Vol.2016-NL-226 No.8
Vol.2016-SLP-111 No.8
2016/5/16

様々な分野における対訳コーパスを用いた構文解析器の自己学習効果の検証

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

概要：本稿では，対訳コーパス，統語ベース翻訳器，機械翻訳の自動評価尺度を用いて，自己学習データを選択した上で構文解析器の自己学習を行う手法を，様々な分野を対象に適用しその効果を検証する．本手法では構文木データを新たに人手で作成する必要が無く，対訳コーパスのみを用いて構文解析器を向上させられる利点がある．実験の結果，11 種類中 4 種類のドメインにおいて，本手法がベースラインと比較して構文解析精度を有意に向上させることが分かった．また，提案手法による性能向上が最も期待できるドメインの特徴について調査した．なお，本実験で作成したモデルは今後公開する予定である．

1. はじめに

統計的手法を用いた構文解析器では，学習に用いる構文木データの量が解析精度に大きく影響する．また，構文解析器の学習データが網羅していない分野に関する文については，解析精度が低くなる傾向があり，これを解決するために様々な分野の学習データが必要とされている [1]．しかし，構文木を作成するためには人手によるアノテーション作業が必要となるため，大規模かつ様々な分野のデータを作成するためには大きなコストがかかってしまう．

このような現状において，構文解析器の精度を高める手法の一つとして自己学習 (Self-Training) が挙げられる [2]．構文解析器の自己学習とは，既存の構文木で学習した構文解析器に，新たな文を入力し構文木を生成し，これらを用いて再度モデルの学習を行う手法である．これにより，追加のアノテーションを必要とせずに学習データが増え，各ドメイン依存のデータが学習データとして取り込まれることで，構文解析精度が向上する．しかしこの手法の問題点として，自動生成した構文木は必ずしも正しくなく，誤った構文木が学習データに混入することで，自己学習の効果が低下する点が挙げられる．

森下ら [3] は，対訳コーパスを利用し自己学習に使用する構文木を選択する手法を提案し，従来の自己学習と比べ高い効果が得られたと報告している．しかし，本手法では単一分野に対してのみ実験を行っており，様々な分野での効果については検証されていない．本稿では，対訳コーパスを用いた構文解析器の自己学習を様々な分野に対して効果が得られるよう適用し，その効果を検証する．具体的には

- 対訳コーパスを用いた構文解析器の自己学習は，様々なドメインに対して適用可能か
- どのような特徴を持ったドメインにおいて，自己学習による効果が得られやすいか

について検証する．

2. 対訳コーパスを利用した構文解析器の標的自己学習

構文解析器の自己学習とは，既存の構文解析器が出力した構文木を，構文解析器の新たな学習データとして再学習を行う [2]．しかし，この際出力される構文木が必ずしも正しいとは限らず，誤りを含んでいる場合学習データ中のノイズとなり，学習の妨げとなる問題がある．この問題を解決するためには，外部の評価指標を基に学習データを選択する標的自己学習 (Targeted Self-Training) が有効である [4]．

森下ら [3] は，対訳コーパスおよび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を提案している．下記では，統語ベース翻訳の一種である Tree-to-String 翻訳および森下らの標的自己学習手法の概要について説明する．

[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)] oda.yusuke.on9@is.naist.jp
[c)] neubig@is.naist.jp
[d)] koichiro@is.naist.jp
[e)] s-nakamura@is.naist.jp

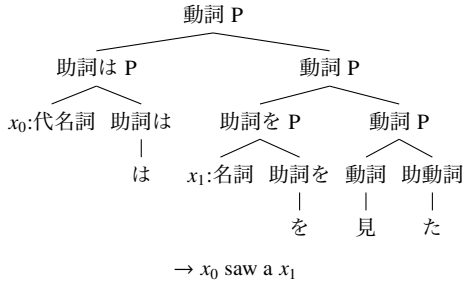

図 1 日英 T2S 翻訳における翻訳ルールの例

2.1 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．

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

様々な手法が提案されている統計的機械翻訳の中でも，Tree-to-String (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，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，式 (6) 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 [*1] によって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 Forest-to-String (F2S) 翻訳 [5] 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 [6]．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．

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

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

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

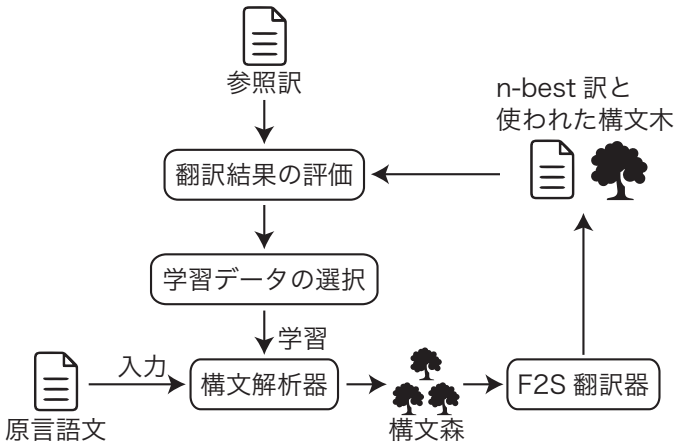

図 2 対訳コーパスを用いた構文解析器の標的自己学習概要

2.2 標的自己学習

本稿では，対訳コーパス，統語ベース翻訳および機械翻訳の評価尺度を利用し，使用するデータを選択した上で，構文解析器の自己学習を行う．森下らの手法概要を図 2 に示す．図 2 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 F2S 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を得る．次に，参照訳と機械翻訳の自動評価尺度を用いて，n-best 訳に対して翻訳精度のスコア付けを行う．F2S 翻訳で正しい翻訳が得られた場合，その翻訳に使われた構文木は正しい可能性が高いと仮定する．この仮定が正しければ，翻訳精度を測定することで間接的に構文木の正しさを測定することができると考えられる．ゆえに本手法では，機械翻訳の自動評価値を基に学習データを選択し，構文解析器の自己学習を行う．

データの選択には，構文木の選択法および文の選択法を組み合わせて使用する．構文木の選択法では，一つの文の構文木候補から誤りの少ない構文木を選択し，文の選択法では，コーパス全体から精度向上に有効な文のみを選択する．3 節ではそれぞれの手法について説明する．

3. 構文木および文の選択法

標的自己学習を行うにあたり，最も重要な点はどのように学習データを選択するかという点である．本節では，一文の構文木候補から誤りの少ない構文木を選択する構文木の選択法，および，コーパス全体から精度向上に有効な文を選択する文の選択法の 2 点を説明する．

3.1 構文木の選択法

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思われる場合が存在する．そこで本手法では，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．

$$\bar{\boldsymbol{e}} = \underset{\boldsymbol{e}\in\boldsymbol{E}}{\operatorname{argmax}}\ \mathrm{score}(\boldsymbol{e}^*, \boldsymbol{e}) \tag{9}$$

3.2 文の選択法

3.2.1 自動評価値上位

3.1 節では，1 つの対訳文の n-best 訳から誤りの少ない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．

F2S 翻訳では，正しく翻訳するためには正しい構文木が必要となる．このため，翻訳文の自動評価値が低い場合，翻訳時に正しい構文木が使われていない可能性があり，これらの構文木を使うと自己学習のノイズとなる可能性が高い．そこで，自動評価値が低い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り，より正確な構文木のみが残ると考えられる．本手法では，Oracle 訳の自動評価値が上位の文に使用された構文木を自己学習に使用する．

3.2.2 文長の分布の保持

文の選択法を使用する際には，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 [7] 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．

表 1 使用するコーパス一覧

コーパス名	文数	翻訳器の学習に使用	自己学習に使用
青空文庫	108k	○	○
BTEC	465k	○	○
KFTT	440k	○	○
法律文書	260k	○	○
例辞郎	424k	○	○
田中コーパス	150k	○	○
TED	97k	○	○
英辞郎	1969k	○	×
WWWJDIC	394k	○	×
Wikipedia	403k	○	×

表 2 使用した JDC 文数詳細

分野	train 文数	test 文数
BCCWJ-OC	1579	491
BCCWJ-OW	1158	340
BCCWJ-OY	1788	491
BCCWJ-PB	2181	485
BCCWJ-PM	2439	395
BCCWJ-PN	2446	471
EHJ	11700	1300
NKN	8747	979
RCP	661	62
JNL	286	30
NPT	1494	208
合計	34479	5252

$$p(|\boldsymbol{e}|+|\boldsymbol{f}|) = \frac{N(|\boldsymbol{e}|+|\boldsymbol{f}|)}{N} \tag{10}$$

4. 実験

4.1 実験設定

本稿では，日本語の構文解析器を用いる日英翻訳を行い，自己学習データを選択した．翻訳器の学習には様々な分野のコーパスを使用し，対応分野を広げた．表 1 に使用したコーパスと，その文数を示す．英辞郎，WWWJDIC，Wikipedia コーパスは全体的に文長が短く，自己学習には適さないため翻訳器の学習のみに使用し，自己学習の対象から除外した [*2]．

F2S 翻訳のデコーダには Travatar [8] を用いた．構文解析器には [9] 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[*3] を用い，日本語係り受けコーパス (JDC) [10] 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 [*4] 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．JDC に含まれる分野および各トレーニングセット，テストセットの文数

[*2] これらの文長が短い文を使用した場合，精度が向上しないことを予備実験により確認した．
[*3] http://code.google.com/p/egret-parser
[*4] 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

を表 2 に示す [*5]．構文森は 100-best 構文木に存在する超辺のみで構成した [*6]．また，構文木選択を行う際に用いる文単位の機械翻訳精度は BLEU+1 [11] または RIBES [12] を用いて評価した．

各構文解析モデルの精度測定時には，JDC のテストセットを使用した．JDC は 11 種類の分野のコーパスが含まれているため，特定の分野にとどまらず様々な分野の解析精度を測定することができる．精度測定には Evalb[*7] を使用し，再現率，適合率，および F 値を測定した．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 [13] (各テストセットを 1/2 にし，1000 回のリサンプリングを行った) により統計的有意差を検証した．

次節では，下記の手法を組み合わせ比較評価する．

構文木の選択法

構文解析器 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．

自動評価尺度 1-best

3.1 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，最も自動評価値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

ランダム

全学習データからランダムに文を選択する．

自動評価値上位

3.2.1 節のように，Oracle 訳とその構文木の中でも，訳の自動評価値が高い文のみを自己学習に使用する．

4.2 各手法の比較

表 1 のコーパスより選択された 20 万文を用いて自己学習を行った際の，構文解析精度測定結果を表 3 に示す．表中の短剣符は，解析精度がベースラインと比較して統計的に有意に高いことを示す (†: $p < 0.05$, ‡: $p < 0.01$)．また，各手法を用いて自己学習に使用する文を 2 万文から 20 万文まで 2 万文ずつ増加させた場合の構文解析精度の変化を図 3 に示す．

構文解析器 1-best を用いた手法では，解析精度は向上しなかった (表 3 (b))．これは，自己学習に使われた構文木に誤りを含んだものが多く混在しており，これらが学習のノイズとなることで正しく学習が行えなかったからだと考えられる．

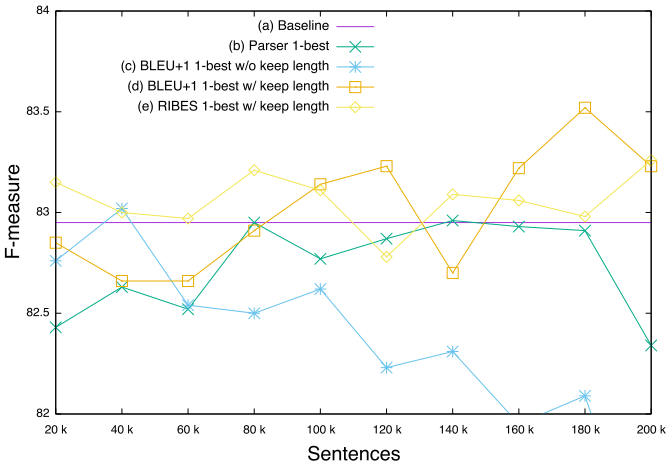

図 3 使用文数による構文解析精度の変化

表 3 (c) には，構文木の選択および文の選択を行い，文長の分布は保たなかった場合の精度を示している．この場合の構文解析精度はベースラインを大きく下回る結果となった．短い文は単語が少し変わっただけでも自動評価値が大幅に変化してしまうため，Oracle 訳の自動評価値が高くなる傾向がある．そのため，文長を考慮せずに自動評価値の高い文だけを選択すると，短い文のみを選択する傾向があり，このように精度が下がってしまったのだと考えられる．

表 3 (d), (e) では，文長を考慮した上で，BLEU+1 上位，RIBES 上位の文を選択した．文長の分布を保つことにより文選択が有効に働き，精度も向上した．

使用文数による精度の変化 (図 3) を見ると，全体的に学習データのサイズに従って精度が変動することが確認できる．表 3 (d), (e) の手法は全体的にベースラインより精度が高い傾向にあり，効果的な自己学習が行われていると思われる．しかし，全体的に精度はばらついており，安定した精度を得ることは今後の課題である．また，文長の分布を考慮しない手法 (表 3 (c)) では，使用する文数が増加するにつれて精度が低下しており，文長の分布を保つことの重要性がうかがえる．

4.3 本手法により性能向上が期待できるドメインの特徴

表 4 に，20 万文を用いて自己学習を行った際の，JDC の各分野での精度測定結果を示す．表中の短剣符は，解析精度がベースラインと比較して統計的に有意に高いことを示す (†: $p < 0.05$, ‡: $p < 0.01$)．本節では，どういったドメインが本手法により精度向上が期待できるのか，要因を複数仮定し検討する．

- 既存モデルの解析精度
  - 既存モデルの精度が低ければ自己学習効果が大きいと考えられる．この関係性について調査した．

[*5] 各分野名と実際の出典 BCCWJ-OC:Yahoo!知恵袋, BCCWJ-OW:白書, BCCWD-OY:Yahoo!ブログ, BCCWD-PB:書籍, BCCWJ-PM:雑誌, BCCWJ-PN:新聞, EHJ:日常会話のための辞書の例文, NKN:日本経済新聞, RCP:クックパッドデータセットの一部, JNL:論文抄録, NPT:特許

[*6] 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．

[*7] http://nlp.cs.nyu.edu/evalb

表 3 20 万文を用いて自己学習を行った際の構文解析精度

	手法名	構文木の選択法	文の選択法	文長の分布の保持	F 値
(a)	Baseline	—	—	—	82.95
(b)	Parser 1-best	構文解析器 1-best	ランダム	なし	82.34
(c)	BLEU+1 1-best w/o keep length	自動評価尺度 1-best (BLEU+1)	自動評価値上位	なし	81.13
(d)	BLEU+1 1-best w/ keep length	自動評価尺度 1-best (BLEU+1)	自動評価値上位	あり	83.23
(e)	RIBES 1-best w/ keep length	自動評価尺度 1-best (RIBES)	自動評価値上位	あり	† 83.26

表 4 各分野での精度測定結果 (F 値)

分野	Baseline	表 3 (e)
BCCWJ-OC	84.09	83.32
BCCWJ-OW	71.89	‡ 74.41
BCCWJ-OY	79.38	80.25
BCCWJ-PB	74.46	‡ 75.90
BCCWJ-PM	78.66	† 80.01
BCCWJ-PN	79.04	† 80.18
EHJ	92.74	91.95
NKN	86.33	85.92
RCP	84.02	82.53
JNL	83.99	82.12
NPT	86.65	86.71
全体	82.95	† 83.26

- 自己学習に使用した文と各分野の類似度
  - 自己学習に使用した文と各分野の類似度が近い場合，自己学習効果が高くなることが考えられる．類似度を計算するために，自己学習に使用した文を基に言語モデルを作成し，この言語モデルと各テストセットとの Perplexity を求めた．言語モデルの作成および Perplexity の測定には KenLM [14] を使用した．
- 文の平均文長
  - McClosky ら [2] は，20 から 50 単語の文において，構文解析器の自己学習がより有効であったと報告している．これを受け，各分野の平均文長と自己学習前後の F 値の差を確認した．
- 自己学習後の未知 bigram の減少率
  - McClosky ら [15] は，構文解析器の自己学習を行った場合，既存モデルで既知の単語が未知の bigram で現れた場合に精度の向上が見られたと報告している．この研究を基に，既存モデルの bigram，自己学習後の bigram を求め，各テストセットについてどの程度未知 bigram が減少したかを確認した．

それぞれの要因について検討した際の結果を表 5 に示す．これらの値を基に，各要因と自己学習前後の F 値の上がり幅との相関係数を求めた．

相関が見られたのは，既存モデルの解析精度および自己学習後の未知 bigram 減少率であり，自己学習に使用した文と各分野の類似度，文の平均文長については相関が見られなかった．

この結果から，BCCWJ-OW のように，既存のモデルで精度が低い分野に対しては自己学習は効果的であるが，既に解析精度が高い分野では，逆に精度が低下してしまう可能性があることが示唆された．また，BCCWJ-PM のように，未知 bigram 減少率が高い分野は自己学習が効果的であった．このことから，解析対象の文がわかっている場合，それらの未知 bigram を減らすように自己学習を行えば，さらに精度が向上する可能性が示唆された．

4.4 単一分野に対して自己学習を行った場合との比較

本節では，単一分野に対して自己学習を行い，様々な分野を含んだ JDC のテストセットで精度を測定し，どの程度の精度になるかを検討する．

実験では，文献 [3] で学習した ASPEC (科学技術論文を抜粋した対訳コーパス) 用モデルについて，JDC のテストセットを使用し精度を測定した．また，ASPEC のテストセット (100 文) についても同様に精度を測定した．実験結果を表 6 に示す．

実験より，ASPEC 用モデルは ASPEC テストセットにおいては最も良い精度を達成できているものの，JDC のテストセットを用いて測定した場合，精度が低下している．また，様々なドメインを対象に自己学習を行った場合，JDC，ASPEC ともに精度の向上が見られるものの，ASPEC だけを対象に学習したモデルの精度には達していない．

このことから，解析対象のドメインが決まっている場合，そのドメインと類似している対訳コーパスを選択した上で自己学習に用いることで，解析精度がより高くなる可能性が示唆される．

5. 関連研究

本研究以前に，様々なドメインについて構文解析器の自己学習を行った研究として，Le Roux らの研究が挙げられる [16]．この研究は Web 上の様々な分野のテキストを構文解析するシェアードタスク “SANCL” [17] に提出されたもので，各参加者は「Yahoo! Answers」，「E メール」，「ニュース」，「レビュー」，「ブログ」の 5 つの分野の文を構文解析し，その精度を競う．Le Roux らは，構文解析器の自己学習を行うことで各分野に特化したモデルを構築し，テスト文がどの分野にあたるかを事前に分類器により分類した上で，構文解析を行った．これにより，SANCL に提出された全システム中 1 位の精度を達成した．

表 5 考えられる要因と相関係数

	F 値上がり幅	既存モデル精度	自己学習文との Perplexity	平均文長	未知 bigram 減少率
BCCWJ-OC	-0.77	84.09	109.614	19.66	0.181
BCCWJ-OW	2.52	71.89	180.061	25.91	0.194
BCCWJ-OY	0.87	79.38	277.745	15.64	0.177
BCCWJ-PB	1.44	74.46	218.737	21.78	0.203
BCCWJ-PM	1.35	78.66	303.472	16.97	0.200
BCCWJ-PN	1.14	79.04	299.449	20.86	0.164
EHJ	-0.79	92.74	42.708	12.65	0.169
NKN	-0.41	86.33	199.356	27.85	0.139
RCP	-1.49	84.02	280.876	18.04	0.131
JNL	-1.87	83.99	212.522	33.76	0.085
NPT	0.06	86.65	307.778	34.91	0.093
F 値上がり幅との相関係数	—	-0.79	0.26	-0.14	0.69
p 値	—	0.0040	0.4360	0.6875	0.0181

表 6 ASPEC に対して自己学習を行った場合の精度測定結果 (F 値)

テストセット	Baseline	Parser 1-best	表 3 (e)	ASPEC 用
ASPEC	84.53	86.40	86.36	88.07
JDC	82.95	82.34	83.26	79.41

Le Roux らの研究と本研究の違いは，Le Roux らは自己学習に使用する構文木の精度を「文の長さ」や「未知語の数」などの特徴を基に予測しているのに対し，本研究では，実際にそれぞれの構文木を基に統語ベース翻訳を行い，その翻訳精度を基に構文木の精度を推定している点にある．Le Roux らは単一言語文のみを使用するのに対し，対訳文が利用できる場合，本手法の方がより正確な精度推定が可能だと思われる．

本研究では，どのドメインにおいても精度が出るよう単一のモデルを構築したが，Le Roux らはそれぞれの分野に対し 1 つずつモデルを構築した方が精度が高いと述べている．実際，4.4 節において ASPEC 単体に対して自己学習を行ったモデルの方が，ASPEC テストセットに対しては精度が向上することが確認できたため，解析対象とする分野を絞った場合に，どのような文選択基準が適切かは今後の課題である．

6. おわりに

本研究では，様々な分野を対象に対訳コーパスを用いて構文解析器の標的自己学習の効果について検討し，自己学習により効果が得られやすいドメインの特徴について調査した．

実験より，本手法で自己学習したモデルは，11 種類中 4 種類のドメインでベースラインより有意に精度の高い解析結果が得られることがわかった．また，精度が向上しやすいドメインの特徴として，様々な要因が考えられるものの，主に既存モデルでの精度が低い分野ほど精度が向上しやすい，また未知 bigram が減少するほど精度が向上しやすいという結果が得られた．

今後の課題としては，解析対象とするドメインを絞った場合に，どのような文を自己学習に使用すると効果が得られやすいかを検討する．

なお，本研究で作成したモデルについては，著者の Web サイト [*8] で公開する予定である．

謝辞 本研究の一部は JSPS 科研費 24240032 および 16H05873 の助成を受けたものです．

参考文献

[1] Gildea, D.: Corpus Variation and Parser Performance, *Proc. EMNLP*, pp. 167–202 (2001).
[2] McClosky, D., Charniak, E. and Johnson, M.: Effective self-training for parsing, *Proc. HLT*, pp. 152–159 (2006).
[3] Morishita, M., Akabe, K., Hatakoshi, Y., Neubig, G., Yoshino, K. and Nakamura, S.: Parser Self-Training for Syntax-Based Machine Translation, *Proc. IWSLT*, pp. 232–239 (2015).
[4] 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).
[5] Mi, H. and Huang, L.: Forest-based translation rule extraction, *Proc. EMNLP*, pp. 206–214 (2008).
[6] Zhang, H. and Chiang, D.: An Exploration of Forest-to-String Translation: Does Translation Help or Hurt Parsing?, *Proc. ACL*, pp. 317–321 (2012).
[7] 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).
[8] Neubig, G.: Travatar: A Forest-to-String Machine Translation Engine based on Tree Transducers, *Proc. ACL Demo Track*, pp. 91–96 (2013).
[9] Neubig, G. and Duh, K.: On the Elements of an Accurate Tree-to-String Machine Translation System, *Proc. ACL*, pp. 143–149 (2014).
[10] Mori, S., Ogura, H. and Sasada, T.: A Japanese Word Dependency Corpus, *Proc. LREC*, pp. 753–758 (2014).
[11] Lin, C.-Y. and Och, F. J.: Orange: a method for evaluating automatic evaluation metrics for machine translation, *Proc. COLING*, pp. 501–507 (2004).
[12] Isozaki, H., Hirao, T., Duh, K., Sudoh, K. and Tsukada, H.:

[*8] http://www.otofu.org

Automatic Evaluation of Translation Quality for Distant Language Pairs, *Proc. EMNLP*, pp. 944–952 (2010).
[13] Koehn, P.: Statistical significance tests for machine translation evaluation, *Proc. EMNLP*, pp. 388–395 (2004).
[14] Heafield, K.: KenLM: Faster and smaller language model queries, *Proc. WMT*, pp. 187–197 (2011).
[15] McClosky, D., Charniak, E. and Johnson, M.: When is Self-training Effective for Parsing?, *Proc. COLING*, pp. 561–568 (2008).
[16] Le Roux, J., Foster, J., Wagner, J., Samad Zadeh Kaljahi, R. and Bryl, A.: DCU-Paris13 systems for the SANCL 2012 shared task, *Notes of SANCL* (2012).
[17] Petrov, S. and McDonald, R.: Overview of the 2012 Shared Task on Parsing the Web, *Notes of SANCL* (2012).