統語ベース翻訳のための構文解析器の自己学習

森下 睦[†]・赤部 晃一[†]・波多腰 優斗[††]・Graham Neubig[†]・
吉野 幸一郎[†]・中村 哲[†]

構文情報を考慮する機械翻訳手法である統語ベース翻訳では，構文解析器の精度が翻訳精度に大きな影響を与えることが知られている．また，構文解析の精度向上を図る手法の一つとして，構文解析器の出力を学習データとして用いる構文解析器の自己学習が提案されている．しかし，構文解析器が生成する構文木には誤りが存在することから，自動生成された構文木が常に精度向上に寄与するわけではない．そこで本論文では，機械翻訳における自動評価尺度を用いて，このような誤った構文木を学習データから取り除き，自己学習の効果を向上させる手法を提案する．具体的には，解析された n-best 構文木それぞれを用いて統語ベース翻訳を行い，それぞれの翻訳結果に対し，自動評価尺度でリスコアリングする．この中で，良いスコアを持つ構文木を自己学習に使用することで，構文構造はアノテーションされていないが，対訳が存在するデータを用いて，構文解析・機械翻訳の精度を向上させることができる．実験により，本手法で自己学習したモデルを用いることで，統語ベース翻訳システムの翻訳精度が 2 つの言語対で有意に向上し，また構文解析自体の精度も有意に向上することが確認できた．

キーワード：機械翻訳, 構文解析, 自己学習

Parser Self-Training for Syntax-Based Machine Translation

Makoto Morishita[†], Koichi Akabe[†], Yuto Hatakoshi[†], Graham Neubig[†],
Koichiro Yoshino[†] and Satoshi Nakamura[†]

In syntax-based machine translation, it is known that the accuracy of parsing greatly affects the translation accuracy. Self-training, which uses parser output as training data, is one method to improve the parser accuracy. However, because automatically generated parse trees often include errors, these parse trees do not always contribute to improving accuracy. In this paper, we propose a method for removing noisy incorrect parse trees from the training data to improve the effect of self-training by using automatic evaluation metrics of translations. Specifically, we perform syntax-based machine translation using n-best parse trees, then we re-scoring parse trees based on the automatic evaluation score of translations. By using the parse trees that have higher score among the candidates for self-training, we can improve parsing and machine translation accuracy by using parallel corpora that are not annotated syntax structure. In experiments, using higher score parse trees for self-training, we found

[†] 奈良先端科学技術大学院大学, Nara Institute of Science and Technology
[††] 奈良先端科学技術大学院大学, 現在セイコーエプソン株式会社, Nara Institute of Science and Technology, currently with Seiko Epson Corporation

自然言語処理 Vol. 13 No. 2 April 2006

that our self-trained parsers significantly improve a state-of-the-art syntax-based machine translation system in two language pairs, and self-trained parsers significantly improve the accuracy of the parsing itself.

Key Words: *Machine Translation, Syntax Parsing, Self-Training*

1 はじめに

統計的機械翻訳 (Statistical Machine Translation, SMT) では，翻訳モデルを用いてフレーズ単位で翻訳を行い, 並べ替えモデルを用いてそれらを正しい語順に並べ替えるフレーズベース翻訳 (Phrase Based Machine Translation) (Koehn, Och, and Marcu 2003)，構文木の部分木を翻訳に利用する統語ベース翻訳 (Yamada and Knight 2001) などの翻訳手法が提案されている．一般的に，フレーズベース翻訳は英仏間のような語順が近い言語間では高い翻訳精度を達成できるものの，日英間のような語順が大きく異なる言語間では翻訳精度は十分でない．このような語順が大きく異なる言語対においては，統語ベース翻訳の方がフレーズベース翻訳と比べて高い翻訳精度を達成できることが多い．

統語ベース翻訳の中でも，原言語側の構文情報を用いる Tree-to-String (T2S) 翻訳 (Liu, Liu, and Lin 2006) は，高い翻訳精度と高速な翻訳速度を両立できる手法として知られている．ただし，T2S 翻訳は翻訳に際して原言語の構文解析結果を利用するため，翻訳精度は構文解析器の精度に大きく依存する (Neubig and Duh 2014)．この問題を改善する手法の一つとして，複数の構文木候補の集合である構文森をデコード時に利用する Forest-to-String (F2S) 翻訳 (Mi and Huang 2008) が挙げられる．しかし，F2S 翻訳も翻訳精度は構文森を作成した構文解析器の精度に大きく依存し，構文解析器の精度向上が課題となる (Neubig and Duh 2014)．

構文解析器の精度を向上させる手法の一つとして，構文解析器の自己学習が提案されている (McClosky, Charniak, and Johnson 2006)．自己学習では，アノテーションされていない文を既存のモデルを使って構文解析し，自動生成された構文木を学習データとして利用する．これにより，構文解析器は自己学習に使われたデータに対して自動的に適応し，語彙や文法構造の対応範囲が広がり，解析精度が向上する．しかし，自動生成された構文木は多くの誤りを含み，それらが学習データのノイズとなることで自己学習の効果を低減させてしまうという問題が存在する．

Katz-Brown ら (Katz-Brown, Petrov, McDonald, Och, Talbot, Ichikawa, Seno, and Kazawa 2011) は構文解析器の自己学習をフレーズベース翻訳のための事前並べ替えに適用する手法を提案している．フレーズベース翻訳のための事前並べ替えとは，原言語文の単語を目的言語の語順に近くなるように並べ替えることによって，機械翻訳の精度を向上させる手法である．この手法では，構文解析器を用いて複数の構文木候補を出力し，この構文木候補を用いて事前並べ

替えを行う．その後，並べ替え結果を人手で作成された正解並べ替えデータと比較することによって，各出力にスコアを割り振る．これらの並べ替え結果のスコアを基に，構文木候補の中から最も高いスコアを獲得した構文木を選択し，この構文木を自己学習に使用する．このように，学習に用いるデータを選択し，自己学習を行う手法を標的自己学習 (Targeted Self-Training) という．Katz-Brown らの手法では，正解並べ替えデータを用いて，自己学習に使用する構文木を選択することで，誤った並べ替えを行う構文木を取り除くことができ，学習データのノイズを減らすことができる．また，Liu ら (Liu, Li, Li, and Zhou 2012) は，単語アライメントを利用して構文解析器の標的自己学習を行う手法を提案している．一般に，構文木と単語アライメントの一貫性が取れている場合，その構文木は正確な可能性が高い．そのため，この一貫性を基準として構文木を選択し，それらを用いて構文解析器を学習することでより精度が向上することが考えられる．

以上の先行研究を基に，本論文では，機械翻訳の自動評価尺度を用いた統語ベース翻訳のための構文解析器の標的自己学習手法を提案する．提案手法は，構文解析器が出力した構文木を基に統語ベース翻訳を行い，その翻訳結果を機械翻訳の自動評価尺度を用いて評価し，この評価値を基にデータを選択し構文解析器の自己学習を行う．統語ベース翻訳では，誤った構文木が与えられた場合，翻訳結果も誤りとなる可能性が高く，翻訳結果を評価することで間接的に構文木の精度を評価することができる．以上に加え，提案手法は大量の対訳コーパスから自己学習に適した文のみを選択し学習を行うことで，自己学習時のノイズを減らす効果がある．

Katz-Brown らの手法と比較して，提案手法は事前並べ替えだけでなく統語ベース翻訳にも使用可能なほか，機械翻訳の自動評価尺度に基づいてデータの選択を行うため，対訳以外の人手で作成された正解データを必要としないという利点がある．これにより，既存の対訳コーパスが構文解析器の標的自己学習用学習データとして使用可能になり，構文解析器の精度や F2S 翻訳の精度を幅広い分野で向上させることができる．また，既に多く存在する無償で利用可能な対訳コーパスを使用した場合，本手法におけるデータ作成コストはかからない．さらに，Liu らの手法とは異なり，翻訳器を直接利用することができる利点もある．このため，アライメント情報を通して間接的に翻訳結果への影響を計測する Liu らの手法に比べて，直接的に翻訳結果への影響を構文木選択の段階で考慮できる．

実験により，提案手法で学習した構文解析器を用いることで，F2S 翻訳システムの精度向上と，構文解析器自体の精度向上が確認できた [1]．

[1] 本論文では, *IWSLT 2015: International Workshop on Spoken Language Translation* で発表した内容 (Morishita, Akabe, Hatakoshi, Neubig, Yoshino, and Nakamura 2015) に加え，翻訳システムの人手評価を実施した結果をまとめた．

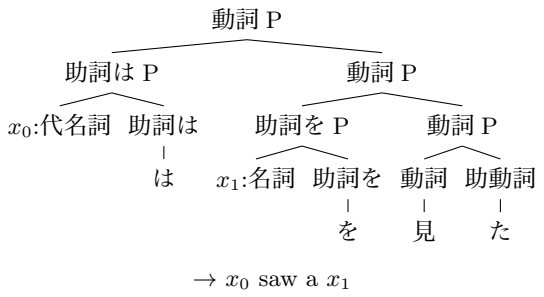


図 1 日英 T2S 翻訳における翻訳ルールの例

2 Tree-to-String 翻訳

SMT では，原言語文 $\boldsymbol{f}$ が与えられた時に，目的言語文 $\boldsymbol{e}$ へと翻訳される確率 $Pr(\boldsymbol{e}|\boldsymbol{f})$ を最大化する $\hat{\boldsymbol{e}}$ を推定する問題を考える．

$$\hat{\boldsymbol{e}} \coloneqq \underset{\boldsymbol{e}}{\operatorname{argmax}} Pr(\boldsymbol{e}|\boldsymbol{f}) \tag{1}$$

様々な手法が提案されている SMT の中でも，T2S 翻訳は原言語文の構文木 $T_{\boldsymbol{f}}$ を使用することで，原言語文に対する解釈の曖昧さを低減し，原言語と目的言語の文法上の関係をルールとして表現することで，より精度の高い翻訳を実現する．T2S 翻訳は下記のように定式化される．

$$\hat{\boldsymbol{e}} \coloneqq \underset{\boldsymbol{e}}{\operatorname{argmax}} Pr(\boldsymbol{e}|\boldsymbol{f}) \tag{2}$$

$$= \underset{\boldsymbol{e}}{\operatorname{argmax}} \sum_{T_{\boldsymbol{f}}} Pr(\boldsymbol{e}|\boldsymbol{f}, T_{\boldsymbol{f}}) Pr(T_{\boldsymbol{f}}|\boldsymbol{f}) \tag{3}$$

$$\simeq \underset{\boldsymbol{e}}{\operatorname{argmax}} \sum_{T_{\boldsymbol{f}}} Pr(\boldsymbol{e}|T_{\boldsymbol{f}}) Pr(T_{\boldsymbol{f}}|\boldsymbol{f}) \tag{4}$$

$$\simeq \underset{\boldsymbol{e}}{\operatorname{argmax}} Pr(\boldsymbol{e}|\hat{T}_{\boldsymbol{f}}) \tag{5}$$

ただし， $\hat{T}_{\boldsymbol{f}}$ は構文木の候補の中で，最も確率が高い構文木であり，下記の式で表される．

$$\hat{T}_{\boldsymbol{f}} = \underset{T_{\boldsymbol{f}}}{\operatorname{argmax}} Pr(T_{\boldsymbol{f}}|\boldsymbol{f}) \tag{6}$$

図 1 に示すように，T2S 翻訳 [2] で用いられる翻訳ルールは，置き換え可能な変数を含む原言語文構文木の部分木と，目的言語文単語列の組で構成される．図 1 の例では，x_0，x_1 が置き換

[2] 具体的には，木トランスデューサ (Tree Transducers) を用いた T2S 翻訳．

え可能な変数である．これらの変数には，他のルールを適用することにより翻訳結果が挿入され，変数を含まない出力文となる．訳出の際は，翻訳ルール自体の適用確率や言語モデル，その他の特徴などを考慮して最も事後確率が高い翻訳結果を求める．また，ビーム探索などを用いることで確率の高い n 個の翻訳結果を出力することが可能であり，これを n-best 訳という．

T2S 翻訳では，原言語文の構文木を考慮することで，語順が大きく異なる言語対の翻訳がフレーズベース翻訳と比べて正確になる場合が多い．しかし，T2S 翻訳は翻訳精度が構文解析器の精度に大きく依存するという欠点がある．この欠点を改善するために，複数の構文木を構文森と呼ばれる超グラフ (Hyper-Graph) の構造で保持し，複数の構文木を同時に翻訳に使用する F2S 翻訳 (Mi and Huang 2008) が提案されている．この場合，翻訳器は複数ある構文木の候補から構文木を選択することができ，翻訳精度の改善が期待できる (Zhang and Chiang 2012)．F2S 翻訳は $\boldsymbol{e}$ と $T_{\boldsymbol{f}}$ の同時確率の最大化として下記のように定式化される．

$$\langle \hat{\boldsymbol{e}}, \hat{T}_{\boldsymbol{f}} \rangle \coloneqq \underset{\langle \boldsymbol{e}, T_{\boldsymbol{f}} \rangle}{\operatorname{argmax}} Pr(\boldsymbol{e}, T_{\boldsymbol{f}} | \boldsymbol{f}) \tag{7}$$

$$\simeq \underset{\langle \boldsymbol{e}, T_{\boldsymbol{f}} \rangle}{\operatorname{argmax}} Pr(\boldsymbol{e} | T_{\boldsymbol{f}}) Pr(T_{\boldsymbol{f}} | \boldsymbol{f}) \tag{8}$$

しかし，1 節で述べたとおり F2S 翻訳であっても翻訳精度は構文森を生成する構文解析器の精度に大きく依存する．そこで，この問題を解決するため，自己学習によって構文解析器の精度を向上する手法について説明する．

3 構文解析の自己学習

3.1 自己学習の概要

構文解析器の自己学習とは，既存のモデルで学習した構文解析器が解析・生成した構文木を学習データとして用いることで，構文解析器を再学習する手法である．言い換えると，自己学習対象の各文に対して，式 (6) に基づいて確率が最も高い構文木 $\hat{T}_{\boldsymbol{f}}$ を求め，この構文木を構文解析器の再学習に用いる．この手法は追加のアノテーションを必要としないため，構文解析器の学習データ量が大幅に増え，解析精度が向上する．

Charniak は，Wall Street Journal (WSJ) コーパス (Marcus, Marcinkiewicz, and Santorini 1993) によって学習された確率文脈自由文法 (Probabilistic Context-Free Grammar, PCFG) モデルを用いた構文解析器では，自己学習の効果は得られなかったと報告している (Charniak 1997)．一方で，潜在クラスを用いることで構文解析の精度を向上させた PCFG-LA (PCFG with Latent Annotations) モデルは自己学習により大幅に解析精度が向上することが知られている (Huang and Harper 2009)．これは，PCFG-LA モデルを用いることで自動生成された構文木の精度が比較的高くなることに加え，PCFG-LA モデルが通常の PCFG モデルと比べて多くのパラメータ

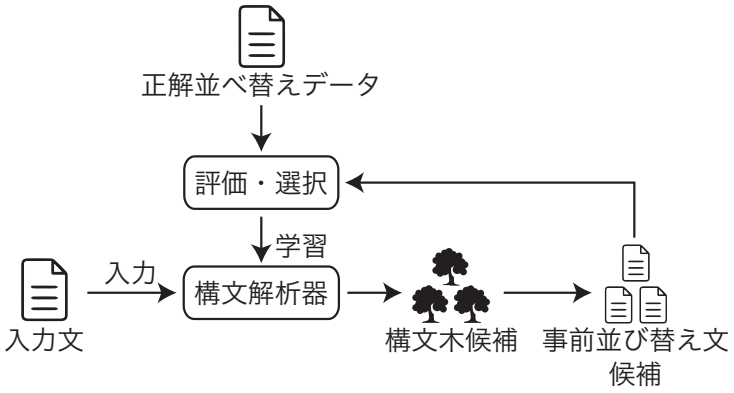


図 2 事前並べ替えのための標的自己学習手法

を持つので，学習データが増加する恩恵が大きいことが理由として挙げられる．これらの先行研究を基にして，本論文では PCFG-LA モデルを用いた構文解析器の自己学習を考える．

3.2 機械翻訳における構文解析器の自己学習と効果

3.2.1 事前並べ替えのための標的自己学習

1 節でも述べたように，構文解析器の自己学習により機械翻訳精度を改善した研究が存在する．Katz-Brown らは，自己学習に使用する構文木を外部評価指標を用いて選択する手法を提案し，これにより翻訳精度自体も向上したと報告している (Katz-Brown et al. 2011)．この手法の概要を図 2 に示す．この研究では，構文解析器により複数の構文木候補を自動生成し，これらの候補を基にした事前並べ替えの結果が人手で作成した正解並べ替えデータに最も近いものを選択し，自己学習に使用する．これにより，構文木の候補からより正しい構文木を選択することができるため，自己学習の効果が増すと報告されている．このように一定の基準を基に学習データを選択し，自己学習を行う手法を標的自己学習 (Targeted Self-Training) という．

事前並べ替えでは，構文木 $T_{\boldsymbol{f}}$ に基づいて，並べ替えられた原言語文 $\boldsymbol{f}'$ を生成する並べ替え関数 reord($T_{\boldsymbol{f}}$) を定義し，システムによる並べ替えを正解並べ替え $\boldsymbol{f}'^{*}$ と比較するスコア関数 score($\boldsymbol{f}'^{*}, \boldsymbol{f}'$) で評価する．学習に使われる構文木 $\bar{T}_{\boldsymbol{f}}$ は，構文木の候補 $\boldsymbol{T}_{\boldsymbol{f}}$ から以下の式によって選択される．

$$\bar{T}_{\boldsymbol{f}} = \underset{T_{\boldsymbol{f}} \in \boldsymbol{T}_{\boldsymbol{f}}}{\operatorname{argmax}} \operatorname{score}(\boldsymbol{f}'^{*}, \operatorname{reord}(T_{\boldsymbol{f}})) \tag{9}$$

本論文ではこれらの先行研究を基に，統語ベース翻訳のための構文解析器の標的自己学習手法を提案する．4 節以降において，提案手法の詳細を示し，実験により提案手法の効果を検証する．

3.2.2 フロンティアノードを利用した構文解析器の標的自己学習

統語ベース翻訳を利用して構文解析器の標的自己学習を行う手法として，フロンティアノード (Frontier Node) を利用する手法が提案されている (Liu et al. 2012).

一般に，構文木と単語アライメントの一貫性が取れている場合，その構文木は正確な可能性が高い．この一貫性を評価する指標として，フロンティアノードの数が挙げられる．フロンティアノードとは，対象ノードから翻訳ルールを抽出できるノードのことを指す．例として，5 つのフロンティアノードを持つ構文木を図 3 に示す．図中の灰色で示されたノードがフロンティアノードである．各ノード中の数字は上から順にスパン (span)，補完スパン (complement span) を示している [3]．ノード N のスパンとは，ノード N から到達可能な全ての目的言語単語の最小連続単語インデックスの集合であり，ノード N の補完スパンとは，N および N の子孫ノード以外のノードに対応する，目的言語単語インデックスの和集合とする [4]．フロンティアノードは，スパンと補完スパンが重複しておらず，かつスパンが null でないという条件を満たすノードのことを指す (Galley et al. 2006)．フロンティアノードの数が多いほど，構文木と単語アライメントの一貫性が取れているといえる．

例えば，「助詞の P」のスパンは 3-5 であり，“of this restaurant” に対応する．また，補完スパンは 1-2, 4 であり，“the speciality”, “this” に対応する．この場合，3-5 のスパンと，4 の補完スパンが部分的に重複しているためフロンティアノードではない．

Liu らの手法では，構文解析結果の 5-best の中からフロンティアノードの数が最も多くなる構文木を選択し，選ばれた構文木を自己学習に使用する．これにより，5-best 中で最も精度が高いと考えられる構文木を選択することができ，従来の自己学習手法よりも効果が高くなる．この手法で自己学習した構文解析器により，統語ベース翻訳の翻訳精度が有意に改善されたと報告されている．本論文では，Liu らの手法も比較対象として実験を行う [5].

4 統語ベース翻訳のための構文解析器の標的自己学習

標的自己学習において，どのように自己学習用のデータを選択するかは最も重要な点である．本論文では F2S 翻訳の精度を向上させるために，自己学習に使用する構文木および文の選択法をいくつか提案する．構文木の選択法を用いることで，一つの文の構文木候補から精度向上に

[3] スパンおよび補完スパンは，文献によってはアライメントスパン (alignment span)，補完アライメントスパン (complement alignment span) とも称される.

[4] フロンティアノードについての詳細は (Galley, Graehl, Knight, Marcu, DeNeefe, Wang, and Thayer 2006) を参照.

[5] Liu らは翻訳器が強制的に参照訳を出力する forced decoding を用いる手法も提案しているが，これを実現するために特殊なデコーダが必要であり，翻訳に用いるデコーダ自体に大幅な変更を加える必要がある．そのため，フロンティアノードに基づく手法や本研究の提案法に比べて実装が困難である．ゆえに，本論文ではフロンティアノードに基づく手法のみを比較対象とした.

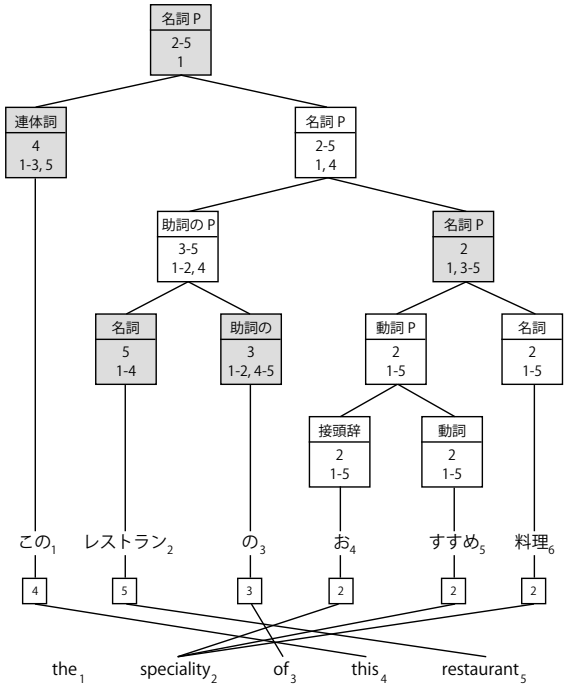


図 3 5 つのフロンティアノードを含む構文木の例

つながる構文木を選択し，文の選択法を用いることで，コーパス全体から精度向上に有効な文のみを選択する．以降ではそれぞれの手法について説明する．

4.1 構文木の選択法

3.2.1 節で述べたように，Katz-Brown らによって提案された標的自己学習手法 (Katz-Brown et al. 2011) では，自動生成された構文木と人手で作成した正解並べ替えデータを比較することにより，n-best 候補の中から最も評価値の高い構文木を選択する．しかし，人手で正解並べ替えデータを作成するには大きなコストがかかるため，この手法のために大規模なデータセットを作成することは現実的でない．一方で，統計的機械翻訳は対訳コーパスの存在を前提としており，対訳データは容易に入手できることが多い．そこで，この問題を解決するために，対訳コーパスのみを使用して構文木を選択する方法を 2 つ提案する．一つは，翻訳器によって選択された 1-best 訳に使われた構文木を自己学習に使用する手法である (翻訳器 1-best)．もう一つは，n-best 訳の中から，最も参照訳に近い訳 (Oracle 訳) を自動評価尺度により選択し，Oracle 訳に使われた構文木を自己学習に使用する手法である (自動評価尺度 1-best)．

4.1.1 翻訳器 1-best

2 節でも述べたように，F2S 翻訳では，構文森から翻訳確率が高くなる構文木を翻訳器が選択する．先行研究では，翻訳ルールや言語モデルの確率を使用することで，F2S 翻訳器は構文森から正しい構文木を選択する能力があることが報告されている (Zhang and Chiang 2012)．翻訳器の事後確率を用いることで，構文解析器だけでは考慮できない特徴を使用して構文木を選択するため，F2S 翻訳器が出力した 1-best 訳に使われた構文木は，構文解析器が出力した 1-best 構文木よりも自己学習に効果的だと考えられる．この際の自己学習に使われる構文木は式 (7) の $\hat{T}_{\boldsymbol{f}}$ となる．

4.1.2 自動評価尺度 1-best

翻訳の際，翻訳器は複数の翻訳候補の中から，最も翻訳確率が高い訳を 1-best 訳として出力する．しかし，実際には翻訳候補である n-best 訳の方が，翻訳器が出力した 1-best 訳よりも翻訳精度が高いと考えられる場合が存在する．そこで本論文では，翻訳候補の集合 $\boldsymbol{E}$ の中から最も参照訳 $\boldsymbol{e}^*$ に近い訳を Oracle 訳 $\bar{\boldsymbol{e}}$ と定義し，$\bar{\boldsymbol{e}}$ に使われた構文木を自己学習に使用する．翻訳候補 $\boldsymbol{e}$ と参照訳 $\boldsymbol{e}^*$ の類似度を表す評価関数 score(·) を用いて，Oracle 訳 $\bar{\boldsymbol{e}}$ は下記の通り表される．

$$\bar{\boldsymbol{e}} = \underset{\boldsymbol{e} \in \boldsymbol{E}}{\operatorname{argmax}} \operatorname{score}(\boldsymbol{e}^*, \boldsymbol{e}) \tag{10}$$

4.2 文の選択法

4.1 節では，1 つの対訳文の n-best 訳から学習に有用だと考えられる構文木を選択する方法について述べた．しかし，正しい訳が n-best 訳の中に含まれていない場合もあり，これらの例を学習に用いること自体が構文解析器の精度低下を招く可能性がある．そのため，n-best 訳の中に良い訳が含まれていない場合その文を削除するように，学習データ全体から自己学習に用いる文を選択する手法を提案する．具体的には，翻訳文の自動評価値が一定の閾値を超えた文のみを学習に使用する (自動評価値の閾値)．また，学習データ全体から翻訳精度を改善すると考えられる構文木を選択する手法も提案する．具体的には，翻訳器 1-best 訳と Oracle 訳の自動評価値の差が大きい文のみを使用する (自動評価値の差)．従来の標的自己学習手法では，構文木の選択手法は提案されていたものの，文の選択手法については検討されていなかった．本論文では，この文の選択手法についても検討を進める．

文の選択法を使用する場合は，構文木の選択法として，自動評価尺度 1-best を使用する．自動評価尺度 1-best を用いて構文木を選択する手法と，文の選択法を組み合わせた提案手法を図にすると，図 4 のようになる．図のように原言語文を構文解析器に入力し，出力された構文森を翻訳器に入力する．これにより n-best 訳と，翻訳に使われた構文木のペアが出力される．その

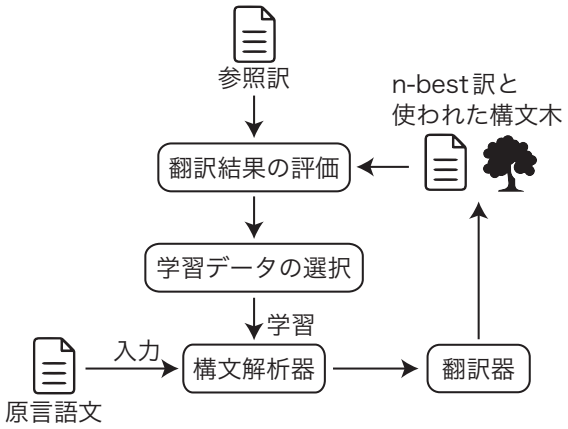


図 4 提案手法の概要

後，参照訳を基に自動評価尺度を用いて n-best 訳をリスコアリングする．これを基に学習データを選択し，自己学習を行う．

4.2.1 自動評価値の閾値

本節では，学習のノイズとなる誤った構文木を極力除外するために，自動評価値を基にデータを選択する手法を提案する．

コーパスの中には，翻訳器が正しく翻訳することができず，自動評価値が低くなってしまう文が存在する．自動評価値が低くなる原因としては以下のような理由が考えられる．

- 誤った構文木が翻訳に使用された．
- 翻訳モデルが原言語文の語彙やフレーズに対応できていない．
- 自動評価値を計算する際に用いられる参照訳が，意訳となっていたり，誤っていたりするため，翻訳器が参照訳に近い訳を出力することができない．

このような場合は，たとえ Oracle 訳であっても自動評価値は低くなってしまうことがある．これらのデータは，F2S 翻訳器が正しい構文木を選択することができていない場合や，自動評価尺度が実際の翻訳品質との相関が低い場合があるため，Oracle 訳で使われた構文木であっても誤った構文情報を持つ可能性がある．そのため，これらのデータを学習データから取り除くことで，学習データ中のノイズが減ると考えられる．そこで，より正確な構文木のみを使用するために，Oracle 訳の自動評価値が一定の閾値を超えた文のみ学習に使用する手法を提案する．自己学習に使用される構文木 $T_{f^{(i)}}$ の集合は下記の式のように定義される．ここで，t は閾値，$\boldsymbol{e}^{*(i)}$ は文 i の参照訳，$\bar{\boldsymbol{e}}^{(i)}$ は文 i の Oracle 訳，$\bar{\boldsymbol{E}}$ は Oracle 訳全体の集合，score(e) は訳の自動

評価関数を示す.

$$\{T_{f^{(i)}} \mid \text{score}(\boldsymbol{e}^{*(i)}, \bar{\boldsymbol{e}}^{(i)}) \geq t,\ \bar{\boldsymbol{e}}^{(i)} \in \bar{\boldsymbol{E}}\} \tag{11}$$

4.2.2 自動評価値の差

本節では，翻訳結果を大きく改善すると考えられる構文木を中心に選択する手法を提案する．この際に着目した指標は，翻訳器 1-best 訳と Oracle 訳の自動評価値の差である．

構文解析器により誤った構文木が高い確率を持った構文森が出力された場合，翻訳器は誤った構文木を選択し，誤った翻訳を 1-best 訳として出力することが多い．一方，Oracle 訳では構文森の中から正しい構文木が使われた可能性が高い．そのため，翻訳器 1-best 訳と Oracle 訳の自動評価値の差が大きい場合，Oracle 訳に使われた構文木を学習データとして使用することで，構文解析器が出力する確率が正しい値へ改善される可能性がある．これにより，自己学習した構文解析器を用いた翻訳システムは正しい訳を 1-best 訳として出力するようになり，翻訳精度が向上すると考えられる．

文を選択するために，翻訳器 1-best 訳 $\hat{\boldsymbol{e}}^{(i)}$ と Oracle 訳 $\bar{\boldsymbol{e}}^{(i)}$ の自動評価値の差を表す関数 $\text{gain}(\bar{\boldsymbol{e}}^{(i)}, \hat{\boldsymbol{e}}^{(i)})$ を定義し，式 (11) と同様に，自動評価値の差が大きい文の構文木を選択する．自動評価値の差を表す関数 $\text{gain}(\bar{\boldsymbol{e}}^{(i)}, \hat{\boldsymbol{e}}^{(i)})$ は下記のように定義される．

$$\text{gain}(\bar{\boldsymbol{e}}^{(i)}, \hat{\boldsymbol{e}}^{(i)}) = \text{score}(\boldsymbol{e}^{*(i)}, \bar{\boldsymbol{e}}^{(i)}) - \text{score}(\boldsymbol{e}^{*(i)}, \hat{\boldsymbol{e}}^{(i)}) \tag{12}$$

本手法ではこれに加えて，学習に用いる文の長さの分布をコーパス全体と同様に保つため，Gascó ら (Gascó, Rocha, Sanchis-Trilles, Andrés-Ferrer, and Casacuberta 2012) によって提案された下記の式を用いて，文の長さに応じて選択数を調節する [6]．以下の式では，$|\boldsymbol{e}|$ は目的言語文 $\boldsymbol{e}$ の長さ，$|\boldsymbol{f}|$ は原言語文 $\boldsymbol{f}$ の長さ，$N_c(|\boldsymbol{e}| + |\boldsymbol{f}|)$ はコーパス全体で目的言語文，原言語文の長さの和が $|\boldsymbol{e}| + |\boldsymbol{f}|$ となる文の数，N_c はコーパス全体の文数を表す．

$$p(|\boldsymbol{e}| + |\boldsymbol{f}|) = \frac{N_c(|\boldsymbol{e}| + |\boldsymbol{f}|)}{N_c}. \tag{13}$$

N_t を自己学習データ全体の文数とすると，自己学習データの内，目的言語文，原言語文の長さの和が $|\boldsymbol{e}| + |\boldsymbol{f}|$ となる文数 $N_t(|\boldsymbol{e}| + |\boldsymbol{f}|)$ は下記の式で表される．

$$N_t(|\boldsymbol{e}| + |\boldsymbol{f}|) = p(|\boldsymbol{e}| + |\boldsymbol{f}|)N_t. \tag{14}$$

[6] 自動評価値の差を基準に文を選択する手法では，文の長さの分布を考慮しない場合，短い文のみを選択する傾向があり，自己学習が正しく行えないことを予備実験で確認した．これは，短い文の場合，文が少し変わっただけでも自動評価値が大幅に上昇してしまうことが原因である．

表 1 ASPEC に含まれる対訳文数

	Train	Dev	DevTest	Test
Ja-En	2,000,000	1,790	1,784	1,812
Ja-Zh	672,315	2,090	2,148	2,107

5 評価

5.1 実験設定

本論文では，日本語の構文解析器を用いる日英・日中翻訳 (それぞれ Ja-En，Ja-Zh と略す) を対象に実験を行った．翻訳データとして，科学論文を抜粋した対訳コーパスである ASPEC[7] を用いた．ASPEC に含まれる対訳文数を表 1 に示す [8]．自己学習の効果を検証するためのベースラインシステムとして，アジア言語間での翻訳ワークショップ Workshop on Asian Translation 2014 (WAT2014) (Nakazawa, Mino, Goto, Kurohashi, and Sumita 2014) において高い精度を示した，Neubig のシステムを用いた (Neubig 2014)[9]．デコーダには Travatar (Neubig 2013) を用い，F2S 翻訳を行った．構文解析器には (Neubig and Duh 2014) で最も高い日英翻訳精度を実現した PCFG-LA モデルに基づく Egret[10] を用い，日本語係り受けコーパス (JDC) (Mori, Ogura, and Sasada 2014) (約 7000 文) に対して Travatar の主辞ルールで係り受け構造を句構造に変換 [11] したものを用いて学習したモデルを，ベースラインの構文解析器として使用した．構文森は 100-best 構文木に存在する hyper-edge のみで構成し，その他については枝刈りした [12]．機械翻訳の精度は BLEU (Papineni, Roukos, Ward, and Zhu 2002) と RIBES (Isozaki, Hirao, Duh, Sudoh, and Tsukada 2010) の 2 つの自動評価尺度，Acceptability (Goto, Lu, Chow, Sumita, and Tsou 2011) という人手評価尺度を用いて評価した．また，文単位の機械翻訳精度は BLEU+1 (Lin and Och 2004) を用いて評価した．自己学習に用いるデータは既存のモデルで使用している JDC に加え，ASPEC のトレーニングデータの中から選択されたものとした．自己学習したモデルは，テスト時に Dev セット，Test セットを構文解析する際のみに使用し，Train セットについては JDC で学習した既存のモデルで行った．Train セットについても自己学習したモデルで構文解析することにより，さらなる精度向上の可能性はあるが，翻訳器を学習し直すには

[7] http://lotus.kuee.kyoto-u.ac.jp/ASPEC
[8] 実際は ASPEC の Ja-En Train セットは 300 万文存在する．しかし，このデータは自動的に文対応が取られたため，対応が誤っている文がある．そのため，信頼できる上位 200 万文のみ使用し，学習データの質を確保した．
[9] http://github.com/neubig/wat2014
[10] http://code.google.com/p/egret-parser
[11] https://github.com/neubig/travatar/blob/master/script/tree/ja-adjust-dep.pl
https://github.com/neubig/travatar/blob/master/script/tree/ja-dep2cfg.pl
[12] Egret は極希に構文解析に失敗し，構文木を出力しない場合がある．そのため，構文解析に失敗した文は学習データから取り除いた．

多くの計算量が必要になってしまう．そのため，本実験では Dev セット，Test セットについてのみ自己学習したモデルで構文解析を行った．実験で得られた結果は，ブートストラップ・リサンプリング法 (Koehn 2004) により統計的有意差を検証した．次節では，下記の手法を比較評価する．

構文木の選択法

Parser 1-best

式 (6) のように，構文解析器が出力した 1-best 構文木を自己学習に用いる．

Frontier Node 1-best

3.2.2 節のように，既存の構文解析器が出力した 5-best 構文木の中から，フロンティアノードの数が最も多くなる構文木を学習に使用する．

MT 1-best

4.1.1 節のように，構文森を翻訳器に入力し，1-best 訳に使われた構文木を自己学習に使用する．

BLEU+1 1-best

4.1.2 節のように，構文森を翻訳器に入力し，翻訳器が出力した 500-best 訳の中から，最も BLEU+1 スコアが高い訳に使われた構文木を選択し，自己学習に用いる．この際，出力される n-best 訳は全て重複が無い文となるようにする．

文の選択法

Random

全トレーニングデータからランダムに文を選択する．この際ランダムに選択される文は手法毎に異ならず，同一になるようにする．[13].

BLEU+1$\geq t$

4.2.1 節のように，Oracle 訳とその構文木の中でも，訳の BLEU+1 スコアが閾値 t を超えた文のみを自己学習に使用する．

BLEU+1 Gain

4.2.2 節のように，Oracle 訳とその構文木の中でも，翻訳器 1-best 訳と Oracle 訳の BLEU+1 スコアの差が大きい文のみを自己学習に使用する．この手法では，文の長さの分布を式 (13)，(14) に従って調節する．

なお文をランダムに抽出する場合は，日英翻訳では全トレーニングデータの 1/20，日中翻訳では 1/10 を抽出した．また，他の手法とほぼ同様の文数となるように，BLEU+1 Gain に関し

[13] 大規模なコーパス全てを用いて構文解析器を学習するには多くの計算量が必要になるため，今回はランダムに抽出した文のみを使用する．

表 2 日英・日中翻訳の実験結果

	文選択手法	構文木選択手法	Ja-En			Ja-Zh		
			文数	BLEU	RIBES	文数	BLEU	RIBES
(a)	—	—	—	23.83	72.27	—	29.60	81.32
(b)	Random	Parser 1-best	96k	23.66	71.77	129k	29.75	‡ 81.55
(c)	Random	Frontier Node 1-best	96k	23.97	71.98	129k	29.83	81.46
(d)	Random	MT 1-best	97k	23.81	72.04	130k	29.76	‡ 81.53
(e)	Random	BLEU+1 1-best	97k	23.93	72.09	130k	‡ 29.89	‡ 81.66
(f)	BLEU+1 $\geq$ 0.7	BLEU+1 1-best	206k	‡ 24.27	72.38	240k	‡ 29.86	⋆ ‡ 81.60
(g)	BLEU+1 $\geq$ 0.8	BLEU+1 1-best	120k	⋆ ‡ 24.26	72.38	150k	‡ 29.91	81.47
(h)	BLEU+1 $\geq$ 0.9	BLEU+1 1-best	58k	⋆ ‡ 24.26	⋆ 72.49	82k	† 29.86	‡ 81.60
(i)	BLEU+1 Gain	BLEU+1 1-best	100k	⋆ † 24.22	72.32	100k	† 29.85	‡ 81.59
(j)	BLEU+1 $\geq$ 0.8 (Ja-En)	BLEU+1 1-best	—	—	—	120k	† 29.87	† 81.58

ては上位 10 万文を抽出した．

以下，5.2 節では，自己学習した構文解析モデルを使用して翻訳を行った際の翻訳器の精度評価，5.3 節では，構文解析器の精度評価を行う．

5.2 翻訳器の精度評価

各手法で自己学習した構文解析器を用いて，翻訳精度の変化を確認する．5.2.1 節では自動評価尺度を用いた評価結果を示し，5.2.2 節で人手評価による評価結果を示す．また，5.2.3 節では提案手法により改善された翻訳例を示し，どのような場合に提案手法が有効かを検討する．

5.2.1 自動評価尺度による翻訳精度の評価

日英・日中翻訳の実験結果を表 2 に示す．表中の短剣符は，提案手法の翻訳精度がベースラインシステムと比較して統計的に有意に高いことを示す ($\dagger : p < 0.05$, $\ddagger : p < 0.01$)．また，表中の星印は，提案手法の翻訳精度が Liu らの手法 (手法 (c)) と比較して統計的に有意に高いことを示す ($\star : p < 0.05$, $\star\star : p < 0.01$)．表 2 中の (b),(c),(d),(e) の手法で自己学習に使用している文は，Egret が構文解析に失敗した場合を除いて同一である [14]．なお，表中の “文数” は自己学習に使用した文数を示し，既存モデルで使用している JDC の文数は含まない．本実験では，BLEU+1 を文や構文木選択を行う際の指標としたため，以降では，主に BLEU スコアに着目して分析を行う．

実験により，以下の 3 つの仮説について検証を行った．

[14] 手法 (d),(e) では Egret が解析に失敗した場合，代替の構文解析器として JDC で学習した Ckylark (Oda, Neubig, Sakti, Toda, and Nakamura 2015) を用いた．

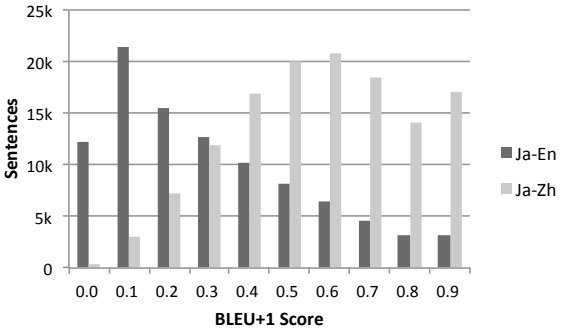


図 5 表 2 手法 (e) の BLEU+1 スコア分布

- 構文木の選択法を用いた標的自己学習 (4.1 節) は翻訳精度向上に効果があるのか
- 文の選択法 (4.2 節) は学習データ中のノイズを減らし，精度を向上させる効果があるのか
- 標的自己学習したモデルは目的言語に依存するのか，多言語に渡って使用できるのか

構文木の選択法による効果: Parser 1-best の構文木を自己学習に使用する手法では，日英，日中翻訳ともに BLEU スコアの向上は見られなかった (表 2 (b))．Frontier Node 1-best を用いた場合，Parser 1-best を用いた手法と比較して多少の精度向上は見られたものの，ベースラインとの有意な差は見られなかった (表 2 (c))．また，日英翻訳で MT 1-best を自己学習に用いた手法では，Parser 1-best を用いた手法と比較すると精度は向上したもののベースラインシステムと比較すると精度は向上しなかった (表 2 (d))．日中翻訳では，MT 1-best を使用した手法は Parser 1-best を用いた手法とほぼ同じ結果となった (表 2 (d))．この際に自己学習に用いられた構文木を確認したところ，正しい構文木もあるが誤った構文木も散見され，精度向上が確認できなかったのは誤った構文木が学習の妨げになったからだと考えられる．

次に BLEU+1 1-best を用いた手法では，Oracle 訳に使われた構文木が選択されることにより，BLEU スコアが日英，日中翻訳ともに向上していることがわかる (表 2 (e))．特に，日中翻訳についてはベースラインより有意に精度が向上している．図 5 に，この手法で自己学習に使われた Oracle 訳の BLEU+1 スコア分布を示す．横軸の値 x は，x 以上 $x+0.1$ 未満の BLEU+1 を持つ文を表しており，縦軸が該当する文の数である．この図からもわかるように，Oracle 訳であっても BLEU+1 スコアが低い文は多く存在する．このため，4.2 節の文選択を実施した．

文の選択法による効果: 次に，BLEU+1 スコアの閾値を用いた文選択手法の効果を確認する．結果から，日英翻訳，日中翻訳ともに，この手法は効果的であることがわかった (表 2 (f), (g), (h))．Liu らの手法 (表 2 (c)) と比較を行った場合でも，提案手法の一部では有意に高い精度が得られている．この結果から，自己学習を行う際には，精度が低いと思われる構文木を極力取り除き，精度が高いと思われる構文木のみを学習データとして使用することが重要であると言え

表 3 人手による日英翻訳精度の評価

	文選択手法	構文木選択手法	Score	(a) との有意差	(b) との有意差
(a)	—	—	2.38	—	—
(b)	Random	Parser 1-best	2.42	なし	—
(c)	BLEU+1 ≥ 0.8	BLEU+1 1-best	2.50	あり ($p < 0.01$)	あり ($p < 0.1$)

る．さらに，翻訳器 1-best 訳と Oracle 訳で BLEU+1 スコアの差が大きい文のみを使用する手法でも，BLEU+1 スコアの閾値を用いた手法と同程度の精度向上を達成することができた (表 2 (i))．

目的言語への依存性: 最後に，日英対訳文で自己学習し，日英翻訳の精度改善に貢献した構文解析器のモデルを，他の言語対である日中翻訳に使用した場合の翻訳精度の変化を検証した．興味深いことに，この場合でも直接日中対訳文で自己学習したモデルとほぼ同程度の翻訳精度の改善が見られた (表 2 (j))．これにより，学習されたモデルの目的言語に対する依存性はさほど強くなく複数の目的言語のデータを合わせて学習データとすることで，さらに効果的な自己学習が行える可能性があることが示唆された．

5.2.2 人手による翻訳精度の評価

提案手法により BLEU スコアは改善されたが，自動評価尺度は完璧ではなく，実際にどの程度の質で翻訳できたか明確に判断することは難しい．そのため，人手評価による翻訳精度の評価を行い，実際にどの程度翻訳の質が改善されたかを確認した．評価基準は，意味伝達と訳の自然性を両方加味する Acceptability (Goto et al. 2011) とした．本研究と関わりが無いプロの翻訳者に各翻訳文に対し評価基準を基に 5 段階のスコアをつけてもらい，これらの平均を評価値とする．評価は日英翻訳システムを対象とし，Test セットからランダムで抽出した 200 文について評価を行った．

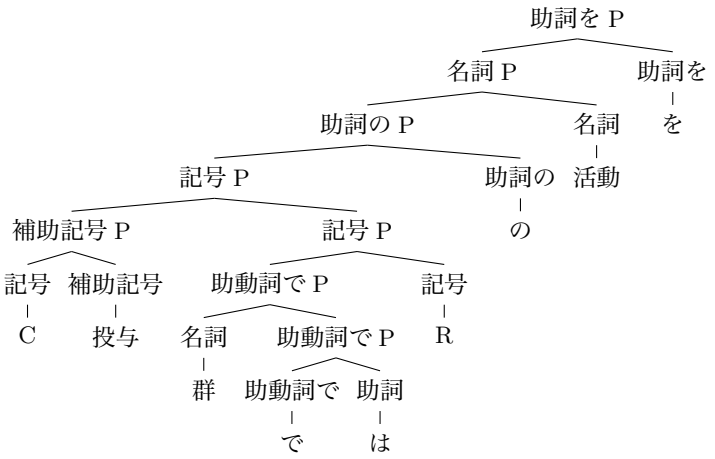
各システムの評価結果を表 3 に示す．既存の自己学習手法で学習したモデルでは，ベースラインシステムと比較して有意な翻訳精度向上は確認できなかった (表 3 (b))．一方，提案手法で自己学習したモデルでは，ベースラインシステムより有意に良い翻訳精度が実現できており，かつ既存の自己学習手法と比較しても $p < 0.1$ 水準ではあるが精度が向上している (表 3 (c))．このように，自動評価尺度だけでなく，人手評価でも提案手法で自己学習したモデルを用いることにより，有意に翻訳精度が向上することが確認できた．

5.2.3 自己学習による訳出改善の例

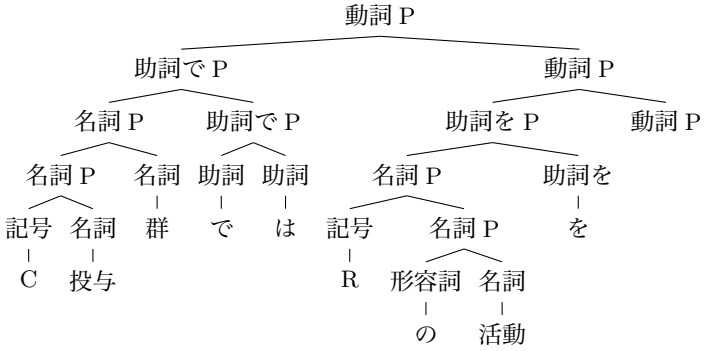
構文解析器の自己学習によって改善された日英訳の例を表 4 に示す．また，表 4 の訳出の際に使用された構文木を図 6 に示す．この文では，「C 投与群」と「R の活動」という名詞句が含

表 4 日英翻訳における訳出改善例

原言語文	C 投与 群 で は R の 活動 を 2 4 0 分 に わた って 明らか に 増強 し た 。
参照訳	in the C - administered group , thermal reaction clearly increased the activity of R for 240 minutes .
ベースライン	for 240 minutes clearly enhanced the activity of C administration group R .
BLEU+1≥0.8	for 240 minutes clearly enhanced the activity of R in the C - administration group .



(a) 自己学習前の構文木の例



(b) 自己学習後の構文木の例

図 6 自己学習により構文解析の精度が改善した例

まれている．ベースラインシステムの構文木は，これらの名詞句を正しく解析できておらず，この構文解析誤りが翻訳結果にも悪影響を与えてしまっている．一方，提案手法で自己学習したシステムでは，これらの名詞句を正しく解析できており，翻訳も正しく行われている．これは McClosky ら (McClosky, Charniak, and Johnson 2008) が報告していたように，既存モデル

表 5 自己学習した日本語構文解析器の精度

	文選択手法	構文木選択手法	再現率	適合率	F 値	(a) との有意差	(b) との有意差
(a)	—	—	84.88	84.77	84.83	—	—
(b)	Random	Parser 1-best	86.52	86.41	86.46	あり ($p < 0.05$)	—
(c)	Random	Frontier Node 1-best	87.77	87.65	87.71	あり ($p < 0.01$)	あり ($p < 0.05$)
(d)	Random	BLEU+1 1-best	87.69	87.58	87.63	あり ($p < 0.01$)	あり ($p < 0.05$)
(e)	BLEU+1 ≥ 0.7	BLEU+1 1-best	88.13	88.01	88.07	あり ($p < 0.01$)	あり ($p < 0.05$)
(f)	BLEU+1 ≥ 0.8	BLEU+1 1-best	88.13	88.01	88.07	あり ($p < 0.01$)	あり ($p < 0.05$)
(g)	BLEU+1 ≥ 0.9	BLEU+1 1-best	87.29	87.13	87.23	あり ($p < 0.01$)	なし

で使用している JDC で既知の単語が，ASPEC で異なる文脈で現れた際に解析精度が向上した結果であると考えられる．

5.3 構文解析器の精度評価

次に，提案手法により自己学習した構文解析器自体の精度を測定した．ASPEC に含まれる日英対訳データの内，Test セット中の 100 文を人手でアノテーションを行い，正解構文木を作成した．その後，各構文解析器の精度を Evalb[15] を用いて測定した．評価には，再現率，適合率，およびそれぞれの調和平均である F 値を用いる．表 5 に構文解析器の精度評価結果を示す．表中の短剣符は，提案手法の F 値がベースラインシステムと比較して統計的に有意に高いことを示す ($\dagger : p < 0.05$, $\ddagger : p < 0.01$)．

表からもわかるように，Parser 1-best を用いて自己学習したモデルはベースラインシステムと比較して $p < 0.05$ 水準で有意に精度が向上している．これに加えて，Frontier Node 1-best を用いた手法や提案手法で自己学習したモデルは $p < 0.01$ 水準で有意に精度が向上している．これらの結果から，提案手法は機械翻訳の精度だけでなく，構文解析器自体の精度もより向上させることがわかった．よって，本手法は構文解析器を分野適応させる場合においても有効であるといえる．

5.4 翻訳器の精度が低い場合の自己学習効果

提案手法では，翻訳結果を用いて間接的に構文木を評価し構文解析器を改良する．そのため，使用する翻訳器の精度によっては十分な学習効果が得られない可能性がある．本節では，精度が低い翻訳器を使用し構文解析器の自己学習を行い，翻訳精度と自己学習効果の依存性について検討する．なお本実験では日英翻訳のみを対象として実験を行った．

表 6 使用した対訳文数と翻訳精度

	Train	Dev	DevTest	Test	BLEU	RIBES
高精度翻訳器	2,000,000	1,790	1,784	1,812	23.83	72.27
低精度翻訳器	250,000	1,790	1,784	1,812	21.91	70.70

表 7 日英翻訳の実験結果 (低精度の翻訳器を用いて自己学習を行った場合)

	使用した翻訳器	文選択手法	構文木選択手法	文数	BLEU	RIBES
(a)	—	—	—	—	23.83	72.27
(b)	高精度	BLEU+1 $\geq$ 0.8	BLEU+1 1-best	120k	‡ 24.26	72.38
(c)	低精度	BLEU+1 $\geq$ 0.7	BLEU+1 1-best	212k	‡ 24.28	72.64
(d)	低精度	BLEU+1 $\geq$ 0.8	BLEU+1 1-best	127k	‡ 24.32	72.08
(e)	低精度	BLEU+1 $\geq$ 0.9	BLEU+1 1-best	66k	24.09	72.06

表 8 自己学習した日本語構文解析器の精度 (低精度の翻訳器を用いて自己学習を行った場合)

	使用した翻訳器	文選択手法	構文木選択手法	再現率	適合率	F 値	(a) との有意差
(a)	—	—	—	84.88	84.77	84.83	—
(b)	高精度	BLEU+1 $\geq$ 0.8	BLEU+1 1-best	88.13	88.01	88.07	あり ($p < 0.01$)
(c)	低精度	BLEU+1 $\geq$ 0.7	BLEU+1 1-best	88.29	88.18	88.24	あり ($p < 0.01$)
(d)	低精度	BLEU+1 $\geq$ 0.8	BLEU+1 1-best	86.32	86.21	86.26	なし
(e)	低精度	BLEU+1 $\geq$ 0.9	BLEU+1 1-best	87.76	87.64	87.70	あり ($p < 0.01$)

5.4.1 実験設定

200 万文使用していた学習データを 25 万文に制限し，低精度の翻訳器を新たに作成した．使用した対訳文数，および翻訳精度を表 6 に示す．学習データが減ったことにより，以前のシステムと比較して BLEU，RIBES ともに低下している．これ以外の条件は全て 5.1 節と同一である．なお，5.2 節の実験結果との比較のために，自己学習後の翻訳精度評価には 200 万文を使用して学習したシステムを用いる．

5.4.2 実験結果，考察

低精度翻訳器を自己学習時に使用し，自己学習した構文解析器を用いて翻訳器を構築しその精度を測定した．この実験結果を表 7 に示す．また，構文解析器自体の精度も 5.3 節と同様に測定した．測定結果を表 8 に示す．

結果は，低精度翻訳器を使用した場合でも，以前の高精度翻訳器を使用して自己学習を行った場合 (表 7 (b)) と遜色ない自己学習効果が得られた．構文解析器の精度自体も，高精度翻訳

[15] http://nlp.cs.nyu.edu/evalb

表 9 日英翻訳の実験結果 (2 回の繰り返し学習)

	自己学習回数	文選択手法	構文木選択手法	文数	BLEU	RIBES
(a)	—	—	—	—	23.83	72.27
(b)	1 回	BLEU+1 $\geq$ 0.8	BLEU+1 1-best	120k	‡ 24.26	72.38
(c)	2 回	BLEU+1 $\geq$ 0.7	BLEU+1 1-best	235k	23.89	72.49
(d)	2 回	BLEU+1 $\geq$ 0.8	BLEU+1 1-best	143k	23.97	72.17
(e)	2 回	BLEU+1 $\geq$ 0.9	BLEU+1 1-best	79k	† 24.20	72.35

表 10 自己学習した日本語構文解析器の精度 (2 回の繰り返し学習)

	自己学習回数	文選択手法	構文木選択手法	再現率	適合率	F 値
(a)	—	—	—	84.88	84.77	84.83
(b)	1 回	BLEU+1 $\geq$ 0.8	BLEU+1 1-best	88.13	88.01	88.07
(c)	2 回	BLEU+1 $\geq$ 0.7	BLEU+1 1-best	88.03	87.91	87.97
(d)	2 回	BLEU+1 $\geq$ 0.8	BLEU+1 1-best	87.29	87.17	87.23
(e)	2 回	BLEU+1 $\geq$ 0.9	BLEU+1 1-best	87.97	87.85	87.91

器を用いた場合 (表 8 (b)) と大きな差は無いことがわかった．これらの結果から，提案手法は既存翻訳器の翻訳精度に依存しないことが示された．これは，翻訳器が出した 500-best の中から Oracle 訳を選択しており，翻訳器が低精度の場合でも 500-best の中にはある程度誤りが少ない訳が含まれているため，比較的正確な構文木が選択できたからだと考えられる．そのため，n-best の n を変えるとこの結果は多少変化する可能性がある．

5.5 自己学習を繰り返し行った場合の効果

構文解析器の自己学習では，1 回自己学習を行った構文解析器をベースラインとして使用し 2 回目の自己学習を行うことで，さらなる精度向上が期待できる．本節では，自己学習を繰り返し行うことで，翻訳精度および構文解析精度にどのような影響が及ぶかを検証する．なお本実験では日英翻訳のみを対象として実験を行った．

本実験では，1 回自己学習を行ったものの構文解析モデルとして，表 2 中の (g) のモデルを用いる．その他の実験設定は 5.1 節と同一である．

自己学習を 2 回行った構文解析モデルを使用して翻訳精度を測定した結果を表 9 に示す．また，構文解析器の精度自体も 5.3 節と同様に測定した．測定結果を表 10 に示す．

実験より，2 回の繰り返し学習を行っても，1 度のみの場合と比較して翻訳精度，構文解析精度ともに向上は見られなかった．これは，学習時に 500-best の中から Oracle 訳を選択しているため，1 度目でも既にある程度精度の高い構文木が選ばれていたことが原因として考えられる．また，スコアを基に学習データを制限しているため，2 度目の学習時に改善された構文木であっ

ても，翻訳結果がスコアの制限を満たさず学習データとして使われなかった可能性がある．そのため，本手法では繰り返し学習の効果は薄いと考えられる．

6 おわりに

本論文では，統語ベース翻訳で用いられる構文解析器の標的自己学習手法を提案し，これにより F2S 翻訳および構文解析の精度が向上することを検証した．具体的には，日英，日中翻訳を対象に実験を行い，本手法で標的自己学習した構文解析器を用いることで，ベースラインシステムと比較して有意に高精度な翻訳結果を得られるようになったことが確認できた．また，日英で自己学習した構文解析器のモデルを，日中の翻訳の際に用いても同様に精度が向上することが確認できた．日英翻訳については訳の人手評価も実施し，人手評価においても有意に翻訳精度の改善が見られた．さらに，提案手法では翻訳精度だけでなく，構文解析の精度自体も向上することを実験により検証した．また，既存翻訳器の精度が十分でない場合でもこの手法は適用可能であることを確認した．本手法の繰り返し適用に関する検討も行ったが，本手法では繰り返し学習の効果は薄いと考えられる．

今後の課題としては，さらに多くの言語対で提案手法が適用可能であることを確認することが挙げられる．また，自己学習による効果は目的言語によらないという可能性が示唆されたため，実際に多言語で学習データを集めて適用することで，より翻訳精度および構文解析精度を向上させることが期待される．さらに，対訳コーパスに対して他の複数の構文解析器を用いて解析し，それらの解析結果が一致している文を正解とみなして構文解析器の学習に使用する tri-training との比較についても検討を行いたいと考えている．

謝　辞

本論文の一部は，JSPS 科研費 25730136 および 24240032 の助成を受け実施したものである．

参考文献

Charniak, E. (1997). “Statistical Parsing with a Context-Free Grammar and Word Statistics.” In *Proc. AAAI*, pp. 598–603.

Galley, M., Graehl, J., Knight, K., Marcu, D., DeNeefe, S., Wang, W., and Thayer, I. (2006). “Scalable inference and training of context-rich syntactic translation models.” In *Proc. ACL*, pp. 961–968.

Gascó, G., Rocha, M.-A., Sanchis-Trilles, G., Andrés-Ferrer, J., and Casacuberta, F. (2012). "Does more data always yield better translations?" In *Proc. ACL*, pp. 152–161.

Goto, I., Lu, B., Chow, K. P., Sumita, E., and Tsou, B. K. (2011). "Overview of the patent machine translation task at the NTCIR-9 workshop." In *Proc. NTCIR*, Vol. 9, pp. 559–578.

Huang, Z. and Harper, M. (2009). "Self-Training PCFG grammars with latent annotations across languages." In *Proc. EMNLP*, pp. 832–841.

Isozaki, H., Hirao, T., Duh, K., Sudoh, K., and Tsukada, H. (2010). "Automatic Evaluation of Translation Quality for Distant Language Pairs." In *Proc. EMNLP*, pp. 944–952.

Katz-Brown, J., Petrov, S., McDonald, R., Och, F., Talbot, D., Ichikawa, H., Seno, M., and Kazawa, H. (2011). "Training a Parser for Machine Translation Reordering." In *Proc. EMNLP*, pp. 183–192.

Koehn, P. (2004). "Statistical significance tests for machine translation evaluation." In *Proc. EMNLP*, pp. 388–395.

Koehn, P., Och, F. J., and Marcu, D. (2003). "Statistical phrase-based translation." In *Proc. HLT*, pp. 48–54.

Lin, C.-Y. and Och, F. J. (2004). "Orange: a method for evaluating automatic evaluation metrics for machine translation." In *Proc. COLING*, pp. 501–507.

Liu, S., Li, C.-H., Li, M., and Zhou, M. (2012). "Re-training Monolingual Parser Bilingually for Syntactic SMT." In *Proc. EMNLP*, pp. 854–862.

Liu, Y., Liu, Q., and Lin, S. (2006). "Tree-to-String Alignment Template for Statistical Machine Translation." In *Proc. ACL*, pp. 609–616.

Marcus, M. P., Marcinkiewicz, M. A., and Santorini, B. (1993). "Building a large annotated corpus of English: The Penn Treebank." *Computational linguistics*, **19** (2), pp. 313–330.

McClosky, D., Charniak, E., and Johnson, M. (2006). "Effective self-training for parsing." In *Proc. HLT*, pp. 152–159.

McClosky, D., Charniak, E., and Johnson, M. (2008). "When is Self-training Effective for Parsing?" In *Proc. COLING*, pp. 561–568.

Mi, H. and Huang, L. (2008). "Forest-based translation rule extraction." In *Proc. EMNLP*, pp. 206–214.

Mori, S., Ogura, H., and Sasada, T. (2014). "A Japanese Word Dependency Corpus." In *Proc. LREC*, pp. 753–758.

Morishita, M., Akabe, K., Hatakoshi, Y., Neubig, G., Yoshino, K., and Nakamura, S. (2015). "Parser Self-Training for Syntax-Based Machine Translation." In *Proc. IWSLT*, pp. 232–239.

Nakazawa, T., Mino, H., Goto, I., Kurohashi, S., and Sumita, E. (2014). "Overview of the 1st

Workshop on Asian Translation." In *Proc. WAT*.

Neubig, G. (2013). "Travatar: A Forest-to-String Machine Translation Engine based on Tree Transducers." In *Proc. ACL Demo Track*, pp. 91–96.

Neubig, G. (2014). "Forest-to-String SMT for Asian Language Translation: NAIST at WAT2014." In *Proc. WAT*.

Neubig, G. and Duh, K. (2014). "On the Elements of an Accurate Tree-to-String Machine Translation System." In *Proc. ACL*, pp. 143–149.

Oda, Y., Neubig, G., Sakti, S., Toda, T., and Nakamura, S. (2015). "Ckylark: A More Robust PCFG-LA Parser." In *Proc. NAACL*, pp. 41–45.

Papineni, K., Roukos, S., Ward, T., and Zhu, W.-J. (2002). "BLEU: a method for automatic evaluation of machine translation." In *Proc. ACL*, pp. 311–318.

Yamada, K. and Knight, K. (2001). "A syntax-based statistical translation model." In *Proc. ACL*, pp. 523–530.

Zhang, H. and Chiang, D. (2012). "An Exploration of Forest-to-String Translation: Does Translation Help or Hurt Parsing?" In *Proc. ACL*, pp. 317–321.

略歴

森下 睦：2015 年同志社大学理工学部インテリジェント情報工学科中途退学 (大学院への飛び入学のため)．現在，奈良先端科学技術大学院大学情報科学研究科博士前期課程在籍．機械翻訳，自然言語処理に関する研究に従事．

赤部 晃一：2015 年奈良先端科学技術大学院大学情報科学研究科博士前期課程修了．機械翻訳，自然言語処理に関する研究に従事．

波多腰 優斗：2015 年奈良先端科学技術大学院大学情報科学研究科博士前期課程修了．現在，セイコーエプソン株式会社にて勤務．

Graham Neubig: 2005 年米国イリノイ大学アーバナ・シャンペーン校工学部コンピュータ・サイエンス専攻卒業．2010 年京都大学大学院情報学研究科修士課程修了．2012 年同大学院博士後期課程修了．同年奈良先端科学技術大学院大学助教．機械翻訳，自然言語処理に関する研究に従事．

吉野 幸一郎：2009 年慶應義塾大学環境情報学部卒業．2011 年京都大学大学院情報学研究科修士課程修了．2014 年同博士後期課程修了．同年日本学術振興会特別研究員（PD）．2015 年より奈良先端科学技術大学院大学情報科学研究科特任助教．京都大学博士（情報学）．音声言語処理および自然言語処理，特に音声対話システムに関する研究に従事．2013 年度人工知能学会研究会優秀賞受賞．IEEE，ACL，情報処理学会，言語処理学会各会員．

中村 哲：1981 年京都工芸繊維大学電子卒．京都大学博士（工学）．シャープ株式会社．奈良先端大　助教授，2000 年 ATR 音声言語コミュニケーション研究所　室長，所長，2006 年 (独) 情報通信研究機構　研究センター長，けいはんな研究所長などを経て，現在，奈良先端大　教授．ATR フェロー．カールスルーエ大学客員教授．音声翻訳，音声対話，自然言語処理の研究に従事．情報処理学会喜安記念業績賞，総務大臣表彰，文部科学大臣表彰，Antonio Zampoli 賞受賞． ISCA 理事、IEEE SLTC 委員，IEEE フェロー．