Vol.2020-NL-245 No.1
2020/9/30

非自己回帰モデルを用いた高速な日本語文法誤り訂正

本間 広樹[1,a)]　小町 守[1,b)]

概要：語学学習者のライティング支援手法として，文法誤り訂正やキーワード推薦が挙げられる．近年，ニューラルネットワークを用いたモデルの発展に伴い，それらを用いた文法誤り訂正の研究が盛んに行われている．例えば，多層畳み込みニューラルネットワークを用いた文法誤り訂正の研究では，統計的手法を用いた文法誤り訂正の訂正精度を超える結果を示している．しかし，ニューラル生成タスクにおいて，トークンを逐次的に出力する一般的な自己回帰型のデコーダでは文長に比例して推論速度が遅くなる問題が存在する．これに対し，全トークンの出力を同時に行うことで高速化を図る非自己回帰型のデコーダが提案され，広く研究され始めている．例えば，文法誤り訂正を局所系列変換タスクと捉え，並列反復編集モデルを用いることで高速な文法誤り訂正を実現している研究や，入力文の各トークンに対する編集操作の反復的なタグ付けにより文法誤り訂正を行い，高速な訂正速度で高い訂正精度を出している研究がある．しかし，これらの先行研究は対象を英語に絞り，言語知識を用いた編集操作を事前に設定しておくことで文法誤り訂正を行っている．そこで，本研究では，事前の規則の作成を必要とせず，修正文を直接高速に生成する手法として，非自己回帰モデルに着目し，日本語文法誤り訂正タスクに取り組む．また，応答時間が短くユーザビリティの高いライティング支援システムの構築に向けて，文長と訂正速度，訂正精度の観点から分析する．

1. はじめに

語学学習者のライティング支援手法として，文法誤り検出や文法誤り訂正，キーワード推薦が挙げられる．この内文法誤りに着目すると，検出と訂正では訂正のほうが難しいタスクであるが，その分有用性も高いことが言える．訂正ができれば，検出と同様に誤り部分を提示できる上に，どのように訂正すべきかの例を提示することが可能となる．

近年，ニューラルネットワークを用いたモデルの発展に伴い，それらを用いた文法誤り訂正の研究が盛んに行われている．Junczys-Dowmun ら [1] は，文法誤り訂正を低リソースの機械翻訳タスクと捉えて，従来の統計的手法を超える訂正性能を出すことに成功しており，清野ら [2] は強力なニューラル機械翻訳モデルである Transformer [3] を用いて文法誤り訂正を行い，疑似データで事前学習を行うことで，当時の最先端の訂正性能を打ち出している．系列全体を見ることが可能なため，ニューラルモデルでは，n-gram を用いる統計的手法ではなし得ない，長距離依存関係のある誤りの訂正を行える可能性がある．しかし，ニューラルネットワークを用いた一般的な系列変換モデルの自己回帰型のデコーダには推論速度が遅いという問題がある．これは推論時にトークンを逐次的に生成するためであり，文長が長いほど遅くなってしまう．ライティング支援システムに応用することを考えると，推論速度が遅ければ遅いほどその使い道が限られるか，使い勝手の悪いものになってしまう．

文長に比例して推論速度が遅くなる問題に対し，全トークンの出力を同時に行うことで高速化を図る非自己回帰型のデコーダが提案され [4]，広く研究され始めている．非自己回帰型のデコーダを用いる系列変換には，各トークンを独立して出力することに起因するマルチモダリティ問題 [4] があり，自己回帰型に対して精度が大きく落ちる問題がある．これに対して，反復校正 [5] や知識蒸留 [6] といった手法を用いることで自己回帰型に近い精度が出るようになってきている．Levenshtein Transformer [7] は，削除と挿入を反復的に行う非自己回帰ニューラル機械翻訳モデルであり，機械翻訳タスクと文書要約における有用性が検証されている．

また，非自己回帰モデルを用いた，系列タグ付けによる高速な文法誤り訂正手法の提案もなされている．Awasthi らが文法誤り訂正を局所系列変換タスクと捉え，編集タグを並列に反復的に適応する非自己回帰モデルを用いることで高速な文法誤り訂正を実現している [8] ほか，Omelianchuk

1 東京都立大学
Asahigaoka 6–6, Hino, Tokyo 191–0065, Japan
a) homma-hiroki@ed.tmu.ac.jp
b) komachi@tmu.ac.jp

ⓒ 2020 Information Processing Society of Japan

らは，入力文の各トークンに対する編集操作の反復的なタグ付けにより非自己回帰的に文法誤り訂正を行い，以前の研究に比べて高速な訂正速度で高い訂正精度を出している [9]．しかし，これらの手法は対象を英語に絞り，言語知識を用いた編集操作をタグとして事前に用意しておくことで訂正を実現している．

本研究では，高速に文法誤り訂正を行う手法として，非自己回帰モデルに着目する．対象言語を日本語とし，事前の変換規則の作成を必要としない非自己回帰モデルを用いた文法誤り訂正を行い，ライティング支援システムへの応用を考慮した分析を行う．特に，訂正性能と推論速度の関係から分析を行い，入力途中の文に対する扱いや，非自己回帰モデル特有のハイパーパラメータの影響の分析も行う．

2. 関連研究

2.1 自己回帰ニューラル機械翻訳

自己回帰ニューラル機械翻訳は，2013 年に Kalchbrenner らによってニューラルネットワークを用いて入力系列から出力系列を直接学習する機械翻訳手法であるニューラル機械翻訳 [10] が提案されて以降，長らく一般的なニューラル機械翻訳におけるデコーディング手法である．このデコーディング手法は，推論時の系列生成時に回帰型言語モデル [11] を利用するものである．

原文を $X = \{x_1, \ldots, x_{T'}\}$，目的文を $Y = \{y_1, \ldots, y_T\}$ とすると，自己回帰型のニューラル機械翻訳モデルは次のような条件付き確率の連鎖により目的文を計算する．

$$p(Y|X;\theta) = \prod_{t=1}^{T+1} p(y_t|y_{0:t-1}, x_{1:T'};\theta) \qquad (1)$$

ここで，y_0 と y_{T+1} はそれぞれ文の先頭と終端を表す特殊トークンとする．また，θ はモデルのパラメータを表す．

2.2 非自己回帰ニューラル機械翻訳

非自己回帰ニューラル機械翻訳は，各トークンを同時に独立に生成するデコーディング手法であり，2018 年に Gu らによって提案されている [4]．この手法はデコーディングを高速化する手法として注目され，近年広く研究がなされている．

Gu らが提案した手法では原文の各単語が目的文側の何単語分に相当するかを予測するファティリティという概念を導入し，次のようにデコーディングを行う．

$$p(Y|X;\theta) = \sum_{f_1,\ldots,f_{T'}\in\mathcal{F}} \left(\prod_{t'=1}^{T'} p_F(f_{t'}|x_{1:T'};\theta) \cdot \prod_{t=1}^{T} p(y_t|x_1\{f_1\},\ldots,x_{T'}\{f_{T'}\};\theta) \right) \qquad (2)$$

ここで，$\mathcal{F}$ はすべてのファティリティ系列の集合であり，系列を合計すると目的文の系列長になる．また，$x\{f\}$ はトークン x を f 回繰り返すことを表す．このように，非自己回帰型のデコーディング手法では，目的文の文長を予測する必要性がある．ファティリティを用いる手法以外にも，トークン数を予測してプレースホルダを挿入し，次のステップでプレースホルダをトークンに置換する手法 [7] が提案されている．この手法の詳細は次節で説明する．

また，非自己回帰ニューラル機械翻訳にはマルチモダリティ問題と呼ばれている問題が存在する [4]．これは翻訳候補が複数存在する場合，各トークンが独立に選択されるためにそれぞれ別の翻訳候補からトークンを選択してしまい，文全体としておかしい文が生成されてしまう問題である．具体的には，主にトークンの繰り返しや，トークンの欠如といった形で現れる．この問題に対して，出力を反復的に校正していく手法 [5][7] や，文をセクションに区切り，その各セクションでは自己回帰的に，全体としては非自己回帰的に出力し，繰り返しが発生したセクションを削除することで解決を行う手法 [12] などが提案されている．更に，非自己回帰モデルを学習する際に，自己回帰モデルの出力を学習データに用いる知識蒸留手法 [13] が有効であることが報告されている [6][14]．これは，自己回帰モデルの出力は多様性が抑制され，マルチモダリティ問題が起きづらく，学習が容易になるからだと言われている [14]．

本研究では，上述の非自己回帰ニューラル機械翻訳における既知の問題の対応策を導入し，自己回帰型からの精度の低下を抑えつつ，推論速度の向上を達成しているモデルである Levenshtein Transformer [7] に焦点を当てる．本モデルの詳細は次節にて述べる．

2.3 Levenshtein Transformer

Levenshtein Transformer [7] は，2019 年に Gu らが提案した反復校正を行う非自己回帰ニューラル機械翻訳モデルである[*1]．機械翻訳タスクと文書要約における有用性は当該論文にて検証されているが，文法誤り訂正における有用性は未検証である．

このモデルは Transformer [3] ブロックを基本的な構成要素とし，原文は各 Transformer ブロックに与えられる．そして，削除分類器とプレースホルダ分類器，トークン分類器を持ち，系列に対してトークンの削除，挿入を停止条件まで繰り返し行うことで文を生成する．停止条件は，設定した最大反復校正回数及び以前と同じ目的文の生成である．最初の反復では削除が行われず，空の系列にプレースホルダを挿入することから始める．削除分類器は各トークンを削除するかどうかを判別し（これによって主にトークンの繰り返し誤りが修正される），プレースホルダ分類器により系列のどこに，何トークン挿入するかが決定され

[*1] 元の論文では「部分的な自己回帰モデル（partially autoregressive model）」と呼んでいるが，デコード時に全トークンを同時に出力するモデルであるため，本稿では非自己回帰モデルと呼ぶ．

（これによって主に必要なトークンの欠如が修正される），トークン分類器によりプレースホルダを実際のトークンに置き換える．つまり，削除分類器によるトークンの削除，プレースホルダ分類器とトークン分類器によるトークンの挿入を交互に繰り返し行うことで，マルチモダリティ問題を回避しつつ文を生成する．

具体的なモデル構造は以下のとおりである．まず，l 番目のブロックからくる状態は次のようになる．

$$\boldsymbol{h}_0^{(l+1)}, \boldsymbol{h}_1^{(l+1)}, \ldots, \boldsymbol{h}_n^{(l+1)} = \begin{cases} E_{y_0} + P_0, E_{y_1} + P_1, \ldots, E_{y_n} + P_n, & l = 0 \\ \mathrm{TransformerB}_l\left(\boldsymbol{h}_0^{(l)}, \boldsymbol{h}_1^{(l)}, \ldots, \boldsymbol{h}_n^{(l)}\right), & l > 0 \end{cases} \quad (3)$$

ここで，E 及び P は，それぞれトークン及び位置埋め込みであり，TransformerB は Transformer ブロックを指す．また，y_0 及び y_n はそれぞれ開始及び終了を表す境界トークンである．次に，このデコーダ出力 $(\boldsymbol{h}_0, \boldsymbol{h}_1, \ldots, \boldsymbol{h}_n)$ により，削除分類とプレースホルダ分類，トークン分類は次のように行われる．削除分類器は $\mathrm{softmax}\left(\boldsymbol{h}_i \cdot A^\top\right), (i = 1, \ldots n-1)$ により，境界トークン以外のトークンに対して削除 (0) か保持 (1) かの二値分類を行い，削除を行う．プレースホルダ分類器は $\mathrm{softmax}\left(\mathrm{concat}\left(\boldsymbol{h}_i, \boldsymbol{h}_{i+1}\right) \cdot B^\top\right), (i = 0, \ldots n-1)$ により，すべてのトークンとトークンの間において，それぞれ何個のプレースホルダを挿入するかを $0 \sim K_{\max}$ で分類し，対応する数の特殊トークン<PLH>を挿入する．ここで $K_{\max}$ は 1 箇所に対して一度に挿入する最大のトークン数であり，基本的にモデルの原文の最大長と同じ 255 が適用される．トークン分類器は $\mathrm{softmax}\left(\boldsymbol{h}_i \cdot C^\top\right), (\forall y_i = \texttt{<PLH>})$ により，すべての特殊トークン<PLH>をそれぞれ語彙 $\mathcal{V}$ の要素に分類し，置換する．ここで，A，B 及び C は状態あるいは状態の結合の次元数をクラス数に線形変換するための行列である．

2.4 文法誤り訂正

文法誤り訂正は，句読点や文法，単語選択などの誤りを修正するタスクであり，様々な手法の研究がなされてきている．近年では，ニューラル機械翻訳の発展に伴い，機械翻訳タスクとして取り組まれていることが多い．これは原文を語学学習者文，目的文を対応する訂正文とする大量の対訳コーパスを元にモデルを学習する手法である．自然言語処理の最先端の研究をまとめている NLP-progress[*2]に掲載されている，英語の文法誤り訂正の共通タスクである BEA Shared Task - 2019 [15] に取り組んだ研究のほぼすべて [9][2][16][17][18][19] が Transformer ベースのモデルを利用した手法になっている．この内，Li ら [19] は畳み込みニューラルネットワークベースのモデルを Transformer ベースのモデルと組み合わせたシステムを用いている．こで，Li らは畳み込みニューラルネットワークのアーキテクチャとして Chollampatt らの手法 [20] を採用し，出力のリランキングは行っていない．

本研究では，文法誤り訂正の高速化に主眼を置くため，Transformer ベースの手法と比較して高速な畳み込みニューラルネットワークベースの手法である Chollampatt らの手法 [20] を自己回帰モデルのベースラインとして使用する．また，Li らと同様に出力のリランキングは行わない．

非自己回帰モデルを用いる高速な文法誤り訂正の既存研究としては次のようなものがある．Awasthi ら [8] は，文法誤り訂正を局所系列変換タスクと捉えて並列反復編集モデルにより高速に解いている．Omelianchuk ら [9] は，系列タグ付けを反復的に行うことで解いている．タグ付けはトークンの生成よりも処理が軽く，高速であることが示されている．しかし，これらの手法はどちらも，事前に用意した接尾辞の変換規則や動詞の活用辞書など，言語知識を利用しており，別言語への適用が容易でない．本研究では，学習コーパスのみを用いた手法を扱う．

また，多くの先行研究と同様に，本研究においても 1 文単位での文法誤り訂正を扱う．つまり，文末表現の不一致など，文を跨いだ修正は考えない．

*2 http://nlpprogress.com/

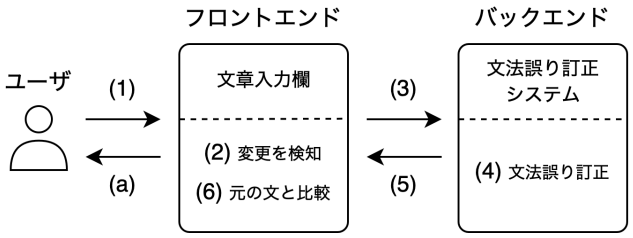

図 1 ライティング支援システムの概略図

3. 提案手法

本論文の目的は，非自己回帰モデルを用いた日本語文法誤り訂正の有効性を，速度や精度の観点から分析することである．ここで，応用に即した分析を行うため，文法誤り訂正システムの応用先にライティング支援システムを仮定する．本章ではこの仮定するライティング支援システムについて説明する．図 1 にシステムの概要図を示す．

3.1 システムの構成

システムは次の 2 つの要素から構成される．

フロントエンド 文章入力欄を持ち，ユーザからの入力を受け付け，文法誤り訂正結果を提示する．

バックエンド 文法誤り訂正システムを持ち，フロントエンドからユーザの入力文を受け取り，その文法誤り訂正結果を返す．

3.2 システムの流れ

システムの流れは以下の通りである．

(1) ユーザが文章入力欄の文章を変更（入力/削除）[*3]
(2) フロントエンドにおいて，ユーザからの変更を検知
(3) 変更が行われた文をバックエンドに送信
(4) バックエンドにおいて，受信した文を文法誤り訂正
(5) 訂正結果をフロントエンドに送信
(6) 訂正結果を元の文と比較
　(a) 異なる場合，訂正結果を提示
　(b) 等しい場合，何も提示しない

3.3 システムの考察

まず，システムの入力について考える．システムの流れの (1) および (2) におけるユーザからの変更に関して次の 2 つの問題が考えられる．

1 つ目は入力途中の文の処理である．ユーザが 1 文を書き終えた場合（多くは改行が入力されたとき）には，その文に対して文法誤り訂正結果を提示すべきなのは明白である．しかし，入力途中の文においては明白ではない．なぜなら入力途中の文は，不完全な文であることや文の短さにより正確な文法誤り訂正が行えない可能性があるからである．この，入力途中の文の文法誤り訂正結果を提示すべきか不明である問題に対し，入力中の文が短い場合は訂正性能が落ちるが，入力中の文が長い場合はある程度の訂正性能が出るという仮説を立て，4.4 節で検証する．

2 つ目は，入力方法の問題である．日本語の場合は英語の入力と異なり，かな漢字変換システムを通して行われる．つまり，かな漢字変換後の状態を受け取るとすると，ユーザからの変更がどの単位（文字か単語か，文節か，あるいは文全体か）で検知可能か自明でない[*4]．本論文では，かな漢字変換確定後の文字列を扱い，単語単位で入力が検知されることを仮定して分析を行う．

次に，システムの応答時間について考える．システムの流れの (2) から (6) の処理速度がシステムの応答時間となり，この長短がユーザ体験に関わる．即応性が求められるだけでなく，ユーザは応答速度が変化するシステムよりも，応答速度が一定であるシステムを好む [21] ことが知られている．(2) から (6) の処理の内，最もウェイトがかかると思われる (4) の処理の時間の分析を 4.5 節にて行う．

4. 実験

4.1 実験設定

4.1.1 データセット

学習データと開発データ，評価データには Lang-8 学習者コーパス [22] 由来のものを用いる．開発データ，評価データは Lang-8 学習者コーパス内の学習者文に対して訂正基準を設けた上で訂正文を付与し直したクリーンなデータ [23] を用いる．このデータは Lang-8 学習者コーパスに元から含まれる訂正文に比べてノイズが少なく，すべての文に対して複数の参照が付与されており，評価に有用であると考えられる．データの前処理やデータの分割は本データを利用した先行研究である小山ら [23] に従う．データの文数を**表 1** に示す．訂正者数は学習者の文 1 文に対する参照文の数を意味する．

表 1 データの文数と訂正者数

	文数	訂正者数
学習データ	1,093,633	1
開発データ	806	2
評価データ	663	3

4.1.2 非自己回帰モデル

本研究では，Transformer ベースの非自己回帰ニューラルモデルである Levenshtein Transformer [7] をそのまま文法誤り訂正に適用する．学習はバッチサイズを 64,000 トークンとして 300,000 回更新し，開発データに対する GLEU スコア [24] が最も高いエポックのモデルを選択する．その他各種ハイパーパラメータは先行研究に従う．実装には公開されている PyTorch ベースのコード[*5]を用いる．本稿ではこれ以降，本モデルを LevT モデルと呼ぶ．

更に，学習データの内の訂正文を自己回帰モデルの出力に置き換えたものを LevT の知識蒸留モデルとして学習する [6]．ハイパーパラメータは LevT モデルと同様とし，自己回帰モデルには次節で説明するモデルを利用する．本稿ではこれ以降，本モデルを LevT+KD モデルと呼ぶ．

また，先行研究における最大反復校正回数は 9 回で実験が行われているが，文法誤り訂正における反復校正回数による性能の変化は明白でない．そのため，最大反復校正回数を変更した場合の性能も評価する．

4.1.3 自己回帰モデル

本研究では，文法誤り訂正の高速化に主眼を置くため，Transformer ベースの手法と比較して高速な畳み込みニューラルネットワークベースの手法である Chollampatt らの手法 [20] をベースラインとして使用する．学習は先行研究に従いバッチサイズを 128 文として 100 エポックまで更新し，開発データに対する GLEU スコアが最も高いエポックのモデルを選択する．また，Levenshtein Transformer と条件を揃えるため，出力のリランキングは行わない．その他の各種ハイパーパラメータは先行研究に従う．実装には公開されている PyTorch ベースのコード[*6]を用いる．本稿ではこれ以降，本モデルを CNN モデルと呼ぶ．

*3 簡単のため 1 行に 1 文の入力がなされるものとする．つまり，ユーザは 1 文ごとに改行を行う．
*4 これはユーザの語学学習レベルや日本語入力レベルによっても変化すると思われる．
*5 https://github.com/pytorch/fairseq/tree/master/examples/nonautoregressive_translation
*6 https://github.com/nusnlp/mlconvgec2018

表 2　各モデルの訂正性能の評価

モデル	GLEU	適合率	再現率	$F_{0.5}$
CNN	73.3	0.159	**0.225**	0.169
LevT	72.1	0.102	0.185	0.112
LevT+KD	**75.2**	**0.213**	0.217	**0.214**

表 3　各モデルが正しく修正した誤りのカテゴリー別の数

モデル	欠落 (242)	誤選択 (441)	余剰 (124)
CNN	**33**	73	**32**
LevT	22	54	5
LevT+KD	**33**	**79**	20

4.1.4　トークン化

すべてのモデルにおいて入力データのトークン化は次のように行う．まず，辞書として現代書き言葉コーパス UniDic[*7]（バージョン 2.2.0）を用いた MeCab[*8]（バージョン 0.996）を用いて形態素単位に分かち書きをする．次に，低頻度語に対処するため，Byte Pair Encoding（BPE）[25] を適用し，サブワード単位に分割する．このとき，NFKC ベースの文字正規化も同時に行い，学習者文側と訂正文側で語彙を共有し[*9]，語彙サイズは 30,000 語とする．実装には sentencepiece[*10]を用いる．

4.1.5　推論速度の計測

推論速度の計測は以下のような設定で行う．計算機環境としては，GPU を用いず，Intel® Xeon® プロセッサー E5-2660 上で動作させる．応用時のパフォーマンスを知るため，バッチサイズを 1 文に設定し，1 文ずつ処理を行う．計時は Python 言語の time モジュールを用いて行う．より詳細には，トークン化を行う前段階の文を入力したときから，文法誤り訂正結果の文が返ってくるまでのシステムクロックの変化を 1 文の推論速度として扱う．

4.1.6　文法誤り訂正の訂正性能の評価尺度

文法誤り訂正の訂正性能の評価尺度として GLEU スコア [24] 及び適合率を再現率の倍重く捉える $F_{0.5}$ スコアを用いる．$F_{0.5}$ スコアを求めるための単語の対応付けは ERRANT[*11]を使用して自動で行う．ただし，ERRANT は英語用のシステムであり直接使用できないため，言語情報を用いないように編集距離を計算する距離関数にレーベンシュタイン距離を指定する．また，$F_{0.5}$ スコアを測る際は，単語単位での一致率を見る．

4.2　訂正性能

完成した文に対する文法誤り訂正の有効性を確認するため，評価データを用いて各モデルを評価する．

結果を**表 2** に示す．GLEU，$F_{0.5}$ スコアともに知識蒸留を行わない LevT は CNN に負けるが，知識蒸留を行った LevT+KD は CNN を上回っている．再現率と適合率に着目すると，知識蒸留によってどちらも上昇しているが，特に適合率が大幅に上がっていることが見て取れる．ここから，知識蒸留は文法誤り訂正において適合率の向上に大きく寄与することが分かる．

知識蒸留が与える影響のより詳細な分析のため，モデルが正しく修正した誤りのカテゴリー別の数を**表 3** に示す．ここで，「欠落」は入れるべき単語を入れていない誤り，「誤選択」は誤った単語を選択する誤り，「余剰」は余分な単語を入れている誤りであり，括弧内の数はその誤りの最大数である[*12]．LevT と LevT+KD を比べると，知識蒸留により，すべての種類の誤りに対する訂正数が増加しており，特に「余剰」に対する訂正能力が増加していることが伺える．これは，CNN の「余剰」に対する訂正能力を知識蒸留により受け継ぐことができたからだと考えられる．

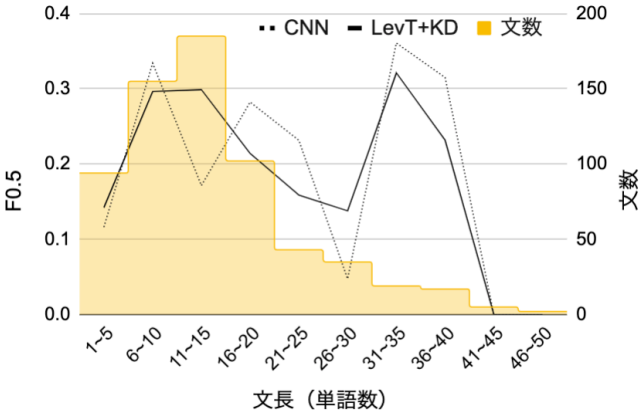

図 2　文長ごとの性能変化

次に，文長ごとの訂正性能を**図 2** に示す．実線と点線はそれぞれ，LevT+KD と CNN の $F_{0.5}$ スコアを，階段面は文数を表している．2 つのモデルにはある程度の相関性が見て取れ，どちらも文長が短い場合（1 から 5 単語）及び文長が長い場合（41 単語以上）は訂正性能が下がることが分かる．ただし，文長が 21 単語以上は文数が少なく，誤差の大きい結果となっている．文長が長い場合に訂正性能が下がる原因として，ニューラルモデルが長文に弱いという問題が考えられる．また，文長が短い場合はモデルが文脈を捉えられず，誤った訂正をしてしまう場合が多いのではないかと考えられる．詳細に見てみると，実際に，どちらのモデルにおいても単語数が短いときは誤った訂正が正しい訂正に比べて 7〜8 倍程度存在することが分かった．

以上より，LevT+KD は CNN に比肩しうる訂正性能を出しており，文法誤り訂正において，知識蒸留を取り入れ

*7 https://unidic.ninjal.ac.jp/
*8 https://taku910.github.io/mecab/
*9 事前実験として学習者文側を文字単位，訂正文側をサブワード単位にした場合も行ったが GLEU の値がわずかに下がる結果であったため，両側ともサブワード単位にトークン化を行う．
*10 https://github.com/google/sentencepiece
*11 https://github.com/chrisjbryant/errant
*12 訂正者 3 人のそれぞれの訂正数の内，最大のものを示す．最小のものはそれぞれ 210，428，103 である．

ⓒ 2020 Information Processing Society of Japan

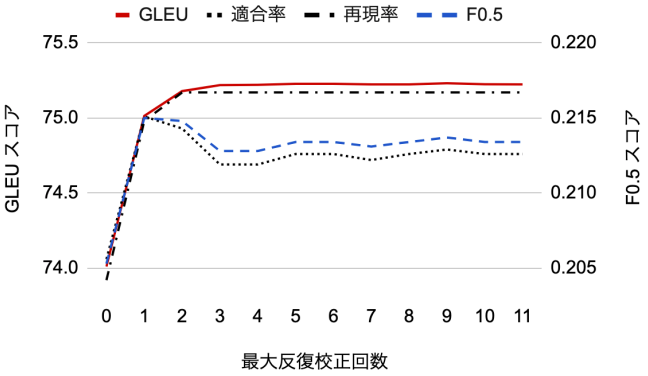

図 3　最大反復校正回数による LevT+KD モデルの性能変化

表 4　各モデルの未完成の文に対する総合的な性能評価

モデル	GLEU	適合率	再現率	$F_{0.5}$
CNN	74.6	0.100	**0.199**	0.111
LevT+KD	**76.9**	**0.125**	0.184	**0.133**

た非自己回帰モデルは自己回帰モデルと同程度かそれを超える訂正性能を出すことができると言える．また，文法誤り訂正に対する非自己回帰モデルにおける知識蒸留の有効性が確認できたため，これ以降の実験では非自己回帰モデルに関しては LevT+KD モデルのみに焦点を当てる．

4.3　反復校正回数

先行研究で主に行われていた機械翻訳タスクと異なり，入力と出力が近い文法誤り訂正タスクでは，必要な反復校正回数が少なくなると思われる．そこで，LevT+KD モデルにおける反復校正回数による性能の変化を評価する．

開発データで選択した最良エポックに対して，評価データを用いて GLEU 及び $F_{0.5}$ スコアで評価した結果を**図 3** に示す．GLEU スコアに着目すると，1 回目以降は最大反復校正回数によって大きく変化することはなく，最大反復校正回数が 3 回程度でほぼ最大となっていることが分かる．また，$F_{0.5}$ スコアを見ると，GLEU スコアと同様に 1 回目以降の最大反復校正回数による変化は小さいが，最大反復校正回数が 1 回より大きい場合は適合率の低下とともに下がり，最大反復校正回数が 1 回のときに最良の結果となっていることが見て取れる．以上より，文法誤り訂正において LevT+KD の最大反復校正回数を大きくする必要性は少なく，1 から 3 回程度で十分であると言える．

4.4　未完成の文に対する性能

ここでは，未完成の文に対する性能を評価するため，テストデータを単語レベルに分かち書きした後，先頭から 1 単語ずつ増やしていく形で未完成の文を作成する．つまり，10 単語の文に対しては 10 個の文ができる形である．そして，元のテストデータの学習者文と訂正文の単語の対応関係を基に，未完成の文に対しても対応する訂正文を作成する．すべてのテストデータに対して上記の処理を行った結果，9,710 文のデータが作成された．このデータを用いて，未完成の文に対する性能を評価する．

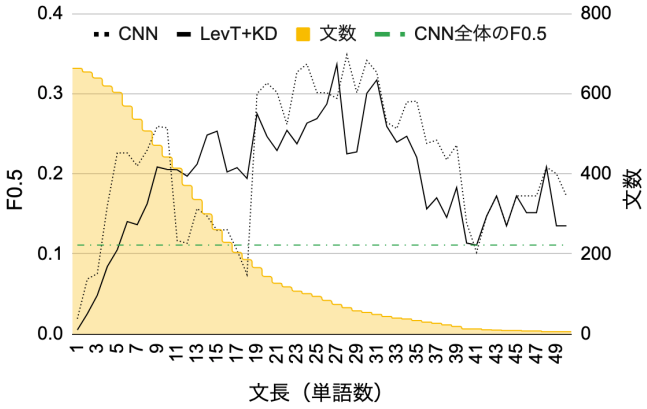

図 4　未完成の文に対する文長ごとの性能

まず，各モデルの総合的な訂正性能を**表 4** に示す．完成した文に対する結果である表 2 と比較すると，全体として同じ傾向，つまり，LevT+KD は GLEU スコアと適合率が高く，CNN は再現率が高いことが分かる．また，どちらのモデルにおいても，GLEU スコアがわずかに上がり，$F_{0.5}$ スコアが下がる結果となっている．総合的に見て，未完成の文に対しても，完成した文のみによって学習した文法誤り訂正モデルがある程度有効であると考えられる．

次に，各モデルの文長ごとの訂正性能を**図 4** に示す．点線と実線がそれぞれ CNN と LevT+KD の $F_{0.5}$ スコア，階段面が文数，そして水平な一点鎖線が CNN の未完成な文全体の $F_{0.5}$ スコアを表している．未完成な文の評価では，完成した文の評価時より多くの文を用いており評価が安定するため，1 単語ごとの結果を載せている．図 2 の完成した文の場合と同様に文長が極端に短いときと極端に長いときには性能が落ちることが見て取れる．また，完成した文の場合に比べて，文長が短いときの訂正性能が特に悪いことが分かる．未完成の文に対しても完成した文と同様に文法誤り訂正結果を提示する場合には，単語数が少ないときには提示しないことが妥当だと考えられる．

4.5　推論速度

推論速度の実験は 4.4 節で説明した 9,710 文の未完成の文を含むテストデータを用いて行う．

モデルごとの推論速度を**図 5** に示す．ここで，モデル名における括弧内の数は最大反復校正回数を表しており，左側は外れ値を考慮せず，右側はひげの長さを四分位範囲の 1.5 倍を超える範囲の値を外れ値としている．平均値は，CNN，最大反復校正回数が 9 回，1 回の LevT+KD でそれぞれ，0.49，0.24，0.19 秒である．CNN に比べて LevT+KD は分散が大きく抑えられているとともに，平均が大幅に下がっていることが分かる．また，最大反復校正

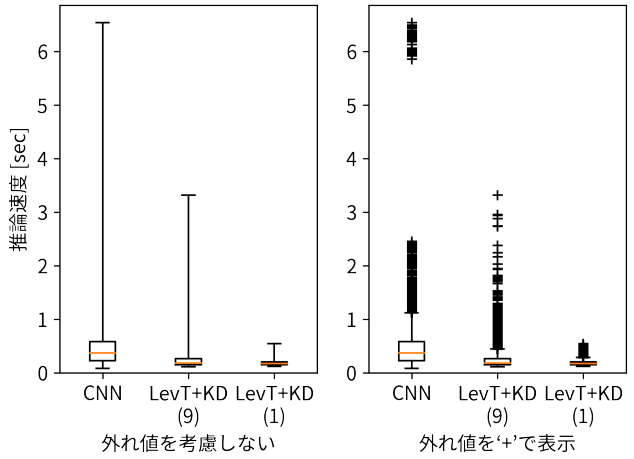

図 5　各モデルの推論速度

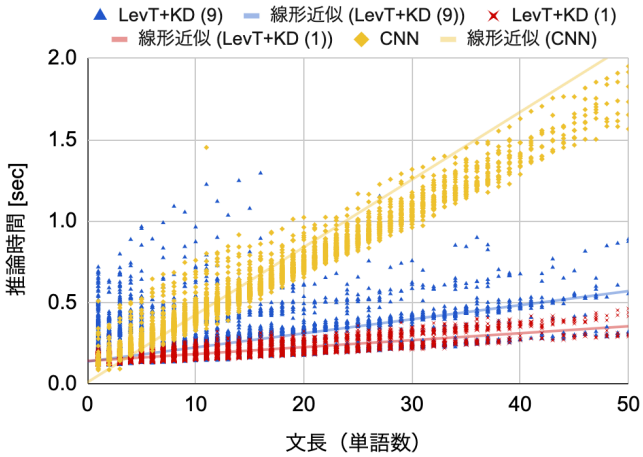

図 6　各モデルの文長ごとの推論速度

回数を減らすことで，更に分散と平均を抑えることができている．外れ値を除いて考えると CNN モデルも 1 秒程度に収まるが，各モデルの推論速度の対応関係は変わらず，LevT+KD モデルのほうが高速であることが分かる．ここで，外れ値となっている文の多くは元の文長が 100 の文から作成された長文が占めており，単純に文長が長いことが推論速度の増加の主な原因だと考えられる．

次に，各モデルの文長ごとの推論速度を**図 6** に示す．直線はそれぞれ線形近似を表しており，モデル名における括弧内の数は上記と同様に最大反復校正回数を表している．線形近似に着目すると，文長が極端に短い（1 から 4 単語）の場合は CNN のほうが各 LevT+KD モデルよりも高速であることが分かる．これは，Levenshtein Transformer モデルが，削除，プレースホルダの挿入及びトークンへの置換の 3 種類の操作を 1 回の反復校正で行っており，CNN モデルに比べてオーバヘッドが大きいためだと考えられる．しかし，文長が 5 単語以上になると，各 LevT+KD モデルのほうが CNN よりも高速になることが見て取れる．

ここで，訂正精度と訂正速度の双方の面から未完成の文に対する文長における影響を分析する．図 4 を見ると，

表 5　各モデルの出力例

例 1	学習者文	これはほんとに大切だか?
	CNN	これはほんとに大切**なの**?
	LevT+KD	これはほんとに大切**です**か?
例 2	学習者文	このせいで、日本語の授業が行けなくなっちゃって、塾に電話しました。
	CNN	このせいで、日本語の授業が行けなくなっちゃって、塾に電話しました。
	LevT+KD	このせいで、日本語の授業**に**行けなくなっちゃって、塾に電話**を**しました。
例 3	学習者文	かわいいくて、安い、素敵な生地です。
	CNN	**かわいいて**、安い、素敵な生地です。
	LevT+KD	**かわいて**、安い、素敵な生地です。

CNN の未完成な文全体の $F_{0.5}$ スコアを最低限の基準としたとき，LevT+KD においては 5 単語目までは下回っていることが分かり，文法誤り訂正結果の提示を行うべきではないと考えられる．そして，図 6 から，6 単語目以降においては LevT+KD が一貫して CNN よりも高速であることが分かる．つまり，LevT+KD モデルを用いて，6 単語以上が入力されたときに文法誤り訂正を行うことで，ある程度の訂正精度を保ちつつ，高速に訂正結果を提示することができる．

4.6　ケーススタディ

システム出力の例を**表 5** に示す．ここで，下線が文法的に誤っている部分，太字がモデルによって変更された部分を表す．例 1 は，だ体に対して終助詞「か」をつけてしまっている誤り文が入力されている．モデルの出力を見ると，CNN，LevT+KD どちらのモデルにおいても文法的に正しく訂正できていることが分かる．ただし，モデルによって訂正の仕方が異なっている．例 2 は，助詞の選択を誤った文が入力されている．CNN のモデルでは訂正できていないのに対し，LevT+KD ではうまく訂正することができている．例 3 は，「かわいい」の連用形である「かわいく」を誤っている文が入力されている．どちらのモデルにおいても修正は行っているが，正しく訂正ができていないことが分かる．これの理由として，学習データ中に含まれる類似の誤り例が少ないことが考えられる．学習データ中に「だか?」という誤りは 172 件存在するのに対し，「かわいいくて」という誤りは 2 件しか存在しない．機械翻訳を用いたエンドツーエンドの文法誤り訂正手法では，学習データに含まれる同種の誤りの数が制約になると考えられる．

LevT+KD モデルにおいて反復校正が行われている例を**表 6** に示す．括弧内の数字はモデルの反復校正回数を表し，太字が挿入された単語，取り消し線が削除された単語を表す．最初の挿入フェーズでは，トークンの欠落や繰り返しの誤りが起きてしまっている．次の削除フェーズで繰

表 6　反復校正が行われる例

入力	き	の	よる	は	たく	やき	パーチイ	ー	い	ます	。
挿入[*13](0)	**きのう**		**よる**	**は**	**たくさん**		**パーティー**	**パーティー**		**し**	**ます** 。
削除 (1)	きのう		よる	は	たくさん		パーティー	~~パーティー~~		し	ます 。
挿入 (1)	きのう		よる	は	たくさん	**やき**	パーティー		**を**	し	ます 。
削除 (2)	きのう		よる	は	たくさん	やき	パーティー		を	し	ます 。
挿入 (2)	きのう	**の**	よる	は	たくさん	やき	パーティー		を	し	ます 。
出力	きのう	の	よる	は	たくさん	やき	パーティー	を	し	ます	。
参照例	きのう	の	よる	は	たこやき	パーティー	に	い	まし	た	。

り返し出力されていたトークン「パーティー」が削除され，更にその次の挿入フェーズでは前後の欠落していたトークン「やき」,「を」が挿入されており，マルチモダリティ問題から回復していることが見て取れる．しかし，「たこやき」と訂正すべきである「たくやき」を「たくさんやき」と誤って修正している箇所や,「きのう」と対応して過去形に訂正すべきである「ます」を訂正できていない箇所も見受けられ，母語話者に比べると訂正能力が劣っている部分があることが分かる．

5. おわりに

本研究では，応答時間が短くユーザビリティの高いライティング支援システムの構築に向けて，非自己回帰モデルの日本語文法誤り訂正システムへの適用可能性を Lang-8 コーパスを用いて調査した．実験から，非自己回帰モデルで，自己回帰型の単一の多層畳み込みニューラルモデルと同等以上の訂正性能を得られることを示した．また，最悪推論時間を 3.2〜6.0 秒程度，平均推論速度を 0.25〜0.3 秒程度短縮し，高速で速度変化の少ない文法誤り訂正が可能であることを示した．更に，入力途中の未完成な文に対しても完成した文と同様に，完成した文で学習したニューラルモデルによる文法誤り訂正が可能なことを示した．ただし，入力単語数が少ないときは完成した文に比べて訂正性能が大幅に落ちるため，訂正結果を提示すべきでないことが分かった．そのとき，非自己回帰モデルを用いる場合，自己回帰モデルの平均の訂正精度を最低限の条件とすると，入力単語が 6 単語以上の場合に訂正結果を提示すれば良いことが分かった．

ライティング支援システムへ応用したときのシステムの外的評価や，大規模事前学習済みモデルを使用した場合の文法誤り訂正の性能への影響の調査，入力文と出力文の大部分が等しいことに着目した非自己回帰モデルの文法誤り訂正への最適化などをフューチャーワークとする．

謝辞　Lang-8 のデータを提供していただいた Lang-8 社長喜洋洋氏に感謝する．

[*13] 挿入は，プレースホルダの挿入と実際のトークンへの置換の両方が行われた結果を示す．また，空の文字列から開始されるため，最初の反復では削除は行われない．

参考文献

[1] Junczys-Dowmunt, M., Grundkiewicz, R., Guha, S. and Heafield, K.: Approaching Neural Grammatical Error Correction as a Low-Resource Machine Translation Task, *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, Association for Computational Linguistics, pp. 595–606 (2018).

[2] Kiyono, S., Suzuki, J., Mita, M., Mizumoto, T. and Inui, K.: An Empirical Study of Incorporating Pseudo Data into Grammatical Error Correction, *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, Association for Computational Linguistics, pp. 1236–1242 (2019).

[3] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. u. and Polosukhin, I.: Attention Is All You Need, *Advances in Neural Information Processing Systems 30* (Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S. and Garnett, R., eds.), Curran Associates, Inc., pp. 5998–6008 (2017).

[4] Gu, J., Bradbury, J., Xiong, C., Li, V. O. and Socher, R.: Non-Autoregressive Neural Machine Translation, *International Conference on Learning Representations* (2018).

[5] Lee, J., Mansimov, E. and Cho, K.: Deterministic Non-Autoregressive Neural Sequence Modeling by Iterative Refinement, *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, Association for Computational Linguistics, pp. 1173–1182 (2018).

[6] Zhou, C., Gu, J. and Neubig, G.: Understanding Knowledge Distillation in Non-autoregressive Machine Translation, *International Conference on Learning Representations* (2020).

[7] Gu, J., Wang, C. and Zhao, J.: Levenshtein Transformer, *Advances in Neural Information Processing Systems 32* (Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E. and Garnett, R., eds.), Curran Associates, Inc., pp. 11181–11191 (2019).

[8] Awasthi, A., Sarawagi, S., Goyal, R., Ghosh, S. and Piratla, V.: Parallel Iterative Edit Models for Local Sequence Transduction, *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, Association for Computational Linguistics, pp. 4260–4270 (2019).

[9] Omelianchuk, K., Atrasevych, V., Chernodub, A. and Skurzhanskyi, O.: GECToR – Grammatical Error Correction: Tag, Not Rewrite, *Proceedings of the Fifteenth Workshop on Innovative Use of NLP for Building Educational Applications*, Association for Computational Linguistics, pp. 163–170 (2020).

[10] Kalchbrenner, N. and Blunsom, P.: Recurrent Continuous Translation Models, *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, Association for Computational Linguistics, pp. 1700–1709 (2013).

[11] Mikolov, T., Karafiát, M., Burget, L., Cernocký, J. and Khudanpur, S.: Recurrent neural network based language model, *INTERSPEECH 2010, 11th Annual Conference of the International Speech Communication Association* (Kobayashi, T., Hirose, K. and Nakamura, S., eds.), International Symposium on Computer Architecture, pp. 1045–1048 (2010).

[12] Ran, Q., Lin, Y., Li, P. and Zhou, J.: Learning to Recover from Multi-Modality Errors for Non-Autoregressive Neural Machine Translation, *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, Association for Computational Linguistics, pp. 3059–3069 (2020).

[13] Kim, Y. and Rush, A. M.: Sequence-Level Knowledge Distillation, *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, Association for Computational Linguistics, pp. 1317–1327 (2016).

[14] Ren, Y., Liu, J., Tan, X., Zhao, Z., Zhao, S. and Liu, T.-Y.: A Study of Non-autoregressive Model for Sequence Generation, *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, Association for Computational Linguistics, pp. 149–159 (2020).

[15] Bryant, C., Felice, M., Andersen, Ø. E. and Briscoe, T.: The BEA-2019 Shared Task on Grammatical Error Correction, *Proceedings of the Fourteenth Workshop on Innovative Use of NLP for Building Educational Applications*, Association for Computational Linguistics, pp. 52–75 (2019).

[16] Kaneko, M., Mita, M., Kiyono, S., Suzuki, J. and Inui, K.: Encoder-Decoder Models Can Benefit from Pre-trained Masked Language Models in Grammatical Error Correction, *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, Association for Computational Linguistics, pp. 4248–4254 (2020).

[17] Grundkiewicz, R., Junczys-Dowmunt, M. and Heafield, K.: Neural Grammatical Error Correction Systems with Unsupervised Pre-training on Synthetic Data, *Proceedings of the Fourteenth Workshop on Innovative Use of NLP for Building Educational Applications*, Association for Computational Linguistics, pp. 252–263 (2019).

[18] Choe, Y. J., Ham, J., Park, K. and Yoon, Y.: A Neural Grammatical Error Correction System Built On Better Pre-training and Sequential Transfer Learning, *Proceedings of the Fourteenth Workshop on Innovative Use of NLP for Building Educational Applications*, Association for Computational Linguistics, pp. 213–227 (2019).

[19] Li, R., Wang, C., Zha, Y., Yu, Y., Guo, S., Wang, Q., Liu, Y. and Lin, H.: The LAIX Systems in the BEA-2019 GEC Shared Task, *Proceedings of the Fourteenth Workshop on Innovative Use of NLP for Building Educational Applications*, Association for Computational Linguistics, pp. 159–167 (2019).

[20] Chollampatt, S. and Ng, H. T.: A Multilayer Convolutional Encoder-Decoder Neural Network for Grammatical Error Correction, *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence* (McIlraith, S. A. and Weinberger, K. Q., eds.), Association for the Advancement of Artificial Intelligence, pp. 5755–5762 (2018).

[21] Shneiderman, B.: Human Factors Experiments in Designing Interactive Systems, *Computer*, Vol. 12, No. 12, pp. 9–19 (1979).

[22] Mizumoto, T., Komachi, M., Nagata, M. and Matsumoto, Y.: Mining Revision Log of Language Learning SNS for Automated Japanese Error Correction of Second Language Learners, *Proceedings of 5th International Joint Conference on Natural Language Processing*, Asian Federation of Natural Language Processing, pp. 147–155 (2011).

[23] Koyama, A., Kiyuna, T., Kobayashi, K., Arai, M. and Komachi, M.: Construction of an Evaluation Corpus for Grammatical Error Correction for Learners of Japanese as a Second Language, *Proceedings of The 12th Language Resources and Evaluation Conference*, European Language Resources Association, pp. 204–211 (2020).

[24] Napoles, C., Sakaguchi, K., Post, M. and Tetreault, J.: GLEU Without Tuning, *eprint arXiv:1605.02592 [cs.CL]* (2016).

[25] Sennrich, R., Haddow, B. and Birch, A.: Neural Machine Translation of Rare Words with Subword Units, *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Association for Computational Linguistics, pp. 1715–1725 (2016).

ⓒ 2020 Information Processing Society of Japan