Retrieval, Masking, and Generation：マスクされた解説文を活用した文法誤り解説文生成

庵 愛　　佐藤 宏　　田中 智大　　増村 亮
日本電信電話株式会社　NTT コンピュータ&データサイエンス研究所
mana.ihori.kx@hco.ntt.co.jp

概要

本稿では，文法誤り解説文生成タスクのための新たな手法を提案する．本タスクは，文法誤りを含む入力文と修正位置が与えられると，修正位置がなぜ誤っているかを解説する文章を生成するため，言語学習者にとって有益なタスクである．従来，本タスクにはデータプールから検索した解説文を入力文に応じて編集する手法が有効だと考えられたが，過剰な編集が起こるために性能が低いという課題があった．そこで本稿では，過剰な編集を防ぐために，検索した解説文の編集すべきトークンをマスクし，そのトークンを予測する *retrieval, masking, and generation* を提案する．文法誤り解説文生成タスクを用いた評価実験により，自動評価と人手評価で従来手法よりも高い性能を達成することを示す．

1　はじめに

文法誤り解説文生成は，文法誤り訂正タスク [1–4] のように誤りを直接修正するのではなく，入力文の修正位置がなぜ誤っているかを解説するタスクである [5]．具体的には，文法誤りを含む入力文（例えば，“He agrees the opinion.”）と文字単位の範囲として与えられる修正位置（3:13）が与えられた場合に，修正位置に対する解説文（“The verb agree is an intransitive verb and cannot take direct objects. Add the appropriate preposition.”）を生成する．本タスクは自身の文法誤りの原因を知ることができるため，言語学習者にとってより有益なタスクだと考えられる．

従来，本タスクに対して retrieval [5], simple generation [6], retrieve-and-edit [7] の 3 つの手法が提案されている．まず，retrieval はデータプールから入力文に対して適切な解説文を検索する手法である．この手法では，モデル化は比較的容易に行えるが，検索された解説文の編集ができないという課題がある．次に，simple generation は入力文から直接解説文を生成する手法である．そのため，入力文に応じて柔軟な解説文の生成が期待できるが，スクラッチで解説文を生成する必要があるため，retrieval よりもモデル化が難しい．最後に，retrieve-and-edit は，これら手法の利点を組み合わせた手法である．具体的には，retrieval によって検索された解説文を simple generation の入力に用いることで，入力文に応じて検索された解説文を書き換える手法であり，simple generation よりも高い性能を達成することが期待されている．しかし，retrieve-and-edit は検索された解説文のうち，編集する必要のないトークンを過剰に書き換えてしまうため，従来手法の中で最も性能が低いことが報告されている [8].

本稿では，retrieve-and-edit の課題を解決するために，この手法を拡張した *retrieval, masking, and generation* を提案する．提案手法は，検索，マスク，生成の 3 つのモジュールから構成され，retrieve-and-edit にマスクモジュールを導入したものである．マスクモジュールは，検索された解説文のうち，正解の解説文に含まれないトークンをマスクするように，各トークンに対してマスクトークンに置換するか，しないかの二値分類を行う．ここで作成されるマスクされた解説文を生成モジュールの入力に用いることで，マスクトークンを予測するように新たな解説文を生成できるため，検索された解説文の過剰な書き換えを防ぐことを期待している．ここで，提案手法は各モジュールの出力をカスケード接続するため，直前のモジュールの性能が後続のモジュールの性能に影響する．この影響を軽減するため，推論時に検索モジュールの上位 1 個のみではなく，k 個の結果を用いる multi-decoding を提案する．文法誤り解説文生成タスクを用いた評価実験により，自動評価で提案手法が従来手法よりも高い性能を達成することを示す．また，提案手法を用いてシェアード

タスクである Generation Challenge 2022[1)]に参加したため，その結果についても報告する．

2 提案手法

図 1 に検索，マスク，生成モジュールから構成される提案手法の概要を示す．本稿では，解説文生成タスクの入力文を $\boldsymbol{X} = \{x_1, \cdots, x_M\}$，解説文を $\boldsymbol{Y} = \{y_1, \cdots, y_N\}$ と定義する．このとき，x_m と y_n は入力，解説文に含まれるトークンを，M と N はそれぞれのトークン数を表す．また，修正位置を位置ラベル $\boldsymbol{P} = \{p_1, \cdots, p_M\}$（このとき，$p_m \in \{0, 1\}$）に変換する．例えば，入力文が {He, agrees, the, opinion, .}，修正位置が 3:13 の場合，$\boldsymbol{P} = \{0, 1, 1, 0, 0\}$ となる．

2.1 検索モジュール

本モジュールでは，$\boldsymbol{X}$ と $\boldsymbol{P}$，解説文の集合から選択された $\boldsymbol{Y}$ をそれぞれベクトルに変換し，ベクトル間の最も高いコサイン類似度を示す $\boldsymbol{Y}$ を出力する．コサイン類似度 s は，式 (1) によって導出される．

$$s = \mathtt{retreival}(\boldsymbol{X}, \boldsymbol{P}, \boldsymbol{Y}; \boldsymbol{\Theta}_{\mathtt{ret}}) \tag{1}$$

ここで，retreival() は検索モジュールの関数，$\boldsymbol{\Theta}_{\mathtt{ret}}$ は学習可能なパラメータを表す．

retreival() で行われる処理を以下に示す．まず，式 (2-3) に従って，$\boldsymbol{X}$ と $\boldsymbol{Y}$ を隠れ表現 $\boldsymbol{Q} = \{\boldsymbol{q}_1, \cdots, \boldsymbol{q}_M\}$ と $\boldsymbol{R} = \{\boldsymbol{r}_1, \cdots, \boldsymbol{r}_N\}$ に変換する．

$$\boldsymbol{Q} = \mathtt{TransformerEncoder}(\boldsymbol{X}; \boldsymbol{\Theta}_{\mathtt{ret}}) \tag{2}$$

$$\boldsymbol{R} = \mathtt{TransformerEncoder}(\boldsymbol{Y}; \boldsymbol{\Theta}_{\mathtt{ret}}) \tag{3}$$

ここで，TransformerEncoder() は [9] と同様の構造である Transformer ベースのエンコーダを表す．本稿では，このエンコーダに事前学習済みの BERT [10] を用いる．次に，式 (4) に従って，$\boldsymbol{Q}$ と $\boldsymbol{P}$ を用いて入力文を単一のベクトルに変換する．

$$\boldsymbol{u} = \sum_{m=1}^{M} \boldsymbol{q_m} \cdot p_m \tag{4}$$

また，データプールから選択された解説文は，文頭の [CLS] トークンの隠れ表現（$\boldsymbol{r}_1$）を用いることで，単一のベクトルに変換する．最後に，$\boldsymbol{u}$ と $\boldsymbol{r}_1$ のコサイン類似度を式 (5) から導出する．

$$\mathtt{retreival}(\boldsymbol{X}, \boldsymbol{P}, \boldsymbol{Y}; \boldsymbol{\Theta}_{\mathtt{ret}}) = \frac{\boldsymbol{u} \cdot \boldsymbol{r}_1}{\|\boldsymbol{u}\| \|\boldsymbol{r}_1\|} \tag{5}$$

学習： 本モジュールは，$\mathcal{D}_{\mathtt{ret}} = \{(\boldsymbol{X}^1, \boldsymbol{P}^1, \mathcal{Y}, \mathcal{S}^1), \cdots, (\boldsymbol{X}^C, \boldsymbol{P}^C, \mathcal{Y}, \mathcal{S}^C)\}$ を用いて学習される．ここで，C は入力文と位置ラベルの総数を，$\mathcal{Y} = \{\boldsymbol{Y}^1, \cdots, \boldsymbol{Y}^D\}$ は総数 D の解説文の集合を表す．また，$\mathcal{S}^c = \{s^{c,1}, \cdots, s^{c,D}\}$ は $\boldsymbol{X}^c$ に対する正解の解説文 $\boldsymbol{Y}^c$ と，$\mathcal{Y}$ から選択した $\boldsymbol{Y}^d$ の Levenshtein 類似度 [11] の集合を表す．このモジュールは，$\boldsymbol{X}^c$ と $\boldsymbol{P}^c$，$\boldsymbol{Y}^d$ から作成されるベクトル間の類似度が $s^{c,d}$ に近づくように式 (6) より学習される．

$$\mathcal{L}_{\mathtt{ret}} = \frac{1}{C \cdot D} \sum_{c=1}^{C} \sum_{d=1}^{D} \left(s^{c,d} - \mathtt{retreival}(\boldsymbol{X}^c, \boldsymbol{P}^c, \boldsymbol{Y}^d; \boldsymbol{\Theta}_{\mathtt{ret}})\right)^2 \tag{6}$$

検索： 式 (7) に従って，$\mathcal{Y}$ から最大のコサイン類似度を示す $\tilde{\boldsymbol{Y}}$ を抽出する．

$$\tilde{\boldsymbol{Y}} = \underset{\boldsymbol{Y} \in \mathcal{Y}}{\arg\max}\, \mathtt{retreieval}(\boldsymbol{X}, \boldsymbol{P}, \boldsymbol{Y}; \boldsymbol{\Theta}_{\mathtt{ret}}) \tag{7}$$

2.2 マスクモジュール

本モジュールでは，$\boldsymbol{X}$ と $\boldsymbol{P}$，$\tilde{\boldsymbol{Y}}$ が与えられた場合に，$\tilde{\boldsymbol{Y}}$ の各トークンに対してマスクするか，しないかの二値分類を行う．具体的には，マスクラベル $\boldsymbol{L} = \{l_1, \cdots, l_N\}$（このとき，$l_n \in \{0, 1\}$）を式 (8) に従って出力する．

$$P(\boldsymbol{L}|\boldsymbol{X}, \boldsymbol{P}, \tilde{\boldsymbol{Y}}; \boldsymbol{\Theta}_{\mathtt{mask}}) = \prod_{n=1}^{N} P(l_n|\boldsymbol{X}, \boldsymbol{P}, \tilde{\boldsymbol{Y}}; \boldsymbol{\Theta}_{\mathtt{mask}}) \tag{8}$$

ここで，$\boldsymbol{\Theta}_{\mathtt{mask}}$ は学習可能なパラメータを表す．

本モジュールは検索モジュールと同様のモデルアーキテクチャで用いることが可能である．そのため，式 (2-4) より，$\boldsymbol{X}$ と $\boldsymbol{P}$ を $\boldsymbol{u}$ に，$\tilde{\boldsymbol{Y}}$ を $\boldsymbol{R} = \{\boldsymbol{r}_1, \cdots, \boldsymbol{r}_N\}$ に変換する．次に，$\tilde{\boldsymbol{Y}}$ の各トークンに対する二値分類を式 (9-10) によって算出する．

$$P(l_n|\boldsymbol{X}, \boldsymbol{P}, \tilde{\boldsymbol{Y}}; \boldsymbol{\Theta}_{\mathtt{mask}}) = \mathtt{softmax}(\boldsymbol{v}_n; \boldsymbol{\Theta}_{\mathtt{mask}}) \tag{9}$$

$$\boldsymbol{v}_n = [\boldsymbol{u}^{\mathrm{T}}, \boldsymbol{r}_n^{\mathrm{T}}]^{\mathrm{T}} \tag{10}$$

ここで，softmax() は活性化関数 softmax を用いた線形変換を表す．

学習： 本モジュールは，$\mathcal{D}_{\mathtt{mask}} = \{(\boldsymbol{X}^1, \boldsymbol{P}^1, \mathcal{Y}, \mathcal{L}^1), \ldots, (\boldsymbol{X}^C, \boldsymbol{P}^C, \mathcal{Y}, \mathcal{L}^C)\}$ を用いて学習される．ここで，$\mathcal{L}^c = \{\boldsymbol{L}^{c,1}, \cdots, \boldsymbol{L}^{c,D}\}$ は，$\mathcal{Y}$ から選択した $\boldsymbol{Y}^d$ のトークンのうち，$\boldsymbol{Y}^c$ に含まれないトークンを 1，それ以外を 0 としたラベル集合を表す．本モジュールは，式 (11) に示すクロスエントロピー損失を用いて学習される．

$$\mathcal{L}_{\mathtt{mask}} = -\sum_{c=1}^{C} \sum_{d=1}^{D} \log P(\boldsymbol{L}^{c,d}|\boldsymbol{X}^c, \boldsymbol{P}^c, \boldsymbol{Y}^d; \boldsymbol{\Theta}_{\mathtt{mask}}) \tag{11}$$

1） https://fcg.sharedtask.org/

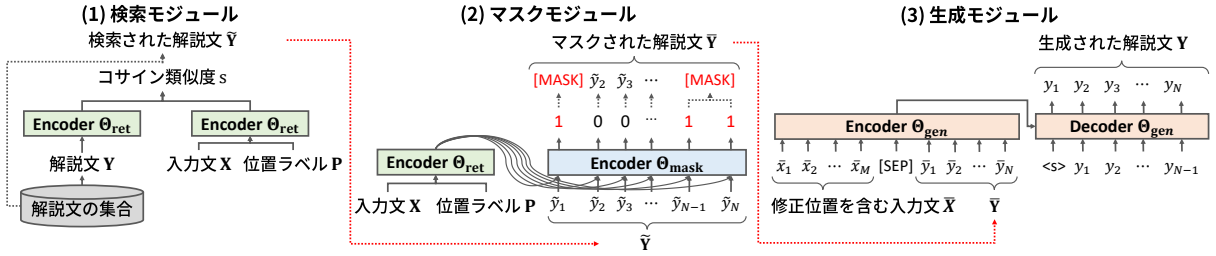

図 1　提案手法の概要

マスク処理：　本モジュールの推論は，式 (12) によって実行される．

$$\hat{\boldsymbol{L}} = \arg\max_{\boldsymbol{L}} P(\boldsymbol{L}|\boldsymbol{X}, \boldsymbol{P}, \tilde{\boldsymbol{Y}}; \boldsymbol{\Theta}_{\mathsf{mask}}) \tag{12}$$

最終的に，式 (13) に従って，$\tilde{\boldsymbol{Y}}$ に含まれるトークンをマスクした $\bar{\boldsymbol{Y}}$ を出力する．

$$\bar{\boldsymbol{Y}} = \mathsf{MASK}(\tilde{\boldsymbol{Y}}, \hat{\boldsymbol{L}}) \tag{13}$$

MASK() は $\hat{\boldsymbol{L}}$ のラベルが 1 のとき，そのラベルに対応する $\tilde{\boldsymbol{Y}}$ のトークンをマスクトークンに置換する処理を表す．このとき，生成モジュールではマスクトークンの数だけトークンを予測する必要があるが，ここで置換したマスクトークンの数と生成すべきトークンの数は必ずしも一致するわけではない．そこで本稿では，1 つのマスクトークンから複数のトークンを予測できる，span masking [12] を用い，連続するマスクトークンは 1 つのマスクトークンに置換する．そのため，$\tilde{\boldsymbol{Y}}$ と $\bar{\boldsymbol{Y}}$ のトークン数は同じであるとは限らない．また，すべてのラベルが 0 の場合，生成モジュールを介さず，直接 $\tilde{\boldsymbol{Y}}$ を出力する．

2.3　生成モジュール

本モジュールでは，$\bar{\boldsymbol{X}} = \{\bar{x}_1, \cdots, \bar{x}_M\}$ と $\bar{\boldsymbol{Y}}$ が与えられた場合に，$\boldsymbol{Y}$ を生成する．ここで，$\bar{\boldsymbol{X}}$ は修正位置の範囲に含まれるトークンを括弧で囲むことにより，あらかじめ修正位置の情報を付与した入力文である．具体的には，$\bar{\boldsymbol{X}}$ と $\bar{\boldsymbol{Y}}$ を特殊トークン [SEP] で連結した，$\boldsymbol{Z} = \{\bar{x}_1, \cdots, \bar{x}_M, [\mathsf{SEP}], \bar{y}_1, \cdots, \bar{y}_N\}$ を入力として，式 (14) に従って $\boldsymbol{Y}$ を生成する．

$$P(\boldsymbol{Y}|\boldsymbol{Z}; \boldsymbol{\Theta}_{\mathsf{gen}}) = \prod_{n=1}^{N} P(y_n|y_1, \cdots, y_{n-1}, \boldsymbol{Z}; \boldsymbol{\Theta}_{\mathsf{gen}}) \tag{14}$$

ここで，$\boldsymbol{\Theta}_{\mathsf{gen}}$ は学習可能なパラメータを表す．

本モジュールは，Transformer ベースの encoder-decoder モデルで構成される．まず，$\boldsymbol{Z}$ を式 (15) によって隠れ表現 $\boldsymbol{H}$ に変換する．

$$\boldsymbol{H} = \mathsf{TransformerEncoder}(\boldsymbol{Z}; \boldsymbol{\Theta}_{\mathsf{gen}}) \tag{15}$$

次に，$y_1, \cdots, y_{n-1}$ と $\boldsymbol{H}$ を用いて，y_n の事後確率を式 (16-17) に基づいて算出する．

$$P(y_n|y_1, \cdots, y_{n-1}, \boldsymbol{Z}; \boldsymbol{\Theta}_{\mathsf{gen}}) = \mathsf{softmax}(w_n; \boldsymbol{\Theta}_{\mathsf{gen}}) \tag{16}$$

$$w_n = \mathsf{TransformerDecoder}(y_1, \cdots, y_{n-1}, \boldsymbol{H}; \boldsymbol{\Theta}_{\mathsf{gen}}) \tag{17}$$

ここで，TransformerDecoder() は [9] と同様の構造である Transformer ベースのデコーダを表す．本稿では，この encoder-decoder モデルに事前学習済みの T5 [12] を用いる．

学習：　本モジュールは，$\mathcal{D}_{\mathsf{gen}} = \{(\boldsymbol{Z}^1, \boldsymbol{Y}^1), \cdots, (\boldsymbol{Z}^{|\mathcal{D}_{\mathsf{gen}}|}, \boldsymbol{Y}^{|\mathcal{D}_{\mathsf{gen}}|})\}$ を用い，式 (18) に示すクロスエントロピー損失によって学習される．

$$\mathcal{L}_{\mathsf{gen}} = -\sum_{(\boldsymbol{Z}, \boldsymbol{Y}) \in \mathcal{D}_{\mathsf{gen}}} \log P(\boldsymbol{Y}|\boldsymbol{Z}; \boldsymbol{\Theta}_{\mathsf{gen}}) \tag{18}$$

デコード：　デコードは式 (19) によって算出される．

$$\hat{\boldsymbol{Y}} = \arg\max_{\boldsymbol{Y}} P(\boldsymbol{Y}|\boldsymbol{Z}; \boldsymbol{\Theta}_{\mathsf{gen}}) \tag{19}$$

2.4　multi-decoding

提案手法では，3 つのモジュールの出力をカスケード接続するため，各モジュールの性能は直前のモジュールの性能に依存する．そこで，最終的に出力される解説文の信頼度を高める方法として，multi-decoding を提案する．multi-decoding は検索モジュールで上位 1 個ではなく，k 個の解説文を検索する．ここで検索された k 個の解説文を用い，1 つの入力文に対して k 個の解説文を生成する．これら k 個の解説文から，1 つの解説文を選択することで最終的な出力を決定する．選択方法として，同じ入力文に対して重複している解説文の数を算出し，最も多い解説文を選択する．しかし，重複している数が同じであったり，すべての解説文が重複していない可能性もある．そのような場合，ある解説文とそれ以外の $k-1$ 個の解説文，および入力文との Levenshtein 類似度を算出し，その平均値が最も高いものを選択することで 1 つの解説文を出力する．

3 評価実験

3.1 実験条件

データ： Generation Challenge 2022 で提供された，入力文，修正位置，解説文がセットになったデータセットを用い，2 章の各データセットを作成する．まず，$\mathcal{D}_{\mathsf{ret}}$ では，$\mathcal{Y}$ は上記データセットの学習セットを用いる．ここで，学習セットのすべての $\boldsymbol{Y}^d$ を用いて $s^{c,d}$ を算出すると，類似度が低いものが多い不均衡なデータセットになる．そこで，類似度の小数点第一位が同じものが 5 個以下になるように $\boldsymbol{Y}^d$ をランダムに削除した．次に，$\mathcal{D}_{\mathsf{mask}}$ では，$\mathcal{Y}$ は $\mathcal{D}_{\mathsf{ret}}$ と同じものを用い，$\boldsymbol{Y}^d$ のうち，$\boldsymbol{Y}^c$ に含まれないトークンを 1，それ以外を 0 として $\boldsymbol{L}^{c,d}$ を作成した．最後に，$\mathcal{D}_{\mathsf{gen}}$ では，$\mathcal{D}_{\mathsf{ret}}$ 作成時に算出した類似度を用い，各 $\boldsymbol{X}^c$ に対して 10 件の $\boldsymbol{Y}^d$ を抽出し，$\mathcal{D}_{\mathsf{mask}}$ の $\boldsymbol{L}^{c,d}$ から $\bar{\boldsymbol{Y}}^d = \mathsf{MASK}(\boldsymbol{Y}^d, \boldsymbol{L}^{c,d})$ を得ることで $\boldsymbol{Z}$ を作成した．表 1 にそれぞれの詳細を示す．

設定： 本稿では，提案手法，検索と生成モジュールを用いた retrieve-and-edit，検索モジュールのみの retrieval，生成モジュールのみの simple generation を実装した．これらの手法は，事前学習済みモデルを fine-tuning することで学習され，BERT には HaggingFace 社 [13] の bert-based-cased を，T5 には t5-base を用いた．retrieve-and-edit の生成モジュールは検索された解説文をマスクしない $\mathcal{D}_{\mathsf{gen}}$，simple generation は提供データセットを用いて fine-tuning した．また，fine-tuning 時は RAdam [14] による最適化を行い，multi-decoding では，$k = 7$ とした．

3.2 結果

表 2 に文法誤り解説文生成の評価結果を示す．表中の値は，BLEU [15] による自動評価と，Generation Challenge 2022 による 1 名のラベラーの人手評価を表す．表より，自動評価，人手評価の両方で multi-decoding を用いた提案手法が最高性能を達成した．提案手法では，multi-decoding によって一つの入力文に対して複数の解説文が生成され，より正解に近い解説文を選択できたために性能が向上したと考えられる．実際に，同じ入力に対して異なるマスクされた解説文を用いることで，異なる解説文が生成されていた．一方，retrieve-and-edit では multi-decoding による性能向上は起こらず，同じ入力文に対して異なる検索された解説文を用いても，全く同じ解説文を生成することがほとんどであった．そのため，retrieve-and-edit は検索された解説文を活用して新たな解説文を生成するのではなく，入力文に対してスクラッチで解説文を生成していると考えられる．また，simple generation よりも性能が高いのは，単純に学習データ量が多かったためだと考えられる．以上より，提案手法では入力のマスクを予測するように解説文を生成でき，検索した解説文を従来手法よりも有効に活用できると考えられる．

表 1 各データセットのデータ数

	学習	開発	テスト
提供データセット	4,868	170	215
$\mathcal{D}_{\mathsf{ret}}$，$\mathcal{D}_{\mathsf{mask}}$	139,687	5,001	-
$\mathcal{D}_{\mathsf{gen}}$	48,309	1,000	-

表 2 文法誤り解説文生成における評価結果

手法	BLEU	人手評価
retrieval	0.423	-
simple generation	0.464	0.479
retrieve-and-edit	0.482	0.502
+ multi-decoding	0.480	-
提案手法	0.482	0.508
+ multi-decoding	**0.494**	**0.517**
提案手法（オラクルマスク）	0.539	-

また，提案手法の推論時にマスクモジュールの予測結果ではなく，オラクルを用いた場合の性能も表に示す．表より，検索された解説文に対してオラクルのマスクを用いると，性能が大幅に向上することが示された．これは，各モジュールを個別に最適化しているために，誤りを含む予測結果の入力が生成の性能を劣化させたためだと考えられる．この性能の劣化は multi-decoding によって低減できるが，より性能を向上させるためには，学習時に予測結果を入力するなど，学習方法を再検討する必要がある．

ここでの実際の出力結果の例は付録に示す．

4 おわりに

本稿では，文法誤り解説文生成のための，*retrieval, masking, and generation* を提案した．提案手法は，マスクされた解説文を入力に用いることで，マスクトークンを予測するように解説文を生成する．また，提案手法は各モジュールの出力をカスケード接続するため，推論時に上位 1 個だけではなく，k 個の検索された解説文を用いる multi-decoding を提案した．文法誤り解説文生成を用いた評価実験の結果，提案手法は従来手法より高い性能を達成したが，学習方法に改善の余地があることが示された．

参考文献

[1] Zheng Yuan and Ted Briscoe. Grammatical error correction using neural machine translation. In **Proc. Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)**, pp. 380–386, 2016.

[2] Courtney Napoles and Chris Callison-Burch. Systematically adapting machine translation for grammatical error correction. In **Proc. Workshop on Innovative Use of NLP for Building Educational Applications (BEA)**, pp. 345–356, 2017.

[3] Marcin Junczys-Dowmunt, Roman Grundkiewicz, Shubha Guha, and Kenneth Heafield. Approaching neural grammatical error correction as a low-resource machine translation task. In **Proc. Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)**, pp. 595–606, 2018.

[4] Shun Kiyono, Jun Suzuki, Masato Mita, Tomoya Mizumoto, and Kentaro Inui. An empirical study of incorporating pseudo data into grammatical error correction. In **Proc. Conference on Empirical Methods in Natural Language Processing and International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)**, pp. 1236–1242, 2019.

[5] Ryo Nagata. Toward a task of feedback comment generation for writing learning. In **Proc. Conference on Empirical Methods in Natural Language Processing and International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)**, pp. 3206–3215, 2019.

[6] Abigail See, Peter J Liu, and Christopher D Manning. Get to the point: Summarization with pointer-generator networks. In **Proc. Annual Meeting of the Association for Computational Linguistics (ACL)**, pp. 1073–1083, 2017.

[7] Tatsunori B Hashimoto, Kelvin Guu, Yonatan Oren, and Percy S Liang. A retrieve-and-edit framework for predicting structured outputs. In **Proc. Advances in Neural Information Processing Systems (NeurIPS)**, p. 10052–10062, 2018.

[8] Kazuaki Hanawa, Ryo Nagata, and Kentaro Inui. Exploring methods for generating feedback comments for writing learning. In **Proc. Conference on Empirical Methods in Natural Language Processing (EMNLP)**, pp. 9719–9730, 2021.

[9] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In **Proc. Advances in neural information processing systems (NIPS)**, pp. 5998–6008, 2017.

[10] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. **arXiv preprint arXiv:1810.04805**, 2018.

[11] Vladimir I Levenshtein, et al. Binary codes capable of correcting deletions, insertions, and reversals. In **Soviet physics doklady**, pp. 707–710, 1966.

[12] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. **Journal of Machine Learning Research**, Vol. 21, pp. 1–67, 2020.

[13] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. Transformers: State-of-the-art natural language processing. In **Proc. Conference on Empirical Methods in Natural Language Processing: System Demonstrations (EMNLP)**, pp. 38–45, 2020.

[14] Liyuan Liu, Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Jiawei Han. On the variance of the adaptive learning rate and beyond. In **Proc. International Conference on Learning Representations (ICLR)**, 2019.

[15] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In **Proc. the Association for Computational Linguistics (ACL)**, pp. 311–318, 2002.

A 付録

3 章の文法誤り解説文生成の評価実験における生成例を表 A.1，A.2，A.3 に示す．ここで，表 A.1 は提案手法を，表 A.2 は retrieve-and-edit を用いて生成した解説文を示す．また，表 A.3 は，提案手法において生成モジュールの入力にマスクモジュールの出力を用いた場合とオラクルを用いた場合の比較を示す．

表 A.1 提案手法の出力結果の例

入力文	正解の解説文	
After all , as a student , he or she needs to put the study ≪at≫ the first place .	≪At≫ is not the correct <preposition> to be used with the set phrase formed using ≪the first place≫ meaning “to prioritize something”. ‘In’ is the <preposition> to be used with the <noun> ≪place≫.	
検索された解説文	マスクされた解説文	生成された解説文
The <preposition> ≪at≫ is normally used to indicate a relatively short period of time such as the time of day. Look up the <noun> ≪time≫ in a dictionary to learn the appropriate <preposition> to be used to indicate a period.	The <preposition> ≪at≫ is normally used to indicate [MASK] relatively short period [MASK] time [MASK] the time [MASK] day. Look up the <noun> ≪**time**≫ in a dictionary to learn the [MASK] <preposition> to [MASK] used to indicate [MASK] **period**.	The <preposition> ≪at≫ is normally used to indicate a relatively short period of time such as the time of day. Look up the <noun> ≪place≫ in a dictionary to learn the appropriate <preposition> to be used to indicate a place.
Using the <preposition> ≪in≫ makes the expression literally mean “to have in one hand”. Look up the <noun> ≪hand≫ in a dictionary to learn the appropriate <preposition> to be used to form an <idiom> introducing one of two ideas.	[MASK] the <preposition> [MASK] **the expression** [MASK] Look up the <noun> [MASK] in a dictionary to learn the [MASK] <preposition> to [MASK] used **to** [MASK]	≪At≫ is not the appropriate <preposition> to be used with the <noun> ≪place≫ to express “to put something in the first place”. Look up the <noun> ≪place≫ in a dictionary to learn the appropriate <preposition> to be used.

マスクされた解説文のうち，**太字の文字**はマスクトークンではないのに変更された文字を表す．生成された解説文のうち，赤色の文字はマスクされた解説文に含まれない文字を表す．

表 A.2 retrieve-and-edit の出力結果の例

入力文	正解の解説文
... colleagues is totally different ≪with≫ the way ...	The <preposition> ≪with≫ is often used to indicate concordance. Consult a ...
検索された解説文	生成された解説文
≪In≫ is not the <preposition> used with ’bad’ to qualify the subsequent ...	The <preposition> ≪with≫ is often used to indicate concordance. Consult a ...
≪to≫ is not the correct <preposition> to be used to refer to the target. Look ...	The <preposition> ≪with≫ is often used to indicate concordance. Consult a ...

表 A.3 提案手法におけるマスクの予測とオラクルの比較の例

入力文	正解の解説文	検索された解説文
... please keep consider ≪about≫ that.	The <verb> ≪consider≫ is a <transitive verb> when used to express “to think about” and does not need the <preposition> ≪about≫.	Since the <verb> encounter is a <transitive verb> when used to express “to meet someone”, a <preposition> does not need to precede the <object>.
	マスクされた解説文	生成された解説文
予測結果のマスク	[MASK] the <verb> [MASK] is <transitive verb> **when used to** [MASK] a <preposition> **does not need to precede the** [MASK]	When the <verb> ≪consider≫ is used as a <transitive verb> in a sense of “care for someone”, a <preposition> is unnecessary.
オラクルのマスク	[MASK] <verb> [MASK] is a <transitive verb> when used to express “to [MASK] does not need [MASK]	The <verb> ≪consider≫ is a <transitive verb> when used to express “to think about” and does not need the <preposition> ≪about≫.

マスクされた解説文のうち，**太字の文字**はマスクトークンではないのに変更された文字を表す．生成された解説文のうち，赤色の文字はマスクされた解説文に含まれない文字を表す．