同期文法を用いた構文木付きコーパスの誤り訂正

加藤 芳秀　　松原 茂樹

名古屋大学情報基盤センター
E-Mail: yosihide@el.itc.nagoya-u.ac.jp

1　はじめに

近年，品詞情報や構文情報などが付与されたコーパス，いわゆるタグ付きコーパスの重要性が高まっており，言語現象の調査や自然言語処理システムの開発など様々な場面で活用されている．しかし，タグ付きコーパスには誤りが少なからず含まれている．タグは，人手による作業，あるいは，解析器による解析結果の人手による編集などにより付与されるが，この過程において誤りが混入されるためである．

このような問題に対して，タグ付きコーパスに含まれる誤りを検出・訂正する手法が提案されている [2, 3, 4, 7, 8]．品詞タグや係り受けタグ，あるいは構文木タグの誤りを訂正する手法である．これらの手法は，コーパスにおいて誤ったタグが付与されている箇所を検出し，それを別のタグに置き換えることにより誤りを訂正するが，構文木付きコーパスの誤り訂正としては不十分である．なぜならば，構文木付きコーパスにおいてはタグが階層的な構造を持つため，構造的な変換を伴う訂正が必要となるからである．

そこで本稿では，構文木付きコーパスに含まれる誤りを検出・訂正する手法を提案する．構造の変換を伴う誤り訂正を実現するために，本手法では，同期文法の一種である synchronous tree substitution grammar (STSG)[5] を用いる．STSG は木構造から木構造への変換を実現するための文法であり，これを用いることにより，誤りを含む構文木を誤りを含まない構文木に変換できる．誤り訂正のための同期文法は，誤り訂正の対象となる構文木付きコーパスから自動的に獲得する．ある単語列がコーパス中の複数の場所に出現し，それらに対して異なる構文構造が割り当てられているとき，一方の構文構造を誤りを含む構文構造，他方を誤りを含まない構文構造とみなし，前者を後者に変換するルールを作成する．頻度情報を利用することにより，高精度な変換ルールを作成できる．構文木付きコーパスである Penn Treebank[6] を用いた実験により，高精度な変換ルールを獲得できることを確認した．

本稿の構成は以下のとおりである．次の 2 節では，タグ付きコーパスの誤り訂正に関する従来の手法を概観する．3 節では，STSG に基づく構文木付きコーパスの誤り訂正手法を提案する．4 節では，構文木付きコーパスを用いた誤り訂正実験について報告する．5 節はまとめである．

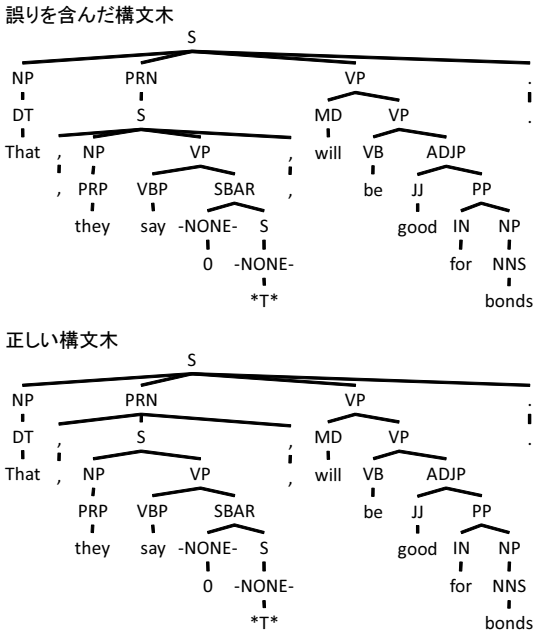

図 1: 構文木の誤りの例

2　タグ付きコーパスの誤り訂正

本節では，タグ付きコーパスの誤り訂正に関する従来の手法を概観し，その問題点について議論する．

タグ付きコーパスの誤り訂正手法は，これまでにいくつか提案されている．Dickinson ら [2] は，品詞タグの誤り訂正を，乾ら [8] は，係り受けタグの誤り訂正を提案している．一般に，品詞タグや係り受けタグは，文中の単語などに対して 1 つだけ与えられる．したがって，このようなタグの誤りを訂正するには，コーパスにおいて与えられている誤ったタグを別のタグに置き換える操作で十分である．

一方，構文木付きコーパスの誤りを訂正する場合，問題はより複雑である．なぜならば，構文木付きコーパスにおいては，タグが階層構造を構成するからである．誤り訂正は，階層構造が誤っている場合にも対処できなければならない．例として，図 1 に示す構文木の誤りについて考える．上の構文木が誤りを含んだ構文木，下が正しい構文木である．この例では，挿入句 “they say” の前後に出現する “,” の構

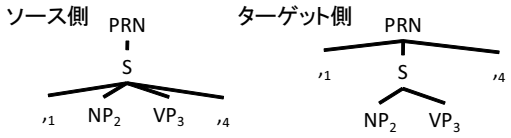

図 2: STSG のルールの例

文木上の位置が誤っている[1]．ここで重要なのは，上の構文木中のタグ（統語範疇）をどのように置き換えようとも，下の正しい構文木は得られないという点である．この誤りを訂正するには，内側に存在する節点 S に直接支配されている木構造 (，,) を移動し，PRN に直接支配されるように構造を変更しなければならない．

このような問題も相まって，従来の手法は，構文木付きコーパスの誤りを訂正するまでには至っていない．Ule らの手法 [7] や，Dickinson らの手法 [4] では，文脈自由規則の出現頻度を元に，誤りである文法規則で構築された構造を検出する．Dickinson らが文献 [3] で提案した手法では，構文木の節点に与えられたタグの誤りを検出する[2]．いずれの手法も，構文木の誤っている箇所を検出するのみであり，その誤りを訂正することはできない．

3 同期文法に基づく誤り訂正

前節で述べた問題を解決するには，構造的な変更を伴う誤り訂正を実現する必要がある．そこで本節では，synchronous tree substitution grammar (STSG) に基づく誤り訂正手法を提案する．STSG は木の対の集合を定める文法であり，木構造の変換を実現できる．本手法では，誤りを含んだ構文木を誤りを含まない構文木に変換する STSG を定める．

3.1 Synchronous Tree Substitution Grammar

STSG は，木の対の集合を定義する文法である [5]．木の対は，ルールを用いて導出する．ルールは

- 基本木と呼ばれる木の対
- 基本木中の節点間の対応関係

から構成される．以下では，基本木の対の一方をソース側の基本木，他方をターゲット側の基本木と呼ぶことにする．ルールは，ソース側の基本木にマッチする構造を，ターゲット側の基本木の構造に変換することを表す．ルールの例を図 2 に示す．このルールは，図 1 で示した構文木の誤りを訂正するルールである．基本木中の節点間の対応関係は，非終端記号，すなわち統語範疇をラベルに持つ葉節点に与えられる．図中では添字により対応関係を示している．以下では節点間の対応関係を明示しないが，非終端記号をラベルに持つ葉節点同士が左から順に対応するものとする[3]．

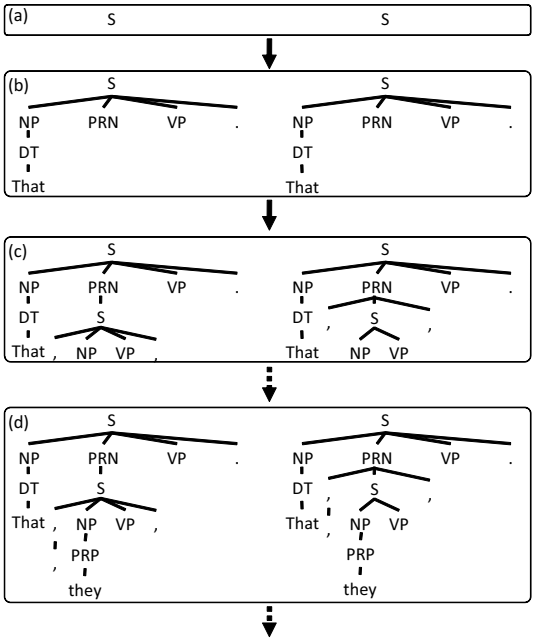

図 3: STSG による木の対の導出過程

木の対は，ルールにおいて対応関係にある葉節点を，それらのラベルと同一のラベルを根に持つ基本木からなるルールで置き換えることにより導出する．図 3 に，木の対の導出過程の例を示す．導出は，開始記号をラベルに持つ節点の対（図 3(a)）から始まる．これらの節点を，同一の基本木

```
(S (NP (DT That)) PRN VP .))
```

の対で同時に置き換えると，(b) の対が得られる．(b) において，PRN でラベル付けさされた節点を，それぞれ，図 2 の基本木で置き換えると，(c) の対が得られる．以下，同様にルールで置き換えることにより，例えば図 1 で示した構文木の対などが得られる．この例が示すように，STSG を用いることにより，誤りを含んだ構文木と正しい構文木の間の対応関係を定めることができる．

以下では，ソース側の基本木とターゲット側の基本木が異なるルールにのみ注目する．ソース側とターゲット側の基本木が同一であるルールは，誤り訂正には貢献しないからである．

3.2 誤り訂正のためのルールの獲得

誤り訂正を実現するためには，そのためのルール集合を用意しなければならない．本節では，誤り訂正の対象となる構文木付きコーパスからルール集合を自動的に獲得する方法を提案する．

[1] 構文木中の 0，及び*T*は，字面上には出現しない空要素である．

[2] 構成素を構成しないことを示す特殊なラベルを用いることにより，階層構造の誤りを検出することもできるが，誤り訂正はできない．

[3] 一般の STSG においては，順に対応しない場合も存在するが，本手法ではそのような場合は扱わない．

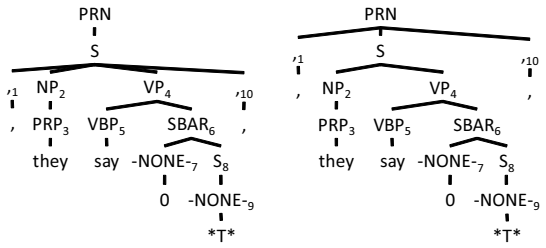

図 4: 擬似パラレルコーパス中の構文木対の例

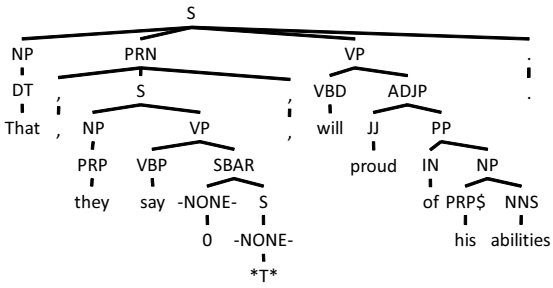

図 5: 挿入句 “they say” を含む別の構文木の例

まず，基本的なアイデアについて述べる．本手法のベースとなるアイデアは，Dickinson らが文献 [3] において提示した以下のような考え方である．

- ある単語列がコーパスの複数の場所に出現し，かつ，それらに対して異なるタグが付与されているとき，いずれかのタグが誤りである可能性が高い

ただし，本手法では，単純にタグの種類の違いを捉えるだけではなく，STSG を用いて階層構造の違いをも捉える．

本節で提案するルール獲得手法は，ある単語列がコーパスの複数の場所で出現し，それらに対して異なる構文木が割り当てられているとき，それらの構文木を対にして擬似的にパラレルコーパスを構築する．これは，誤りを含む構文木と正しい構文木との関係を捉えるためのパラレルコーパスである．次に，このパラレルコーパスから STSG のルールを抽出する既存の手法を適用することにより，ルールを自動的に抽出する．以下では，それぞれの手続きについて順に説明する．

3.2.1 部分構文木の対応付けによる擬似パラレルコーパスの構築

擬似パラレルコーパスを構築する手順は以下の通りである．コーパス中に出現する構文木の集合を T とする．構文木 σ に含まれる部分構文木の集合を $Sub(\sigma)$ とする．このとき，以下のように定義される構文木対の集合（擬似パラレルコーパス）を獲得する．

$$\begin{aligned} Para(T) \;=\; \{\langle \tau, \tau' \rangle \mid \tau, \tau' \in \bigcup_{\sigma \in T} Sub(\sigma) \\ \wedge\, \tau \neq \tau' \\ \wedge\, yield(\tau) = yield(\tau') \\ \wedge\, root(\tau) = root(\tau')\} \end{aligned}$$

ここで，$yield(\tau)$ は，τ の葉のラベル，すなわち単語を左から順に並べた列，$root(\tau)$ は，τ の根のラベルである．

例えば，図 4 の構文木対は，擬似パラレルコーパス中の要素の一例である．図 1 に示した誤りを含んだ構文木と正しい構文木がコーパス中に存在するとき，この対は擬似パラレルコーパスの要素となる．部分構文木に注目しているので，同一の文に対する正しい構文木でなくともよい．すなわち，図 1 の正しい構文木の代わりに，挿入句 “they say” を含む別の構文木，例えば，図 5 に示すような構文木が存在する場合でも，図 4 の構文木対は，擬似パラレルコーパスの要素となる．

3.2.2 擬似パラレルコーパスからのルールの獲得

本稿で提案する手法では，Cohn らの手法 [1] を用いて STSG ルールを抽出する．Cohn らの手法では，パラレルコーパスの対応関係を表現できるような STSG ルールを獲得する．この手法では，対となる構文木の単語間に対応関係が存在することを前提としているが，前節の方法で作成した擬似パラレルコーパス中の対において，単語列は同一であるため，対応関係を一意に定めることができる．ルール抽出の手順は 2 つのステップに分けられる．まず，単語間の対応関係を用いて，構文木対の節点の間の対応関係を定める．次に，節点間の対応関係を用いて，構文木対からルールを抽出する．

構文木対 $\langle \tau, \tau' \rangle$ の節点間の対応関係を次のように定義する．

$$\begin{aligned} C(\tau, \tau') = \{\langle \eta, \eta' \rangle \mid \eta \in Node(\tau) \wedge \eta' \in Node(\tau') \\ \wedge\, label(\eta) = label(\eta') \\ \wedge\, yield(\eta) = yield(\eta')\} \end{aligned}$$

ここで，$Node(\tau)$ は，τ 中の節点の集合，$label(\eta)$ は，節点 η のラベル，$yield(\eta)$ は，η が支配する葉節点のラベルを左から順に並べた列である．例えば，図 4 の構文木対の節点間の対応関係は，添字のように定められる．

ルールは，構文木対から対応関係の存在する節点の子供以下を削除し，残った部分を抽出する．例えば，図 4 の構文木対からは，図 2 の基本木の対が抽出される．

3.3 頻度情報に基づくルールの選択

上述の手順で得られたルールは，擬似パラレルコーパスから得たものであるが，擬似パラレルコーパス中の部分構文木の対には，誤りを含んだ部分構文木

と誤りを含まない部分構文木の対もあれば，そうでない対もある．したがって，得られたルール集合には，誤り訂正に使用できるルールも存在すれば，そうでないルールも存在することになる．そこで，本手法では，コーパスにおける基本木の出現頻度に基づき，誤り訂正のためのルールを選別する．

基本的な考え方は以下の通りである．コーパスには誤りが含まれるとはいえ，その頻度は非常に小さいと考えられる．したがって，出現頻度の少ない基本木を出現頻度の多い基本木に変換するようなルールが，誤り訂正を実現するルールである可能性が高いと考えられる．この考え方に基づき，以下の評価尺度を定義する．

$$Score(\langle s,t\rangle) = \frac{f(t)}{f(s)+f(t)}$$

ここで，s はソース側の基本木，t はターゲット側の基本木を表し，$f(\cdot)$ はコーパスにおける基本木の出現頻度を表す．$Score(\langle s,t\rangle)$ は，$(0,1)$ の範囲の値をとる．ソース側の基本木の出現頻度が，ターゲット側の基本木の出現頻度と比べて小さいほど，$Score(\langle s,t\rangle)$ は大きな値となる．この値が大きいとき，$\langle s,t\rangle$ は誤り訂正を実現するルールである可能性が高いといえる．

4 評価実験

提案手法の有効性を確認するために，構文木付きコーパスである Penn Treebank[6] を用いた実験を行った．

実験の概要を以下に示す．実験対象として，Penn Treebank の Wall Street Journal コーパスの全 49208 文を使用した．これに対して提案手法を適用することにより誤り訂正ルールを獲得した．獲得されたルールの総数は，8776 個であった（なお，ルール $\langle s,t\rangle$ が存在するとき，ルール $\langle t,s\rangle$ も必ず存在するが，これらは別のルールとしてカウントしている）．

Score 上位 100 位以内のルールについて，各ルールをコーパスの構文木に適用し，誤り訂正の精度を測定した．精度の定義は以下の通りである．

$$\text{精度} = \frac{\text{ルールが誤りを訂正した回数}}{\text{ルールが適用された回数}}$$

ルールが適用された回数の総数は 331 回であり，そのうち誤りが訂正できたのは，238 回であった．誤り訂正の精度は 71.9%である．各ルールの精度をそれぞれ評価したところ，70 個のルールが精度 100%に達しており，多くのルールが正しく誤りを訂正している．また，これら 70 個のルールのうち，30 個のルールが構造の変更を伴う誤り訂正であった．このことは，構造を変更しなければ訂正できない誤りがコーパスに実在し，そのような誤りが本手法により訂正できることを示している．

獲得された誤り訂正ルールの例を図 6 にいくつか挙げる．(1) は，前置詞句 PP を付加する位置の誤りを訂正するルールである．(2) は，主語の位置に余分に挿入されている名詞句 NP を取り除くルールである．(3) は，ラベルの誤りを訂正するルールである．

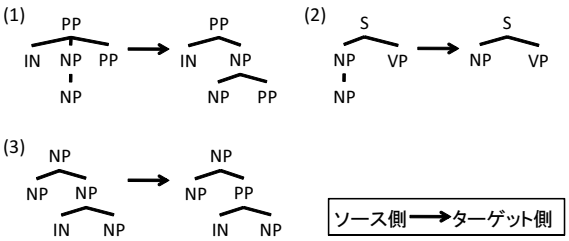

図 6: 獲得された誤り訂正ルールの例

5 おわりに

本稿では，同期文法を用いた構文木付きコーパスの誤り訂正手法を提案した．提案手法では，対象となるコーパスから擬似的にパラレルコーパスを作成し，そこから誤り訂正のためのルールを獲得する．実験により，高精度な誤り訂正ルールを獲得できることを確認した．

本手法により獲得した誤り訂正ルールのカバーできる範囲は大きくないと考えられるが，その範囲を拡大するのが今後の課題である．

参考文献

[1] T. Cohn and M. Lapata, “Sentence Compression as Tree Transduction,” Journal of Artificial Intelligence Research, 34(1):637–674, 2009.

[2] M. Dickinson and D. Meurers, “Detecting Errors in Part-of-Speech Annotation,” Proc. 10th EACL, pp. 107–114, 2003.

[3] M. Dickinson and D. Meurers, “Detecting Inconsistencies in Treebanks,” Proc. 2nd Workshop on Treebanks and Linguistic Theories, 2003.

[4] M. Dickinson and W. D. Meurers, “Prune Diseased Branches to Get Healthy Trees! How to Find Erroneous Local Trees in a Treebank and Why It Matters,” Proc. 4th Workshop on Treebanks and Linguistic Theories, 2005.

[5] J. Eisner, “Learning Non-Isomorphic Tree Mappings for Machine Translation,” Proc. 41st ACL, Companion Volume, pp. 205–208, 2003.

[6] M. P. Marcus, B. Santorini, and M. A. Marcinkiewicz, Building “A Large Annotated Corpus of English: the Penn Treebank,” Computational Linguistics, 19(2):310–330, 1993.

[7] T. Ule and K. Simov, “Unexpected Productions May Well be Errors,” Proc. 4th LREC, pp. 1795–1798, 2004.

[8] 乾孝司, 乾健太郎, “統計的部分係り受け解析における係り受け確率の利用法 — コーパス中の構文タグ誤りの検出 —,” 情処研報 NL-134, pp. 15–22, 1999.