ユーザ定義の翻訳ルールにより語彙と構文が制御可能なニューラル機械翻訳

徐聖源 [1]　宮田玲 [1]　佐藤理史 [1]
[1] 名古屋大学大学院工学研究科
seo.sung.won.e3@s.mail.nagoya-u.ac.jp

概要

ユーザが翻訳ルールを定義することで制御可能なニューラル機械翻訳 URNMT (**U**ser-defined **R**ule constrained **N**eural **M**achine **T**ranslation) を提案する。本システムはまず、翻訳ルールによって、語彙だけでなく語順など構文に関する制約の情報を含めた未完成の翻訳（部分翻訳）を生成する。そして、その部分翻訳を語彙制約付 NMT の一つである EDITOR に制約として与えることで、統制された翻訳を生成する。EDITOR には、部分翻訳を制約として効果的に活用するための機能を追加した。擬似部分翻訳を用いた自動評価により、提案手法が適切に部分翻訳を活用して翻訳を生成できることを確認した。

1　はじめに

ニューラル機械翻訳 (neural machine translation; NMT) を制御する手法の一つとして語彙制約付 NMT の研究が行われている [1, 2, 3, 4]。語彙制約付 NMT は、制約として与えた単語やフレーズを含む翻訳を生成することができるが、出力文に合わせた語形変化が難しいことが指摘されている [4]。また、マニュアル等の産業文書の翻訳では、使用すべき語彙だけでなく、語順などの構文的なパターンも特定の形に統制したいという要求がある [5]。

Jon ら [4] は、Transformer [6] 学習時にレンマ化した制約をソース文と共に学習データとすることで、基本形の制約を与えてもモデルが正しい表層形を選択できるようにした。この手法は、語形変化に柔軟に対応できる形で語彙レベルで制約された翻訳を可能にするが、構文レベルの制御は対象としない。このほか、構文を制御して文を生成する研究 [7] もあるが、翻訳タスクで語彙と構文を同時に制御する研究は不足している。

本研究では、ユーザが定義した翻訳ルールに従って NMT の出力を制御する URNMT (User-defined Rule constrained Neural Machine Translation) を提案する。URNMT は、翻訳ルールを用いて翻訳できる部分だけ翻訳して、翻訳できない部分は原言語のまま残した部分翻訳を生成する。その後、その部分翻訳を語彙制約付 NMT に制約として与えることで翻訳を完成させる。部分翻訳に語順や文型などの情報を組み込むことができるため、語彙制約付 NMT の枠組みを用いながら構文的な制約も課すことができる点が特徴である。

本稿では、現在日英翻訳を対象として開発中の URNMT の全体構成と各機構を説明する（2 節）。また、擬似的な部分翻訳を用いた実験により、提案手法が部分翻訳を制約として適切に活用できることを示す（3 節）。

2　提案システム

URNMT は大きく、翻訳ルールを用いて部分翻訳を生成する部分翻訳生成器と、部分翻訳を制約として用いる NMT で構成される（図 1）。

2.1　部分翻訳生成器

部分翻訳生成器は、事前に定義された翻訳ルールが適用できる部分だけ翻訳し、残りは原言語のまま保持した、不完全な翻訳を生成する。具体的にはまず、**解析器**を使用して入力文の情報を持つ構文木 (Source Tree) を構築する。続いて、**辞書ルール**に従って、語句の訳し方に関する情報を Source Tree に追加する。最後に、Source Tree がどのような条件を満たすときどのような翻訳を生成するか記述する**組み立てルール**に従って、目標言語側の木 (Target Tree) を生成した上で、部分翻訳として出力する[1)]。

解析器　入力文を Stanza [8] を用いて構文解析した結果に基づき、Source Tree を構築する。このと

1）　Source Tree、Target Tree、辞書ルール、組み立てルールの例は付録 A に示す。

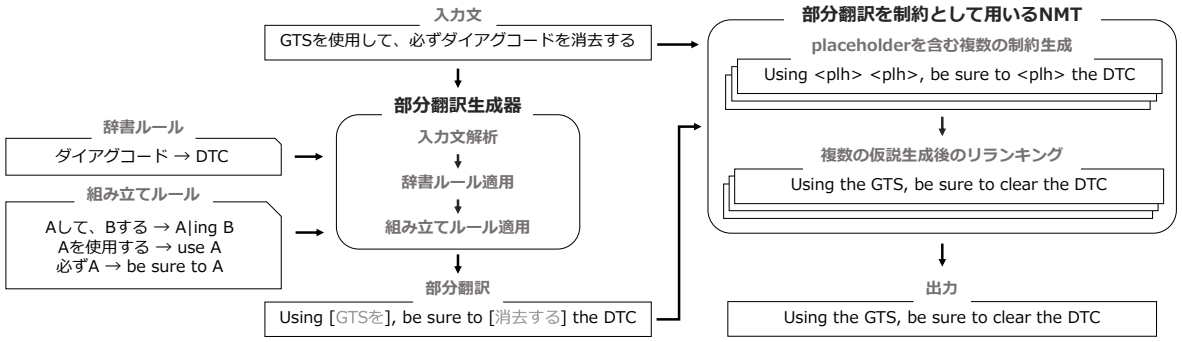

図 1　提案システム URNMT の全体構成と翻訳例

き、Stanza で得られる情報の一部を削除し、翻訳ルールの定義に必要な情報のみを残した。また、日本語の動詞の後ろに付く各種の機能表現の情報を翻訳時に用いるため、日本語文末解析器 Panzer [9] で解析した情報を追加する。

辞書ルール　辞書ルールは、ルールの適用条件、変換方法、語句の属性、適用優先度で構成される。解析器が生成した Source Tree に対し、適用条件を満たすノードが検索された場合、ルールで定義された情報を追加する。追加する情報は、図 1 の「ダイアグコード → DTC」のような変換方法だけでなく、語句の属性（意味カテゴリなど）も含む[2)]。Source Tree の各ノードには最大 1 つの辞書ルールが適用される。複数のルールが適用可能な場合は、適用優先度が最大のものを選ぶ。それでも 1 つに絞り込めない場合は、適用条件のより複雑なルールを優先する。

組み立てルール　組み立てルールは、ルールの適用条件、Target Tree の組み立て方法、適用優先度から構成される。図 1 では、「A して、B する → A|ing B」のように簡略化して記述しているが、実際の組み立てルールは、適用条件・組み立て方法ともに木構造で定義され、語彙だけでなく構文の変換方法も柔軟に表現できる。組み立てルールの適用条件に完全にマッチする Source Tree 中の部分（ノード集合）に対して、組み立て方法を用いて Target Tree の部分木を生成し、上位ノードから順次 Target Tree を組み立てる。ある Source Tree ノードに対し、複数の組み立てルール（適用条件）のルートノードが重なる場合は、適用優先度と適用条件の複雑度を考慮して、1 つに絞り込む。なお、ユーザが定義する比較的少数の組み立てルールでは、Source Tree の全体をカバーできないため、ユーザ定義ルールとは別に、バックアップ用の基本ルールをあらかじめ定義してある。

2)　辞書ルールで語句の属性を定義しておくことで、組み立てルールの柔軟な定義が可能となる。

2.2　部分翻訳を制約として用いる NMT

語彙制約付 NMT の一つである EDITOR [3] を使用して、不完全な部分翻訳から完全な翻訳を生成する。EDITOR は編集を繰り返すことで文を生成するモデルである。編集時は、与えられた制約を初期文とした後、トークンを削除したり位置を変更したりする再配置操作、単語をどこにいくつ追加すべきか予測して placeholder を挿入する placeholder 挿入操作、placeholder をトークンに変換するトークン予測操作を繰り返すことで翻訳を完成させる。

EDITOR に制約を与える方法には、制約の編集を許容するソフトな制約設定と、制約の削除や複数の制約間へのトークン挿入を禁止するハードな制約設定がある。提案システムでは、制約を厳密に守ることを重視するためハードな制約設定を用いるが、オリジナルの EDITOR に準拠して単純な単語の羅列を制約にしてしまうと、複数の制約はひと続きの固定表現として訳されてしまう。そこで、本研究では、未翻訳部分の情報を含んだ部分翻訳を制約として効果的に活用するため、EDITOR に以下の 3 つの機能を追加した (図 1 の例も適宜参照されたい)。

1. placeholder を含む制約生成　部分翻訳中の未翻訳部分をあらかじめ placeholder に置き換えて、EDITOR のデコード時の中間表現に合わせた制約を生成する。このとき、未翻訳部分の文字列から、翻訳を完成するときに埋めるべきトークン (すなわち placeholder) の数を予測する。具体的には未翻訳部分を NMT 学習時と同じ方法でトークナイズして得られたトークン数を基準とする。なお、EDITOR はトークンの追加や削除が可能であるため、placeholder を最初に 1 つ入れておき、あとは EDITOR に任せればよいとも考えられるが、経験的な観察の結果から、適切な数の placeholder を最初から制約に含めることが有効であると判断した。

また、オリジナル EDITOR のハードな制約設定では、制約に含まれる placeholder の編集も禁止されてしまうが、本システムではそれを許可した。

2. リランキング　上記で予測する placeholder の数は、あくまで原文のトークン数に基づくもので、必ずしも適切な翻訳文に必要なトークン数と一致するわけではない。この問題を解決するため、予測した placeholder の数にノイズを与えて複数の制約を生成した後、それぞれの制約に対して EDITOR の出力を生成する。その後、生成された複数の出力をリランキングし、最も良いものを選ぶ。

placeholder の数は、予測値（未翻訳部分のトークン数）を中心に、0 から 2 倍の範囲内でランダムで決定し、1 文ごとに 50 の制約を生成する。異なる制約が 50 個生成できない場合は、0 から 2 倍の範囲を超える placeholder を挿入する。

リランキング手法には、雑音のある通信路モデルに基づく手法 [10] を使用した。この手法は、順方向翻訳モデルの生成確率、逆方向翻訳モデルの生成確率、言語モデルの生成確率を線型結合して翻訳にスコアを付けた後、順位付けする。提案システムでは、順方向翻訳モデルとして EDITOR、逆方向翻訳モデルとして逆方向 Transformer モデル、言語モデルとしてターゲット側 Transformer 言語モデルを使用する。

3. スキップ機能　オリジナルの EDITOR は制約が与えられた時、トークンの再配置と placeholder 挿入を行った後、トークン予測を行う。そのため、制約に含まれる placeholder に対するトークン予測が行われる前に、placeholder の削除や追加が行われる可能性がある。最初に予測した placeholder の情報をなるべく生かすために、制約に placeholder が含まれている場合、再配置と placeholder 挿入の操作を初回のみスキップしてトークン予測を先に行うようにした。

3　疑似部分翻訳を用いた実験

2.2 節で提案した NMT が、部分翻訳を効果的に制約として活用できるのか、また追加した 3 つの機能が有効であるかを評価した。評価では、実際の使用環境を再現して、開発データを用いてユーザが翻訳ルールを定義した後、評価データを用いて翻訳ルールに基づき生成された部分翻訳を用いることが望ましい。しかし、このような方法では大量かつ多様なデータの準備が難しいため、その前段階として、参照文の一部を特殊トークンに置換することで自動生成した擬似部分翻訳を用いて NMT の評価を行った。

3.1　擬似部分翻訳の生成方法

翻訳ルールに基づいて組み立てられた部分翻訳は、語句や節など文法的な単位の未翻訳部分を持つ。なるべく実際の部分翻訳の形を模倣するために、以下の手順で擬似部分翻訳を生成した。

1. 参照文を Stanza で係り受け解析する。
2. 係り受け解析木における、特定の依存関係[3]にある単語間のパスを分割することで、文を一定のまとまりを持つ要素単位に分解する。
   例）`Using / the GTS / clear / the DTC`
3. 制約に含める文要素をランダムで選ぶ。ある要素を制約に含める確率を p とし、制約に含めない要素はトークンに分割し、<tok> に置換する。実際の部分翻訳では、未翻訳部分は原言語トークン列であるが、本実験では <tok> を未翻訳部分とみなす[4]。
   例）`Using <tok> <tok> clear <tok> <tok>`

部分翻訳を制約として用いる NMT は 2.2 節で説明した方法で <tok> の数を基準にランダムな数の placeholder を挿入した制約を複数生成し、使用する。

例）`Using <plh> clear <plh> <plh> <plh>`

3.2　実験方法

設定　部分翻訳の完成度合いに多様性を持たせるために、3.1 節の手順 3 における確率 p は、0 から 0.9 まで 0.1 間隔に合計 10 種類を試した。EDITOR は、ハードな制約設定に加えてソフトな制約も試した。また、EDITOR に追加した 3 つの機能の有効性を確かめるために、各機能を使わない設定での実験も行った。なお、placeholder を使わない設定では、<tok> を削除した疑似部分翻訳を用いた。リランキングを行わない設定では、ランダムな数の placeholder を含む制約を複数ではなく、1 つのみ使用する。

データセット　NTCIR-9 特許機械翻訳テストコレクション [11] の英日、日英データを使用した。学習データは約 320 万文、検証データは 914 文、テストデータは 2000 文を使用した。

3）　本研究で対象とした依存関係：root, obj, nsubj, iobj, csubj, ccomp, xcomp, obl, nmod, parataxis, conj, cc, advcl

4）　参照文におけるトークン数を既知のものとして利用しているため、原文におけるトークン数から予測するよりも易しいタスクとなっている。

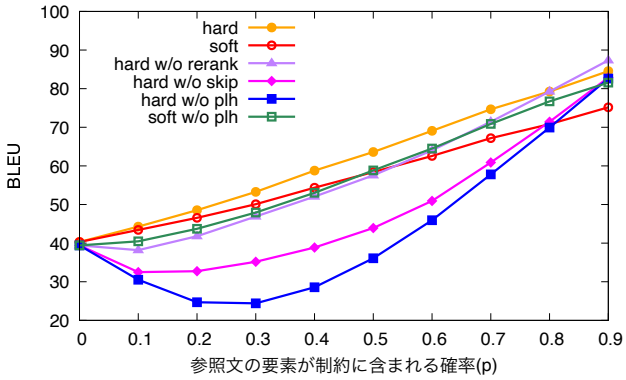

図 2　擬似部分翻訳を制約として与えた NMT の性能

モデルの学習　EDITOR の学習には教師モデルを学習データで学習した後、学習データの参照文を教師モデルの出力に置き換えて学生モデルを学習するシーケンスレベル知識蒸留法 [12] を用いた。本実験では、Transformer 教師モデルから EDITOR 学生モデルを学習した。

本実験では、日英方向を順方向としたとき、順方向 EDITOR モデル、知識蒸留のための順方向 Transformer モデル、リランキングのための逆方向 Transformer モデルとターゲット側 Transformer 言語モデルの合計 4 つのモデルを学習した。学習設定の詳細は付録 B に示す。

評価　翻訳出力に対して、SentencePiece [13] と NLTK detokenizer [14] によるデトークナイズを行った上で、SacreBLEU [15] により BLEU を計算した。

3.3　実験結果

図 2 に結果を示す。基本的には、参照文の要素が制約に含まれる確率 p が高くなるほど、文に対する制約トークンの割合は大きくなる。図中の `hard` は、ハードな制約設定の EDITOR に 3 つの拡張機能を追加した、本研究の提案手法である。制約に含まれる参照文の要素が増えるほど BLEU が安定的に上昇するため、制約を効果的に活用できていることがわかる。ソフトな制約を使用した場合 (`soft`) も BLEU は安定的に上昇しているが、制約が守られないことが多くなるため、`hard` に比べると BLEU が低くなる。

`hard w/o plh` と `soft w/o plh` はそれぞれ、ハードな制約設定とソフトな制約設定で placeholder を持たない部分翻訳を制約として与えた場合である。placeholder が存在しないので、自動的にリランキングとスキップ機能も適用対象外となり、オリジナルの EDITOR と同等の設定である。ハードな制約では、ひと続きの制約トークン間に新しいトークンの挿入が禁止される。そのため、p が比較的小さい時、すなわち複数の制約要素が文中で散在する条件下で、placeholder を削除してしまうと、BLEU が大きく下がる。ソフトな制約では、生成の過程で新しいトークンの挿入も行われるので、このような問題は生じないが、提案手法より一貫して BLEU が低い。

提案手法からリランキングの機能を除いた場合 (`hard w/o rerank`) は、$p < 0.8$ の区間で、提案手法と比べて BLEU が低い。特に、p の値が低い区間、すなわち制約が比較的短い場合では、提案手法と比べ BLEU が大きく下回り、$p = 0.1$ のときは、制約なしの場合 ($p = 0$) よりも BLEU が低い。大部分の区間で提案手法はより高い BLEU を達成し、下がる区間なく安定的な上昇を見せるため、リランキング機能は有効であったと考えられる。

提案手法からスキップ機能を外した場合 (`hard w/o skip`)、BLEU が大きく下がる区間が存在する。これは、2.2 節で紹介した placeholder の数が勝手に変わってしまう現象によるものである。このような現象が起こる理由は、EDITOR モデルが placeholder を含む制約に対して再配置と placeholder 挿入操作を学習していないためだと考えられる。EDIOTR モデルの学習の仕方自体を変える方法もあるが、より簡便なスキップ機能の導入により、大きく性能を向上できている。

以上のように、提案手法が大部分の区間で最も高い BLEU を達成したことから、追加した 3 つの機能が有効であることを確認できた。

4　おわりに

本稿では、ユーザ定義の翻訳ルールによって翻訳を語彙的・構文的に制御できるシステム URNMT を提案した。URNMT は、翻訳ルールに基づく部分翻訳の生成器と、部分翻訳を制約とした NMT から構成される。部分翻訳を効果的に活用するために、語彙制約付き NMT である EDITOR の機能を拡張した。擬似部分翻訳を用いた実験から、部分翻訳が効果的に活用できること、追加した各機能が有効であることを確認した。

提案システムのプロトタイプはすでに完成しているため、今後はユーザによる翻訳ルールの定義とそれに基づき生成した部分翻訳を用いた評価実験を行う予定である。また、ユーザによるシステム全体の利用を通じたユーザビリティ評価も行う。

謝辞

本研究は、トヨタ自動車株式会社との共同研究の枠組みで行われた。また、本研究の一部は、JSPS 科研費（課題番号：19H05660）および KDDI 財団調査研究助成の支援を受けた。

参考文献

[1] Matt Post and David Vilar. Fast lexically constrained decoding with dynamic beam allocation for neural machine translation. In **Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies**, pp. 1314–1324, New Orleans, Louisiana, USA, June 2018.

[2] Raymond Hendy Susanto, Shamil Chollampatt, and Liling Tan. Lexically constrained neural machine translation with Levenshtein Transformer. In **Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics**, pp. 3536–3543, Online, July 2020.

[3] Weijia Xu and Marine Carpuat. EDITOR: An edit-based transformer with repositioning for neural machine translation with soft lexical constraints. **Transactions of the Association for Computational Linguistics**, Vol. 9, pp. 311–328, March 2021.

[4] Josef Jon, João Paulo Aires, Dusan Varis, and Ondřej Bojar. End-to-end lexically constrained machine translation for morphologically rich languages. In **Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing**, pp. 4019–4033, Online, August 2021.

[5] Takahiro Makino, Rei Miyata, Seo Sungwon, and Satoshi Sato. Designing and building a Japanese controlled language for the automotive domain: Toward the development of a writing assistant tool. In Annette Klosa-Kückelhaus, Stefan Engelberg, Christine Möhrs, and Petra Storjohann, editors, **Dictionaries and Society. Proceedings of the XX EURALEX International Congress**, pp. 409–422. Mannheim, Germany, July 2022.

[6] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In **Proceedings of the 31st International Conference on Neural Information Processing Systems**, pp. 6000–6010, Red Hook, New York, USA, December 2017.

[7] Ashutosh Kumar, Kabir Ahuja, Raghuram Vadapalli, and Partha Talukdar. Syntax-guided controlled generation of paraphrases. **Transactions of the Association for Computational Linguistics**, Vol. 8, pp. 329–345, 2020.

[8] Peng Qi, Yuhao Zhang, Yuhui Zhang, Jason Bolton, and Christopher D. Manning. Stanza: A Python natural language processing toolkit for many human languages. In **Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations**, pp. 101–108, Online, July 2020.

[9] 佐野正裕, 佐藤理史, 宮田玲. 文末述語における機能表現検出と文間接続関係推定への応用. 言語処理学会 第 26 回年次大会 発表論文集, pp. 1483–1486, オンライン, March 2020.

[10] Kyra Yee, Yann Dauphin, and Michael Auli. Simple and effective noisy channel modeling for neural machine translation. In **Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing**, pp. 5696–5701, Hong Kong, China, November 2019.

[11] Isao Goto, Bin Lu, Ka Po Chow, Eiichiro Sumita, and Benjamin Tsou. Overview of the Patent Machine Translation Task at the NTCIR-9 Workshop. In **Proceedings of NTCIR-9 Workshop Meeting**, pp. 559–578, Tokyo, Japan, December 2011.

[12] Yoon Kim and Alexander M. Rush. Sequence-level knowledge distillation. In **Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing**, pp. 1317–1327, Austin, Texas, USA, November 2016.

[13] Taku Kudo and John Richardson. SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In **Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations**, pp. 66–71, Brussels, Belgium, November 2018.

[14] Steven Bird, Ewan Klein, and Edward Loper. **Natural Language Processing with Python**. O'Reilly Media Inc., 2009.

[15] Matt Post. A call for clarity in reporting BLEU scores. In **Proceedings of the Third Conference on Machine Translation**, pp. 186–191, Brussels, Belgium, October 2018.

[16] Taku Kudo and Yuji Matsumoto. Japanese dependency analysis using cascaded chunking. In **Proceedings of the 6th Conference on Natural Language Learning**, pp. 63–69, Stroudsburg, Pennsylvania, USA, 2002.

[17] Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. fairseq: A fast, extensible toolkit for sequence modeling. In **Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations**, pp. 48–53, Minneapolis, Minnesota, USA, June 2019.

A　部分翻訳生成器の例

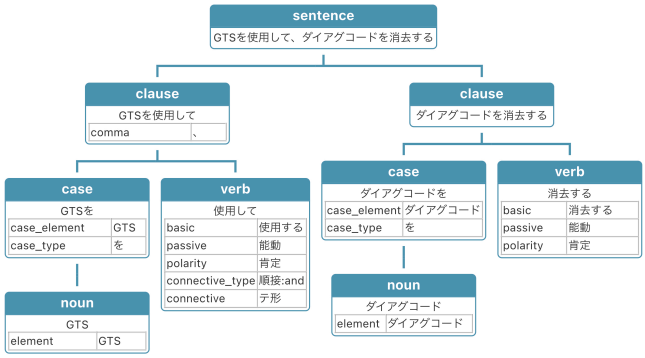

図 3　Source Tree の例

図 3 は Source Tree の例である。2 つの節で構成されている、「使用する」という動詞がテ形で使われている、といった情報が保持されている。

図 4　辞書ルールの例

図 4 は辞書ルールの例である。この例は、Source Tree 中の「ダイアグコード」という要素に対して、「DTC」に変換する（変換方法）、「code」という属性を持つ（語句の属性）、という情報を追加するルールである。

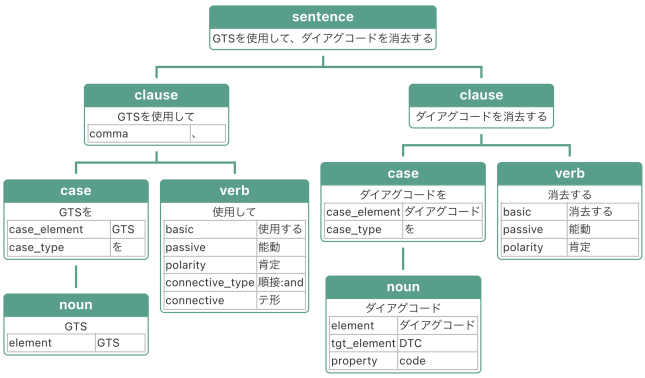

図 5　辞書ルールを適用した Source Tree の例

図 5 は、図 4 の辞書ルールを図 3 の Source Tree に適用した例である。「ダイアグコード」のノードに、辞書ルールで定義した変換方法と語句の属性情報が付加されている。

図 6 は組み立てルールの例である。この例は、「A して、B する」という構文は「A|ing B」という構文に変換することを定めている。図 6 の適用条件と図 5 の Source Tree を比較すると、条件の全ての要素

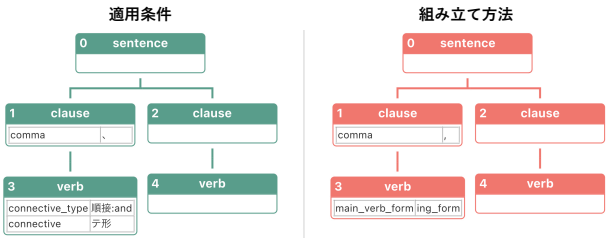

図 6　組み立てルールの例

が Source Tree に含まれるため、この組み立てルールが適用される。適用条件と組み立て方法で同じ番号を持つノードは互いに繋がり、組み立て方法を Target Tree に足す位置を決定する。

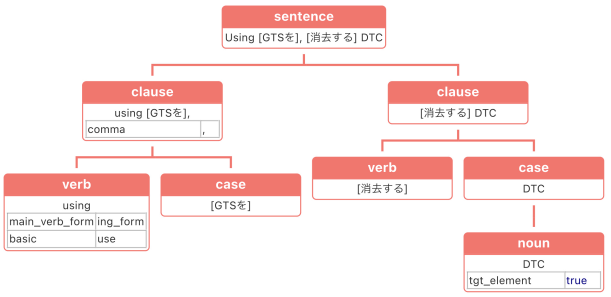

図 7　Target Tree の例

図 7 は、複数の組み立てルールを用いて生成された Target Tree の例である。「GTS を」や「消去する」など翻訳ルールが定義されていない部分は、原文文字列のまま Target Tree に組み込まれている。Target Tree のルートノードに表示されている文が、Target Tree を組み立てて生成された部分翻訳である。

B　モデルの学習設定

学習データは日本語を MeCab [16]、英語を Stanza [8] でトークナイズした後、SentencePiece [13] でサブワード分割した。SentencePiece の語彙サイズは日英共有で 16000 にした。

全てのモデルは fairseq [17] で学習しており、Vaswani ら [6] の Transformer (base) の設定に従っている。ドロップアウト率は 0.3 を使用した。

Transformer モデル [6] では、バッチサイズを 44000 とした。最大学習率は 0.0006 とし、検証データの BLEU をエポック毎に評価して、10 エポックの間改善が見られない場合、学習を終了した。

EDITOR モデル [3] では、バッチサイズを 64200 とした。最大学習率は 0.0005 とし、アップデート回数 30 万まで学習を行った。10000 アップデート毎に保存したチェックポイントの中で検証データの BLEU が最も高いものを選んだ。