The 33rd Annual Conference of the Japanese Society for Artificial Intelligence, 2019

Pointer Networksを用いた文内述語項構造解析

Predicate argument structure analysis with Pointer Networks

髙橋啓吾　大森光　小町守
Keigo Takahashi　Hikaru Omori　Mamoru Komachi

首都大学東京
Tokyo Metropolitan University

Recently, neural network models are used in intra-sentential predicate argument structure analysis (PASA). However, the sequence labeling approach may assign the same case for multiple arguments. Thus, we propose a novel PASA method by using the Pointer Networks and alleviate the problem of multiple case assignments.

1　2　3　4
私が**友人**に**お餅**を**あげる**と、
5　6
友人はその場で**食べ**た.

述語	ガ格	ヲ格	ニ格
あげる（4）	私（1）	お餅（3）	友人（2）
食べる（6）	友人（5）	お餅［ゼロ照応］（3）	-

図 1: 文内ゼロ照応を含む述語項構造解析の例

1. はじめに

述語項構造解析とはテキスト中に存在する述語とその項との意味構造を認識するタスクである．述語項構造解析は機械翻訳や含意関係認識など複雑な文脈の解析を必要とするタスクに有用である．そのうち，文内述語項構造解析は述語と同一文内に存在する項のみを同定する述語項構造解析タスクである．述語項構造解析では述語に対するガ格，ヲ格，ニ格の必須格や任意格を同定する．述語項構造解析の例を図 1 に示す．図 1 は必須格が省略されている文内ゼロ照応を含むものである．数字は単語の区別のため記載した．本研究では，文内の必須格の述語項構造解析を対象とする．

先行研究では，Recurrent Neural Networks (RNN) を用いた end-to-end なモデルが優れた解析精度を示しているが，これらは述語項構造解析を系列ラベリングタスクとして取り扱っている [Matsubayashi 18, Ouchi 17]．述語項構造解析を系列ラベリングで解く場合，文内に共参照関係や並列関係等の特別な文構造が存在しない場合でも，複数の項候補に対して同じ必須格をラベリングしてしまうという問題が起こりうる．

本研究では，この問題を解決するために述語項構造解析に対して Pointer Networks [Vinyals 15] を応用した手法を提案する．Pointer Networks は入力シーケンスの特定の位置を指すポインタを出力する Neural Network を用いたモデルである．本タスクは入力文の中から述語に対応した必須格の単語を同定するタスクであるので，Pointer Networks を用いて入力文の中から項の位置を表すポインタを出力し，予測に用いることで，同じ必須格を複数出力してしまう問題を解決する．

連絡先: 髙橋啓吾，takahashi-keigo@ed.tmu.ac.jp

2. 系列ラベリングによる述語項構造解析

本研究は文内述語項構造解析タスクに取り組むものである．実験の設定は Ouchi ら [Ouchi 17] のものに合わせ，述語である単語は既知，述語に対応した必須格である単語は未知とした．文内係り受け，文内ゼロ照応の推定を対象とし，文間ゼロ照応，外界ゼロ照応は対象としなかった．同様に，どの単語が述語であるかは既知，必須格である単語は未知として述語に対応した必須格を同定するタスクとした．また，一つの文に対して複数の述語が存在する場合もあるが，述語に対応する必須格はこのタスク設定において高々一つずつである．

本研究では，先行研究 [Matsubayashi 18, Ouchi 17] で行われている系列ラベリングによる述語項構造解析手法をベースラインとして，我々の提案手法である Pointer Networks を応用した手法と比較する．それぞれの手法で用いるモデルは入力層，RNN 層，出力層から構成される．ここではまずベースラインとした系列ラベリングを用いた述語項構造解析手法について記載する．図 2 にベースラインの構成を示す．

2.1 入力層

本研究では素性として述語の表層形，項候補の表層形と合わせて，述語と項候補の単語換算距離，述語を含む文節と項候補を含む文節の文節換算距離，項候補が述語かどうかを用いた．

1. 述語の表層形
2. 項候補の表層形
3. 述語と項候補の単語換算距離（述語単語の文内位置 − 項候補単語の文内位置）
4. 述語を含む文節と項候補を含む文節の文節換算距離（述語単語の文内文節位置 − 項候補単語の文内文節位置）
5. 項候補が述語かどうか

ここで項候補は文の各単語 $w_t \in [w_1, \cdots, w_T]$ とし，述語はその中の一つの単語である．述語の表層形と文中全ての単語を項候補として一つずつ組み合わせ，さらに項候補に対応した追加の素性 3〜5 を入力する．これらの素性は変換行列を介してベクトル化した．特に，1. 述語の表層形，2. 項候補の表層形のベクトル化には学習済み Word Embedding[*1] を用いた．

それぞれをベクトル化したものを結合し，x_t と表す．t は時系列の時刻とする．

*1 http://www.asahi.com/shimbun/medialab/word_embedding

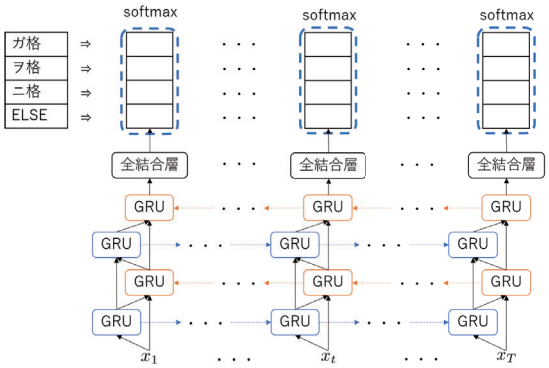

図 2: 系列ラベリングによる述語項構造解析モデル

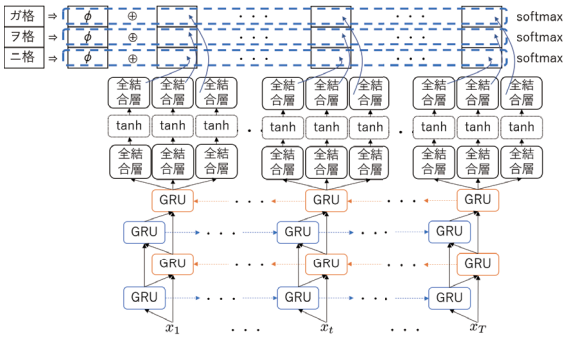

図 3: Pointer Networks による述語項構造解析モデル

2.2 RNN 層

RNN 層は 2 層の Stacked bi-directional Gated Recurrent Unit (GRU) [Cho 14] を用いた．また，Ouchi ら [Ouchi 17], Matsubayashi & Inui [Matsubayashi 18] の研究に従い，Residual 接続を用いる．各層のインデックスを $l \in [1, \cdots, L]$ とした場合の隠れ状態を $h_t^l \in \mathbb{R}^{d^h}$ とする．d^h は隠れ層の次元数である．これらを用いると各隠れ層からの出力は以下となる．

$$h_t^l = \begin{cases} \mathrm{gru}^l(h_t^{l-1}, h_{t-1}^l) & (l = odd) \\ \mathrm{gru}^l(h_t^{l-1}, h_{t+1}^l) & (l = even) \end{cases} \quad (1)$$

ここで，$\mathrm{gru}^l(\cdot)$ は l 層目の GRU であり，$h_t^0 = x_t$ である．

2.3 出力層

出力層は，RNN 層の最終層の隠れ層のテンソル h_t^L を全結合層の入力にとる．そして，その出力に softmax 関数をかけて確率ベクトル o_t^{base} を算出する．これらの処理はまとめて以下のように表される．

$$o_t^{base} = \mathrm{softmax}(W_1^{base} h_t^L + b_1^{base}) \quad (2)$$

ここで，$W_1^{base} \in \mathbb{R}^{4 \times d^h}$ は出力層の重み，b_1^{base} は出力層に用いた全結合層のバイアス項を表す．

また，ベースラインの出力層では必須格 [ガ格, ヲ格, ニ格, ELSE] の 4 種類に対応する推定確率を出力する．ELSE は述語項候補がどの必須格にも該当しない場合の要素である．

3. Pointer Networks による述語項構造解析

既存の系列ラベリング手法を用いた場合に生じる，複数の項候補に同じ必須格ラベルを割り当ててしまうという課題に対して，必須格をそれぞれ高々一つずつ割り当てるようなモデルを提案する．Pointer Networks を用いた提案手法では出力層に Pointer Networks [Vinyals 15] を応用した手法を取り入れた．入力層，RNN 層はベースラインと同じものを用いた．

3.1 出力層

提案手法ではベースラインモデルのように項候補ごとに必須格の推定確率を算出するのではなく，述語に対してどの項候補が最も必須格らしいかを表す推定確率 o_t^{ptr} を算出するようにした．図 3 に提案手法の構成を示す．以下に算出式を示す．

$$c_t^{ptr,i} = \tanh(W_1^{ptr,i} h_t^L + b_1^{ptr,i}) \quad (3)$$

$$s_t^{ptr,i} = W_2^{ptr,i} c_t^{ptr,i} + b_2^{ptr,i} \quad (4)$$

$$o^{ptr,i} = \mathrm{softmax}(\phi \oplus s_1^{ptr,i} \oplus \ldots \oplus s_T^{ptr,i}) \quad (5)$$

ここで，添字の i はそれぞれの必須格を表し，出力層は必須格ごとに用意した．右下の添字は出力層内の全結合層の layer 数を表す．$W_1^{ptr,i} \in \mathbb{R}^{d_1^{ptr} \times d^{hidden}}$ は 1 層目の全結合層の重み，d_1^{ptr} は 1 層目の全結合層の次元数，b_1^{ptr} は 1 層目の全結合層のバイアス項を表す．同様に，$W_2^{ptr,i} \in \mathbb{R}^{1 \times d_1^{ptr,i}}$ は 2 層目の全結合層の重み，d_2^{ptr} は 2 層目の全結合層の次元数，$b_2^{ptr,i}$ は 2 層目の全結合層のバイアス項を表す．2 層目からの出力はスカラー値である．ϕ は，対象となる必須格が存在しない場合に指す要素として用意したスカラー値であり，ϕ は常に 0.0 とした．

訓練時には正解の必須格の位置を教師あり学習のラベルと合わせて学習するため，モデルは必須格の位置さえ出力できればよいが，最終的に各必須格の項候補を決定するにはデコーディングを行う必要がある．この時，必須格ごとに他の必須格の予測結果を考慮せずにそれぞれの必須格で独立して項候補を選択していった場合，一つの項候補に複数の必須格ラベルを付与する場合が起こりうる．

そこで，今回はそれを防ぎ，一つの項候補には一つの必須格を割り当てるために制約付きデコーディングとして global-argmax による手法を用いた．擬似コードを Algorithm 1，7〜15 行目に示す．全ての必須格に対する各項候補の確率ベクトルを結合して，それらの中で値の高いものから必須格として割り当てていく．このとき，既に割り当てられた項候補は別の必須格の項候補に割り当てないようにした．同様に既に割り当てる項候補が決まっている必須格には複数の項候補を割り当てないようにした．それぞれ，値が高いものから順に割り当てを行なった．

文の中から最適な項候補をそれぞれの必須格に対して独立して選択する操作は，既存の系列ラベリングモデルでは単語に対して必須格ごとの推定確率を算出しているため実現できないが，本提案手法では Pointer Networks モデルを用いて文長方向の推定確率を算出できるようになったことで global な argmax を取る操作が可能となった．

4. 実験

4.1 実験設定

データは NAIST テキストコーパス 1.5 を用いた．訓練, 開発, 評価データへの分割は Taira ら [Taira 08] の方法に従った．

Algorithm 1 デコードアルゴリズム

Given:
s_{wo}: 各項候補に対するヲ格の推定確率ベクトル
s_{ga}: 各項候補に対するガ格の推定確率ベクトル
s_{ni}: 各項候補に対するニ格の推定確率ベクトル
l: 上記ベクトル長（ヲ，ガ，ニで同じ）

Do (global argmax):
$ret = [0, 0, 0]$
$s = \text{concat}(s_{wo}, s_{ga}, s_{ni})$
$s, type1, pos1 = \text{get\_index}([], [], s, l)$
$ret[type1] = pos1$
$s, type2, pos2 = \text{get\_index}([type1], [pos1], s, l)$
$ret[type2] = pos2$
$s, type3, pos3 = \text{get\_index}([type1, type2], [pos1, pos2], s, l)$
$ret[type3] = pos3$

---

procedure get_index($types, poses, s, l$)
    $index = \text{argmax}(s)$
    while is_on_restricted($types, poses, index, l$) **do**
        $s[index] = -\infty$
        $index = \text{argmax}(s)$
    end while
    $type = \text{int}(index \div l)$
    $pos = index \mod l$
    return $s, type, pos$
end procedure

---

procedure is_on_restricted($types, poses, index, l$)
    $t = \text{int}(index \div l)$
    $p = index \mod l$
    if $p == 0$ **then** return False
    end if
    if $t \in types$ **then** return True
    end if
    if $p \in poses$ **then** return True
    end if
    return False
end procedure

詳細な数は表 1 に記載する．単語境界，文節境界は NAIST テキストコーパスに付与されているものを使用した．

モデルは pytorch ver.1.0.0 で訓練，評価を行った．それぞれのモデルで利用したハイパーパラメータおよび探索範囲を表 2 に示す．ハイパーパラメータの選定には pytorch ver.0.4.1.post2, hyperopt [Bergstra 13] を用いた．hyperopt は提案手法は ver.0.1，ベースラインは ver.0.1.1 を用いた．

4.2 結果

結果を表 3 に示す．スコアは全て F1 値である．本研究で算出したスコアはモデルの重みパラメータを生成するシードの異なるモデルを 5 つ作成し，その平均と分散を示した．Dep は述語と項候補が直接の係り受け関係にあるものを表し，Zero は文内ゼロ照応によって項が省略されたものを表す．

先行研究 [Matsubayashi 18] では各必須格ごとのスコアが提示されていないが，本研究は全体，Dep の ALL において M&I [Matsubayashi 18] の手法を，各必須格に対しても Ouchi ら [Ouchi 17] の手法を上回るスコアを出すことができた．

項目	訓練	開発	評価
文数	23,218	4,628	8,816
述語数	62,489	12,724	23,981
ガ格の数 [a]	37,615/11,546	7,520/2,552	14,230/4,764
ヲ格の数 [a]	24,997/1,802	5,105/394	9,532/783
ニ格の数 [a]	5,855/359	1,637/112	2,547/211

[a] Dep/Zero の数を記載した

表 1: NAIST テキストコーパス 1.5 の文内の述語項の数

本来、百億円は自分で**借り**たものであり、
自ら返済すべきカネではないのか。

分類	ガ格	ヲ格	ニ格
正解	自分	もの	ー
ベースライン	ー	円、もの	ー
提案手法	ー	もの	ー

図 4: 格の出現を予測できなかった例

両親の選択は**誤っ**ていなかったようだ。

分類	ガ格	ヲ格	ニ格
正解	両親	選択	ー
ベースライン	ー	ー	ー
提案手法	選択	ー	ー

図 5: 格の出現は予測できたが，単語を間違えた例

5. 考察

まず各モデルの正誤の集計結果について述べる．結果はモデルパラメータ初期値の生成シードを変えたモデルを 5 つ作成し，そのうち 4 つ以上のモデルが同じ回答を出したものを用いた．正答件数はベースライン 16,037 件に対して，提案手法 16,670 件であった．同一の必須格に複数の項候補を割り当てたものはベースラインでは 76 件であったが，それらは提案手法で解消された．

提案手法の誤答例を図 4，5 に示す．図 4 は格の出現を予測できなかったものの例である．述語「借り」に対して，正解はガ格「自分」とヲ格「もの」である．ベースラインは「円」と「もの」をヲ格と出力し，ガ格を出現できておらず，また一つの必須格に複数の項候補を割り当てている．一方で，提案手法では「もの」をヲ格と正しく出力しているが，ベースライン同様にガ格の出現を予測することができなかった．同類の誤りは 2,147 件あった．また，図 5 は格の出現は予測できたが，単語を間違えたものの例である．述語「誤っ」に対して，ガ格の正解は「両親」であるが，提案手法は「選択」と誤答している．同類の誤りは 3,071 件あった．

これらは述語に関する知識（例えばガ格に有生性のものがきやすい述語）や，文脈に関する情報が足りていない事が要因であると考えられる．図 4 の述語「借り」は，主語に有生性のものがきやすい述語であるが，そのような格フレームに関する知識はモデルに与えていない．また，図 5 はガ格の「両親」は述語に直接かかっていないため難易度が高く，格フレームなどの選択選好相当の知識や文全体を考慮に入れた手法を用いる必要があると考える．特に述語の知識をモデルに組み込む研究は Neural Network 以前に盛んに行われており，それらと Neural Network を組み合わせた研究を進める必要があると考える．

パラメータ	ベースライン	提案手法	探索範囲
最適化手法	RMSprop	SGD	Adam, Adadelta, Adagrad, SGD, RMSProp から選択
学習率	0.0001	0.1	0.2, 0.1, 0.05, 0.01, 0.005, 0.001, 0.0001 から選択
バッチサイズ	10	16	2 から 200（ベースライン）, 256（提案手法）を 2 刻み
出力層サイズ: d_1^{ptr}	-	96	64 から 256 を 16 刻み
勾配クリップ	10	5	0 から 10 を 1 刻み
ドロップアウト率	0.2	0.3	0.0 から 0.7 を 0.1 刻み

表 2: ハイパーパラメータ

	全体		Dep				Zero			
手法	ALL	SD	ALL	ガ格	ヲ格	ニ格	ALL	ガ格	ヲ格	ニ格
Ouchi+ 17 Single_Seq	81.15	-	88.10	88.32	93.89	65.91	46.10	49.51	35.07	9.83
M & I 18 BASE	83.39	±0.13	89.90	-	-	-	54.37	-	-	-
Ouchi+ 17 Multi_Seq	81.42	-	88.17	88.75	93.68	64.38	47.12	50.65	32.35	7.52
M & I 18 MP_POOL_SELFATT	83.94	±0.12	90.26	90.88	94.99	67.57	**55.55**	**57.99**	**48.9**	**23**
ベースライン	83.70	±0.28	90.31	90.65	94.79	**72.09**	52.53	54.98	45.21	11.09
提案手法	**84.37**	±0.18	**90.73**	**91.01**	**95.04**	72.01	54.06	56.66	45.16	9.38

表 3: 評価データに対する F1 スコア（ベースライン，提案手法は 5 モデルの平均値）

6. 関連研究

文内述語項構造解析の研究はこれまでも盛んに行われてきている．Taira ら [Taira 08] は，ガ格，ヲ格，ニ格のそれぞれの必須格ごとにモデルを用意した手法を提案した．Yoshikawa ら [Yoshikawa 11] は，Markov Logic という統計的関係学習の枠組みで項の間の関係性を考慮した手法を提案した．Ouchi ら [Ouchi 15] は，二部グラフを用いて複数の述語と項の関係性を考慮しつつ解析する手法を提案した．本研究ではこれらの研究とは異なり，必須格ごとにモデルを構成するのではなく，出力層を除いて全ての必須格を同じモデルで構成した．また，Pointer Networks を用いて項の関係性を表現した．

今日では，本研究と同様に Neural Network を用いた手法が盛んに用いられている．Ouchi ら [Ouchi 17] は，ある 1 つの述語に対する述語項構造情報を別の述語の構造解析に役立てるために，複数の述語を同時に考慮する Grid Recurrent Neural Networks (Grid RNN) を用いた手法を提案した．また，Matsubayashi & Inui [Matsubayashi 18] は，Ouchi ら [Ouchi 17] の研究を更に発展させ，Grid RNN に対して self-attention を適用した手法を提案した．

本研究ではこれらの研究とは異なり，系列ラベリングで解く場合のように系列中の各単語がどの必須格らしいかではなく，系列の中からどの単語が最も必須格らしいかを推定するタスクとして，述語項構造解析を解く．また，先行研究 [Ouchi 17, Matsubayashi 18] では複数の述語の関係を考慮に入れたモデルが提案されているが，本研究では複数の述語の関係は考慮に入れていない．

7. おわりに

本研究では系列ラベリング手法で問題であった複数の項に同じ必須格を割り当ててしまうという問題に対し，Pointer Networks と制約付きデコーディングを組み合わせた手法を用いることで解決することができ，同時に F1 値も向上した．

一方で，提案手法でも解決できていない例も確認された．これらは述語の格フレームなどの選択選好を考慮に入れたモデルや，文脈や文全体を考慮に入れたモデルを用いることで解決できるのではないかと考える．これらを今後の課題としたい．

参考文献

[Bergstra 13] Bergstra, J., Yamins, D., and Cox, D.: Making a Science of Model Search: Hyperparameter Optimization in Hundreds of Dimensions for Vision Architectures, in *Proceedings of ICML*, pp. 115–123 (2013)

[Cho 14] Cho, K., Merriënboer, van B., Gülçehre, Ç., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y.: Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation, in *Proceedings of EMNLP*, pp. 1724–1734 (2014)

[Matsubayashi 18] Matsubayashi, Y. and Inui, K.: Distance-Free Modeling of Multi-Predicate Interactions in End-to-End Japanese Predicate Argument Structure Analysis, in *Proceedings of COLING*, pp. 94–106 (2018)

[Ouchi 15] Ouchi, H., Shindo, H., Duh, K., and Matsumoto, Y.: Joint Case Argument Identification for Japanese Predicate Argument Structure Analysis, in *Proceedings of the ACL–IJCNLP*, pp. 961–970 (2015)

[Ouchi 17] Ouchi, H., Shindo, H., and Matsumoto, Y.: Neural Modeling of Multi-Predicate Interactions for Japanese Predicate Argument Structure Analysis, in *Proceedings of ACL*, pp. 1591–1600 (2017)

[Taira 08] Taira, H., Fujita, S., and Nagata, M.: A Japanese Predicate Argument Structure Analysis Using Decision Lists, in *Proceedings of EMNLP*, pp. 523–532 (2008)

[Vinyals 15] Vinyals, O., Fortunato, M., and Jaitly, N.: Pointer Networks, in *Proceedings of NIPS*, pp. 2692–2700 (2015)

[Yoshikawa 11] Yoshikawa, K., Asahara, M., and Matsumoto, Y.: Jointly Extracting Japanese Predicate-Argument Relation with Markov Logic, in *Proceedings of IJCNLP*, pp. 1125–1133 (2011)