\mathbf{x}_c^2 = \overrightarrow{\mathbf{x}}_c^2 \oplus \overleftarrow{\mathbf{x}}_c^2. \quad (5)$$

5. 得られたベクトルの系列を単方向の前向き LSTM と後向き LSTM を用いてエンコードする．

$$\overrightarrow{\mathbf{y}}_c^2 = \mathbf{LSTM}_f^2(\mathbf{x}_c^2) \quad (6)$$

$$\overleftarrow{\mathbf{y}}_c^2 = \mathbf{LSTM}_b^2(\mathbf{x}_c^2). \quad (7)$$

6. その結果を結合する．

$$\mathbf{x}_c^{MLP} = \overrightarrow{\mathbf{y}}_c^2 \oplus \overleftarrow{\mathbf{y}}_c^2. \quad (8)$$

7. $c-1$ 番目の文字と c 番目の文字の間に単語境界があるかを予測する．

$$\mathbf{s}_c^{seg} = \mathbf{W}^{seg}\mathbf{x}_c^{MLP} + \mathbf{b}^{seg}. \quad (9)$$

ここで，$\mathbf{W}^{seg} \in \mathbb{R}^{2\times 2h}$，$\mathbf{b}^{seg} \in \mathbb{R}^2$ である．h は LSTM 出力の次元である．

8. 単語境界がある場合，c 番目の文字が所属する語の品詞を予測する．

$$\mathbf{s}_c^{POS} = \mathbf{W}^{POS}\mathbf{x}_c^{MLP} + \mathbf{b}^{POS}. \quad (10)$$

ここで，$\mathbf{W}^{POS} \in \mathbb{R}^{|S^{POS}|\times 2h}$，$\mathbf{b}^{|S^{POS}|} \in \mathbb{R}^{|S^{POS}|}$ である．

4 提案手法

4.1 バイト埋め込みの加法による文字表現

提案手法では， Unicode の符号空間における符号位置において面・区・点が取り得る値それぞれに埋め込み表現を与える．面が 10 進数で取りえる値 i に与えられた埋め込み表現を $\mathbf{V}^{men} \in \mathbb{R}^{16\times d}$ の第 i 行目の成分とし，$\mathbf{v}_i^{men}$ であらわす[1]．ここで d は埋め込みの次元である．同様に区・面に与えられた埋め込み表現の行列 $\mathbf{V}^{ku} \in \mathbb{R}^{256\times d}$，$\mathbf{V}^{ten} \in \mathbb{R}^{256\times d}$ も定義する．

位置 c の文字の符号位置の面・区・点が i, j, k であるとき，c の文字表現 $\mathbf{x}_c$ を以下のように表す．

$$\mathbf{x}_c = \mathbf{v}_i^{men} + \mathbf{v}_j^{ku} + \mathbf{v}_k^{ten}. \quad (11)$$

例として,「木」という文字を考える．Unicode において「木」の符号位置は +u6728 である．ここで面・区・点がとる値はそれぞれ 10 進数で 0, 103, 40 となる．よって「木」の文字表現は式 (12) で表される．

$$\mathbf{v}_0^{men} + \mathbf{v}_{103}^{ku} + \mathbf{v}_{40}^{ten}. \quad (12)$$

このようにして作られた文字表現は文字埋め込みと同様，$n=1$ として，ネットワークモデル 1 層目でエンコードされる．提案手法のアイディアは非常に簡潔であり，様々なタスクで文字埋め込みの代わりとして利用できる．

他にも結合や巡回畳み込みなど多様な構成方法が考えられる．我々の事前調査において，結合や巡回畳み込みも比較したが，加法の方が良好な結果を示したため，本稿では加法による実験結果のみ報告する．

[1]現時点では Unicode の第 4 ～ 13 面は使用されていないため，実装上は $(10 \times d)$ 行列を用いる．

4.2 RNN 言語モデルによる初期化

近年，モデルの一部を教師なしコーパスで訓練された言語モデルによって初期化する手法が様々なタスクで用いられている．それによって言語の構造を陰に学習した状態でタスク依存の訓練を開始することができる．しかし，そういった訓練に用いられるコーパスが大規模になると入力として対応すべき語彙数が大きくなり，その語彙に含まれる語それぞれに埋め込み表現を与えた場合は結果としてモデル全体のパラメタ数が増える原因となる．

それに対して我々の手法は Unicode の文字集合に含まれている文字全てを 519 個のバイト埋め込みの組み合わせで表すため，訓練データが増えても必要なパラメタ数は一定である．

実験では提案手法のモデルのバイト埋め込み及び 1 層目の単方向 LSTM について言語モデルとして予め訓練し，形態素解析タスク学習のための初期値として用いた．

5 関連研究

Costa-jussà et al. [1] は機械翻訳タスクでソース文とターゲット文を UTF-8 によってバイト列に変換し，バイト列の入力に対してバイト列を出力するタスクとして解いた．実験は英語やスペイン語など文字の語彙数が少ない言語で行なわれており，バイト埋め込みを多言語で共通の空間に埋め込むことによってソース言語とターゲット言語で共通の表現を使うことに焦点を置いている点で我々の手法とは異なっている，

Kitagawa and Komachi [2] は，ニューラルネットを用いた日本語の単語分割において，漢字・平仮名・片仮名といった文字種に関する素性を付与することで解析性能が向上することを示した．我々の手法では Unicode における符号位置によって文字種に意識的な表現を得ているため，日本語の文字に限らない文字種情報を扱うことが可能である．

Tolmachev et al. [3] は，大規模な生文コーパスを高性能な形態素解析器 Juman++[2]で解析し，その解析結果を用いてよりメモリ量が小さいニューラル形態素解析器を訓練することで，高性能で省メモリ形態素解析器が作れることを示した．しかし，大規模コーパスで訓練しているため，文字埋め込みのサイズは依然として大きい．

[2]http://nlp.ist.i.kyoto-u.ac.jp/index.php?JUMAN++

データセット	UD_Japanese				京大コーパス			
	文字埋め込み		提案手法		文字埋め込み		提案手法	
	ランダム	言語モデル	ランダム	言語モデル	ランダム	言語モデル	ランダム	言語モデル
単語分割	92.77	**93.09**	92.46	92.93	97.35	**97.61**	97.27	97.39
UPOS / 大分類	89.81	**90.18**	89.08	89.92	96.62	**96.90**	96.41	96.66
XPOS / 細分類	88.92	**89.70**	88.25	89.17	95.04	**95.87**	93.51	95.40
パラメタ数（単語埋込）	237600	300000	**52200**	**52200**	272000	300000	**52200**	**52200**
パラメタ数（全体）	650657	713057	**465257**	**465257**	684253	712253	**464453**	**464453**
増減率	0%	+9.59%	**-28.49%**	**-28.49%**	0%	+4.1%	**-32.12%**	**-32.12%**

表 3: UD_Japanese-GSD および京大コーパスでの結果

6 実験

6.1 言語モデル訓練

言語モデルの訓練には日本語 Wikipedia の一部を用いた．訓練文の選定は，日本語 Wikipedia に含まれる文書のうち，UD_Japanese-GSD の訓練データ中の文それぞれに類似する文書のトップ 10 を重複を許さずに抽出することで行なった．コーパス中の文数は 471111 文となった．学習は 10 エポック行った．類似度計算のための文書表現作成には BPEmb ライブラリ[3]によって提供されているトークナイザーおよびトークン埋め込みを用いた．トークナイザーの語彙数は 20 万でトークン埋め込みの次元は 50 のものを使った．

6.2 結果

表 3 に結果を示す． 増減率はランダム初期化された文字埋め込みを用いるモデルと比べたパラメタ数の変化を表す．実験には UD_Japanese-GSD[4]および京都大学テキストコーパス[5]を用いた．全ての指標で文字埋め込みを用いて一層目 LSTM の初期化に訓練済み言語モデルを用いたものが最も高いスコアを出した．しかし，言語モデルの訓練によって文字の語彙数が増えたため，パラメタ数が大きく増えている．ランダム初期化した文字埋め込みを使うモデルと言語モデルによる事前訓練を行なった提案法を比べると，提案法はパラメタ数を大きく減らしつつ全ての指標でランダム初期化した文字埋め込みを使うモデルを上回っている．

7 おわりに

日本語のように多様な文字を含む言語に対するニューラル形態素解析において，文字の語彙数が大きいことによるモデルサイズの増大や多様な文字種に意識的な解析を行いにくいことが課題であった．本稿では Unicode 符号位置を利用したマルチバイト埋め込みによって文字表現を獲得することで，これらの課題を解決した．実験結果から提案手法がこれまで標準的に使われてきた文字埋め込みより少ないメモリ使用量で同等の解析性能を達成できることが示された．

今後の課題として，中国語や韓国語などのマルチバイト言語に対しても実験を行い，多言語で提案法の有効性を検証したい．その際に，Unicode が言語共通である特性を活かして，全言語での同時学習も興味深い研究課題となる．また，構文解析や機械翻訳などのタスクでも提案法の有効性を検証したい．

参考文献

[1] Marta R. Costa-jussà, Carlos Escolano, and José A. R. Fonollosa. Byte-based neural machine translation. In *Proceedings of the First Workshop on Subword and Character Level Models in NLP*, pages 154–158, Copenhagen, Denmark, September 2017. Association for Computational Linguistics. doi: 10.18653/v1/W17-4123. URL https://www.aclweb.org/anthology/W17-4123.

[2] Yoshiaki Kitagawa and Mamoru Komachi. Long short-term memory for japanese word segmentation. 09 2017.

[3] Arseny Tolmachev, Daisuke Kawahara, and Sadao Kurohashi. Shrinking Japanese morphological analyzers with neural networks and semi-supervised learning. 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)*, pages 2744–2755, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1281. URL https://www.aclweb.org/anthology/N19-1281.

[3] https://github.com/bheinzerling/bpemb
[4] https://universaldependencies.org/treebanks/ja_gsd/index.html
[5] http://nlp.ist.i.kyoto-u.ac.jp/index.php?京都大学テキストコーパス