学修番号 19860636

修士論文

高速な文法誤り訂正機能を持つ
日本語ライティング支援システムの構築

本間　広樹

2021 年 2 月 19 日

東京都立大学大学院
システムデザイン研究科 情報科学域

本間　広樹

審査委員：

小町 守 准教授　（主指導教員）

山口 亨 教授　　（副指導教員）

高間 康史 教授　（副指導教員）

高速な文法誤り訂正機能を持つ日本語ライティング支援システムの構築*

本間　広樹

修論要旨

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．このうち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いる．また，清野らは強力なニューラル機械翻訳モデルである Transformer を用いて文法誤り訂正を行い，逆翻訳により作成した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．ニューラルモデルでは系列全体を見ることが可能なため，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，文長が長いほど推論に時間がかかる．ライティング支援システムに応用することを考えても，推論速度が遅ければ遅いほどその使い道が限られるか，ユーザビリティの低下を招くことになり，問題である．

今日，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，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

*東京都立大学大学院 システムデザイン研究科 情報科学域 修士論文, 学修番号 19860636, 2021 年 2 月 19 日.

力することに起因するマルチモダリティ問題があり，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，反復校正や知識蒸留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．Gu らによって提案された Levenshtein Transformer 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ほか，Omelianchuk 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．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った．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った．

更に，本研究では文法誤り訂正機能付きのライティング支援システムを構築し，その有効性を確認した．具体的には，学習した文法誤り訂正モデルをバックエンドシステムに埋め込み，ライティング支援環境のためのプロトコル及びフレームワークである TEASPN を用いて，Visual Studio Code などの既存のエディタで利用可能な文法誤り訂正機能付きの日本語ライティング支援システムを構築した．

本稿の構成は次のとおりである．第 1 章では本研究の概要について述べる．第 2 章では文法誤り訂正の関連研究を説明する．第 3 章では提案する文法誤り訂正モデル及びライティング支援システムについて述べる．第 4 章では第 3 章で説明した文法誤り訂正モデルの評価実験について述べる．第 5 章では第 3 章で説明したライティング支援システムの評価実験について述べる．最後に，第 6 章で本研究のまとめを行う．

Japanese Writing Support System with Fast Grammatical Error Correction*

Hiroki Homma

Abstract

Grammatical error correction (GEC) is a writing support method for language learners. In recent years, neural GEC has been actively researched owing to its ability to produce fluent text. For example, Kiyono et al. used Transformer, a powerful neural machine translation model, to achieve state-of-the-art correction accuracy. Because the neural model can see the entire sequence, it can correct errors with long-range dependencies; these errors cannot be corrected by a statistical method that uses n-grams.

However, considering the application of GEC in a writing support system, we must consider how to handle incomplete sentences. It is easy to present the GEC result when the user finishes writing a sentence. However, in case of an incomplete sentence, the user will not know how to fix the sentence while writing it. If the system can perform GEC correctly for incomplete sentences, the results can be presented to the user. In most previous studies, complete sentences have been evaluated, and the performance of GEC for incomplete sentences has not been researched.

In addition, there is a problem that inference speed is slow in a conventional autoregressive (AR) decoder of a sequence-to-sequence model. Considering the application of GEC in a writing support system, a slower inference speed would restrict its utility or lower the usability of the model. Gu et al. was proposed a non-autoregressive (NAR) decoder that speeds up inference time by outputting

*Master's Thesis, Department of Computer Science, Graduate School of System Design, Tokyo Metropolitan University, Student ID 19860636, February 19, 2021.

all tokens simultaneously. Following the success of NAR models, Levenshtein Transformer, an NAR NMT model that iteratively deletes and inserts inputs, was proposed by Gu et al. Its usefulness was verified in machine translation and document summarization tasks.

Moreover, fast GEC methods with sequence tagging using an NAR model have been proposed. Awasthi et al. was regarded GEC as a local sequence conversion task and achieved high-speed GEC by using an NAR model that iteratively adapted editing tags in parallel. Omelianchuk et al. was performed NAR GEC by repetitive tagging of editing operations on each token of an input sentence and achieved higher correction accuracy and faster correction speed than in previous studies. However, these methods exhibited good performance by narrowing down the target language to English and preparing the editing operations as tags using language knowledge in advance.

In this study, I focus on the NAR model as a method for high-speed GEC. We perform GEC in Japanese using the NAR model that does not need to prepare editing operations in advance. I analyze the proposed method considering its application to writing support systems. In particular, I analyze the relationship between the correction accuracy and the inference speed, focusing on incomplete sentences, and evaluate the impact of hyperparameters on NAR models.

Furthermore, in this study, I constructed a writing support system with a grammatical error correction function and confirmed its effectiveness. Specifically, the trained grammatical error correction model was embedded in the back-end system. TEASPN, a protocol, and framework for the writing support environment, was used to construct a Japanese writing support system with a grammatical error correction function that can be used with existing editors such as Visual Studio Code.

The structure of this paper is as follows. Chapter 1 outlines this research. Chapter 2 describes the related researches on grammatical error correction. Chapter 3 describes the proposed grammatical error correction model and writing support system. Chapter 4 describes the evaluation experiment of the gram-

matical error correction model explained in Chapter 3. Chapter 5 describes the evaluation experiment of the writing support system explained in Chapter 3. Finally, Chapter 6 summarizes this research.

目次

図目次

第 1 章　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．このうち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いる．また，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，逆翻訳により生成した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．ニューラルモデルでは系列全体を見ることが可能なため，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，文長が長いほど推論に時間がかかる．ライティング支援システムに応用することを考えても，推論速度が遅ければ遅いほどその使い道が限られるか，ユーザビリティの低下を招くことになり，問題である．

今日，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．Gu ら [7] によって提案された Levenshtein Transformer 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．しかし，文法誤り訂正における有用性は未検証であるため，本研究で検証を行う．

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 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初めて行い，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った．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った．

更に，本研究では文法誤り訂正機能付きのライティング支援システムを構築し，その有効性を確認した．具体的には，学習した文法誤り訂正モデルをバックエンドシステムに埋め込み，ライティング支援環境のためのプロトコル及びフレームワークである TEASPN [10] を用いて，Visual Studio Code などの既存のエディタで利用可能な文法誤り訂正機能付きの日本語ライティング支援システムを構築した．

本稿の構成は次のとおりである．第 1 章では本研究の概要について述べる．第 2 章では文法誤り訂正の関連研究を説明する．第 3 章では提案する文法誤り訂正モデル及びライティング支援システムについて述べる．第 4 章では第 3 章で説明した文法誤り訂正モデルの評価実験について述べる．第 5 章では第 3 章で説明したライティング支援システムの評価実験について述べる．最後に，第 6 章で本研究のまとめを行う．

第 2 章　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 [11] 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 [12] を利用するものである．1 トークンごとに仮説を計算するため，目的文全体の出力における計算量は文長に比例したものとなる．

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．

$$p(Y|X;\theta) = \prod_{t=1}^{T+1} p(y_t|y_{0:t-1}, x_{1:T'};\theta) \tag{2.11}$$

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．各トークンの仮説を同時に独立に出力するため，目的文全体の出力時の計算量は文長に比例しない．

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ニューラル機械翻訳に導入し，次のようにデ

コーディングを行う．

$$p(Y|X;\theta) = \sum_{f_1,\ldots,f_{T'}\in\mathcal{F}} \left(\prod_{t'=1}^{T'} p(f_{t'}|x_{1:T'};\theta) \cdot \prod_{t=1}^{T} p(y_t|x_1\{f_1\},\ldots,x_{T'}\{f_{T'}\};\theta) \right) \tag{2.21}$$

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，トークンの繰り返しや欠如のある不適切な文が生成されてしまう問題である．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 [13] 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 [14] が有効であることが報告されている [6][15]．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 [15].

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 3.4.3 項にて述べる．

2.3 文法誤り訂正

文法誤り訂正は, 句読点や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データを対訳コーパスとして利用し，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*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 [16] に取り組んだ研究のほぼすべて [9][2][17][18][19][20] が，機械翻訳におけるベースラインモデルである Transformer をベースとしたモデルを利用している．この内，Li ら [20] 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 [21] 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文法誤り訂正手法である Chollampatt らの手法 [21] 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．

*http://nlpprogress.com/（閲覧日：2021/01/18）

第 3 章　日本語学習者向けのライティング支援システム

本研究の目的の 1 つは，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．本章ではこのライティング支援システムについて説明する．図 3.1 にシステムの概要図を示す．

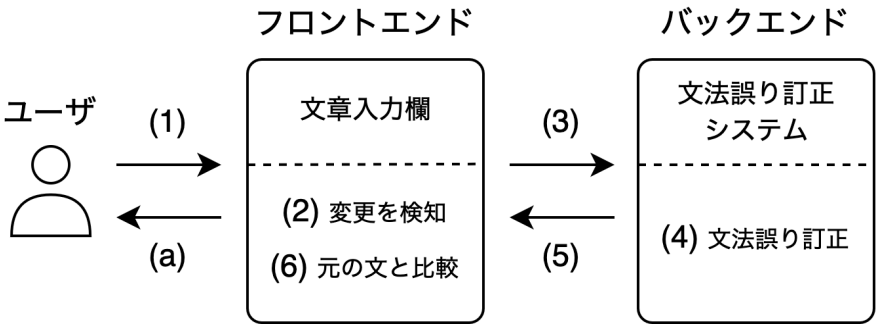


図 3.1: ライティング支援システムの概略図

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）*
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，4.2.3 節で検証する．

2 つ目は，入力方法の問題である．入力途中の文に対する文法誤り訂正の訂正精度を評価する際に，どの単位で入力が与えられるかで結果が変わることが考えられる．特に，日本語入力は英語の場合と異なり，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†．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．

*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．
†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．

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 [22] 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 4.2.4 節にて行う．

3.4 システムの構築

本研究では実際にシステムを構築し，その有用性をユーザ実験により検証する．フロントエンドの実装の詳細を 3.4.2 項で，利用する文法誤り訂正モデルの詳細を 3.4.3 項で説明する．

3.4.1 バックエンドの実装

バックエンドシステムでは，フロントエンドから受信したユーザの入力文に対して文法誤り訂正を行い，その訂正結果をフロントエンドに送信する．ここで，5 章で説明する実験の結果から，文長が短い場合は提示するべきでないことが分かったため，極端に短い文に対しては文法誤り訂正を行わない．また，フロントエンドでユーザにわかり易い提示を行うため，訂正文とともに，トークン化した訂正文及びトークン化した入力文を併せて送信する．上述の処理を含めたバックエンドの処理の流れは以下の通りである．

1. 入力文を受け取る．
2. 入力文のトークン化を行う．
3. トークン化後の入力文のトークン数を最小トークン数[‡]と比較する．
   （a）最小トークン数より少ない場合，入力文をそのまま訂正文とする．
   （b）最小トークン数以上の場合，入力文を文法誤り訂正モデルに入力し，その結果を訂正文とする．
4. 訂正文のトークン化を行う．

[‡]5 章の実験結果より，最小トークン数は 6 とした．

5. 訂正文，トークン化後の訂正文及びトークン化後の入力文を返す．

トークン化の処理をフロントエンドで行うことも考えられるが，トークン化の処理をバックエンドにまとめることで，フロントエンドではトークン化を行うための辞書やモデルを保持する必要がなくなるという利点があるため，バックエンド側で行うこととした．

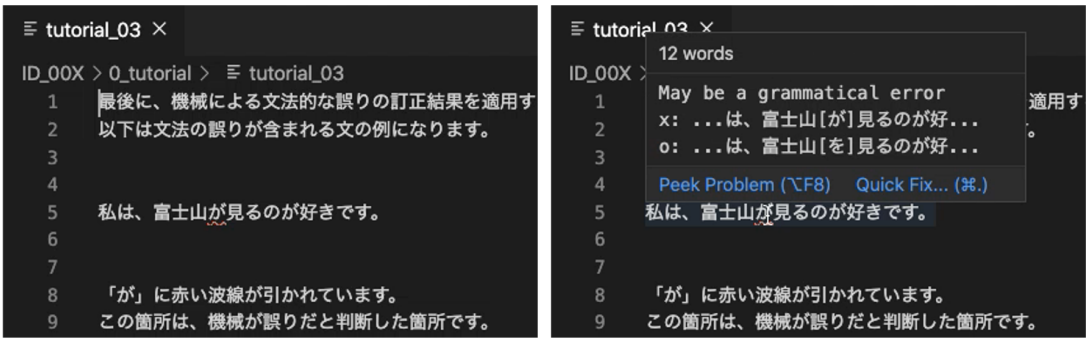
3.4.2 フロントエンドの実装

フロントエンドの実装には，ライティング支援環境を実現するプロトコル及びフレームワークである TEASPN [10] を用いた．この TEASPN は言語処理モジュールとエディタとの橋渡しをする仕組みを標準化したものであり，エディタ上でユーザに対する文法誤り訂正結果を提示する等の処理を比較的容易に行うことが可能である．エディタには Visual Studio Code[§]を用いた．

フロントエンドのシステムでは，ユーザからの入力を受け付け，バックエンドシステムに送信し，バックエンドから受信した訂正文をユーザに提示する．エディタへの入力検知は TEASPN で行う．ただし，変更された行の特定を行うため，エディタ上のテキストを行ごとに保持しておく．保持されたテキストと比較を行うことで，テキストの途中への入力時に，それ以降のすべての行のテキストを再度バックエンドシステムに送信することを防ぐことができる．また，ユーザに訂正文を提示するときに，文全体を提示すると，誤り箇所が分かりづらい問題や，複数箇所の修正を持つ訂正の場合に，個別での訂正が困難になるという問題がある．そこで，入力文と訂正文に対して動的計画法を用いて最長共通部分列を求め，不一致箇所を別々の誤りとして認識する．そのとき，バックエンドから受信したトークン化された文対を用い，トークン単位での一致を見ることで，効率的に，正しい分割箇所での訂正を求めることができる．また，不一致箇所が連続する場合はまとめて一つの訂正として提示する．

実際の訂正結果の提示例を図 3.2 に示す．図 3.2a は誤り箇所を提示している図で，モデルが誤りと認識した箇所を赤い下線で示している．ここではモデルが「富

[§]https://code.visualstudio.com/



(a) 誤り箇所の提示　　(b) 訂正案の提示

図 3.2: 実際の文法誤り訂正の結果の提示例

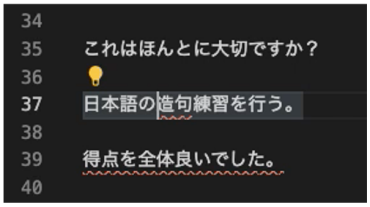


図 3.3: 文法誤り訂正の結果の適用方法（電球ボタン）

士山が」の助詞「が」を文法誤りと判断し，提示している．図 3.2b は訂正案を提示している図で，ユーザが誤り箇所にカーソルオーバーを行うと，このように訂正案が表示される．ここではモデルが「が」を「を」に訂正していることを提示している．また，訂正箇所の認識を容易にするため，前後 5 文字を併せて表示するようにしている．

訂正結果の適用方法がいくつか存在する．1 つ目は図 3.2b のポップアップの右下にある「Quick Fix...」をクリックする方法である．2 つ目は訂正案の提示にカーソルを移動してからショートカットキーを入力する方法である．3 つ目は図 3.3 に示すように，訂正案の提示にカーソルを移動したときに表示される電球ボタンをクリックする方法である．

3.4.3　システムに利用する文法誤り訂正モデル

文法誤り訂正システムには，日本語学習者が書いた文章を日本語母語話者が訂正したデータで学習したニューラル機械翻訳モデルを用いる．提案手法となるシステムには推論速度が高速な非自己回帰ニューラル機械翻訳モデルを，その比較用として自己回帰ニューラル機械翻訳モデルをそれぞれ組み込む．本研究では非自己回帰ニューラル機械翻訳モデルに Levenshtein Transformer [7] を採用し，自己回帰ニューラル機械翻訳モデルに多層畳み込みニューラルネットワーク [21] を採用した．これらのモデルの詳細を本項で述べる．これらのモデルの文法誤り訂正に対する有効性の検証実験は 4 章で行う．

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¶．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（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．

¶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．

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．

$$\boldsymbol{h}_0^{(l+1)}, \boldsymbol{h}_1^{(l+1)}, \ldots, \boldsymbol{h}_n^{(l+1)} = \begin{cases} E_{y_0} + P_0, E_{y_1} + P_1, \ldots, E_{y_n} + P_n, & l = 0 \\ \text{TransformerB}_l \left(\boldsymbol{h}_0^{(l)}, \boldsymbol{h}_1^{(l)}, \ldots, \boldsymbol{h}_n^{(l)} \right), & l > 0 \end{cases} \tag{3.41}$$

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 $\text{softmax}\left(\boldsymbol{h}_i \cdot A^\top\right), (i = 1, \ldots n-1)$ 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 $\text{softmax}\left(\text{concat}\left(\boldsymbol{h}_i, \boldsymbol{h}_{i+1}\right) \cdot B^\top\right), (i = 0, \ldots n-1)$ 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 $\text{softmax}\left(\boldsymbol{h}_i \cdot C^\top\right), (\forall y_i = \texttt{<PLH>})$ 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．

多層畳み込みニューラルネットワーク

Chollampatt ら [21] は 2018 年に多層畳み込みエンコーダデコーダニューラルネットワークモデルを用いて文法誤り訂正タスクに取り組んでいる．彼らは英語における文法誤り訂正の一般的なベンチマークテストデータである CoNLL-2014 [23] 及び JFLEG [24] で，当時の最先端の性能を達成している．このモデルはエンコーダとデコーダの隠れ状態を計算するために畳み込みニューラルネットワークを用いており，2017 年の Gehring らの畳み込みニューラルネットワークをテキスト系列変換に適用した研究 [25] に基づいている．

第 4 章　非自己回帰モデルを用いた文法誤り訂正の評価実験

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 [26] 由来のものを用いた．開発データ，評価データには 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 [27] を用いた．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期待できる．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 [27] に従った．データの文数を表 4.1 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．ここで，学習データに関しては，1 つの学習者の文に対して複数の訂正文が付与されている場合，その訂正文の数のデータとして数えている．

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した．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

表 4.1: データの文数と訂正者数

用途	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

[24] が最も高いエポックのモデルを選択した．その他各種ハイパーパラメータは先行研究に従った．実装には公開されている PyTorch ベースのコード*を用いた．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した [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した．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した．

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 [21] をベースラインとして使用した．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した．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かった．その他の各種ハイパーパラメータは先行研究に従った．実装には公開されている PyTorch ベースのコード†を用いた．本稿ではこれ以降，本モデルを CNN モデルと呼ぶ．

*https://github.com/pytorch/fairseq/tree/master/examples/nonautoregressive_translation

†https://github.com/nusnlp/mlconvgec2018

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った．まず，NFKC ベースの文字正規化[‡]を行った後, 辞書として現代書き言葉コーパス UniDic[§]（バージョン 2.2.0）を用いた MeCab[¶]（バージョン 0.996）を用いて形態素単位に分かち書きを行う．次に，低頻度語に対処するため，Byte Pair Encoding（BPE）[28] を適用し，サブワード単位に分割する．このとき，学習者文側と訂正文側で語彙を共有し[‖]，語彙サイズは 30,000 語とする．実装には sentencepiece[**]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った．計算機環境としては，GPU を用いず，Intel® Xeon® プロセッサー E5-2660 上で動作させた．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 文ずつ処理を行った．計時は Python 言語の time モジュールを用いて行った．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った．

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 $F_{0.5}$ スコアを用いた．$F_{0.5}$ スコアを求めるための単語の対応付けは ERRANT[††]を使用して自動で行った．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

[‡]NFKC ベースの文字正規化は Unicode 正規化の一種であり，Unicode の互換等価な文字列に分解した後，正準等価な文字列に再度合成する文字の正規化処理である．

[§]https://unidic.ninjal.ac.jp/

[¶]https://taku910.github.io/mecab/

[‖]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．

[**]https://github.com/google/sentencepiece/

[††]https://github.com/chrisjbryant/errant/

表 4.2: 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 4.3: 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

離関数にレーベンシュタイン距離を指定した．また，$F_{0.5}$ スコアを測る際は，単語単位での一致率を見た．

4.2 実験結果

4.2.1 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した．

結果を表 4.2 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 4.3 に示す．ここで，「欠落」は入れるべき単語を入れていな

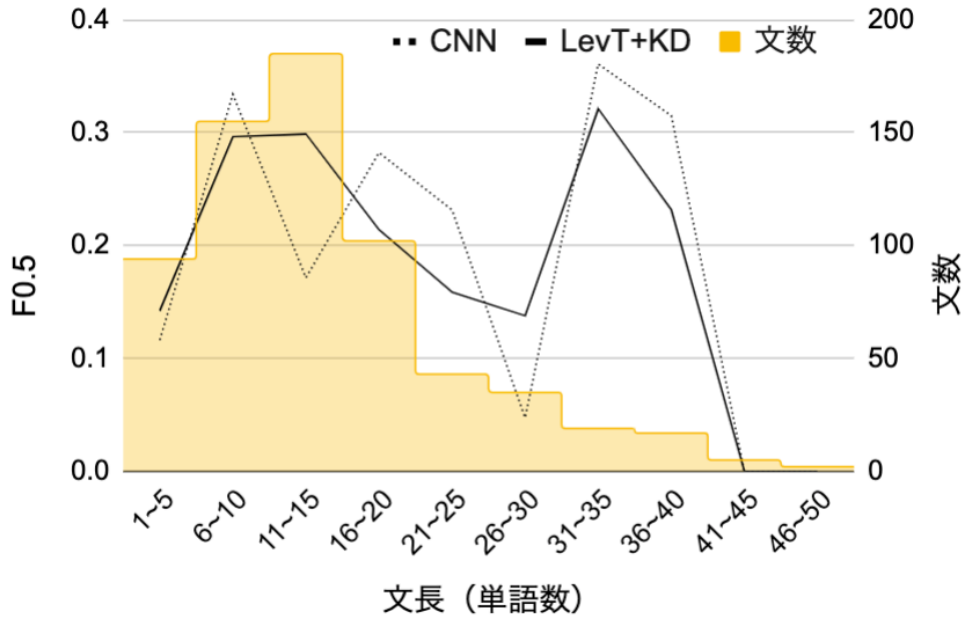


図 4.1: 文長ごとの性能変化

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‡‡．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る．

次に，文長ごとの訂正性能を図 4.1 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

‡‡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．

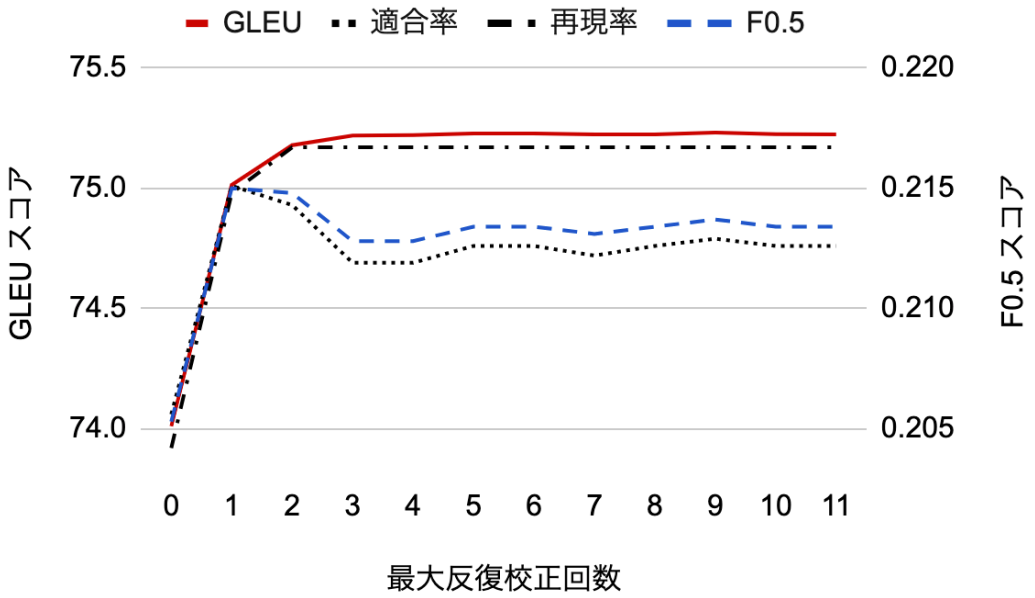


図 4.2: 最大反復校正回数による LevT+KD モデルの性能変化

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．

4.2.2 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した．

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 4.2 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

表 4.4: 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**

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．

4.2.3 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した．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．

まず，各モデルの総合的な訂正性能を表 4.4 に示す．完成した文に対する結果である表 4.2 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る．

次に，各モデルの文長ごとの訂正性能を図 4.3 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

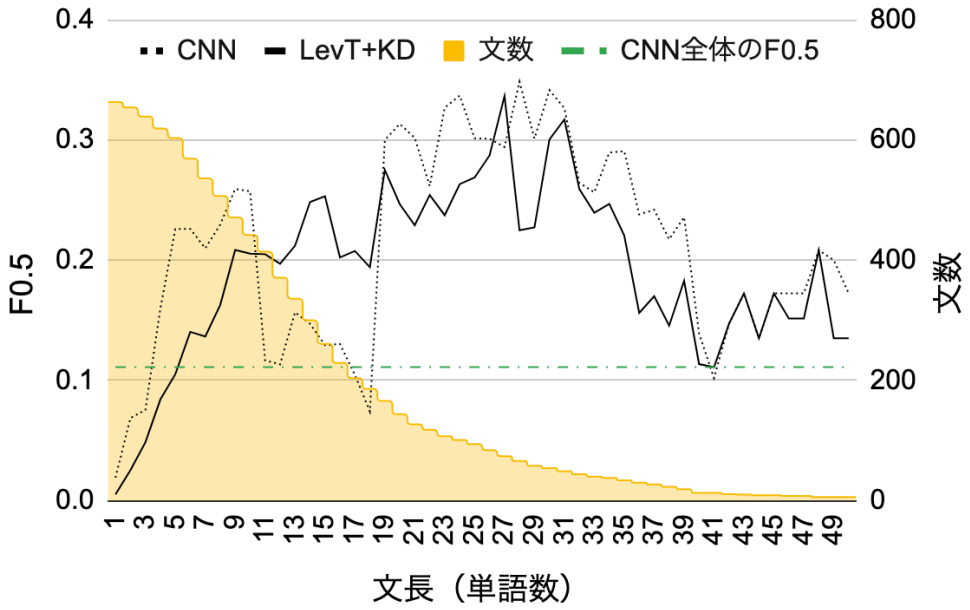


図 4.3: 未完成の文に対する文長ごとの性能

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 4.1 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．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る．

4.2.4 推論速度

推論速度の実験は 4.2.3 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った．

モデルごとの推論速度を図 4.4 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

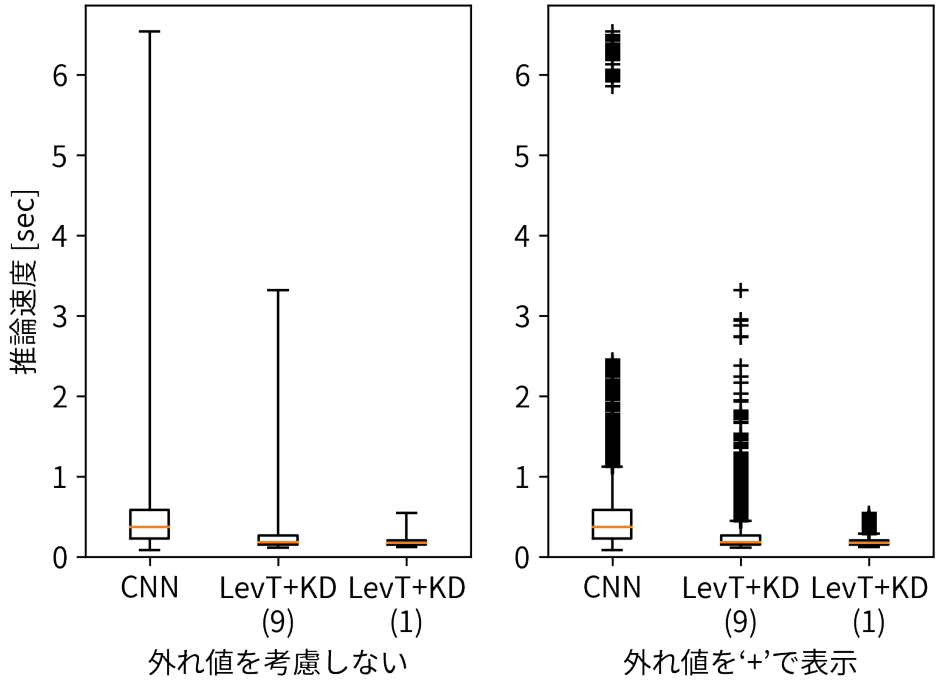


図 4.4: 各モデルの推論速度

下がっていることが分かる．また，最大反復校正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る．

次に，各モデルの文長ごとの推論速度を図 4.5 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る．しかし，文長が 5 単語以上になると，各

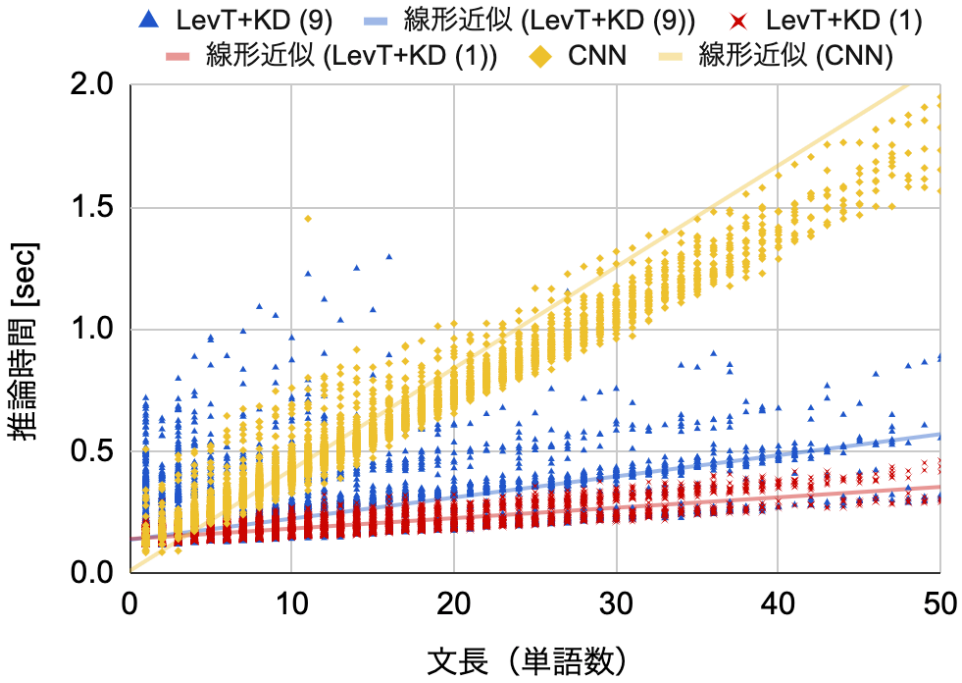


図 4.5: 各モデルの文長ごとの推論速度

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 4.3 を見ると，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る．そして，図 4.5 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．

4.2.5 ケーススタディ

システム出力の例を表 4.5 に示す．ここで，赤い下線が文法的に誤っている部分，**青い太字**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

表 4.5: 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。

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る．

LevT+KD モデルにおいて反復校正が行われている例を表 4.6 に示す．括弧内の数字はモデルの反復校正回数を表し，**青い太字**が挿入された単語，~~赤い取り消し線~~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

表 4.6: 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[§§](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	。	

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る．

[§§]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．

[¶¶]すべてのトークンを保持（削除しない）と分類したため，トークンの削除が行われていない．

第 5 章　ライティング支援システムの評価実験

5.1　実験設定

ライティング支援システムの評価のために，ユーザ実験を行う．まず，システムの有効さの評価として実験協力者に作文タスクに取り組ませる．そしてその結果を人手で評価する．

作文タスクは，全部で 9 種類用意し，実験協力者にはすべてのプロンプトに対する作文に取り組ませた．プロンプトの一覧を表 5.1 に示す．これらのプロンプトは作文タスクの評価を行っている新井らの研究 [29] を参考にして作成した．作文を行うにあたり，おおよその長さを揃えるために各プロンプトに対して 3 文程度で回答するように指示し，条件を揃えるために辞書を使わないで取り組ませた．

また，対面でのユーザ実験の実施が厳しい社会情勢であったため，ビデオ会議ツールの Zoom*を用いて非対面で行い，実験協力者には Zoom を介して筆者の手元にあるコンピュータを遠隔操作させた．日本語入力システムには Google 日本語入力†を利用した．Google 日本語入力の環境設定として，プライバシーのシークレットモードをオンにすることで,「学習機能」及び「入力履歴からのサジェスト機能」を無効化し，条件を揃えた．

5.1.1　実験協力者

日本語を第 2 言語とする研究生・大学生・大学院生の 6 人に，後述する作文タスクに取り組ませる．6 人の実験協力者の内，日本語能力試験‡の認定者が 4 人おり，その全員がレベル N1 である§．

*https://zoom.us/

†https://www.google.co.jp/ime/

‡https://www.jlpt.jp/

§日本語能力試験によると，N1 の認定の目安は「幅広い場面で使われる日本語を理解することができる」とされている．

表 5.1: 作文タスクのプロンプト

1	あなたの住んでいる町を紹介してください。
2	夏と冬とどちらが好きですか？その理由は？
3	日本語の難しいところはどこですか？
4	テレビニュースと新聞は何が違いますか？
5	大学生活で一番楽しかったことは何ですか？
6	自由な時間があったら、何をしたいですか？
7	自分の国の魅力を紹介してください。
8	嘘をつくのはいいことですか？
9	あなたが健康のためにしていることは何ですか？

5.1.2 ライティング支援システム

非自己回帰モデルと自己回帰モデルの比較，さらに文法誤り訂正機能の有効性の確認のため，以下の 3 つのライティング支援システムを用いてユーザ実験を行った．文法誤り訂正モデル以外のシステムは同じものである．

LevT（提案手法） 文法誤り訂正モデルとして 3.4.3 項で説明した Levenshtein Transformer を用いたシステム

CNN 文法誤り訂正モデルとして 3.4.3 項で説明した畳み込みニューラルネットワークを用いたシステム

None 文法誤り訂正を行わないシステム

5.1.3 実験方法

作文の順番による影響を抑えるため，実験協力者ごとに実験の順番を変更した．実験協力者を 3 つのグループに等分し，それぞれ「LevT → CNN → None」,「CNN → None → LevT」,「None → LevT → CNN」の順番でシステムを用いて，作文のプロンプトを「1,2,3 → 4,5,6 → 7,8,9」の順番で見せ，3 文程度の作文に取り組

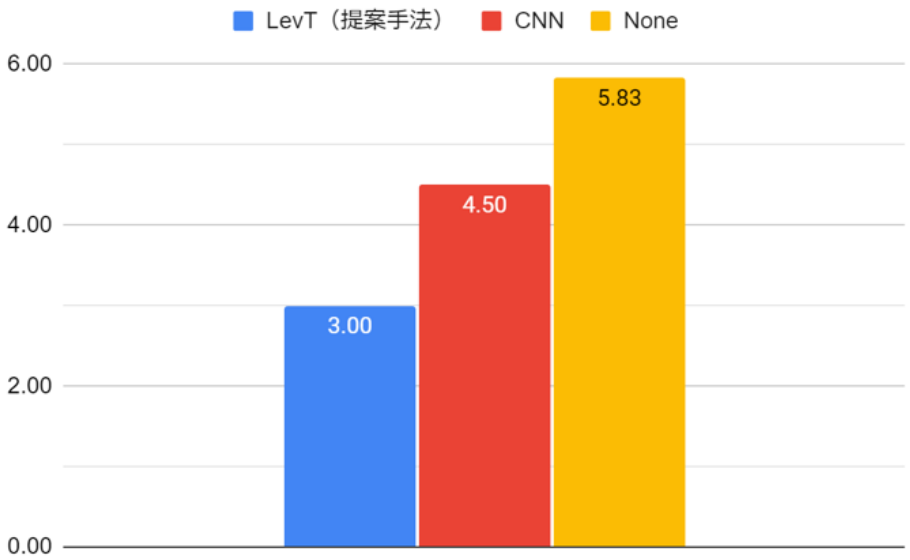


図 5.1: 各システムの 1 人あたりの平均文法誤り数

ませた．

また，実験協力者にシステムへ慣れてもらうため，10 分程度のチュートリアルを用意し，実験前に行わせた．そのチュートリアルは遠隔操作状態での文字の入力方法の確認，提示内容の確認方法の確認及び提示内容の適用方法の確認を含む．

5.1.4 評価尺度

システムの有効さの評価を作文タスクの結果の文法性を元に行い，システムの満足度の評価をアンケート調査により行う．作文タスクの評価は，1 人あたりの文法誤り数の平均で求める．文法誤り数は日本語母語話者により人手で数える．

表 5.2: 各実験協力者のシステム毎の文法誤り数

実験協力者	LevT	CNN	None
A	4	**1**	**1**
B	**7**	8	8
C	**3**	**3**	7
D	**2**	**2**	6
E	**1**	10	5
F	**1**	3	8

5.2 実験結果

5.2.1 実験協力者の作文の評価

図 5.1 に各システムの 1 人あたりの平均文法誤り数を示す．数値が少ないほど，つまり文法誤り数が少ないほど作文結果がよく，システムが有効であると言える．文法誤り訂正機能を持たないシステムである None に比べて，LevT 及び CNN は 1 人あたりの平均誤り数が少なくなっており，文法誤り訂正機能の有効性が確認できる．また，提案手法である LevT は CNN と比べて平均文法誤り数が 1.5 個少なく，None と比べて平均文法誤り数が 5.83 から 3.00 と約半分程度に抑える事ができており，その有効性が明らかとなった．

表 5.2 に各実験協力者のシステム毎の文法誤り数を示す．各実験協力者は 1 つのシステムで 3 つのプロンプトに対してそれぞれ 3 文程度の作文を書くため，表の数値は 3 作文，9 文程度における文法誤り数である．実験協力者毎に，最も文法誤り数が少なかったシステムを太字で表している．実験協力者 A 以外のすべての実験協力者が，提案手法である LevT を用いたときに最も良い作文を書けている．

ここで，システムが誤り箇所を提示したときに，ユーザが取れる選択肢について考える．システムからの提示を受けたとき，ユーザは，「訂正案を確認した上で修正する」，「訂正案を確認した上で放置する」，「訂正案を確認せずに自分で修正する」，「訂正案を確認せずに放置する」の内のいずれかの行動を取ることになる．表 5.2 の実験協力者 B に注目すると，文法誤り訂正機能付きの 2 つのシステムでも文法誤

り数が 7 箇所あるいは 8 箇所と比較的多いが，この実験協力者はすべての訂正案に対して，「訂正案を確認せずに放置する」の選択肢を取っていた．「訂正案を確認した上で放置する」あるいは「訂正案を確認せずに放置する」という選択肢が取られる理由として，採用すべきか，あるいは訂正すべきかどうか分からず放置していた可能性が考えられる．この問題は，例えば新井らの研究 [29] のような用例検索システムと組み合わせた利用や，訂正ごとに文法誤り訂正モデルの確信度を表示することによって低減できるかもしれない．

更に，ユーザの個々人の言語能力による有効性の違いを考察する．実験協力者 F は，日本語能力試験を受けておらず，自己申告ではレベル N3[¶]程度の日本語能力とアンケートで答えており，レベル N1 である実験協力者 B から E に比べて日本語能力が低い．実験協力者 F は，None よりも LevT のほうが文法誤り数が 7 箇所減って 1/8 になっており，個数と割合のどちらにおいても全実験協力者の中で最大の改善結果となっている．これは，None における文法誤り数が多く，また，LevT において多くの誤りに対して正しい訂正を提示できたからである．正しい訂正を提示できた理由として，本実験協力者がその言語能力により比較的容易な日本語を用いた文を書き，文法誤り訂正モデルは容易な日本語における文法誤りの訂正が得意であったことが考えられる．つまり提案システムは，日本語能力が低いユーザに対して，より強力な文法誤り低減の効果が期待できる．

表 5.3 に実験協力者の作文の例を示す．下線のある赤い文字は文法的に誤っている箇所を表す．**太字の青い文字**は文法誤り訂正機能によって提示された訂正案を実験協力者が受け入れた部分であり，その右下の文字は訂正の適用内容を示す．文字列が挿入された場合を「挿入」，文字列が置換された場合を「~~置換前の文字列~~」で表している．改行は削除している．例 1 はシステム None，つまり文法誤り訂正機能無しのシステムで書かれた作文の例である．助詞の間違いや表記の誤り，語彙選択の誤り，漢字の誤変換などのミスが見受けられる．例 2 及び例 3 は，実験協力者がシステムの提案した修正を適用し，より文法的に正しい作文となった例である．例 2 及び例 3 では，不足している助詞や助動詞の挿入がなされており，例 3 ではさら

[¶]N3 の日本語能力試験による認定の目安は「日常的な場面で使われる日本語をある程度理解することができる」である．

表 5.3: 実験協力者の作文の例

	システム	実験協力者の作文
例 1	None	敬語の使い方は苦手です。漢字の他にかたがなによく使われるので、中国人に対しては難しいです。場合によって、同じ表現の言い方は代わりますので、めんどくさいと思います。
例 2	LevT	たくさんの 挿入 人々に会えます. そして, おもしろい研究をすることができます. 大学生だ 挿入 から, たくさんの 挿入 機会があると思います.
例 3	CNN	私はインドネシアから来ました. インドネシアはとてもきれいだ 挿入 と思います. たくさんの 挿入 山や, 海や, 風景が ~~や~~ あります.
例 4	LevT	個人的には嘘をつくのは反対します。いい結果を達成する嘘もあるけど、それは少数です。結果より動機の方は大切ので、嘘をつくことを反対します。
例 5	CNN	授業を取ることが一番楽しかった。高校と違っていろいろなことを勉強できます。専門が異なっても履修できる科目があるから楽しかった。

に助詞の誤選択の修正‖が正しくなされている．日本語学習者の誤りの傾向として助詞や助動詞関連の誤りが多く，それに伴い，文法誤り訂正システムによる修正も助詞や助動詞の挿入や置換，削除が多い結果となった．例 4 及び例 5 は，文法誤り訂正機能付きシステムにおいて，文法誤りの含む作文が書かれた例である．まず，例 4 の 1 文目は，明確な誤りではないが不自然な表現となっており，システムは修正案を提示していなかった．例 5 の 2 文目の文法誤りは，ですます調の不統一によるものである．これは，1 文ごとに評価を行う文法誤り訂正モデルでは訂正不可能な誤りであり，複数文にまたがった文法誤りの訂正を行うためには別の手法を取る

‖副助詞「や」による列挙の後に続けてしまった副助詞「や」を格助詞「が」に修正できている．

必要がある．次に，例 4 の 3 文目の複数箇所の誤りに関しては，システムが誤り箇所を提示していたが，実験協力者が修正案を確認せず，修正しなかった箇所である．文法誤り訂正機能付きのシステムを用いた作文中の誤りは，その多くが複数文にまたがった誤りか，システムの修正案を実験協力者が放置した箇所であった．システムの提示する誤り箇所・修正案をユーザが放置する原因の 1 つとして，修正案が妥当かどうかをユーザが判断できないことが考えられる．この問題は，例えば新井らの研究 [29] のような用例検索システムと組み合わせた利用や，訂正ごとに文法誤り訂正モデルの確信度を表示することによって低減できるかもしれない．

5.2.2 アンケート評価

アンケートでは日本語能力を聞いたほか，文法誤り訂正機能及びシステム全体での 10 段階評価，入力途中での訂正の提示が役に立ったかどうか等を聞いた**．実験協力者は各実験をどのシステムか分からない状態で行い，各実験のシステムを評価している．以下に示す結果は実際のシステムごとにまとめ直したものである．

図 5.2 に実験協力者が各システムの総合評価と訂正機能の評価を 10 段階でつけた結果の平均を示す．総合評価を見ると，None に比べて LevT と CNN がどちらも高く，文法誤り訂正機能の存在が実験協力者にとってシステムの満足度を向上させていることが確認できる．また，LevT と CNN を比べると総合評価と訂正機能評価のどちらにおいても LevT の方が高く，CNN の訂正結果よりも LevT の訂正結果のほうがユーザの満足度の向上に役立つ事が分かる．

また，図 5.3 に各システムの入力途中の文に対する訂正提示の評価を示す．ほぼすべての実験協力者が訂正提示機能を持つどちらのシステムにおいても「とても役に立った」あるいは「まあまあ役に立った」に入れており，入力途中における訂正の提示が有効であることが分かる．さらに LevT と CNN を比較すると，LevT のほうが入力途中における訂正の提示が役に立つと感じている実験協力者が多いことも分かる．

** ここで，実験協力者のうちの 1 人のアンケートが矛盾をはらむ内容を含み，すべての数字を同じ値にしており，信頼のおけない結果であったため，以降の結果からは省いている．

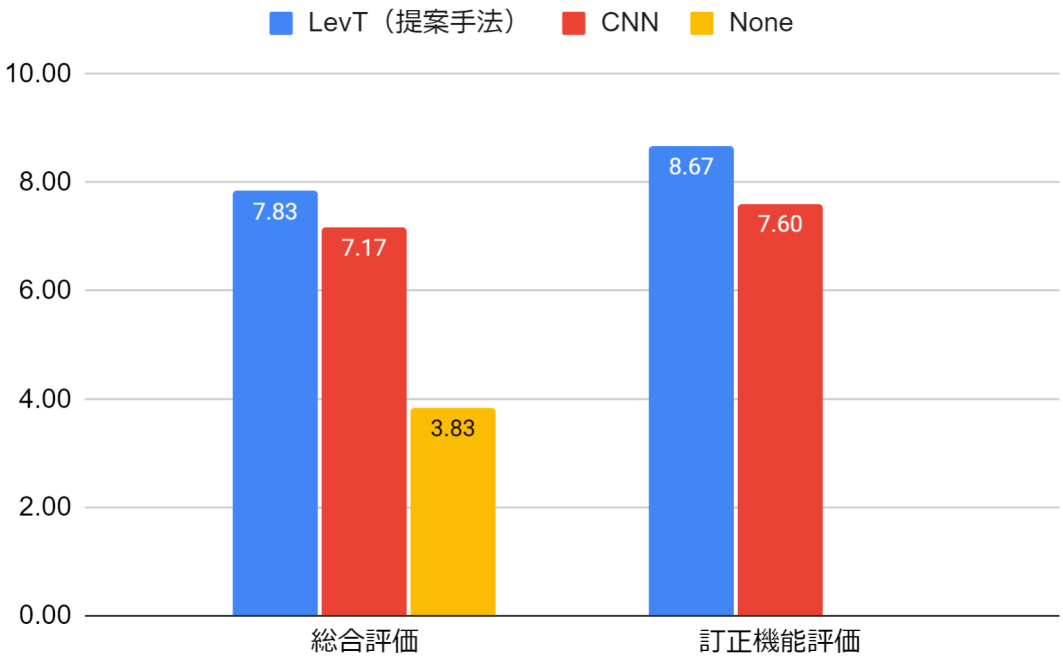


図 5.2: 各システムの総合及び訂正機能の 10 段階評価の平均

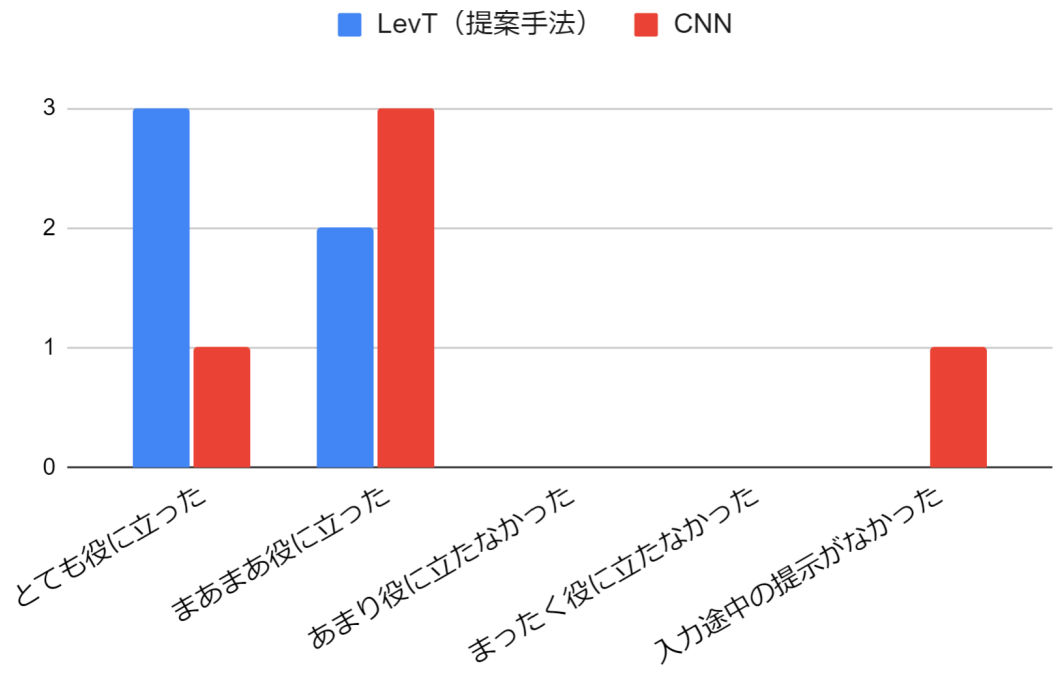


図 5.3: 各システムの入力途中の文に対する訂正提示の評価

第 6 章　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．

また，エディタを通してユーザの書いた文に対する文法誤り訂正案を提示可能な，言語学習者向けのライティング支援システムを構築し，その評価を行った．作文タスクでは，文法誤り訂正機能の有効性が確認でき，提案手法のシステムが文法誤り機能無しのエディタに比べて文法誤り数を半分程度に抑えられることを示した．アンケート調査では，文法誤り訂正機能の満足度が高く，文法誤り訂正機能無しのシステムに比べて総合評価が大きく上回ることを示した．また，アンケート調査では，提案手法の訂正機能がベースラインのモデルよりも高い満足度であることや，入力途中の文に対する訂正提示機能が日本語学習者のユーザにとって役に立つと感じられることも示した．

発表リスト

国際会議

1. Hiroki Homma and Mamoru Komachi. **Non-Autoregressive Grammatical Error Correction Towards a Writing Support System.** In The 6th Workshop on Natural Language Processing Techniques for Educationa Application (NLPTEA 2020). December 4, 2020.

国内会議

1. 本間広樹, 小町守. **非自己回帰モデルを用いた高速な日本語文法誤り訂正.** 情報処理学会第 245 回自然言語処理研究会. September 30, 2020.

謝辞

本論文の執筆に際して，研究室に配属されてからの 3 年間に渡り，ご指導ご鞭撻をいただきました小町守准教授に深く感謝いたします．また，シンポジウムや研究会，学会等での聴講・発表の機会，企業の方との共同研究など，多くの得難い経験をさせていただきまして，感謝しております．そして，副査を引き受けてくださった山口教授，高間教授にも心より感謝いたします．加えて，Lang-8 のデータを提供していただいた Lang-8 社長喜洋洋氏にも感謝いたします．さらに，被験者実験にご協力いただいた方々は，忙しい中時間を割いて参加してくださいました．ここに感謝いたします．

最後に，多くのアドバイスやご協力をいただいた研究室の皆様，共同研究でお世話になりました企業の方々，学会等で様々なアドバイスをいただいた方々など，研究生活において関わったすべての方々に，深く感謝の意を表します．

参考文献

[1] M. Junczys-Dowmunt, R. Grundkiewicz, S. Guha, and K. Heafield, “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), pp.595–606, Association for Computational Linguistics, 2018.

[2] S. Kiyono, J. Suzuki, M. Mita, T. Mizumoto, and K. Inui, “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), pp.1236–1242, Association for Computational Linguistics, 2019.

[3] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A.N. Gomez, L.u. Kaiser, and I. Polosukhin, “Attention is all you need,” Advances in Neural Information Processing Systems 30, eds. by I. Guyon, U.V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, pp.5998–6008, Curran Associates, Inc., 2017.

[4] J. Gu, J. Bradbury, C. Xiong, V.O. Li, and R. Socher, “Non-autoregressive neural machine translation,” International Conference on Learning Representations, 2018.

[5] J. Lee, E. Mansimov, and K. Cho, “Deterministic non-autoregressive neural sequence modeling by iterative refinement,” Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pp.1173–1182, Association for Computational Linguistics, 2018.

[6] C. Zhou, J. Gu, and G. Neubig, “Understanding knowledge distillation in non-autoregressive machine translation,” International Conference on Learning Representations, 2020.

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

[8] A. Awasthi, S. Sarawagi, R. Goyal, S. Ghosh, and V. Piratla, “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), pp.4260–4270, Association for Computational Linguistics, 2019.

[9] K. Omelianchuk, V. Atrasevych, A. Chernodub, and O. Skurzhanskyi, “GEC-

ToR – grammatical error correction: Tag, not rewrite," Proceedings of the Fifteenth Workshop on Innovative Use of NLP for Building Educational Applications, pp.163–170, Association for Computational Linguistics, 2020.

[10] M. Hagiwara, T. Ito, T. Kuribayashi, J. Suzuki, and K. Inui, "TEASPN: Framework and protocol for integrated writing assistance environments," Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing - System Demonstrations, pp.229–234, 2019.

[11] N. Kalchbrenner and P. Blunsom, "Recurrent continuous translation models," Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pp.1700–1709, Association for Computational Linguistics, 2013.

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

[13] Q. Ran, Y. Lin, P. Li, and J. Zhou, "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, pp.3059–3069, Association for Computational Linguistics, 2020.

[14] Y. Kim and A.M. Rush, "Sequence-level knowledge distillation," Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pp.1317–1327, Association for Computational Linguistics, 2016.

[15] Y. Ren, J. Liu, X. Tan, Z. Zhao, S. Zhao, and T.-Y. Liu, "A study of non-autoregressive model for sequence generation," Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pp.149–159, Association for Computational Linguistics, 2020.

[16] C. Bryant, M. Felice, Ø.E. Andersen, and T. Briscoe, "The BEA-2019 shared task on grammatical error correction," Proceedings of the Fourteenth Workshop on Innovative Use of NLP for Building Educational Applications, pp.52–75, Association for Computational Linguistics, 2019.

[17] M. Kaneko, M. Mita, S. Kiyono, J. Suzuki, and K. Inui, "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, pp.4248–4254, Association for Computational Linguistics, 2020.

[18] R. Grundkiewicz, M. Junczys-Dowmunt, and K. Heafield, "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, pp.252–263, Association for Computational Linguistics, 2019.

[19] Y.J. Choe, J. Ham, K. Park, and Y. Yoon, "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, pp.213–227, Association for Computational Linguistics, 2019.

[20] R. Li, C. Wang, Y. Zha, Y. Yu, S. Guo, Q. Wang, Y. Liu, and H. Lin, "The LAIX systems in the BEA-2019 GEC shared task," Proceedings of the Fourteenth Workshop on Innovative Use of NLP for Building Educational Applications, pp.159–167, Association for Computational Linguistics, 2019.

[21] S. Chollampatt and H.T. Ng, "A multilayer convolutional encoder-decoder neural network for grammatical error correction," Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, eds. by S.A. McIlraith and K.Q. Weinberger, pp.5755–5762, Association for the Advancement of Artificial Intelligence, 2018.

[22] B. Shneiderman, "Human factors experiments in designing interactive systems," Computer, vol.12, no.12, pp.9–19, 1979.

[23] H.T. Ng, S.M. Wu, T. Briscoe, C. Hadiwinoto, R.H. Susanto, and C. Bryant, "The CoNLL-2014 shared task on grammatical error correction," Proceedings of the Eighteenth Conference on Computational Natural Language Learning: Shared Task, pp.1–14, Association for Computational Linguistics, 2014.

[24] C. Napoles, K. Sakaguchi, M. Post, and J. Tetreault, "Ground truth for grammatical error correction metrics," Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers), pp.588–593, Association for Computational Linguistics, 2015.

[25] J. Gehring, M. Auli, D. Grangier, D. Yarats, and Y.N. Dauphin, "Convolutional sequence to sequence learning," Proceedings of the 34th International Conference on Machine Learning, pp.1243–1252, 2017.

[26] T. Mizumoto, M. Komachi, M. Nagata, and Y. Matsumoto, "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, pp.147–155, Asian Federation of Natural Language Processing, 2011.

[27] A. Koyama, T. Kiyuna, K. Kobayashi, M. Arai, and M. Komachi, "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, pp.204–211, European Language Resources Association, 2020.
[28] R. Sennrich, B. Haddow, and A. Birch, "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), pp.1715–1725, Association for Computational Linguistics, 2016.
[29] 新井美桜，金子正弘，小町　守，"日本語学習者向けの文法誤り検出機能付き作文用例検索システム，" 人工知能学会論文誌，vol.35，no.5，pp.A–K23_1–9，2020.