訂正難易度を考慮した文法誤り訂正のための性能評価尺度

五藤　巧†　永田　亮†,††　三田　雅人††　塙　一晃††
† 甲南大学知能情報学部　†† 理研 AIP
E-mail: †{nagata-nlp2020,nagata-nlp2020}@ ml.hyogo-u.ac.jp.,
††{masato.mita,kazuaki.hanawa}@riken.jp

1. はじめに

本稿では，訂正難易度を考慮して，文法誤り訂正システムの性能を評価する手法を提案する．従来この分野では $F_{0.5}$ や GLEU [10] などの評価尺度が利用されてきたが，いずれも訂正難易度は考慮していない．言うまでもなく，訂正が容易な誤り（例：主語動詞の一致誤り）もあれば，難しい誤り（例：冠詞誤り）もある．また，同じカテゴリの誤りでも，事例ごとに訂正難易度が異なることもある．例えば，a/an の使い分けの誤りよりも，定冠詞/無冠詞の使い分けの誤りのほうが訂正は難しい．評価の目的にもよるが，通常は，訂正が容易な誤りよりも，難しい誤りを訂正できるほうが価値が高いであろう．以上を考慮すると，訂正難易度を考慮して性能評価を行えることが好ましいといえる．

詳細は 5. で述べるが，図 1 に可視化したように，提案手法では訂正難易度の推定を行う．例えば，図 1 では，定冠詞/無冠詞の使い分けの誤りの訂正が最も難しく，一方，不定冠詞で複数名詞を修飾した誤りの訂正は比較的容易であると推定している．提案手法は，このような訂正難易度を反映した性能評価を実現する．そのため，提案性能尺度で高い性能を達成するためには，訂正が難しい誤りを積極的に訂正する必要がある．このことは，訂正が難しい誤りに取り組むことを促すという副次的な利点をもたらす．

本研究の貢献は次の三点である．第一に，訂正難易度を考慮して性能評価を行うためのアルゴリズムを提案する．第二に，提案性能評価尺度を 6 コーパス，8 システムに適用し，その性質を定量的，定性的に分析する．例えば，提案性能評価尺度は，コーパスによらず，評価結果が一貫する傾向にあることを経験的に示す．第三に，提案性能評価尺度を計算するためのツールとデータを公開 (注 1) する．このツールとデータを用いると，任意のシステムに対して，訂正難易度を考慮した性能評価が容易に行える．また，図 1 のような訂正難易度の可視化機能も提供する．

2. 関連研究

文法誤り訂正の性能評価には，主に recall，precision，$F_{0.5}$，GLEU [10] が使われてきた．また，性能評価ツールとして，MaxMatch（M^2）スコアラ [2] や ERRANT [1] がある．

評価用コーパスも多数利用可能である．例えば，CoNLL-2013 [12]，CoNLL-2014 [13]，FCE [14]，JFLEG [11]，KJ [9]，ICNALE [4] などがある．これらのコーパスは，ライティングトピックや書き手の習熟度など様々な点で異なる．以上のコーパスとツールが文法誤り訂正技術の発展に大きく寄与してきたことは疑いようがない．

3. 提案性能評価尺度

3.1 基本アイデア

提案性能評価尺度の基本アイデアを，クイズを例にして説明する．いま，A と B という二種類のクイズがあるとする．更に，それぞれ 10 人に回答してもらったところ，クイズ A は正解者 10 人，クイズ B は正解者 1 人となったとする．このとき，直感的には，クイズ B のほうがより難しいといえるであろう．問題の難易度は様々に定義できるが，この考え方では難易度を問題の正解率に関連付けている．

提案手法も，この考え方を利用して訂正難易度を定義する．誤りの訂正を行うということは，システムが問題を解いていると捉えられる．そのように捉え，提案手法では，問題の正解率を訂正の成功率に置き換えて訂正難易度を定義する．

訂正難易度を定式化するために，次の記号を導入する．いま，システムの総数を N で表す．また，i 番目の誤りを正しく訂正したシステム数を n_i で表す．

このとき，訂正難易度をハイパーパラメタ a, b, c を用いて

$$w_i = a - \frac{n_i + b}{N + c} \tag{1}$$

と定義する．特に，本稿では $a = 1, b = 0, c = 0$ として，$0 \leqq w_i \leqq 1$ となる $w_i = 1 - \frac{n_i}{N}$ を考える．この式の第二項は訂正の成功率であるので，基本アイデアを実現している．

提案手法では，この訂正難易度で各誤りを重み付けして性能評価を行う（具体的には recall と precision を重み付けする）．理論上は，式 (1) の値は，訂正に成功したシステム数を数えるだけで計算できる．

しかしながら，このアイデアには実用上の問題がある．なぜなら，文法誤り訂正では，単語の削除と挿入があるため，一般には，各システムにおける訂正試行数が一定にならないからである．クイズを例にとると，解答者により解答したク

(注 1)：https://github.com/gotutiyan/GTS/

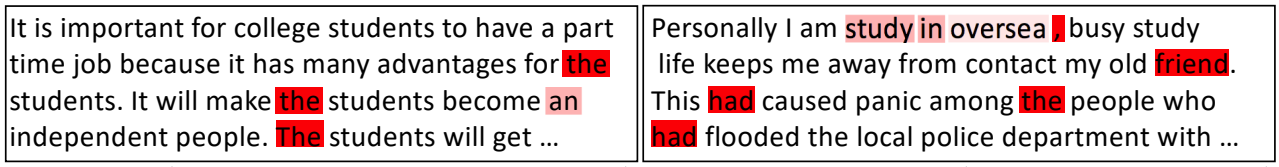

図 1： 提案手法で計算される訂正難易度を可視化した例.

イズの数が一定でないという状況が起こる．したがって，何らかの工夫を行い，訂正数を揃える必要が生じる．次節では，この問題を解決するためのアルゴリズムを示す．

3.2 アルゴリズム

図 2 にアルゴリズムの流れを示す．また，表 1 に，アルゴリズムの説明に用いる記号とその意味を示す．

アルゴリズムの概要は次の通りである．まず，誤りを正しく訂正した文（以降，正解文と呼ぶ）をチャンク列という別の形式に変換する．次に，各システムの訂正結果（以降，訂正文と呼ぶ）も同様にチャンク列に変換する．最後に，チャンク列を通じて，正解文と各訂正文を比較することで，各システムが訂正に成功したかどうかを表すバイナリ列を得る．このとき，バイナリ列をシステムによらず同じ長さに揃える．このバイナリ列に基づいて訂正難易度と性能値を計算する．

具体的なアルゴリズムは次の 3 ステップからなる：

Step（1）：正解文 $T^{(g)}$ を正解チャンク列 $C^{(g)}$ に変換

Step（2）：各訂正文 $T^{(s)}$ をチャンク列 $C^{(s)}$ に変換

Step（3）：$C^{(s)}$ をバイナリ列 $B^{(s)}$ に変換

Step（1）では，正解文 $T^{(g)}$ を原文 $T^{(o)}$ と比較することでチャンク $C^{(g)}$ に変換する．まず，Damerau-Levenshtein 距離が最小となるように $T^{(g)}$ と $T^{(o)}$ とのアライメントを取る（図 2 の Step（1）a）．このとき，$T^{(g)}$ におけるアライメントの境界からなる区間（図 2 の Step（1）a の縦棒で区切られた区間）をチャンクの基礎とする．更に，単語の挿入に相当するシステムの訂正に対応するため，チャンクの境界に仮想的なチャンクを挿入する．ただし，基礎のチャンクが単語の挿入に相当する場合，その境界には挿入しない（例：図 2 の Step（1）b の *have been*）．以上より得られた系列をチャンク列 $C^{(g)}$ とする（図 2 の Step（1）b）．

Step（2）では，各システムから得られる訂正文 $T^{(s)}$ をチャンク列 $C^{(s)}$ に変換する（図 2 の Step（2））．手順は，Step（1）と同様である．

Step（3）では，$C^{(g)}$ と $C^{(s)}$ を比較し，訂正の成否を表すバイナリ列 $B^{(s)}$ を得る．まず，コスト最小の弾性マッチングにより，$C^{(g)}$ と $C^{(s)}$ のマッチングを行う．コストは，$C^{(g)}$ のチャンク $c_i^{(g)}$ と $C^{(s)}$ のチャンク $c_j^{(s)}$ が一致していれば 0，そうでなければ 1 とする．ここで，$c_i^{(g)}$ と $c_j^{(s)}$ が一致するとは，次の二つの条件を満たすこととする：(i) 両チャンクとも，原文 $T^{(o)}$ 中の同じ単語列へアライメントされている；

図 2： アルゴリズムの流れと具体例.

表 1： アルゴリズムで使用する記号の意味.

記号	意味
s	ある誤り訂正システム
$T^{(o)}$	原文のトークン列
$T^{(g)}$	正解文のトークン列
$T^{(s)}$	システム s の出力（トークン列）
$C^{(g)}$	$T^{(g)}$ に対するチャンク列
$C^{(s)}$	$T^{(s)}$ に対するチャンク列
$c_i^{(g)}$	$C^{(g)}$ の i 番目の要素
$c_i^{(s)}$	$C^{(s)}$ の i 番目の要素
$B^{(s)}$	システム s の訂正の成否を表すバイナリ列
$b_i^{(s)}$	$B^{(s)}$ の i 番目の要素

(ii) 両チャンク内の単語列が同一である．以上の手順により得られるバイナリ列を $B^{(s)}$ とする．$B^{(s)}$ は $C^{(g)}$ を基準に決定するため，システム s によらず一定の長さとなる（$C^{(g)}$ 中のチャンク数と等しい）．よって，訂正システムにより訂正数が一定ではないという問題を解決する．

この $B^{(s)}$ から，訂正成功数 n_i が容易に求まる（i.e., $n_i = \sum_s b_i^{(s)}$）．更に，n_i と式 (1) より w_i が求まる．

最後に，w_i で重み付けした recall と precision をそれぞれ次のように定義する（注 2）：

$$R = \frac{\sum_{i\in\mathcal{E}} w_i b_i}{\sum_{j\in\mathcal{E}} w_j}, P = \frac{\sum_{i\in\mathcal{E}} w_i b_i}{\sum_{j\in\mathcal{E}} w_j b_j + \sum_{k\in\mathcal{C}} w_k (1-b_k)}. \quad (2)$$

（注 2）：冗長になるため，式中の $b^{(s)}$ は b と省略している．

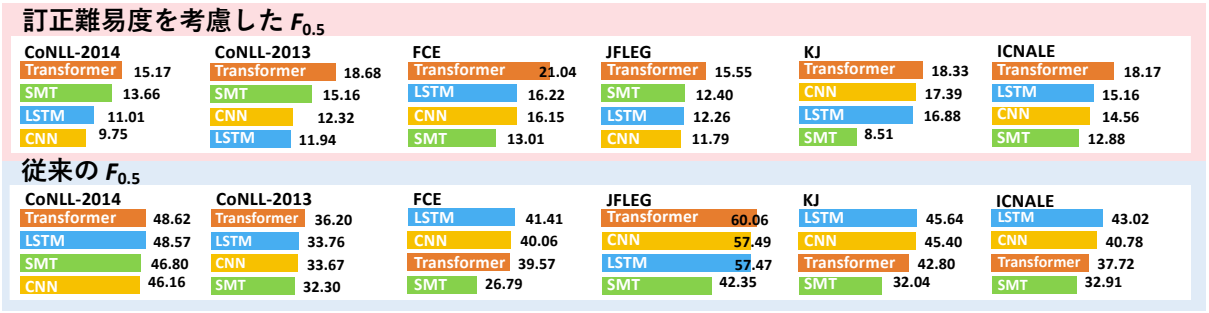

図 3： 提案性能評価尺度（重み付き $F_{0.5}$）と従来の $F_{0.5}$ の比較.

ただし，$\mathcal{E}$ は，$C^{(g)}$ 中の誤りを含むチャンクのインデックスからなる集合である．また，$\mathcal{C}$ は，システム s が誤り訂正を適用した，$C^{(g)}$ 中のチャンクのインデックスからなる集合である．なお，Step（1），(2）のアライメント結果から，誤りがあるチャンクと誤り訂正が適用されたチャンクに関する情報が得られることに注意されたい．また，R と P を用いて，$F_{0.5}$ 値も通常の定義で計算できることにも注意されたい．

4. 評価実験

提案手法を評価するために二種類の実験を行った．一つ目はコーパス重視の実験で，二つ目はシステム重視の実験である．いずれの場合も，提案評価尺度（重み付き $F_{0.5}$）と従来の $F_{0.5}$ を比較した．従来の $F_{0.5}$ は M^2 スコアラを用いて算出した．

一つ目の実験では，6 コーパス（CoNLL-2013，CoNLL-2014，FCE，JFLEG，KJ，ICNALE) に対する 4 システム（CNN ベース，LSTM ベース，Transformer ベース，SMT ベース）の訂正結果を用いた．これは，文法誤り訂正の性能を調査した文献 [8] と同じ条件である（詳細は，同文献を参照されたい）.

図 3 に実験結果を示す．従来の $F_{0.5}$ では，コーパスにより最高性能となるシステムが異なる．一方で，重み付き $F_{0.5}$ では，全てのコーパスで Transformer が 1 位である．このことは，Transformer が他の 3 システムと異なる訂正傾向を持つことを意味する．なぜなら，重み付き $F_{0.5}$ は，その定義より，他が訂正できない誤りを訂正できるシステムを重視するからである．他システムと訂正傾向が異なるということは，コーパスの種類とは（ある程度）独立した事象であると予想されるため，最高性能となるシステムが一貫すると分析できる．

二つ目の実験では，標準的な評価データである CoNLL-2014 に対する 8 システムの訂正結果を用いた．上述の 4 システムに加え，State-of-the-Art を含む体表的な誤り訂正システム（Kiyono ら [7], Junczys ら [5], [6],Ge ら [3]）を用いた．

表 2 に実験結果を示す．表中の「変動」とは，重み付き $F_{0.5}$ と従来の $F_{0.5}$ で評価したときの順位の差である．表 2 より，Kiyono らのシステムが最高性能を達成しており，一つ目の実験の Transformer と同様に，他とは異なる訂正傾向を持つことを示唆する．実際，Kiyono らのシステムのみが，疑似訓練データの生成（大量の母語話者コーパスと back translation により得た疑似誤りコーパス）を利用する．一方，従来，文法誤り訂正では，同じような訓練データ（多くの場合 CoNLL のデータセットと Lang-8）を用いることが多い（ここでの他の 7 システムの訓練も同様である）．したがって，Kiyono らのシステムは，正しい英文と誤りのある英文の両面で，他システムにはない知識源を利用している．このことが，他にない訂正傾向を生み出していると分析できる．

また，表 2 では，2 システムの順位が上昇している．この 2 システムのみ SMT に基づいているという点で，他とは訂正傾向が異なる（他は全て深層学習ベースである）.

5. 考　　察

本節では，提案性能評価尺度が，訂正難易度を反映できているかどうかを考察する．そのため，まず，提案手法で計算される訂正難易度（w_i）を可視化する．具体的には，w_i に応じて各誤りを五段階に色付けする（最も訂正が難しい誤りを最も濃い赤とする）．用いるデータは，ICNALE に対する 4 システムの訂正結果と CoNLL-2014 に対する 8 システムの訂正結果である．

図 1（2 ページ目に掲載）に可視化結果の一部を示す．可視化結果は，人間の直感によく一致する．例えば，無冠詞/定冠詞の使い分けや過去形/過去完了形の使い分けに関する誤りの訂正が最も難しいと認識されている．これらは，文外文脈や書き手の意図の理解を必要とする訂正難易度の高い誤りである．一方で，比較的狭い文脈で訂正可能な誤り（例：*an independent people*，*oversea_*）は難易度が低いと認識されている．

更に，誤りをカテゴリごとに w_i の平均値で順位付けする．誤りカテゴリはツール ERRANT から得られるものを利用する．表 3 に，出現頻度 20 以上の誤りカテゴリを対象にした結果を示す．表 3 の結果も人間の直感によく合うことがわかる．例えば,「主語動詞の一致」や「綴り誤り」が，訂正が容易な誤りであると判定されている．逆に，難易度が高いものは，主に，語彙選択にかかわるものである．また，訂正難易

表 2： 重み付き $F_{0.5}$ による順位変動．

システム	重み付き $F_{0.5}$	従来 $F_{0.5}$	順位（変動）
Kiyono ら [7]	26.09	65.00	1 (—)
Junczys-Dowmunt ら [5]	21.40	57.53	2 (↑ 1)
Ge ら [3]	19.14	61.34	3 (↓ 1)
Junczys-Dowmunt ら [6]	15.76	49.49	4 (—)
Transformer	15.05	48.62	5 (—)
SMT	12.78	46.80	6 (↑ 1)
LSTM	10.70	48.57	7 (↓ 1)
CNN	9.81	46.16	8 (—)

度は中程度であるが，標準偏差が大きい誤り（例：「冠詞」，「前置詞」）もあり興味深い．これは，同じカテゴリの誤りでも事例により難易度がばらつくことを意味する．

また，表 3 は，文法誤り訂正の語学学習支援への示唆も与える．「形容詞」や「副詞」の出現頻度は比較的低いが訂正難易度は高い．出現頻度が低い誤りは，従来の性能評価尺度では過少に評価される傾向にある（通常の recall や precision では高頻度な誤りが支配的になる）．そのため，低頻度な誤りの訂正に取り組むことへの動機付けが少ない．一方で，提案性能評価尺度では，訂正難易度で重み付けされるため，中低頻度でも訂正難易度が高い誤りは無視できない．言い換えれば，高い性能を達成するためには，そのような誤りを積極的に訂正する必要がある．豊かな文章を書く上で形容詞/副詞の適切な使用は重要であることを考慮すると，形容詞/副詞の誤りを訂正することを促すことは，語学学習支援の観点から好ましいといえる．

残された課題の一つに，訂正成功率と訂正難易度のバランスの調整がある．本稿では，訂正難易度を $w_i = 1 - \frac{n_i}{N}$ と定義した．この定義（バランス）が良いかどうかは，性能評価の目的に依存するため一概にいうことは難しいが，更なる探求が必要なことは間違いない．他の方法として，例えば，訂正成功率に応じて，指数関数的／対数関数的に訂正難易度を変化させることも可能である．

ここで重要であるのは，提案手法を用いると訂正難易度に関連付けてシステムの性能評価を行えるという点である．更に，本稿は文法誤り訂正を対象にしたが，原理的には，提案手法は，各種分類問題，系列ラベリング，生成問題にも応用できるという点でも重要である．

6. おわりに

本稿では，訂正難易度を考慮した，文法誤り訂正のための性能評価尺度を提案した．6 コーパス，8 システムを用いた実験により，提案評価尺度の特徴を定量的，訂正的に分析した．特に，提案評価尺度の訂正難易度は人間の直感によく合うことを示した．今後は，訂正成功率と訂正難易度のバランスの調整や訂正難易度を用いた誤り訂正モデルの学習方法の考案などに取り組む予定である．

表 3： 誤りの種類と訂正難易度（CoNLL-2014，8 システム）

誤りの種類	平均 w_i	標準偏差	出現頻度
形容詞	0.982	0.074	28
その他	0.972	0.120	440
句読点	0.966	0.114	109
正書法	0.940	0.164	42
名詞	0.937	0.180	93
代名詞	0.923	0.177	63
副詞	0.911	0.181	55
動詞	0.891	0.254	160
時制	0.876	0.213	191
冠詞	0.747	0.292	356
前置詞	0.724	0.343	226
形態	0.644	0.370	78
動詞の活用	0.590	0.393	89
名詞の数	0.539	0.340	210
綴り誤り	0.533	0.342	64
主語動詞の一致	0.499	0.365	97

参 考 文 献

[1] C. Bryant et. al., “Automatic Annotation and Evaluation of Error Types for Grammatical Error Correction,” Proc. of ACL, 2017.

[2] D. Dahlmeier et. al., “Better Evaluation for Grammatical Error Correction,” Proc. of NAACL, 2012.

[3] T. Ge et. al., “Reaching Human-level Performance in Automatic Grammatical Error Correction: An Empirical Study,” arXiv, 2018.

[4] S. Ishikawa et. al., “The ICNALE and Sophisticated Contrastive Interlanguage Analysis of Asian learners of English,” Learner Corpus Studies in Asia and the World, vol.1, 2013.

[5] M. Junczys-Dowmunt et. al., “Phrase-based Machine Translation is State-of-the-Art for Automatic Grammatical Error Correction,” Proc. of EMNLP, 2016.

[6] M. Junczys-Dowmunt et. al., “Approaching Neural Grammatical Error Correction as a Low-Resource Machine Translation Task,” Proc. of NAACL, 2018.

[7] S. Kiyono et. al., “An Empirical Study of Incorporating Pseudo Data into Grammatical Error Correction,” Proc. of EMNLP-IJCNLP, 2019.

[8] M. Mita et. al., “Cross-Corpora Evaluation and Analysis of Grammatical Error Correction Models — Is Single-Corpus Evaluation Enough?,” Proc. of NAACL, 2019.

[9] R. Nagata et. al., “Creating a Manually Error-Tagged and Shallow-Parsed Corpus,” Proc. of ACL, 2011.

[10] C. Napoles et. al., “Ground Truth for Grammatical Error Correction Metrics,” Proc. of ACL, 2015.

[11] C. Napoles et al., “JFLEG: A Fluency Corpus and Benchmark for Grammatical Error Correction,” Proc. of EACL, 2017.

[12] H. Ng et. al., “The CoNLL-2013 Shared Task on Grammatical Error Correction,” Proc. of CoNLL 2013 Shared Task, 2013.

[13] H. Ng et. al., “The CoNLL-2014 Shared Task on Grammatical Error Correction,” Proc. of CoNLL 2014 Shared Task, 2014.

[14] H. Yannakoudakis et. al., “A New Dataset and Method for Automatically Grading ESOL Texts,” Proc. of ACL, 2011.