処理途中での非文生成の回避を考慮した係り受け解析・語順整序・読点挿入の同時実行

荒木 駿介 [1]　大野 誠寛 [1]　松原 茂樹 [2]
[1] 東京電機大学 大学院未来科学研究科　[2] 名古屋大学 情報基盤センター
23fmi04@ms.dendai.ac.jp　ohno@mail.dendai.ac.jp
matsubara.shigeki.z8@f.mail.nagoya-u.ac.jp

概要

日本語において，語順や読点の使用法は比較的自由に書き手に任されるが，実際には選好が存在しているため，意味は伝わるものの読みにくい文が作成されることがある．本稿では，推敲支援のための要素技術として，Shift-Reduce アルゴリズムを拡張した，日本語文に対する係り受け解析・語順整序・読点挿入の同時実行手法を提案する．提案手法の特徴は，従来手法において処理途中の語順が非文になるという問題を回避するため，同じ文節に係ると決定された 2 文節間でのみ語順入替を行う点にある．

1　はじめに

日本語は語順が比較的自由であるが，実際には選好が存在しているため，文法的には間違っていないものの読みにくい語順を持った文が無意識に作成される．また読点についても同様に，その有無や位置によっては文が読みにくくなる．このような読みにくい文は，語順を整え，適切に読点を挿入し直せば読みやすくなる．

語順整序や読点挿入に関する研究は，推敲支援や文生成などに応用でき，いくつか行われている（例えば，[1, 2, 3, 4]）．その中でも係り受け解析・語順整序・読点挿入を同時実行する手法として，宮地ら [3] の Shift-Reduce アルゴリズムを拡張した手法がある．我々の先行研究 [4] では，宮地らの手法 [3] の操作選択において BERT[5] を用いる手法を提案し、大幅な精度の向上を達成している．しかし，両手法が採用しているアルゴリズムでは，隣接文節間における順序関係が逆であった場合にバブルソートのように順次，語順入替を行うため，処理途中において，一時的に非文が生成されることがある．その非文となる語順に基づいて，入力文とは意味が変わるような係り受け関係が誤って同定される場合があるなど，精度向上に向けて改善の余地がある．

そこで本稿では，処理途中で非文が生成されることを回避しつつ，係り受け解析・語順整序・読点挿入を同時実行する手法を提案する．具体的には，従来手法 [3, 4] の拡張 Shift-Reduce アルゴリズムにおいて，同じ文節に係る 2 文節間でのみ語順入替を許すように変更する．

2　先行研究のアルゴリズム [3, 4]

従来手法 [3, 4] では，Shift-Reduce アルゴリズムにおいて，Shift と Reduce の他に，読点挿入のための Shift-Comma と Reduce-Comma，語順変更のための Swap という 3 つの操作を追加するとともに，キューやスタックの他に，語順変更のためのスタック（以下，語順変更スタック）を新たに用意するという拡張を行い，係り受け解析・語順整序・読点挿入の同時実行を実現している．その概要は、語順変更スタックが空であればスタック Top の文節とキュー Front の文節を，語順変更スタックが空でなければスタック Top の文節と語順変更スタック Top の文節を，それぞれ操作対象として，これらの間に対する操作（Shift, Shift-Comma, Reduce,Reduce-Comma, Swap のいずれか）を選択することを繰り返すというものである。以下では，操作対象の 2 文節のうち，スタック Top の文節を前方文節，もう一方の文節を後方文節と呼び，各操作を下記で説明する．

Shift：前方文節が後方文節に係らないことを決定し，後方文節をスタックに移す．

Shift-Comma：Shift の操作に加えて，前方文節と後方文節の間に読点挿入することを決定する．

Reduce：前方文節が後方文節に係ることを決定し，前方文節をスタックから削除した上で，後方文節の子ノードとして追加し係り受け木を構成する．

Reduce-Comma：Reduce の操作に加えて，前方文節と後方文節の間に読点挿入することを決定する．
Swap：前方文節と後方文節の語順入替を決定し，前方文節，後方文節の順に語順変更スタックにプッシュする．なお，前方文節や後方文節が子ノードを持っている（すなわち，別の文節から係られると決定されている）場合は，その語順を保つ形で，全ての子孫を語順変更スタックにプッシュする．また，語順変更スタックにプッシュされる文節に読点が付与されている場合，その読点は削除される．

従来手法の動作例を図 1 に示す．図 1 では，入力文「おせちを家族のことを想って選んだ。」に対して，係り受け解析と語順整序，読点挿入を同時的に施し，読みやすい文「家族のことを想って、おせちを選んだ。」に整形する過程が示されている（最適な操作選択ができるとして）．まず入力文の文節列[1)]がキューに格納され，時刻 1 で Shift により，キューの先頭「おせちを」がスタックにプッシュされる．次に時刻 2 で Swap により，スタックのトップ「おせちを」とキューの先頭「家族の」がこの順で語順変更スタックにプッシュされる．時刻 3 で Shift により，語順変更スタックのトップ「家族の」がスタックにプッシュされる．なお，語順変更スタックの要素はキューの要素より優先的に操作対象となる．時刻 6 で Reduce により，「家族の」がスタックから削除され，「家族の」が「ことを」に係るとする係り受け木が構成される．時刻 12 で Shift-Comma により，「想って」と「おせちを」の間に読点が付与されるとともに，「おせちを」がスタックにプッシュされる．最後に時刻 13 で「想って、」と「おせちを」が「選んだ。」の子ノードとして Reduce され[2)]，終了する．

ここで，時刻 3 においてスタック・語順変更スタック・キューに格納された文節列を結合した文は「家族のおせちをことを想って選んだ。」となり，非文となる．これは時刻 2 で Swap により，「おせちを」と「家族の」の語順が入れ替わり，「家族の」と「ことを」の間に「おせちを」が挿入されたことが原因である．さらに，時刻 4 では，その非文となる語順に基づいて，操作選択が行われることになり，操作対象の「家族の」が「おせちを」に係ると誤って判定される（図 1 に示す正解の Shift ではなく、Reduce が選択される）可能性が高まると考えられる．このように，非文となる語順が一時的に生成さ

1) 入力文にある読点はすべて事前に削除される．
2) 入力文の語順と構文的制約 [6] から，1 回の Reduce によって複数の文節の係り先が一度に決定されることもある．

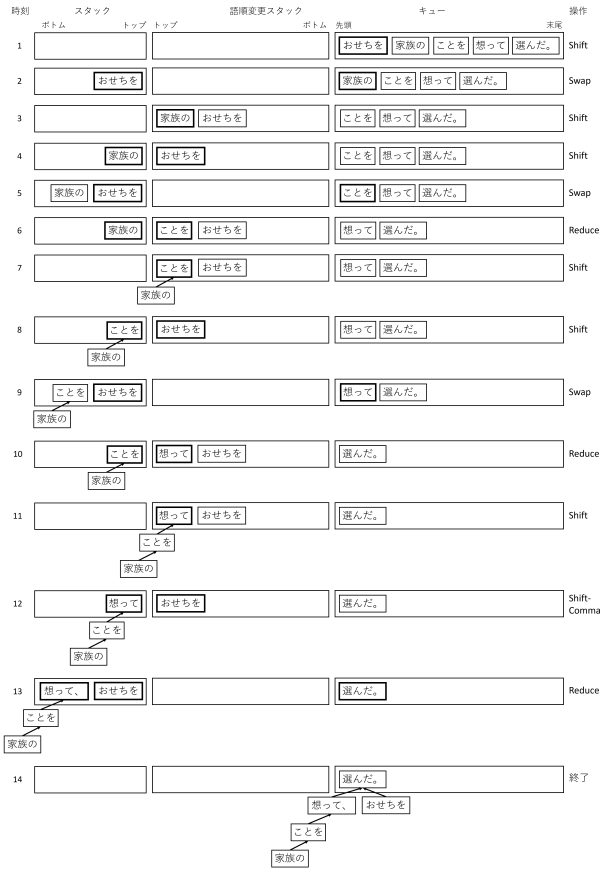

図 1　先行研究の動作例

れることにより，入力文とは意味が変わるような係り受け構造が同定される危険性が高まり，精度低下の一因となっていると考えられる。

3　提案手法

提案手法では，図 1 で説明した従来アルゴリズム [3, 4] を基に，解析の途中段階において語順入替後の文が非文になることを防ぐ方法を導入する．なお，本研究の問題設定は先行研究 [3, 4] と同様であり，意味は伝わるものの読みにくい語順を持った文の文節列が入力されることを想定する．

3.1　提案手法のアルゴリズム

提案手法では，同じ文節に係ると判定された 2 文節間でのみ語順入替を行うようにアルゴリズムを変更する．この実現のため，操作 Swap の代わりに，新たに定義した操作 Reduce-Swap を導入する．Reduce-Swap は，前方文節が後方文節に係ることと，前方文節と「既に後方文節に係ると決定されている文節のうち，最も先頭に位置する文節（以下，Swap 候補文節）」の語順入替を決定し，前方文節，Swap

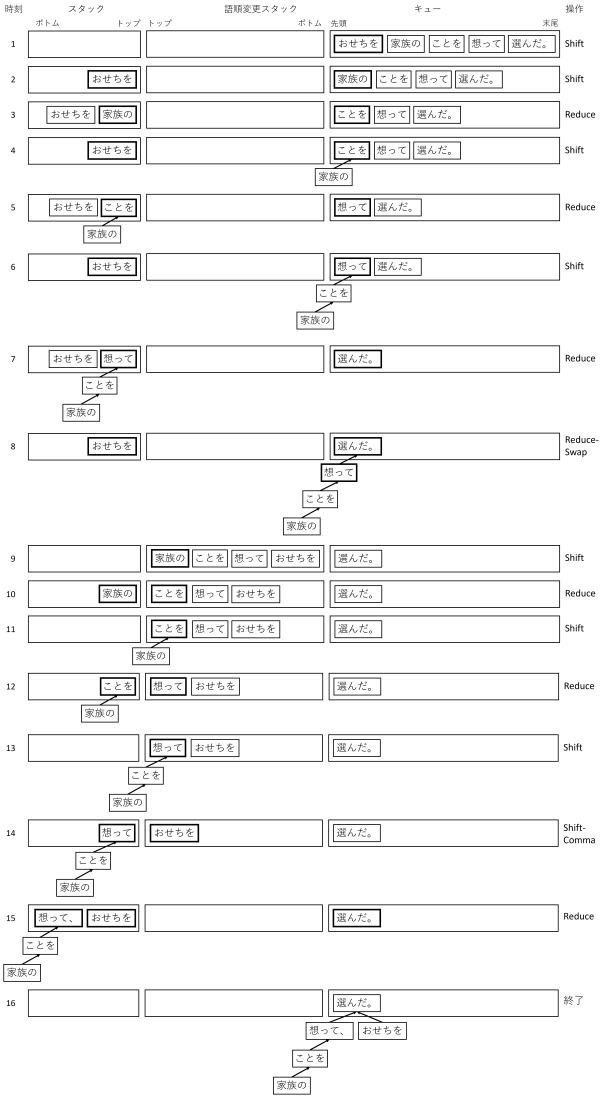

図 2　提案手法の動作例

候補文節の順に語順変更スタックにプッシュする操作である．なお，前方文節や Swap 候補文節に子ノードがある場合は，その語順を保つ形で，全ての子孫を語順変更スタックにプッシュする．また，語順変更に伴い読点位置を再度判定するため，Swap と同様に，語順変更スタックにプッシュされる文節に読点が付与されている場合は削除する．

図 1 と同じ入力文に対して提案手法を適用した場合の動作例を図 2 に示す．時刻 2 で選択される操作は，図 1 では Swap だが，図 2 では Shift である．これは，「家族の」に係ることが決定された文節が時刻 2 では存在せず，語順入替の条件を満たさないことが理由である．時刻 3 で「家族の」が「ことを」の子ノードとして Reduce される．時刻 4 では「おせちを」が「ことを」に係らないことが決定され，語順入替の条件を満たさないことから，Swap

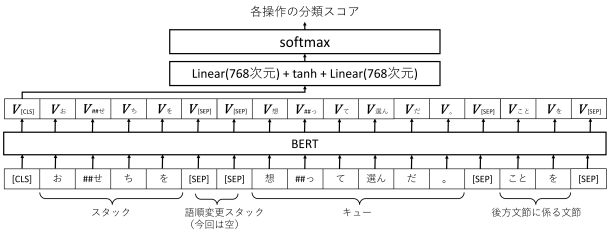

図 3　提案手法の操作選択モデル（図 2 の時刻 6）

ではなく Shift が選択される．時刻 8 で Reduce-Swap により，「おせちを」が「選んだ。」に係ることと，「おせちを」と「想って」の語順入替が決定され，「おせちを」，「想って」，「ことを」，「家族の」がこの順で語順変更スタックにプッシュされる．時刻 14 で Shift-Comma により，「想って」と「おせちを」の間に読点が付与されるとともに，「おせちを」がスタックにプッシュされる．最後に時刻 15 で「想って、」と「おせちを」が「選んだ。」の子ノードとして Reduce され[2)]，終了する．

3.2　BERT を用いた操作選択モデル

図 2 では，説明のため，各時刻で最適な操作が選択されているが，実際には機械的に選択する必要がある．提案手法の操作選択モデル（図 2 の時刻 6 での計算例）を図 3 に示す．スタック・語順変更スタック・キューに格納された各文節列，および後方文節に直接係る文節列を結合し，先頭に [CLS] を，各々の後に [SEP] を付与したうえで，サブワード分割を施したものを各時刻での計算状況とみなし，BERT に入力する．BERT の出力のうち，[CLS] に対応する出力のみを取り出し，2 層の Linear 層と Softmax を介し分類スコアを得て，分類スコアが最も高い操作を選択する．

本モデルを学習するには，各時刻での計算状況と，それに対する適切な操作の組が大量に必要となる．このような学習データを構築するには，読みにくい文と，それを読みやすく整形した文に対して係り受け構造を付与したデータが大量に必要となるが，そのようなデータはない．本研究では，従来手法 [4] と同様に、係り受け構造が付与された読みやすいと想定される文を収録したコーパスを元に，疑似的な読みにくい文を機械的に生成し，それを提案アルゴリズムで元の読みやすい文に戻す過程を自動生成し，計算状況と適切な操作の組を大量に構築した．

表 1 係り受け解析の正解率

手法	係り受け単位	文単位
従来	84.55%	49.60%
手法 [4]	(6,415/7,587)	(496/1,000)
提案	90.68%	59.00%
手法	(6,880/7,587)	(590/1,000)

表 2 語順整序の正解率

手法	2 文節単位	文単位
従来	86.11%	48.90%
手法 [4]	(27,350/31,760)	(489/1,000)
提案	86.10%	52.40%
手法	(27,344/31,760)	(524/1,000)

表 3 読点挿入の再現率・適合率・F 値

手法	再現率	適合率	F 値
従来	70.47%	83.28%	76.34
手法 [4]	(284/403)	(284/341)	
提案	72.31%	76.69%	74.43
手法	(329/455)	(329/429)	

4 評価実験

提案手法の有効性を確認するために，新聞記事文から擬似的に作成した読みにくい文を用いた実験を行った．

4.1 実験概要

テストデータには，京大テキストコーパス [7] を元に人手を介して疑似的に作成された読みにくい語順の文 1,000 文（従来研究 [3, 4] と同一）を用いた．学習データには，京大テキストコーパス Ver.4.0[7] の 32,506 文[3] に対して，3.2 節の手順を適用して得られる計算状況と適切な操作の組 1,467,980 件を用いた．学習データとテストデータの間で，元となった新聞記事文に重複はない．

評価では，従来手法 [4] と同じ指標を用いた．係り受け解析では，係り受け正解率（文末文節以外の全文節のうち，正解と係り先が一致している文節の割合）と文単位正解率（正解の係り受け構造と完全一致している文の割合）を測定した．語順整序では，2 文節単位正解率（文末文節以外の文節を 2 つずつ取り上げたとき，それらの前後関係が正解と一致している割合）と文単位正解率（正解の語順と完全一致している文の割合）を測定した．読点挿入では，語順が正解と完全一致している文のみを対象に，読点位置に関する再現率，適合率，F 値を測定した．

モデルは PyTorch を用いて実装し，BERT の事前学習モデルには，東北大学が公開しているモデル[4] を用いた．学習アルゴリズムは Adam を用い，パラメータの更新はミニバッチ学習 (学習率 1e-5，バッチサイズ 16) により行った．損失関数には Cross Entropy Loss を使用した．エポック数は 1 とした．

3） 構文的制約から 1 通りの語順しか考えられない文や，構文的制約を満たさない文は事前に排除した．

4） https://github.com/cl-tohoku/bert-japanese

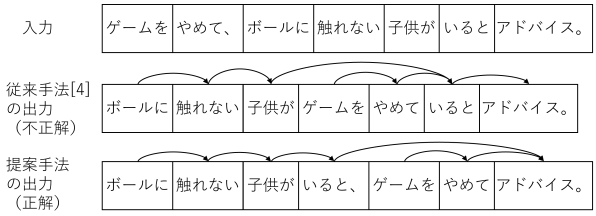

図 4 提案手法の成功例

4.2 実験結果

実験結果を表 1 から表 3 に示す．提案手法は，従来手法 [4] と比較して，読点挿入の F 値で下回ったものの，係り受け解析の正解率と語順整序の文単位正解率で大幅に上回っており，提案手法の有効性を確認した．

図 4 に従来手法 [4] では不正解だったが，提案手法では正解した例を示す．従来手法 [4] では，「やめて」が「いると」に係ると誤って解析しており，それに伴って，これらの文節に関わる語順整序や読点挿入に失敗している．これは，「ゲームを」と「やめて」の語順を変更するために，これらの位置を「ボールに」，「触れない」，「子供が」の後に順次入れ替えていった結果，目的の位置に達する前に，非文となる語順が生成されたことが原因と考えられる．一方，提案手法は，共に「アドバイス」に係ると解析された「いると」と「やめて」の間でのみ語順入替が行われた結果，処理途中で入力と文意が変わることがなく，正しく解析が行われたと考えられる．

5 おわりに

本稿では，Shift-Reduce アルゴリズムを拡張し，処理途中で非文になることを回避しつつ，日本語文に対する係り受け解析・語順整序・読点挿入を同時実行する手法を提案した．読みにくい文に対する評価実験を実施した結果，従来手法と比べ，係り受け正解率や語順整序の文単位正解率が大幅に向上しており，本手法の有効性を確認した．今後は，操作選択モデルの精緻化などにより，更なる精度向上を図りたい．

謝辞

本研究は，一部，科学研究費補助金基盤研究 (C) No. 19K12127 により実施した．

参考文献

[1] 内元清貴, 村田真樹, 馬青, 関根聡, 井佐原均. コーパスからの語順の学習. 自然言語処理, Vol. 7, No. 4, pp. 163–180, 2000.

[2] 大野誠寛, 吉田和史, 加藤芳秀, 松原茂樹. 係り受け解析との同時実行に基づく日本語文の語順整序. 電子情報通信学会論文誌 D, Vol. J99-D, No. 2, pp. 201–213, 2016.

[3] 宮地航太, 大野誠寛, 松原茂樹. 係り受け解析との同時実行に基づく日本語文の語順整序と読点挿入. 言語処理学会第 26 回年次大会発表論文集, pp. 243–246, 2020.

[4] 荒木駿介, 大野誠寛, 松原茂樹. 日本語文に対する係り受け解析・語順整序・読点挿入の同時実行. 情報処理学会第 85 回全国大会講演論文集, No. 2, pp. 735–736, 2023.

[5] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. 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)**, pp. 4171–4186, 2019.

[6] 内元清貴, 関根聡, 井佐原均. “最大エントロピー法に基づくモデルを用いた日本語係り受け解析. 情報処理学会論文誌, Vol. 40, No. 9, pp. 3397–3407, 1999.

[7] 黒橋禎夫, 長尾眞. 京都大学テキストコーパス・プロジェクト. 言語処理学会第 3 回年次大会発表論文集, pp. 115–118, 1997.