リカレントニューラルネットワークを用いた述語項構造解析

大内 啓樹　　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
{ouchi.hiroki.nt6, shindo, matsu}@is.naist.jp

1 はじめに

述語項構造解析[1]は，「誰が何をどうした」という述語を中心とした意味関係を同定するタスクである．典型的な述語項構造解析手法では，品詞タグ付けと構文解析を行ってから，パーセプトロンなどの線形分類器でそれらを素性として用い，予測を行っていた．品詞や構文木などの「構文情報」が述語項構造を予測するのに有効であることが報告されているが，人手による素性設計コストの上昇と構文解析の誤り伝搬などが問題となっている．

これらの問題を回避するため，構文情報を用いず，生テキストの情報のみから述語項構造を予測する手法が提案されている．Collobert らの手法 [5] では，畳込みニューラルネットワーク (Convolutional Neural Network; CNN) を用いて，句構造に基づいた英語述語項構造解析タスク (CoNLL-2005 Shared Task) に取り組み，F 値で 74.15%の解析精度を達成している．同様のタスクにおいて，Zhou ら [10] は，リカレントニューラルネットワーク (Recurrent Neural Network; RNN) の Long Short-Term Memory (LSTM) ユニットを利用することにより，これまでの最高精度にあたる F 値 81.07%を達成した．

Zhou らの実験で，述語項構造解析タスクにおける RNN の有効性が示唆されたが，RNN の学習は，用いる再帰隠れユニットに大きく左右されると指摘されており [4]，LSTM 以外のユニットがどのような挙動を示すか定かではない．本研究では，LSTM ユニットに加え，標準的に用いられる双曲線正接 ($tanh$) ユニットや，LSTM と同等以上の性能を持つことが示唆されている Gated Recurrent Unit (GRU) の 3 つを用いた述語項構造解析における比較実験を行う．

2 リカレントニューラルネットを利用した述語項構造解析モデル

本研究では，CoNLL-2005 Shared Task[2] の句構造に基づいた英語述語項構造解析タスクに取り組む．

[1]本稿では，「述語項構造解析」と「意味役割付与」を同義として扱う．

She	did	not	attend	the	meeting	.
A0		AM-NEG		A1		
B-A0	O	B-AM-NEG	B-V	B-A1	I-A1	O

⬇

	features				label
	arg	pred	context	mark	
1	She	attend	not attend the	0	B-A0
2	did	attend	not attend the	0	O
3	not	attend	not attend the	1	B-AM-NEG
4	attend	attend	not attend the	1	B-V
5	the	attend	not attend the	1	B-A1
6	meeting	attend	not attend the	0	I-A1
7	.	attend	not attend the	0	O

図 1: Zhou ら [10] のモデルで使用する素性とラベル

2.1 述語項構造解析タスクの定式化

本タスクでは，文と解析対象の述語が与えられ，各述語の項の範囲を同定する．図 1 は，[10] で用いられる素性と予測するラベルの例を示している．図 1 中の上表の 1 行目は入力文を表しており，ターゲットの述語は `attend` である．2 行目は `attend` の項である句を示している．3 行目は項となる句の範囲を BIO タグで表しており，各単語に対応するこれらの BIO タグを予測することによって項の範囲を同定する．

図 1 の下表は，BIO タグの予測に用いる素性の例を示している．素性の種類は，項候補の単語のベクトル (arg)・述語の単語のベクトル (pred)・述語の周りの単語のベクトル (context)[2]・項候補単語が context の範囲に入っているか否か (mark)，の 4 種類であり，構文情報は用いず，生テキストから抽出できる情報のみ利用する．これらの各ベクトルを 1 つに結合したものを，各単語に対する素性ベクトルとして用いる．例えば，図 1 中の `She` の素性として，`She` 自身の単語ベクトル (arg)，ターゲットの述語 `attend` の単語ベクトル (pred)，述語 `attend` とそのまわりの単語 `not` と `the` のベクトル (context)，`She` は context に含まれていないもとを表す $mark = 0$，の 4 つの素性を結合したベクトルを用いる．各単語に対して，これらの結合した素性ベクトル $\mathbf{x}_t$ をつくり，モデルのネットワークに入力として与える．

[2]context のウィンドウサイズはハイパーパラメータであり，本研究では 5 に設定．

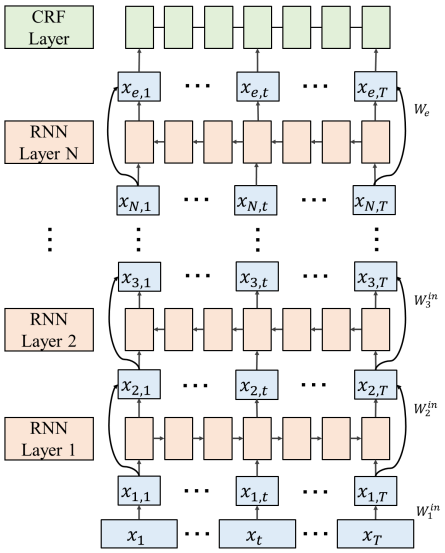

図 2: Zhou ら [10] のモデル

2.2 モデルのネットワーク構造

本研究で用いる Zhou ら [10] が提案したネットワーク構造を図 2 に示す．基本的な構造は，入力ベクトルを受け取り，RNN を用いた複数の層 (RNN Layer) で再帰的に状態ベクトルを計算し，出力層の条件付確率場 (CRF) でラベルを予測する．RNN 層は，奇数番目のものは系列を左から右に，偶数番目は右から左に処理する**両方向型リカレントニューラルネット** [6] を採用している．これらの RNN 層を重ねることによって，ネットワーク構造の深さを変えることが可能である．

2.3 リカレントニューラルネットワーク

n 層目の RNN は，各時刻 t において，入力系列 $\{\mathbf{x}_{n,t}\}_1^T$ のうちの 1 つの入力 $\mathbf{x}_{n,t}$ を受け取り，1 つの出力 $\mathbf{h}_{n,t}$ を返す．

$$\mathbf{h}_{n,t} = g\left(\mathbf{W}_n\, \mathbf{x}_{n,t} + \mathbf{W}_n\, \mathbf{h}_{n,t-1}\right) \tag{1}$$

関数 $g(\cdot)$ は任意の非線形関数であり，本研究では双曲線正接関数 ($tanh(\cdot)$) を用いる．ここで，入力ベクトル $\mathbf{x}_{n,t}$ は，$n-1$ 層目の RNN の入力ベクトル $\mathbf{x}_{n-1,t}$ と出力ベクトル $\mathbf{h}_{n-1,t}$ を結合したベクトルを次の式で計算したものである．

$$\mathbf{x}_{n,t} = u(\mathbf{W}_n^{in}\, [\mathbf{x}_{n-1,t}, \mathbf{h}_{n-1,t}])$$

ただし，1 層目のみ，入力された各単語の素性ベクトル $\mathbf{x}_t$ を単独で用いて $\mathbf{x}_{1,t}$ を計算する．関数 $u(\cdot)$ は任意の関数であり，本研究では正規化線形関数 (rectified linear function) を用いる．

RNN は勾配の消失・爆発などの問題から，学習が困難であることが指摘されている [9]．これらの問題を回避し，系列の長期的な記憶の保持を目的として LSTM[7] や GRU[3] が提案されている．

2.3.1 LSTM

LSTM は (1) 式を以下のように変更したものである．

$$\mathbf{h}_{n,t} = \mathbf{o}_{n,t} \circ f(\mathbf{c}_{n,t})$$

LSTM のユニットは，時刻 t においてメモリセル $\mathbf{c}_{n,t}$ を保持する．$\mathbf{o}_{n,t}$ は出力ゲートと呼ばれ，メモリセルの値をどの程度出力するかを調整する働きをする．この出力ゲートは，入力 $\mathbf{x}_{n,t}$，前の時刻 $(t-1)$ の出力 $\mathbf{h}_{n,t-1}$，メモリセル $\mathbf{c}_{n,t}$ を用いて次のように計算される．

$$\mathbf{o}_{n,t} = \sigma(\mathbf{W}_n^o\, \mathbf{x}_{n,t} + \mathbf{U}_n^o\, \mathbf{h}_{n,t-1} + \mathbf{V}_n^o\, \mathbf{c}_{n,t})$$

上式において，$\sigma(\cdot)$ はロジスティックシグモイド関数が用いられる．メモリセルは，入力ゲート $\mathbf{i}_{n,t}$ と忘却ゲート $\mathbf{f}_{n,t}$ を用いて次のように更新される．

$$\mathbf{c}_{n,t} = \mathbf{f}_{n,t} \circ \mathbf{c}_{n,t-1} + \mathbf{i}_{n,t} \circ \tilde{\mathbf{c}}_{n,t}$$

$$\tilde{\mathbf{c}}_{n,t} = g(\mathbf{W}_n^c\, \mathbf{x}_{n,t} + \mathbf{U}_n^c\, \mathbf{h}_{n,t-1})$$

入力ゲートは，入力をどの程度メモリセルに伝えるかを調整し，忘却ゲートはメモリセルの値をどの程度保持しておくかを調整する働きをする．それぞれのゲートは以下のように計算される．

$$\mathbf{i}_{n,t} = \sigma(\mathbf{W}_n^i\, \mathbf{x}_{n,t} + \mathbf{U}_n^i\, \mathbf{h}_{n,t-1} + \mathbf{V}_n^i\, \mathbf{c}_{n,t-1})$$

$$\mathbf{f}_{n,t} = \sigma(\mathbf{W}_n^f\, \mathbf{x}_{n,t} + \mathbf{U}_n^f\, \mathbf{h}_{n,t-1} + \mathbf{V}_n^f\, \mathbf{c}_{n,t-1})$$

標準的なリカレントユニットと異なり，LSTM は記憶を保持しておくか否かをゲートを用いることによって調整することが可能となっている．

2.3.2 GRU

GRU は (1) 式を以下のように変更したものである．

$$\mathbf{h}_{n,t} = (1 - \mathbf{z}_{n,t})\, \mathbf{h}_{n,t-1} + \mathbf{z}_{n,t}\, \tilde{\mathbf{h}}_{n,t}$$

GRU のユニットは，$t-1$ 時刻の出力 $\mathbf{h}_{n,t-1}$ と t 時刻の出力候補 $\tilde{\mathbf{h}}_{n,t}$ の線形補間となっている．線形補間に使われる $\mathbf{z}_{n,t}$ は更新ゲートと呼ばれ，情報の保持の調整を行う．$\mathbf{z}_{n,t}$ は以下のように計算される．

$$\mathbf{z}_{n,t} = \sigma(\mathbf{W}_n^z)\, \mathbf{x}_{n,t} + \mathbf{U}_n^z\, \mathbf{h}_{n,t-1}$$

出力候補の $\tilde{\mathbf{h}}_{n,t}$ はリセットゲート $\mathbf{r}_{n,t}$ を用いて以下のように計算される．

$$\tilde{\mathbf{h}}_{n,t} = g(\mathbf{W}_n\, \mathbf{x}_{n,t} + \mathbf{U}_n\, (\mathbf{r}_{n,t} \circ \mathbf{h}_{n,t-1}))$$

リセットゲートは，1 時刻前の出力をどの程度忘却するかを調整する役割を担い，更新ゲートと同様，以下のように計算される．

$$\mathbf{r}_{n,t} = \sigma(\mathbf{W}_n^r)\, \mathbf{x}_{n,t} + \mathbf{U}_n^r\, \mathbf{h}_{n,t-1}$$

GRU は LSTM と異なり，記憶を保持するための明示的なメモリセルを持たず，更新・リセットゲートを用いることで入出力情報の保持・忘却を調整する．

2.4 モデルの学習

Collobert ら [5] と同様に，以下の誤差関数を最小化することによって，ネットワークのパラメータ集合 θ を学習する．

$$E(\theta) = -\sum_{d=1}^{D} \log P(\mathbf{Y}_d|\mathbf{X}_d;\theta) + \frac{\lambda}{2}||\theta||^2$$

上式において，D は学習サンプル数，$\mathbf{X}$ は入力文の単語列，$\mathbf{Y}$ は正解のタグ列を表している．学習の詳細は 3.1 で述べる．

3 実験

述語項構造解析における $tanh$，LSTM，GRU ユニットの効果を調査するため，CoNLL-2005 Shared Task で用いられた訓練・開発・評価データを用いて評価実験を行った．

3.1 実験設定

すべてのモデルの実装は深層学習ライブラリ Theano[1] を利用し，CPU(Intel 6 Core Xeon E5-4617) で実行した．エポック数は 100 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．パラメータの最適化は，ミニバッチ (バッチサイズ = 8) を利用した確率的勾配降下法 (SGD) で行った．学習係数は Adam[8] を用いて自動調整した．各ハイパーパラメータは，Zhou ら [10] の実験設定を参考に以下のように選んだ．

単語ベクトル ：SENNA[5] によって事前に学習された 50 次元のベクトル[3]を用いた．

ユニットのパラメータの次元 ：[10] における実験で最も良い結果を出した 128 次元に設定．

パラメータ初期値 ： パラメータ行列の値として，$[-0.08, 0.08]$ から一様分布に従ってサンプリングした値を設定した．

正則化項 ：正則化項のハイパーパラメータ λ は [0.01, 0.001, 0.0001] の中から開発データの精度が最大となるものを選んだ．

3.2 実験結果

3.2.1 解析精度

表 1 に RNN 層の深さごとの予測結果の精度 (P)，再現率 (R)，F 値 (F) を示した．LSTM と GRU は各 RNN 層の深さで同等の結果を示しており，RNN 層の深さが増すごとに結果が向上している．一方，$tanh$ は LSTM・GRU と比べ結果が劣っている．RNN 層の深さが 2 の場合は 1 より向上しているが，4 の場合は下がっており，層が深くなった場合にパラメータが上手く学習されていない．

表 2 に，各ユニットにおいて，最も結果の良かった RNN 層数 ($tanh$ は 2，LSTM・GRU は 4) における各ラベルの解析結果を示す．LSTM と GRU は，各ラベルの解析結果も同等の結果となっている．

表 3: 1 エポックの学習にかかった時間 (秒数)

	RNN 層数		
	1	2	4
$tanh$	1361	1480	1716
LSTM	1943	2585	3685
GRU	1763	2054	2771

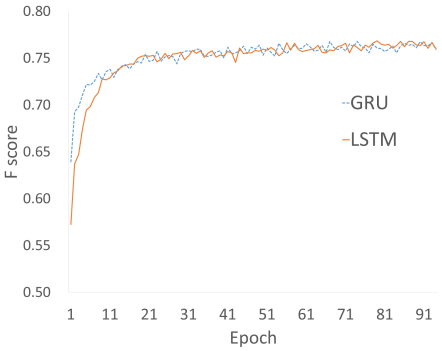

図 3: 開発データにおける GRU/LSTM の F 値の遷移

3.2.2 学習にかかる実行時間

表 3 に，各ユニットが 1 エポックの学習にかかる実行時間を示す．各ユニットのパラメータ数は，「$tanh$ < GRU < LSTM」，となっているため，その影響が学習にかかる実行時間に反映され，$tanh$，GRU，LSTM の順に時間がかからず，高速に学習が行えている．また，表 3 が示すように，GRU と LSTM は学習の収束がほぼ同等の速さであるため，実行速度の速い GRU の有用性が示唆される．

4 おわりに

本研究では，リカレントニューラルネットワークにおける，$tanh$，LSTM，GRU ユニットの比較評価実験を，述語項構造解析タスクに対して行った．標準的な $tanh$ ユニットでは上手く学習できない深層両方向型リカレントニューラルネット構造も，LSTM・GRU ユニットでは適切にパラメータの学習が進むことが分かった．また，GRU は LSTM と同等の性能であり，かつ，学習をより早く終えることができるという利点が明らかになった．今後の課題として，教師ラベルありのデータに加え，ラベルなし大規模データを効率的に利用する手法の考案などが挙げられる．

[3] http://ronan.collobert.com/senna/

表 1: 述語項構造解析結果

	RNN 層	開発			評価		
		P	R	F	P	R	F
$tanh$	1	63.98	50.96	56.73	65.40	51.61	57.69
	2	67.69	62.29	64.88	68.25	61.74	64.84
	4	65.74	56.93	61.02	66.81	57.70	61.92
LSTM	1	68.15	58.97	63.23	69.43	59.25	63.94
	2	75.02	73.71	74.36	74.99	72.78	73.86
	4	76.88	**76.24**	**76.56**	76.53	**75.27**	**75.89**
GRU	1	68.73	58.42	63.16	68.71	57.81	62.79
	2	74.75	72.86	73.79	75.03	72.56	73.77
	4	**76.99**	76.06	76.52	**76.70**	75.00	75.84

表 2: 各ラベルの解析結果

	$tanh$			LSTM			GRU		
	P	R	F	P	R	F	P	R	F
A0	80.12	76.24	78.13	**86.88**	87.26	**87.07**	86.20	**87.89**	87.04
A1	64.29	67.12	65.68	**76.08**	**77.60**	**76.83**	75.97	76.92	76.45
A2	47.63	37.55	41.99	**61.14**	**57.84**	**59.44**	59.93	57.84	58.87
A3	47.37	14.59	22.31	58.10	32.97	42.07	**75.00**	**35.68**	**48.35**
AM-ADV	48.01	38.98	43.03	64.62	47.00	54.42	**69.33**	**48.07**	**56.78**
AM-DIS	63.64	47.08	54.12	68.89	**72.51**	**70.66**	**70.29**	69.88	70.09
AM-LOC	**62.79**	18.08	28.08	53.95	**53.35**	**53.65**	51.75	52.68	52.21
AM-MNR	47.29	26.87	34.27	**58.46**	**50.22**	**54.03**	53.93	45.37	49.28
AM-MOD	94.83	94.24	94.53	95.74	98.13	96.92	**95.76**	**98.60**	**97.16**
AM-NEG	97.72	91.79	94.66	95.44	**97.14**	96.28	**97.83**	96.79	**97.31**
AM-TMP	60.61	59.30	59.95	74.23	73.98	74.10	**76.81**	**75.15**	**75.97**
R-A0	87.83	81.12	84.34	**91.39**	89.56	**90.47**	89.33	**90.76**	90.04
R-A1	71.08	66.67	68.80	80.65	**84.75**	**82.64**	**84.62**	74.58	79.28

参考文献

[1] Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, James Bergstra, Ian J. Goodfellow, Arnaud Bergeron, Nicolas Bouchard, and Yoshua Bengio. Theano: new features and speed improvements. Deep Learning and Unsupervised Feature Learning NIPS 2012 Workshop, 2012.

[2] Xavier Carreras and Lluis Marquez. Introduction to the conll-2005 shared task: semantic role labeling. In *Proceedings of CoNLL*, 2005.

[3] Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of EMNLP*, 2014.

[4] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv: 1412.3555*, 2014.

[5] Ronan Collobert, Jason Weston, Leon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. Natural language processing (almost) from scratch. In *Journal of Machine Learning Research*, 2011.

[6] Alan Graves, Navdeep Jaitly, and Abdel-rahman Mohamed. Hybrid speech recognition with deep bidirectional lstm. In *Automatic Speech Recognition and Understanding (ASRU), 2013 IEEE Workshop*, 2013.

[7] Sepp Hochreiter and Jurgen Schmidhuber. Long short-term memory. In *Neural Computation*, 2013.

[8] D.P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv: 1412.6980*, 2014.

[9] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. In *Proceedings of ICML*, 2013.

[10] Jie Zhou and Wei Xu. End-to-end learning of semantic role labeling using recurrent neural networks. In *Proceedings of ACL-IJCNLP*, 2015.