反復改良法を用いた日本語述語項構造解析

宮脇 峻平 [1]　清野 舜 [2,1]　松林 優一郎 [1,2]
今野 颯人 [1]　高橋 諒 [1,2]　大内 啓樹 [2,1]　乾 健太郎 [1,2]
[1] 東北大学　[2] 理化学研究所
{miyawaki.shumpei, ryuto, ryo.t, inui}@ecei.tohoku.ac.jp
{shun.kiyono, hiroki.ouchi}@riken.jp
y.m@tohoku.ac.jp

1 はじめに

述語項構造解析は自然言語処理における意味解析タスクの一つであり，文章内の述語に対して，「誰が，何を」というような格関係を持つ項を同定する．例えば以下，

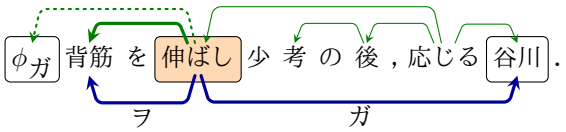


図1: 係り受け関係と述語項関係 例文上部の矢印は係り受け関係を，下部の矢印は述語「伸ばし」に関する格関係をそれぞれ表す．なお $\phi_{ガ}$ はゼロ代名詞を表し，項「谷川」を参照する．（NTC1.5: 950112-0140-950112199.ntc）

という文に対して，「伸ばし」という述語に着目する．このとき，主格（ガ格）には「谷川」，対格（ヲ格）には「背筋」という項が対応する．いま，述語「伸ばし」と項「背筋」間には直接的な係り受け関係が存在する．このような述語と項間の関係を DEP と定義する．一方で項「谷川」と述語の間には直接的な係り受け関係は存在せず，$\phi_{ガ}$ で示されるように項が省略されている．このように省略された項はゼロ代名詞と呼ばれる．ゼロ代名詞の照応解析を ZERO と定義する．

述語項構造解析の特徴に，係り受け関係によって解析難易度が大きく異なる点がある．例えば，ZERO の解析は DEP に比べて困難であることが知られている．ZERO では述語と項の間に直接的な係り受け関係が存在せず，構文が比較的複雑になるためである．実際，述語と項が同一文内に存在する格関係を解析対象とした際に，DEP と ZERO の解析精度を比較すると，DEP での F_1 値は 91% 程度であるのに対して，ZERO での F_1 値は 58% 程度に留まることが報告されている [4]．

我々はゼロ照応解析をはじめとする，予測が困難とされる格関係の解析に焦点を当てる．我々の出発点は easy-first アプローチ [1] である．具体的には，容易に解析可能である格関係の情報を，困難な格関係の予測の「手がかり」として用いることを試みる．手がかりが有効的に作用する例として，先の例文の解析について考える．いま，ガ格に対応する「谷川」という項は ZERO に分類される．また，通常「伸ばす」という述語には様々な格フレームの候補が挙げられる．例えば「企業が売り上げを伸ばす」や「優しさが成長を伸ばす」などのように，ガ格に割り当たる項は，必ずしも「谷川」のような，人に相当する項が選択されるとは限らない．そのため，ガ格に相当するような項候補は複数存在する．いま，ガ格の項を予測する際に，「伸ばす」という述語に対して，「背筋」というヲ格が既に割り当てられていた場合を考える．このとき，「（背筋を）伸ばす」という述語句に対して，ガ格には人に相当する項が割り当たる，ということが容易に予測できると考えられる．

Easy-first アプローチの実現に向けた第一歩として，本研究は反復改良法 [3] を用いたモデルを提案する．反復改良法では，モデルは一つの事例に対して繰り返し（反復的に）予測を行う．このとき，毎回の予測において前回の予測結果を入力に取り入れる．具体的には，モデルの出力した事後確率を確信度とみなし，高い確信度で予測した項の情報を次ステップの入力に用いることで，予測の困難な格関係の予測が改善されることを期待する．実験では提案手法による F_1 値の向上は見られなかったものの，分析を通してその原因を明らかにする．

2 タスク設定

本研究では，日本語述語項構造解析で利用されるデータセットの一つである NAIST テキストコーパス 1.5 版（NTC1.5）[6] の注釈の仕様に基づき実験を行う．既存研究 [4] に習い，述語と項が同一文内に存在するものを解析対象とする．本稿のモデルは，文の単語列と対象述語の位置を入力として受け取り，ガ，ヲ，ニ格に対応する項を高々一つ出力する．評価の際は与えられた格関係と一致したものを正解とする．

3 ベースラインモデル

本節では，本研究のベースラインモデルである Matsubayashi ら [4] の `Base` ついて述べる．`Base` は，k 層の双方向 RNN に基づくモデルである．まず，入力とし

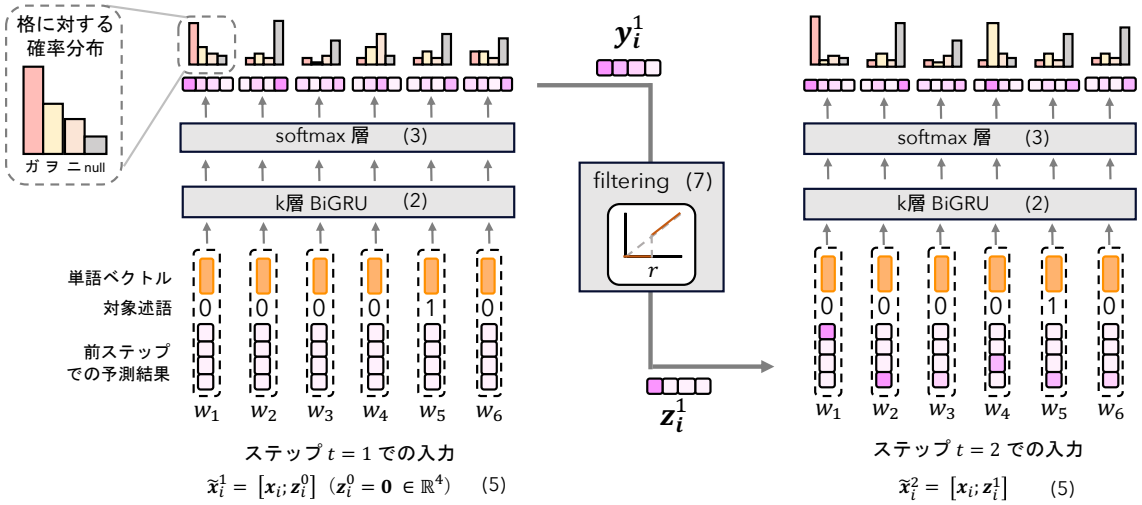


図2: 提案手法の概要：ステップ $t=0$ から $t=1$ の計算過程を表す．

て，単語列 $\boldsymbol{w} = w_1, \ldots, w_i, \ldots, w_I$ と，対象述語の位置 $j \in \{1, \ldots, I\}$ が与えられる．単語 w_i に対する入力素性 $\boldsymbol{x}_i \in \mathbb{R}^{D+1}$ は，単語 w_i の単語ベクトルと対象述語の位置 j を示すバイナリ値の結合である．

$$\boldsymbol{x}_i = [\boldsymbol{e}(w_i); p] \tag{1}$$

ここで，$[\boldsymbol{a}; \boldsymbol{b}]$ はベクトル $\boldsymbol{a}$ と $\boldsymbol{b}$ の結合を表す．また，$\boldsymbol{e}(w_i) \in \mathbb{R}^D$ は単語 w_i に対応する単語ベクトルを取得する関数で，D は単語ベクトルの次元数である．$p \in \{0, 1\}$ は $i = j$ のときに 1，それ以外は 0 となる．

いま，単語列 $\boldsymbol{w}$ に対応する入力素性の列を $\boldsymbol{X}$ で表す．つまり，$\boldsymbol{X} = \boldsymbol{x}_1, \ldots, \boldsymbol{x}_I$ である．入力 $\boldsymbol{X}$ を k 層双方向 RNN を用いて以下のようにエンコードする．

$$\boldsymbol{H}_k = \mathrm{BiRNN}(\boldsymbol{X}) \tag{2}$$

ここで $\boldsymbol{H}_k$ は第 k 層の隠れ層の系列で，$\boldsymbol{H}_k = \boldsymbol{h}_{1,k}, \ldots, \boldsymbol{h}_{I,k}$ である．$\boldsymbol{h}_{i,k} \in \mathbb{R}^H$ であり，H は隠れ層の次元を表す．また BiRNN は，層ごとに向きの変わる k 層双方向 RNN を表す[*1]．また RNN セルとして GRU を用いる．

次に，各時刻の隠れ層 $\boldsymbol{h}_{i,k}$ を softmax 層に入力し，確率分布を計算する．

$$\boldsymbol{y}_i = \mathrm{softmax}(\boldsymbol{W}_h \boldsymbol{h}_{i,k}) \tag{3}$$

ここで，$\boldsymbol{W}_h \in \mathbb{R}^{4 \times H}$ は重み行列である．また，$\boldsymbol{y}_i \in \mathbb{R}^4$ は単語 w_i が項となる確率を表す確率分布であり，それぞれの値は三つの格（ガ格，ヲ格，ニ格）と項ではないことを表す “null” に対応する．

訓練時には，正解ラベル $\boldsymbol{y}_i^*$ と予測結果 $\boldsymbol{y}_i$ との交差エントロピー誤差 $\ell(\boldsymbol{y}_i^*, \boldsymbol{y}_i)$ が最小となるようにモデルのパラメータを更新する．

$$\mathcal{L}(\mathcal{D}) = \frac{1}{|\mathcal{D}|} \sum_{(\boldsymbol{w}, j) \in \mathcal{D}} \sum_{i=1}^{I} \ell(\boldsymbol{y}_i^*, \boldsymbol{y}_i) \tag{4}$$

[*1] BiRNN の詳細については，文献 [4] を参照せよ

ここで $\mathcal{D}$ は訓練データを表す集合である．

推論時には，出力された確率分布から対象述語に対して格となる項を決定する．この際，それぞれの格に対して最大の確率値を持つ単語を対応する項として選択する．また，選択された項の確率値が 0.5 を超えない場合には，null として扱う．

4 提案手法

提案手法の概要を図2に示す．提案手法の核となるアイデアは以下の二つである．

- **自己反復**：同一モデルを用いて複数回の予測を行う．その際，各ステップの予測結果を次ステップの入力として与える．比較的簡単な格関係の予測結果が，困難な格関係の予測の手がかりとなることを期待する．
- **確信度に基づくフィルタリング**：予測結果を入力として用いる際に，確信度の低い予測はフィルタリングによって取り除く．

自己反復　我々の提案手法である自己反復では，同じモデルを使って合計 T 回の予測を行う．今，第 t ステップでの予測について考える．第3節で述べたベースラインモデルとの重要な違いは，入力に前ステップ $t-1$ の予測を用いる点である．いま，単語 w_i に対する入力素性 $\tilde{\boldsymbol{x}}_i^t$ は，式1で定義した $\boldsymbol{x}_i$ と，前ステップの予測情報から構成される実数ベクトル $\boldsymbol{z}_i^{t-1}$ の結合である．つまり，

$$\tilde{\boldsymbol{x}}_i^t = [\boldsymbol{x}_i; \boldsymbol{z}_i^{t-1}] \tag{5}$$

である．また，$\boldsymbol{z}_i^0$ はゼロベクトル $\mathbf{0}$ と定義する．その後，新しい入力素性列 $\tilde{\boldsymbol{X}}^t = \tilde{\boldsymbol{x}}_1^t, \ldots, \tilde{\boldsymbol{x}}_I^t$ を用いて，ベースラインと同様の演算を式2と3で行う．

モデルの学習にあたっては，正解ラベル $\boldsymbol{y}_i^*$ とステップ t での予測結果 $\boldsymbol{y}_i^t$ との交差エントロピー誤差 $\ell(\boldsymbol{y}_i^*, \boldsymbol{y}_i^t)$ を毎ステップ計算する．ステップ T が終了後，各ステッ

プの誤差を合計し，モデルのパラメータを更新する．

$$\mathcal{L}'(\mathcal{D}) = \frac{1}{|\mathcal{D}|} \sum_{(\boldsymbol{w},j)\in\mathcal{D}} \sum_{t=1}^{T} \sum_{i=1}^{I} \ell(\boldsymbol{y}_i^*, \boldsymbol{y}_i^t) \qquad (6)$$

確信度に基づくフィルタリング　ステップ $t-1$ での予測結果をステップ t の入力に取り入れるにあたって，いくつかの選択肢が考えられる．言い換えると，式5における実数ベクトル $\boldsymbol{z}_i^{t-1}$ の定義が複数考えられる．最も単純な手法としては，前ステップの予測結果 $\boldsymbol{y}_i^{t-1}$ をそのまま入力するということが挙げられる．つまり，$\boldsymbol{z}_i^{t-1} = \boldsymbol{y}_i^{t-1}$ である．しかしこの場合，確信度の低い予測結果が次ステップの予測においてノイズとして作用する可能性がある．

この問題に対処するため，我々はフィルタリング機構を提案する．フィルタリングは，確信度の高い予測結果のみを次のステップの入力として用いることを目的とする．本研究では，モデルの出力した事後確率をモデルの確信度と仮定する．

いま，次のように関数 filter を定義する．

$$v = \mathrm{filter}(u|r) = \begin{cases} u, & \text{if } r \le u \\ 0, & \text{otherwise} \end{cases} \qquad (7)$$

ここで $r \in \mathbb{R}$ は閾値として作用するハイパーパラメータである．モデルの出力 $\boldsymbol{y}_i^{t-1} \in \mathbb{R}^4$ の各要素に filter を適用し，次ステップの入力 $\boldsymbol{z}_i^{t-1} \in \mathbb{R}^4$ を得る．これは，各予測確率 $\boldsymbol{y}_i^{t-1}$ に対し，閾値 r を下回る値を 0 へ変換することで，低確信度の予測を削除することに相当する．

5　実験

5.1　実験設定

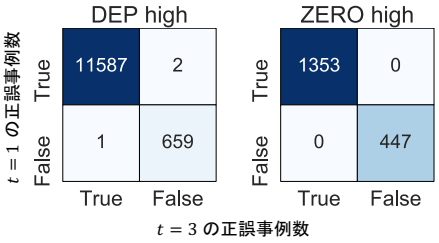
データセット・評価指標　実験には NTC1.5 を用いる．既存研究 [5] に従い，訓練セット，開発セットと評価セットに分割した．評価指標には，適合率，再現率と F_1 値の三つを用いた．各実験では，三つの異なるシード値を用いてモデルを学習し，平均値と偏差を報告する．

ハイパーパラメータ　単語埋め込み次元数 D および隠れ層の次元数 H を 256 とし，BiRNN の層数 k を 10，dropout 率を 0.1 とした．単語ベクトルの初期値には，Matsubayashi ら [4] と同様に，日本語 Wikipedia から学習した word2vec ベクトルを用いた[*2]．最適化には Adam[2] を用いた．学習率は α は 0.002 とし，ミニバッチサイズは 512 とした．自己反復モデルのステップ数は $T = 3$ とした．またフィルタリング機構の閾値 r は，開発セット上で調整を行い 0.5 に設定した．

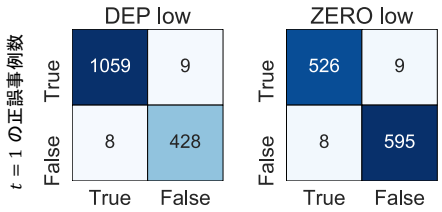
5.2　実験結果

ベースラインモデル（Base）と提案モデル（self-refine）の評価セット上での性能を表1に示す．まず，Base と self-refine（$t = 3$）を比較する．いま

[*2] 2016 年 9 月 1 日のダンプデータを用いた



(a) 提案モデルが高い確率値で予測した例



(b) 提案モデルが低い確率値で予測した例

図3: 提案モデルによる予測ステップでの正誤事例数の変化: 予測確率別の DEP と ZERO の正誤事例数．確率値の高低の境界値は 0.5 とする．

全体の F_1 値（ALL）に着目すると，ベースラインと提案手法はほぼ同等の値となり，提案手法による性能の改善は見られなかった．また ZERO においては，提案手法の F_1 値がベースラインよりも悪化した．反復改良の仕組みを取り入れることで，ZERO のような難しい問題の性能向上を狙ったが，期待通りの効果は得られなかった．次に提案手法におけるステップ間の性能に着目する．予測を繰り返した場合も，ALL，DEP や ZERO の全てで F_1 値に大きな変化は見られなかった．特にステップ $t = 1$ において，既にベースラインと同じ精度で予測できており，反復改良による予測結果の改善は確認できなかった．

5.3　分析

第5.2節の実験では，提案手法による性能の改善は見られなかった．本節では，提案手法の分析を行うことで将来的な改善案を考えたい．

フィルタリング機構の有効性　提案手法では，前ステップの予測を入力に取り入れる際に，フィルタリング機構を用いて高い確信度の予測結果のみを利用した．フィルタリング機構の効果を検証するため，閾値を $r = 0.0$ と設定し，フィルタを適用しない場合の実験を行った．結果を表2に示す．表より，フィルタリング機構を用いない場合 DEP と ZERO のいずれにおいても性能が低下した．モデルの性能が低下した理由として，フィルタリングによって本来除去されるはずの予測が，ノイズとして作用している可能性が示唆される．

反復改良による改善及び改悪事例数　確信度が低い予測が反復改良においてノイズとして作用している可能性を

表1: 評価データを用いたベースライン（Base）と提案手法（self-refine）の性能比較

モデル	ALL F_1	ALL 適合率	ALL 再現率	DEP F_1	DEP 適合率	DEP 再現率	ZERO F_1	ZERO 適合率	ZERO 再現率
Base	82.74 ± 0.24	86.38	79.40	89.69 ± 0.16	91.99	87.50	53.30 ± 0.58	59.33	48.40
self-refine ($t=1$)	82.77 ± 0.11	87.21	78.76	89.73 ± 0.11	92.38	87.23	52.51 ± 0.19	60.77	46.28
self-refine ($t=2$)	82.78 ± 0.12	87.18	78.82	89.75 ± 0.13	92.39	87.25	52.58 ± 0.24	60.65	46.46
self-refine ($t=3$)	82.78 ± 0.12	87.15	78.84	89.75 ± 0.12	92.37	87.28	52.55 ± 0.25	60.62	46.44

表2: フィルタリングの有無における F_1 値の比較

モデル		ALL	DEP	ZERO
Base		83.13 ± 0.19	89.91	**54.43**
self-refine $r=0.5$	($t=1$)	**83.22 ± 0.08**	89.92	54.05
	($t=2$)	83.21 ± 0.08	89.92	54.05
	($t=3$)	83.21 ± 0.09	**89.93**	54.04
self-refine $r=0.0$	($t=1$)	83.00 ± 0.08	89.81	53.64
	($t=2$)	83.00 ± 0.06	89.81	53.75
	($t=3$)	83.00 ± 0.06	89.81	53.74

検証するため，確信度別の正誤事例数に関する分析を行う．具体的には，モデルの出力を正誤で分類した混同行列を確信度別に分析する（図3）．縦軸は $t=1$ での予測を，横軸は $t=3$ での予測を表す．ここでは，モデルの出力した確率値が閾値 $r=0.5$ 以上のものを high，それ未満のものを low と定義する．

まず確信度に基づく予測結果の正解率から，モデルの予測を手がかりとして利用する際の影響について考える．図3より確率値の高い予測のほとんどは $t=1$ と $t=3$ の両方で True となっている．一方で確信度の低い予測については，$t=1$ と $t=3$ の両方で False となっている事例の割合が大きい．これは，確信度の低い予測の正解率が，確信度の高い予測の正解率よりも低いことを意味する．つまり，確信度の低い予測は手がかりとして信頼するべきではないと考えられる．フィルタリング機構を用いることで，信頼できない予測をうまく取り除くことができると考えられる．

また，図3からは提案手法で性能が改善しなかった原因も読み取れる．確率値によらず，$t=3$ だけが True となるような事例（自己反復による改善事例）数，および $t=3$ だけが False となるような事例（自己反復による改悪事例）数は，全て 10 件未満と低い値となった．このことから，提案法ではステップをまたいだ予測の変化はほとんど生じていない可能性が示唆される．表1において，提案手法で性能が改善していないこと，また $t=1$ と $t=3$ の間でほとんど F_1 値が変化していないのは，これが原因だと考えられる．

今後の改善案　将来的な改善案を二つ述べる．第一に，複数述語を考慮した反復改良が考えられる．本研究では一つの対象述語に閉じた問題に対して解析を行なった．これは，一つの対象述語の格関係間に難易度の差があることを暗黙的に仮定している．しかし実際には，難易度の差は述語内だけではなく，述語間にも存在すると考えられる．例えば「尋ねる」と「答える」などのように対義に当たる述語間では，しばしば動作主と被動作主に当たる項が共有される．ここでは，片方の述語の格関係を予測した結果を，項を共有するもう片方の述語の予測の手がかりとして用いる手法が考えられる．将来的には，複数述語を考慮する解析モデル [4] に反復改良の枠組みを取り入れることで予測精度の向上を試みたい．

次に，より精緻な easy-first アプローチの実現が考えられる．本研究のモデルは easy-first に解くことを狙いとしていたが，実際には予測の順番はモデルに任されていた．今後は格関係における難易度の違いを考慮しつつ，簡単な項から予測していくモデルを考えたい．

6　おわりに

本稿では，日本語述語項構造解析のための反復改良モデルを提案した．結果としてはベースラインと同程度の性能となった．分析では，反復改良モデルが自らの予測を更新できていないことが分かった．今後の展望として，反復的に予測を改良するために，異なる述語間での項の共有を考慮する方法や，より精緻に easy-first を取り入れる工夫などが必要であると考えられる．

謝辞　本研究は JSPS 科研費 JP19H0416,JP19K12112, JP19K20351 の助成を受けたものである．

参考文献

[1] Yoav Goldberg and Michael Elhadad. “An Efficient Algorithm for Easy-First Non-Directional Dependency Parsing”. In: *ACL*. 2010, pp. 742–750.
[2] Diederik Kingma and Jimmy Ba. “Adam: A Method for Stochastic Optimization”. In: *ICLR*. 2015.
[3] Jason Lee, Elman Mansimov, and Kyunghyun Cho. “Deterministic Non-Autoregressive Neural Sequence Modeling by Iterative Refinement”. In: *EMNLP*. 2018, pp. 1173–1182.
[4] Yuichiroh Matsubayashi and Kentaro Inui. “Distance-Free Modeling of Multi-Predicate Interactions in End-to-End Japanese Predicate-Argument Structure Analysis”. In: *COLING*. 2018, pp. 94–106.
[5] Hirotoshi Taira, Sanae Fujita, and Masaaki Nagata. “A Japanese Predicate Argument Structure Analysis using Decision Lists”. In: *EMNLP*. 2008, pp. 523–532.
[6] 飯田 龍 et al. “述語項構造と照応関係のアノテーション：NAIST テキストコーパス構築の経験から”. In: 自然言語処理 (2010), pp. 225–250.