機械翻訳の精度を考慮した構文解析器の自己学習

森下 睦[1,a)] 赤部 晃一[1,b)] Graham Neubig[1,c)] 吉野 幸一郎[1,d)] 中村 哲[1,e)]

概要：構文情報を考慮する機械翻訳では，構文解析器の精度が翻訳精度に大きく影響することが知られている．そこで，構文解析の精度向上を図る手法として，構文解析器の解析結果を学習データとして用いる自己学習が提案されている．自己学習により機械翻訳に悪影響を及ぼす解析結果が減少し，翻訳精度が向上することが先行研究により確認されている．しかし，自動的な構文解析により得られる構文木は必ずしも翻訳精度の向上に寄与するとは限らず，自己学習した際にモデルにノイズが混入してしまう問題がある．本稿では，自己学習を行う際に機械翻訳の自動評価尺度を用いて学習データを選択する手法を提案する．これにより，機械翻訳の精度向上に寄与する学習データのみが選別され，翻訳精度を向上するような自己学習を行うことができると考えられる．実験により，本手法で構築した構文解析器を用いることで，最先端の機械翻訳システムの翻訳精度が複数の言語対で有意に向上した．

1. はじめに

統計的機械翻訳では，フレーズ単位で翻訳を行い，それらを並び替えるフレーズベース翻訳 (Phrase Based Machine Translation, PBMT)[1]，構文木の部分木を翻訳の際に利用する統語ベース翻訳 [2] などの翻訳手法が提案されている．フレーズベース翻訳は英仏間のように，語順が近い言語間では高い翻訳精度を達成しているものの，日英などの語順が大きく異なる言語間での翻訳精度は十分でない．一方，統語ベース翻訳は，日英などの大幅な並び替えが起こりやすい言語間で，フレーズベース翻訳と比べて翻訳精度が高くなることが多い．統語ベース翻訳の中でも，原言語側の構文情報を用いた Tree-to-String 翻訳 [3] は，語順が大きく異なる言語間で高速かつ高精度な翻訳を実現することで知られている．ただし，Tree-to-String 翻訳は翻訳に際して構文解析器が出力した情報を用いるため，この構文解析器の精度が翻訳精度に大きく依存することがわかっている [4]．

構文解析器の精度向上の手法として，構文解析器の自己学習が提案されている [5]．自己学習では，既存のモデルで学習した構文解析器が出力した構文木を，再度学習データとして用いることで，構文解析器が対象とするデータに適応する．構文解析器の自己学習は統語ベース翻訳においても効果があり，自己学習したモデルを用いることで，翻訳精度が向上することが報告されている [6]．さらに，機械翻訳の単語並び替えに用いられる構文解析器の自己学習では，構文解析器の出力と正解データとを比較しスコア付けし，付与されたスコアに基づいて学習に用いる構文解析木のデータを選択すると，より効果的な学習ができることがわかっている [7]．このように，学習に用いるデータを選択し，自己学習を行う方法を標的自己学習 (Targeted Self-Training) という．しかし，[7] の手法で用いられた標的自己学習は，並び替えの正解データを人手で作成する必要があり，コストが大きい．

そこで本研究では，機械翻訳の自動評価尺度を選択時に用いることで，より少ないコストで効果的に構文解析器の自己学習を行う手法を提案する．自動評価尺度を用いることで，多くの既存の対訳コーパスを自己学習用の学習データとして用いることができ，より幅広い分野で低コストな構文解析器の精度向上と，それに伴う Tree-to-String 翻訳の翻訳精度向上を期待できる．実験の結果，機械翻訳の精度を考慮した構文解析器の自己学習を行うことにより，最先端の翻訳器の翻訳精度が複数の言語対で有意に向上することがわかった．

2. Tree-to-String 翻訳

統計的機械翻訳では，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．

[1] 奈良先端科学技術大学院大学 情報科学研究科
Graduate School of Information Science, Nara Institute of Science and Technology
[a)] morishita.makoto.mb1@is.naist.jp
[b)] akabe.koichi.zx8@is.naist.jp
[c)] neubig@is.naist.jp
[d)] koichiro@is.naist.jp
[e)] s-nakamura@is.naist.jp

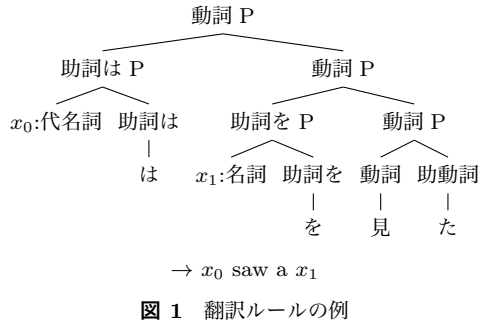


図 1 翻訳ルールの例

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

統計的機械翻訳の中でも，Tree-to-String 翻訳は原言語側の構文木 $T_{\boldsymbol{f}}$ を翻訳に用いることで，2 言語間の複雑な関係がルールとして表現可能となり，より精度の高い翻訳が可能となる．Tree-to-String 翻訳は下記のように定式化できる．

$$\begin{aligned}\hat{\boldsymbol{e}} &\coloneqq \underset{\boldsymbol{e}}{\operatorname{argmax}} Pr(\boldsymbol{e}|\boldsymbol{f}) \\ &= \underset{\boldsymbol{e}}{\operatorname{argmax}} \sum_{T_{\boldsymbol{f}}} Pr(\boldsymbol{e}|\boldsymbol{f}, T_{\boldsymbol{f}}) Pr(T_{\boldsymbol{f}}|\boldsymbol{f}) && (2)\\ &\simeq \underset{\boldsymbol{e}}{\operatorname{argmax}} \sum_{T_{\boldsymbol{f}}} Pr(\boldsymbol{e}|T_{\boldsymbol{f}}) Pr(T_{\boldsymbol{f}}|\boldsymbol{f}) && (3)\\ &\simeq \underset{\boldsymbol{e}}{\operatorname{argmax}} Pr(\boldsymbol{e}|\hat{T}_{\boldsymbol{f}}) && (4)\end{aligned}$$

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．

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

Tree-to-String 翻訳で用いられる翻訳ルールは，図 1 に示すように，置き換え可能な変数 x を含む部分木と目的言語の組で表される．図 1 の例では，x_0, x_1 が置き換え可能な変数である．訳出の際は，翻訳ルール自体の確率と各変数に入る候補の確率を考慮して，最も確率の高い訳を求める．また，確率の高い n 個の翻訳結果を出力する場合もあり，これを n-best 訳という．

Tree-to-String 翻訳では原言語文の構文木を考慮することで，フレーズベース翻訳と比べてより精度の高い翻訳が実現できる．しかし，構文木を翻訳に利用するため，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，構文森を翻訳に使用する Forest-to-String 翻訳 [8] も提案されている．構文森を翻訳に用いることで，複数の構文木の候補の中から，翻訳精度の高い訳出を行う構文木を選択することができ，翻訳精度の改善につながる [9]．Forest-to-String 翻訳は下記のように定式化できる．

$$\langle \hat{\boldsymbol{e}}, \hat{T}_{\boldsymbol{f}} \rangle = \underset{\langle \boldsymbol{e}, T_{\boldsymbol{f}} \rangle}{\arg\max} Pr(\boldsymbol{e}|T_{\boldsymbol{f}}) Pr(T_{\boldsymbol{f}}|\boldsymbol{f}) \tag{6}$$

3. 構文解析器の自己学習

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出力した構文木を，モデルの学習データとして用いることで，構文解析器を解析対象のデータに適応させ精度を向上させる手法である．つまり，自己学習対象の文に対して，式 (5) 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学習に用いる．

構文解析器の自己学習は，Charniak により初めて検証され，WSJ コーパス [10] 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されている [11]．一方，構文解析モデルの中でも，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 [12]．これは，PCFG-LA モデルが高精度なモデルなため，自己学習に用いる構文木がより高精度なものとなるほか，EM アルゴリズムによって，正解木と自動で生成された構文木から，複雑な文法規則を抽出できることを理由にあげている．本研究では，これをもとに 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

構文解析器の自己学習を用いて，機械翻訳の精度を向上させる先行研究はいくつか存在する．Katz-Brown らは，事前並べ替え [13] に用いる構文解析器の自己学習を行うことで，機械翻訳システム自体の翻訳精度が向上したと報告している [7]．この研究では，構文解析器が出力した構文木の候補の中から，並び替えの精度が最も高くなる構文木を自己学習に利用する標的自己学習を用いて，通常の自己学習より効果的な自己学習を実現している．

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 $\hat{T_{\boldsymbol{f}}}$ は，構文木の候補 $\boldsymbol{T_f}$ から以下の式によって選択される．

$$\hat{T_{\boldsymbol{f}}} = \arg\max_{T_{\boldsymbol{f}} \in \boldsymbol{T_f}} \text{score}(\boldsymbol{f}'^{*}, \text{reord}(T_{\boldsymbol{f}})) \tag{7}$$

また，波多腰らは Tree-to-String 翻訳における構文解析器の自己学習の効果を検証している [6]．この研究によると，Tree-to-String 翻訳においても構文解析器の自己学習は一定の効果を示している．ただし，この研究では構文解析器が出力した 1-best 木を学習に利用する通常の自己学習の枠組みを用いており，標的自己学習については検証されていない．

本研究では，これらの研究をもとに，Tree-to-String 翻訳

における構文解析器の標的自己学習の枠組みを提案する．次節以降では，この手法の詳細，および効果を検証する．

4. 機械翻訳の精度を考慮した自己学習

標的自己学習において重要な点は，学習に使用するデータをいかに選択するかという点である．以降では本研究で扱う，自己学習に用いる効果的な構文木の選択法，および文の選択法について述べる．

4.1 構文木の選択法

3.2 節で述べた，Katz-Brown らの標的自己学習 [7] では，人手で作成された単語対応に基づく正解並び替えデータと構文解析器が出力した並び替えデータとの比較を行い，最も精度が高いものを学習データとして選択する．しかし，人手で単語対応を作成するためには大きなコストがかかってしまい，大規模なデータを人手で作成することは現実的ではない．この問題を解決するために，本研究では対訳コーパスのみを使用し標的自己学習を行う手法を提案する．具体的には，翻訳器によって選択された 1-best 訳の構文木を学習データとして用いる手法，自動評価尺度を用いて選択された Oracle 訳の構文木を用いる手法の 2 つを検証する．

4.1.1 翻訳器 1-best

2 節でも述べたように，構文森を翻訳器の入力とした場合，複数の構文木の候補の中から翻訳精度が高くなると思われる構文木が翻訳器によって選択される．これにより，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 の構文木より自己学習に効果的であると考えられる．この際の自己学習に使われる構文木は式 (6) 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出を 1-best として出力する．しかし，実際には翻訳器が出力した 1-best 訳よりも，候補となった他の n-best 訳の方が参照訳に近く，より翻訳精度が高い場合が存在する．翻訳候補 $\boldsymbol{E}$ の中から，最も参照訳 $\boldsymbol{e}^*$ に近い訳を Oracle 訳 $\bar{\boldsymbol{e}}$ と言い，参照訳との誤差を表すエラー関数 error$(\cdot)$ を用いて下記の通り表される．

$$\bar{\boldsymbol{e}} = \underset{\boldsymbol{e} \in \boldsymbol{E}}{\operatorname{argmin}} \operatorname{error}(\boldsymbol{e}^*, \boldsymbol{e}) \tag{8}$$

本研究では，n-best 訳に対して自動評価尺度を用いてスコア付けを行い，その際のスコアが最も高い訳を Oracle 訳とする．Oracle 訳に用いられた構文木は翻訳器 1-best の構文木よりも，さらに翻訳精度の改善に効果的な構文木であると考えられる．

4.2 文の選択法

4.1 節では，1 つの対訳文から学習に有用な構文木を抽出する方法について述べた．しかし，そもそも正しい訳が n-best の翻訳候補に含まれていないような場合は，これらの文が学習のノイズとなる可能性がある．そのため，自己学習の学習データとなる翻訳器が出力した候補の中から，学習に用いるデータを選択すると更なる精度向上が実現可能であると考えられる．本研究では，訳の自動評価値が閾値を超えた文のみ学習に使用する手法，翻訳器 1-best と Oracle 訳の自動評価値の差が大きい文のみを使用する手法の 2 つを検証する．

4.2.1 自動評価値の閾値

コーパスの中には，翻訳器が上手く翻訳することができず，自動評価尺度が低くなってしまう文が多く存在する．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．
- 翻訳器の学習が十分でない．
- 自動評価値を計算する際に用いられた参照訳が意訳となっており，機械翻訳が出力しにくい訳となっている．
- コーパスに誤りがあり，正しい対訳データとなっていない．

このような例は，例え Oracle 訳であったとしても自動評価値が低い場合，自己学習の際に学習のノイズとなってしまう可能性が高く，学習データから除外する必要がある．また Tree-to-String 翻訳において，高精度の翻訳結果を出力するためには，正しい構文木が必要となるため，自動評価値が高い訳に使われた構文木は，正しい構文木である可能性が高い．これらの理由から，自動評価値が一定の閾値を上回ったもののみを学習に使用することで，より効果的な学習が行えると考えられる．閾値を t，文 i の Oracle 訳を $\bar{\boldsymbol{e}}^{(i)}$ とすると，Oracle 訳全体の集合 $\boldsymbol{E}$ の中から，学習に使われる文は自動評価尺度によるスコア関数 score(e) を用いて以下の集合で表される．

$$\{i \mid \operatorname{score}(\bar{\boldsymbol{e}}^{(i)}) \geq t,\ \bar{\boldsymbol{e}}^{(i)} \in \boldsymbol{E}\} \tag{9}$$

4.2.2 自動評価値の差

次に着目した点は，翻訳器 1-best と Oracle 訳の自動評価値の差である．構文解析器により，正解とは異なる構文木が高い確率を持つ構文森が出力される場合，翻訳器 1-best では誤った構文木を選択し，誤訳となる場合が多い．一方，Oracle 訳では構文森の中から正しい構文木が使用される可能性が高い．そのため，Oracle 訳に用いられた構文木を学習データとして用いることで，構文解析器のモデルが翻訳に適した方向に修正される．これにより，自己学習した構文解析器を用いた翻訳器は，正しい翻訳結果を 1-best として出力しやすくなり，結果として翻訳精度の改善につながると期待できる．

これを文選択に反映させるために，1-best 訳 $\hat{\boldsymbol{e}}^{(i)}$ と Oracle 訳 $\bar{\boldsymbol{e}}^{(i)}$ の間の評価値の向上を表す関数

$$\mathrm{gain}(\bar{\boldsymbol{e}}^{(i)}, \hat{\boldsymbol{e}}^{(i)}) = \mathrm{score}(\bar{\boldsymbol{e}}^{(i)}) - \mathrm{score}(\hat{\boldsymbol{e}}^{(i)}) \qquad (10)$$

を定義し，式 (9) と同様に，向上の大きな文を選択する．

本手法では，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 [14] 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．ここで，$N(|\boldsymbol{e}|+|\boldsymbol{f}|)$ は，目的言語文 $\boldsymbol{e}$ の長さを $|\boldsymbol{e}|$，原言語文 $\boldsymbol{f}$ の長さを $|\boldsymbol{f}|$ とした時に，その和 $|\boldsymbol{e}|+|\boldsymbol{f}|$ が一致する文がコーパス内に存在している数であり，N はコーパス内の文の総数を表す．

$$p(|\boldsymbol{e}|+|\boldsymbol{f}|) = \frac{N(|\boldsymbol{e}|+|\boldsymbol{f}|)}{N} \qquad (11)$$

5. 実験的評価

5.1 実験設定

実験は，構文解析誤りが発生しやすい日本語の構文解析器を用いる日英・日中翻訳を対象とした．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[*1] を用いた．自己学習の効果を検証するための最先端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 WAT2014[15] において高評価を得た，Neubig のシステムを用いた [16][*2]．デコーダには Travatar[17] を用い，Forest-to-String 翻訳を行った．構文解析は PCFG-LA モデルを用いた Egret[*3] により行い，日本語係り受けコーパス JDC[18](約 7000 文) で学習したモデルを，既存のモデルとして使用した．Egret は極稀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．機械翻訳の精度は，BLEU[19]，RIBES[20] の 2 つの自動評価尺度を用いて評価した．また，文単位の機械翻訳の精度は BLEU+1[21] 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ある JDC に加え，ASPEC のトレーニングデータの中から，ランダムまたは一定の基準で抽出されたものとした．また，自己学習したモデルはテスト時のみ使用し，翻訳モデルの学習は JDC で学習した既存のモデルで行った．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 [22] 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．

Parser 1-best
: 式 (5) のように，Egret が出力した 1-best 構文木を自己学習に用いる．自己学習に用いる文はランダムに抽出する．

MT 1-best
: 4.1.1 節のように，Egret が出力した構文森を Travatar に入力し，Travatar の 1-best 訳に使われた構文木を自己学習に用いる．自己学習に用いる文はランダムに抽出する．

Oracle
: 4.1.2 節のように, MT 1-best と同様の入力で, Travatar が出力した 500-best の訳の中から，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 は全て重複が無い文となるようにした．自己学習に用いる文はランダムに抽出する．

Oracle (BLEU+1≥t)
: 4.2.1 節のように，Oracle と同様の方法で選択された訳，構文木の中でも，翻訳結果の BLEU+1 スコアが一定値以上であった文の構文木のみを自己学習に用いる．

BLEU+1 Gain
: 4.2.2 節のように，Oracle (BLEU+1≥t) と同様の方法で選択された訳，構文木の中でも，1-best 訳と Oracle 訳間で BLEU+1 スコアの差が大きい文を自己学習に用いる．

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 に関しては上位 10 万文を抽出した．

5.2 実験結果

日英翻訳での実験結果を表 1 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 ($\dagger: p < 0.05$, $\ddagger: p < 0.01$)．表 1 中の (b),(c),(d) 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．なお，表中の Sentences は自己学習に使用した文数を示し，既存モデルである JDC の文数は含まない．

波多腰ら [6] の手法である Parser 1-best を学習データとする方法では，精度の向上を得ることができなかった (表 1(b))．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．

MT 1-best では，構文森を翻訳した結果として，翻訳精度が高いと思われる構文木が翻訳器内部で選択される．しかし，実験結果からは Parser 1-best と比較した場合翻訳精度は向上しているが，ベースラインシステムと比較した場合，精度の向上は見られなかった (表 1(c))．

翻訳候補の中から Oracle 訳の構文木を学習データとして用いると，MT 1-best よりもさらに翻訳精度が高い木が選択され，結果として自己学習後の翻訳精度が若干向上している (表 1(d))．この際の自己学習に用いられた文の BLEU+1 スコアの分布を図 2 に示す．図は横軸を x とし

[*1] http://lotus.kuee.kyoto-u.ac.jp/ASPEC
[*2] http://github.com/neubig/wat2014
[*3] http://code.google.com/p/egret-parser

表 1　自己学習手法と日英翻訳の精度

	Sentence selection	Tree selection	Sentences (k)	BLEU	RIBES
(a) Baseline	—	—	—	23.83	72.27
(b) Parser 1-best	Random	Parser 1-best	96	23.66	71.77
(c) MT 1-best	Random	MT 1-best	97	23.81	72.04
(d) Oracle	Random	BLEU+1 1-best	97	23.93	72.09
(e) Oracle (BLEU+1≥0.7)	BLEU+1 ≥ 0.7	BLEU+1 1-best	206	‡ 24.27	72.38
(f) Oracle (BLEU+1≥0.8)	BLEU+1 ≥ 0.8	BLEU+1 1-best	120	‡ 24.26	72.38
(g) Oracle (BLEU+1≥0.9)	BLEU+1 ≥ 0.9	BLEU+1 1-best	58	‡ 24.26	72.49
(h) BLEU+1 Gain	BLEU+1 Gain	BLEU+1 1-best	100	† 24.22	72.32

表 2　自己学習手法と日中翻訳の精度

	Sentence selection	Tree selection	Sentences (k)	BLEU	RIBES
(a) Baseline	—	—	—	29.60	81.32
(b) Oracle	Random	BLEU+1 1-best	130	‡ 29.89	‡ 81.66
(c) Oracle (BLEU+1≥0.8)	BLEU+1 ≥ 0.8	BLEU+1 1-best	150	‡ 29.91	81.47
(d) Oracle (BLEU+1≥0.9)	BLEU+1 ≥ 0.9	BLEU+1 1-best	82	† 29.86	‡ 81.60
(e) BLEU+1 Gain	BLEU+1 Gain	BLEU+1 1-bset	100	† 29.85	‡ 81.59
(f) Oracle (BLEU+1≥0.8, Ja-En)	BLEU+1 ≥ 0.8	BLEU+1 1-best	120	† 29.89	† 81.58

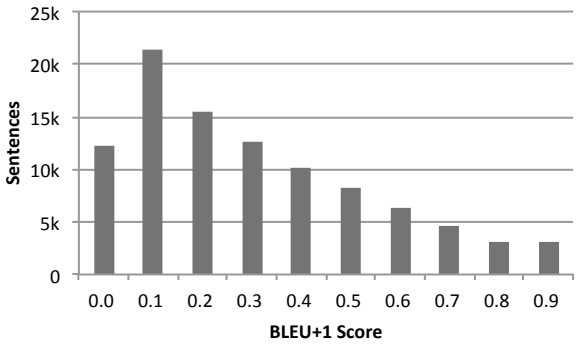


図 2　Oracle の自己学習に用いられた文の BLEU+1 スコアの分布

て，x 以上 $x+0.1$ 未満の BLEU+1 スコアを持つ文の数を表している．この図からわかるように，例え Oracle 訳でも学習に用いられている文の多くは BLEU+1 スコアが低く，これらが学習のノイズとなり自己学習の妨げになっていると考えられる．

そこで，BLEU+1 スコアに閾値を定め，高精度なデータのみを学習データとすると (表 1(e),(f),(g))，さらに翻訳精度が向上した．このことから構文解析器の自己学習を行う場合，低精度の構文木を取り除き，高精度の構文木のみを残すことが重要だと考えられる．その選択法として BLEU+1 スコアの閾値を用いる方法は有効であると確認できた．

また，MT 1-best と Oracle 訳の BLEU+1 スコアの差が大きいデータのみを用いると，従来の構文解析器が誤りやすく，翻訳精度の改善に寄与する構文木のみを選択することができると考えられる．実際にこの手法でも BLEU+1 スコアに閾値を定めて文を選択する方法と，ほぼ同等の効果が得られることを確認できた (表 1(h))．

日中での実験結果を表 2 に示す．日英と同様に，自己学習に用いる構文木を選択することにより，構文解析器がより対象のデータに適応し翻訳精度が向上していることが確認できた．

興味深いことに，日英のデータで自己学習し日英翻訳の精度改善に貢献したモデルを，そのまま日中翻訳に用いたところ，日中翻訳においても日中で学習したモデルと同程度の精度向上が見られた．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．

5.3　自己学習による訳出改善の例

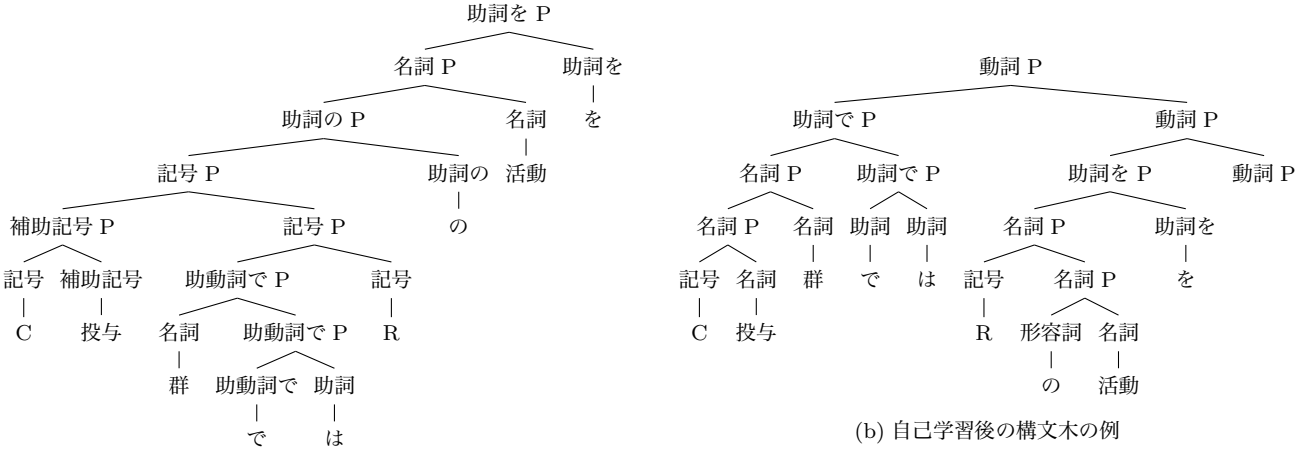
構文解析器の自己学習によって改善された日英訳の例を表 3 に示す．また，表 3 の訳出の際に使用された構文木を図 3 に示す．この例からわかるように，Tree-to-String 翻訳では構文解析結果の誤りが，翻訳精度に大きく影響してしまう．実際にベースラインシステムの構文木は名詞句を正しく捉えられておらず，大きく誤った構文木となっている．一方，自己学習を行った構文解析器は，正しく名詞句を分割することができており，その結果翻訳内容も修正されている．

6.　おわりに

本研究では，機械翻訳の精度を考慮し自己学習に用いる文を選択することで，Tree-to-String 翻訳において，より効果的な構文解析器の自己学習が行えることを検証した．日英，日中の 2 つの言語対に対して実験を行い，本手法で自己学習した構文解析器を用いることで，最先端の翻訳器がより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

表 3 構文解析器の自己学習により改善された日英訳の例

Source	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 。
Reference	in the C - administered group , thermal reaction clearly increased the activity of R for 240 minutes .
Baseline	for 240 minutes clearly enhanced the activity of C administration group R .
Oracle (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 3 自己学習による構文木の改善例

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．

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，より翻訳精度を向上させることが期待される．さらに，この自己学習を繰り返し行うことで，翻訳精度にどのような影響が及ぶかを検証したいと考えている．

謝辞 本研究の一部は，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

[1] Koehn, P., Och, F. J. and Marcu, D.: Statistical phrase-based translation, *Proc. HLT*, pp. 48–54 (2003).
[2] Yamada, K. and Knight, K.: A syntax-based statistical translation model, *Proc. ACL* (2001).
[3] Liu, Y., Liu, Q. and Lin, S.: Tree-to-String Alignment Template for Statistical Machine Translation, *Proc. ACL* (2006).
[4] Neubig, G. and Duh, K.: On the Elements of an Accurate Tree-to-String Machine Translation System, *Proc. ACL*, pp. 143–149 (2014).
[5] McClosky, D., Charniak, E. and Johnson, M.: Effective self-training for parsing, *Proc. HLT-NAACL*, pp. 152–159 (2006).
[6] 波多腰優斗，Neubig, G.，Sakti, S.，戸田智基，中村 哲：Tree-to-String 翻訳における構文解析器の自己学習の効果，言語処理学会第 21 回年次大会 (2015).
[7] Katz-Brown, J., Petrov, S., McDonald, R., Och, F., Talbot, D., Ichikawa, H., Seno, M. and Kazawa, H.: Training a Parser for Machine Translation Reordering, *Proc. EMNLP*, pp. 183–192 (2011).
[8] Mi, H., Huang, L. and Liu, Q.: Forest-Based Translation, *Proc. ACL*, pp. 192–199 (2008).
[9] Zhang, H. and Chiang, D.: An Exploration of Forest-to-String Translation: Does Translation Help or Hurt Parsing?, *Proc. ACL*, pp. 317–321 (2012).
[10] Marcus, M. P., Marcinkiewicz, M. A. and Santorini, B.: Building a large annotated corpus of English: The Penn Treebank, *Computational linguistics*, Vol. 19, No. 2, pp. 313–330 (1993).
[11] Charniak, E.: Statistical Parsing with a Context-Free Grammar and Word Statistics, *Proc. AAAI*, pp. 598–603 (1997).
[12] Huang, Z. and Harper, M.: Self-Training PCFG grammars with latent annotations across languages, *Proc. EMNLP*, pp. 832–841 (2009).
[13] Xia, F. and McCord, M.: Improving a statistical MT system with automatically learned rewrite patterns, *Proc. COLING* (2004).
[14] Gascó, G., Rocha, M.-A., Sanchis-Trilles, G., Andrés-Ferrer, J. and Casacuberta, F.: Does more data always yield better translations?, *Proc. ACL*, pp. 152–161 (2012).
[15] Nakazawa, T., Mino, H., Goto, I., Kurohashi, S. and Sumita, E.: Overview of the 1st Workshop on Asian Translation, *Proc. WAT* (2014).
[16] Neubig, G.: Forest-to-String SMT for Asian Language Translation: NAIST at WAT2014, *Proc. WAT* (2014).
[17] Neubig, G.: Travatar: A Forest-to-String Machine Translation Engine based on Tree Transducers, *Proc. ACL Demo Track*, pp. 91–96 (2013).
[18] Mori, S., Ogura, H. and Sasada, T.: A Japanese Word Dependency Corpus, *Proc. LREC* (2014).
[19] Papineni, K., Roukos, S., Ward, T. and Zhu, W.-J.: BLEU: a method for automatic evaluation of machine translation, *Proc. ACL*, pp. 311–318 (2002).
[20] Isozaki, H., Hirao, T., Duh, K., Sudoh, K. and Tsukada,

H.: Automatic Evaluation of Translation Quality for Distant Language Pairs, *Proc. EMNLP*, pp. 944–952 (2010).
[21] Lin, C.-Y. and Och, F. J.: Orange: a method for evaluating automatic evaluation metrics for machine translation, *Proc. COLING*, pp. 501–507 (2004).
[22] Koehn, P.: Statistical significance tests for machine translation evaluation, *Proc. EMNLP* (2004).