非局所的依存関係同定の木構造を用いた近似解法

加藤 芳秀　　松原 茂樹

名古屋大学情報連携統括本部
yoshihide@icts.nagoya-u.ac.jp

1 はじめに

非局所的依存関係はゼロ代名詞や構成素の移動といった言語現象を表現しており，意味解析において重要な役割をもつ．一般に，非局所的依存関係を含む構文構造はグラフとなるが，本稿では，このグラフを木構造に変換する手法を提案する．提案する変換手法は近似的なものであり，木構造から元のグラフが復元できることを保証しないが，言語学的な制約を用いることにより高い精度でグラフを復元できる．これまでに提案されている構文解析の多くはその構文構造として木構造を想定しているが，提案する変換手法により，非局所的依存関係同定と構成素解析の 2 つのタスクをそのような構文解析を用いて同時に解くことが可能となる．本稿で提案するアプローチが，非局所的依存関係同定において従来の手法よりも高い精度・再現率を達成できることを実験により確認した．

2 非局所的依存関係とその同定

本節では，Penn Treebank (PTB) における非局所的依存関係について説明し，次に，これまでに提案された非局所的依存関係同定手法について概観する．

2.1 PTB における非局所的依存関係

PTB において非局所的依存関係は，構文構造中のノードの対，すなわちエッジとして表現される．一方のノードを**空要素**と呼び，字面上は現れない構文構造中の要素を表す．もう一方のノードを**フィラー**と呼ぶ．非局所的依存関係は，空要素の位置に，対応するフィラーが存在するものとして解釈すべきであることを意味している．PTB の構文構造において構成素に関する情報は木構造をなしているが，非局所的依存関係も考慮すると，その構造はグラフである．以下では，PTB の構文構造を，その構造がグラフであることを明確にするために **PTB グラフ**と呼ぶことにする．図 1 左に PTB グラフの例を示す．-NONE- は空要素を表すラベルである．0 や *T* は空要素のタイプを表しており，0 は関係代名詞の省略を，*T* は wh 移動の痕跡を表している．空要素のタイプに番号が与えられているとき，対応するフィラーが PTB グラフ中に存在し，それに対しても同一の番号が付与される．例えば，空要素 *T*-1 に対応するフィラーは，WHNP-1 であることを意味する．

2.2 非局所的依存関係の同定

これまでに PTB に基づく構文解析は多数提案されているが，その大半は，PTB グラフから非局所的依存関係を表すエッジ及び，空要素を取り除いた木構造（以下では，**PTB 木**と呼ぶ）を対象としている．木構造を対象とすることにより，その解析アルゴリズムは簡潔かつ効率的になるが，非局所的依存関係の情報は当然失われてしまう．

これに対して，非局所的依存関係を同定する手法がこれまでに提案されている．それらは以下の 2 つに大別できる．

- 非局所的依存関係を同定するために特殊な操作を構文解析に導入する手法．
- 後処理により PTB 木から PTB グラフを復元する手法．

構文解析に特殊な操作を導入する手法として，空要素を挿入する操作を導入する手法 [5, 7] や，グラフ構造を直接的に扱う手法 [4, 9] が提案されている．いずれの手法においても，従来の構文解析とは異なる操作を導入した結果として，そのアルゴリズムは複雑になる．また，解析のためのモデルも構文解析アルゴリズムに合わせて別途，設計する必要が生じる．

後処理に基づく手法は，PTB 木に基づく構文解析を実行した後，解析結果から非局所的依存関係を復元する．復元の方法として，パターンマッチングに基づく手法 [6]，ルールに基づく手法 [2]，複数の識別器を用いる手法 [10] が提案されている．

3 PTB グラフの木構造による近似

本節では，PTB グラフを，非局所的依存関係を近似的に表現した木構造へと変換する手法を提案する．その狙いは，非局所的依存関係同定，及び構成素解析の 2 つのタスクを，木構造に基づく構文解析をそのまま用いて同時に解くことにある．変換は，次の 2 つの変換から構成される．

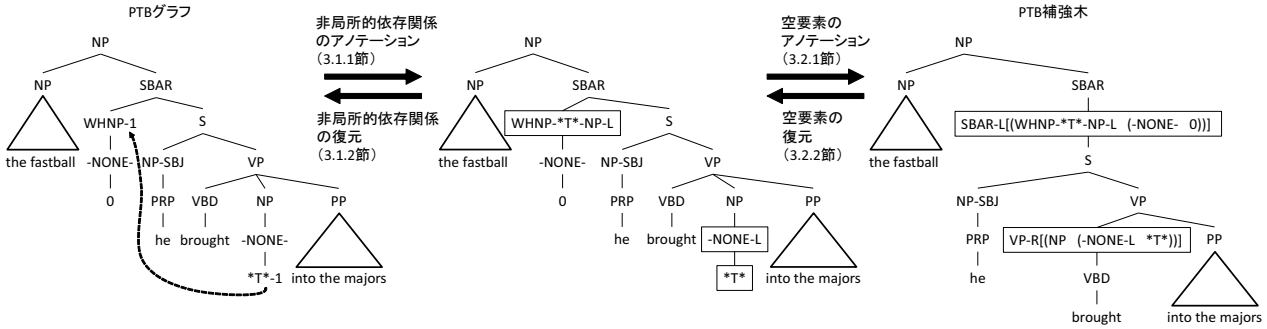

図 1: PTB グラフとその PTB 補強木

非局所的依存関係の削除 非局所的依存関係を表すエッジを削除し，これを復元するための情報をノードのラベルに付加する．

空要素の削除 空要素の情報を構文構造中にノードとして埋め込み，空要素を削除する．

以下では，2 つの変換を順に行った結果として得られる木構造を **PTB 補強木**と呼ぶ．次節以降で変換について説明するが，まず図 1 に変換の例を示す．

3.1 非局所的依存関係の削除

非局所的依存関係を表すエッジの情報は，Kato ら [7] と同様のアノテーションを用いて近似的に表現する．この方法では，空要素とフィラーのラベルに対してタグを付加し，エッジ自体は削除する．非局所的依存関係を表すエッジを削除することにより，PTB グラフは木構造に変換される．得られた木構造から元の非局所的依存関係が一意に復元できることは保証されないが，c-統御の関係を用いたルールベースの方法により大半の非局所的依存関係がアノテーションから復元できる．

3.1.1 アノテーション

タイプが *EXP*，*ICH*，*RNR*，*T* であるフィラーと空要素に対しては，以下のようにタグを付加する．

- フィラーに対して空要素のタイプ，及び空要素の親ノードの範疇を付加する．
- フィラー及び空要素に対して，フィラーの空要素に対する相対的な位置を付加する．位置を表す値は A，L，R のいずれかであり，それぞれ，フィラーが空要素の祖先，左，右に位置することを表す．

タイプが * の空要素に対しては，以下のようにタグを付加する．フィラーに対してはアノテーションは行わない．

- 空要素に対してフィラーの空要素に対する相対的な位置を付加する．
- 対応するフィラーに機能タグ SBJ が付与されていない場合，空要素にタグ OBJCTRL を付加する．

タグ OBJCTRL は，主語コントロールと目的語コントロールを区別するためのタグである．

図 1 の中央の木構造は，図左の PTB グラフに対して変換を施した結果である．非局所的依存関係を表すエッジは削除され，対応関係にある空要素 (-NONE- *T*-1) 及びフィラー WHNP-1 に対してアノテーションがなされる．対応関係を持たない空要素 (-NONE- 0) に対してはアノテーションはなされない．

3.1.2 非局所的依存関係の復元

本節では，前節で述べたアノテーションから非局所的依存関係を復元する方法について提案する．提案する手法はルールベースであり，そのルールは Kato らと類似した考え方に基づいているが，従来のルールが構文解析の中間解析結果に適用する目的で設計されていたため複雑であったのに対して，ここで提案するルールは構文木を適用の対象とした結果，より簡単なものとなっている．

非局所的依存関係復元ルールは，ノードと制約の対として与えられる．構文木中に「ノード」にマッチする空要素，あるいはフィラーが存在するとき，「制約」を満たすノードをその対応するフィラーあるいは空要素とする．制約を満たすノードが複数存在する場合は，最も近いものを選択する．表 1 にそのルールをまとめる．ここで，$\mathtt{A}(x, y)$ は，ノード x がノード y の祖先であることを意味する．$\mathtt{L}(x, y)$ は，x が y の左に出現することを，$\mathtt{R}(x, y)$ は右に出現することを意味する．$\mathtt{c\text{-}cmd}(x, y)$ は c-統御の関係であり，以下のように定義する．

- x, y を任意のノードとする．1. と 2. を満たすノード z が存在するとき，x は y を c-統御するという．
  1. z は x の兄弟である．
  2. $\mathtt{A}(z, y)$.

$\mathtt{SBJ}(x)$ は x のラベルがタグ SBJ を持つことを表し，$\mathtt{cat}(x)$，及び $\mathtt{par}(x)$ はそれぞれ，x の範疇，親ノー

表 1: 非局所的依存関係復元ルール

ノード	対応するノード x が満たすべき制約
$e =$ (-NONE-L $*$)	$\texttt{L}(x, e) \wedge \texttt{c-cmd}(x, e) \wedge \texttt{SBJ}(x)$
$e =$ (-NONE-R $*$)	$\texttt{R}(x, e) \wedge \texttt{c-cmd}(x, e) \wedge \texttt{SBJ}(x)$
$e =$ (-NONE-L-OBJCTRL $*$)	$\texttt{L}(x, e) \wedge \texttt{c-cmd}(x, e) \wedge \texttt{cat}(x) \in \{\texttt{NP}, \texttt{PP}\} \wedge \texttt{cat}(\texttt{par}(x)) = \texttt{VP}$
$e =$ (-NONE-L $*$T$*$)	$\texttt{L}(x, e) \wedge \texttt{c-cmd}(x, e) \wedge \texttt{match}(x, e)$
$e =$ (-NONE-A $*$T$*$)	$\exists y \Big(\texttt{A}(x, y) \wedge \texttt{A}(y, e) \wedge \texttt{cat}(y) = \texttt{PRN}\Big) \wedge \texttt{cat}(\texttt{par}(e)) = \texttt{cat}(x)$
$e =$ (-NONE-R $*$RNR$*$)	$\texttt{R}(x, e) \wedge \texttt{c-cmd}(x, e) \wedge \texttt{match}(x, e)$
$e =$ (-NONE-L $*$ICH$*$)	$\texttt{L}(x, e) \wedge \texttt{match}(x, e)$
$e =$ (-NONE-R $*$ICH$*$)	$\texttt{R}(x, e) \wedge \texttt{match}(x, e)$
$f =$ (X-$*$EXP$*$-R $\cdots$)	$\texttt{R}(f, x) \wedge \texttt{c-cmd}(x, f) \wedge x =$ (NP (PRP it))

ドを表す．$\texttt{match}(f, e)$ はフィラー f にタグ付けされたタイプと範疇が，空要素 e のそれと一致し，かつ f と e に付加された位置のタグも一致することを表す．

3.2 空要素の削除

3.1 節で述べた変換により非局所的依存関係を表すエッジが取り除かれるが，依然として空要素は残されている．構文解析時に空要素を扱う操作を不要とするために，本節では，空要素の情報を内部ノードとして構文木に埋め込み，空要素は削除する変換を提案する．

3.2.1 アノテーション

空要素削除の変換は，3.1 節の変換の後に行う．なお，以下ではノード x が支配するすべての葉ノードが空要素であるとき，x は**空**であるということにする．すべての空ノード n に対して，場合に応じて以下のいずれかの変換を適用する．

- n の左の兄弟に空でないノードがあるとき，以下の処理を行う．
  1. 最も左に出現する空でない兄弟ノードを c_s とし，n の左隣の兄弟を c_e とする．また，p を n の親ノードとする．このとき，$c_i (s \leq i \leq e)$ を p から切り離す．
  2. 親を p，子を $c_i (s \leq i \leq e)$ とするような新たなノード t を生成する．
  3. t のラベルとして，p の範疇を与え，タグとして R，及び n を根とする部分木を表現するようなタグを付加する．
- n に左の兄弟に空でないノードがない場合，上述の処理において，c_i を n の右に出現するすべての兄弟とし，R を L に置き換えた処理を実施する．

図 1 右が，中央の構文木にこの変換を実施した結果である．この変換において注意すべき点は，n を根とする部分木を表現するような文字列は単なるラベルの一部に過ぎず，最終的に得られた PTB 補強木のノードではない点である．

Kummerfeld らの手法 [9] でも類似した方法で空要素が表現されているが，空要素の位置を保存しないこと，非局所的依存関係の情報が保存されていることが我々の手法とは異なる．また，Kummerfeld らの手法では，構文解析時に循環を含むグラフを構成しないようにするために，特別に設計された主辞決定ルールを必要とするが，我々の手法ではそもそも主辞決定ルール自体が不要である．

3.2.2 空要素の復元

前節の方法によりノードのラベルとして埋め込まれた空要素の情報は，逆の操作で一意に復元できる．すなわち，挿入されたノード t を削除し，ラベルのタグから部分木を復元し，それをタグ L がある場合は左隣，R の場合は右隣の兄弟として挿入すればよい．

4 評価実験

提案手法の有効性を確認するために，PTB グラフを PTB 補強木に変換し構文解析モデルを学習し，構文解析が出力する PTB 補強木から PTB グラフを復元し，非局所的依存関係同定の精度・再現率を評価した．評価法は，Johnson が提案した方法 [6] を用いた．この評価法では，非局所的依存関係を「空要素のタイプ」,「空要素の範疇」,「空要素の位置」,「フィラーの範疇」「フィラーの位置」の 5 項組として表現し，精度，再現率を評価する．

構文解析には Kitaev ら [8] のものを用いた．この構文解析には，外部データとして ELMo を使用することができるが，使用する場合と，使用しない場合の両者で実験した．PTB の WSJ コーパスのセクション 2-21，22，23 をそれぞれ学習データ，バリデーションデータ，テストデータとして使用した．ハイパーパラメータは，Kitaev ら [8] と同一である．PTB 補強木を構成素木とみなして，バリデーションデータにおいて F 値が最大となったモデルを選択した．

非局所的依存関係同定の精度・再現率を表 2 に示す．提案手法は，従来の手法と比較して大幅に精度・再現率が向上している．その主要な理由は，本実験において用いた構文解析の性能によるものと考えられるが，ここで重要なポイントは，本稿で提案した PTB グラフの PTB 補強木への変換がそのような構文解析の利用を可能にしたという点である．構文解析に何らかの

表 2: 非局所的依存関係同定の精度・再現率

	空要素検出（フィラーを考慮せず評価）			非局所的依存関係同定			非局所的依存関係同定（フィラーがあるもののみで評価）		
	精度	再現率	F	精度	再現率	F	精度	再現率	F
Johnson (2002) [6]	85	74	79	73	63	68	–	–	–
Dienes ら (2003) [3]	–	–	–	81.5	68.7	74.6	–	–	–
Campbell (2004) [2]	85.2	81.7	83.4	78.3	75.1	76.7	–	–	–
Schmid (2006) [11]	86.0	82.3	84.1	–	–	–	81.7	73.5	77.4
Cai ら (2011) [1]	90.1	79.5	84.5	–	–	–	–	–	–
Hayashi ら (2016) [5]	90.3	81.7	85.8	–	–	–	–	–	–
Kato ら (2017) [7]	88.5	82.1	85.2	81.4	75.5	78.4	79.8	73.8	76.7
Kummerfeld ら (2017) [9]	89.5	81.6	85.4	74.3	67.3	70.6	–	–	–
PTB 木から PTB グラフを復元する手法（正解 PTB 木を使用した場合）									
Johnson (2002) [6]	93	83	88	80	70	75	–	–	–
Campbell (2004) [2]	94.9	91.1	93.0	90.1	86.6	88.4	–	–	–
提案手法 (ELMo なし)	92.6	87.7	90.1	88.1	83.4	85.7	88.4	81.1	84.6
提案手法 (ELMo あり)	94.2	90.3	92.3	89.9	86.2	88.0	90.4	84.1	87.2

操作を導入するアプローチでは，このように従来の構文解析を直接的に利用することはできない．一方，後処理による手法では，その前処理にあたる構文解析の性能が向上すれば，非局所的依存関係同定の性能も向上するが，構文解析が全く誤らない場合でも，その精度・再現率は提案手法と同程度である．

復元された PTB グラフを PTB 木に変換し構成素解析として評価したところ，ELMo を用いる場合，用いない場合の F 値は，それぞれ 94.99 と 93.31 であった．文献 [8] の実験結果では，その F 値はそれぞれ，95.13，93.55 であり，提案手法は構成素解析にほとんど悪影響を与えていないことが確認された．

5 おわりに

本稿では，PTB グラフの PTB 補強木への変換に基づき，非局所的依存関係同定を近似的に解く手法を提案した．提案した枠組において用いる構文解析には特定のものを想定しておらず，様々な構文解析を用いることができる．今後，構文解析技術が発展し，より高精度な構文解析が開発されれば，非局所的依存関係同定の精度も併せて向上すると期待できる．

謝辞

本研究は一部，科研費基盤研究 (C)(No. 17K00303) により実施した．

参考文献

[1] Shu Cai, David Chiang, and Yoav Goldberg. Language-independent parsing with empty elements. In *Proc. of ACL-HLT 2011*, pp. 212–216, 2011.

[2] Richard Campbell. Using linguistic principles to recover empty categories. In *Proc. of ACL 2004*, pp. 645–652, 2004.

[3] Péter Dienes and Amit Dubey. Antecedent recovery: Experiments with a trace tagger. In *Proc. of EMNLP 2003*, pp. 33–40, 2003.

[4] Kilian Evang and Laura Kallmeyer. PLCFRS parsing of English discontinuous constituents. In *Proc. of IWPT 2011*, pp. 104–116, 2011.

[5] Katsuhiko Hayashi and Masaaki Nagata. Empty element recovery by spinal parser operations. In *Proc of ACL 2016*, pp. 95–100, 2016.

[6] Mark Johnson. A simple pattern-matching algorithm for recovering empty nodes and their antecedents. In *Proc. of ACL 2002*, pp. 136–143, 2002.

[7] Yoshihide Kato and Shigeki Matsubara. Left-corner parsing for identifying PTB-style nonlocal dependencies. *Journal of Natural Language Processing*, Vol. 24, No. 3, pp. 371–394, 2017.

[8] Nikita Kitaev and Dan Klein. Constituency parsing with a self-attentive encoder. In *Proc. of ACL 2018*, pp. 2676–2686, 2018.

[9] Jonathan K. Kummerfeld and Dan Klein. Parsing with traces: An $O(n^4)$ algorithm and a structural representation. *TACL*, Vol. 5, pp. 441–454, 2017.

[10] Roger Levy and Christopher Manning. Deep dependencies from context-free statistical parsers: Correcting the surface dependency approximation. In *Proc. of ACL 2004*, pp. 327–334, 2004.

[11] Helmut Schmid. Trace prediction and recovery with unlexicalized PCFGs and slash features. In *Proc. of COLING-ACL 2006*, pp. 177–184, 2006.