Wikipediaの修正履歴を用いた日本語入力誤りデータセットの構築

田中 佑　村脇 有吾　河原 大輔　黒橋 禎夫
京都大学 大学院情報学研究科
{ytanaka, murawaki, dk, kuro}@nlp.ist.i.kyoto-u.ac.jp

1 はじめに

文章にはしばしば入力誤りが出現する．入力誤りによって，人はスムーズにテキストを読むことを妨げられ，計算機は正しく解析することを妨げられる．人が文章を書く際，入力誤り修正を行うシステムを用いれば，入力誤りの発生を抑えることができる．また，計算機においては，入力誤り修正システムを事前に適用することによって，解析誤りを減らし，種々の解析精度向上が見込めるため，NLP においても重要である．

入力誤り修正の一種であるスペルミス修正において，ニューラルネットワークは成功を収めており [5]，入力誤り修正システムを構築するのに有望であると考えられる．ニューラルモデルは多量の学習データを必要とすることが知られており，ニューラル入力誤り修正システムの実現への重要課題は多量の入力誤りとその修正ペアを用意することである．

本研究では，Wikipedia の修正履歴から，日本語 Wikipedia 入力誤りデータセット (JWTD; Japanese Wikipedia Typo Dataset) を作成した．JWTD は 50 万を超える入力誤りと修正文ペアからなり，公開予定である．日本語入力誤りを誤字・脱字・衍字・漢字誤変換の 4 種類に分類し集めた．クラウドソーシングによって，JWTD を評価し，JWTD を用いて入力誤り修正を学習する実験を行った．

2 関連研究

Max ら [4] は Wikipedia の修正履歴からフランス語スペルミスデータセットを構築している．彼らの手法は編集された単語が特定できることを前提としている．単語分割の必要がある日本語などの言語では，入力誤りによって，単語分割が狂い，複数の単語が変化する場合があり，その場合編集された単語を特定できない．そのため，彼らの手法を日本語に直接適用することはできない．

金山ら [7] は日本語 Wikipedia の編集履歴から書き換えパターンの抽出を行っている．彼らは修正を，表記修正・内容修正に分類し，その分類器を作成している．このうち表記修正は，本研究で扱う入力誤り修正に近いと考えられるが，両者は異なる．「コンピュータ」と「コンピューター」などのスタイルに関する修正を，彼らは表記修正として扱うが，本研究ではどちらにも誤りはないと考え，入力誤り修正として扱わない．そのため，彼らの分類器を入力誤り修正の抽出に利用するのは難しい．また，彼らのデータは公開されておらず，利用できない．

誤字	兄の部隊【の → に】所属していた兵士で ···
脱字	··· 組織をもっていること【+ で】知られる。
衍字	特に免疫力の差などがそう【– う】である。
誤変換	··· 全てが大学院に【以降 → 移行】して ···

表 1: 入力誤りとその修正例

Hagiwara ら [2] は多言語において Github の commit ログから入力誤りを収集した Github typo corpus を作成し，公開している．これには日本語も含まれているが，サイズは 1,000 文程度でニューラルモデルの学習には十分ではない．

3 日本語における入力誤りの分類

本研究では日本語における入力誤りを誤字・脱字・衍字・漢字誤変換（以下，誤変換と呼ぶ）の 4 種類に分類した．それぞれの入力誤りとその修正例を表 1 に示す．

誤字は文字の入れ替え，脱字は文字の抜け，衍字は余分な文字の挿入，誤変換は以下で説明するような，同一の読みを持つ漢字の入れ替えの誤りである．

一般的な日本語の漢字入力は，漢字の読みであるひらがなを入力し，その後インプットメソッドによって表示される変換候補リストから，入力したい漢字を選択する．この際，同じ読みの漢字は無数に存在するため，漢字選択を誤る場合がある．このような漢字選択の際の誤りが誤変換である．

誤字と誤変換は文字の入れ替えという点では同質であるが，単純なタイプミスである誤字と，漢字選択の誤りである誤変換は，異なる分布で発生すると考え，本研究では区別している．

4 データセット作成手法

データセット作成は，Wikipedi の修正履歴から入力誤りとその修正文ペアを取り出し，その後，入力誤り修正ではないと考えられるペアをフィルタリングした．

4.1 Wikipedia マイニング

Wikipedia の修正履歴から，入力誤り修正を有している可能性のある文ペアを以下の方法で取り出した[1]．

1. 全ての記事の全ての版から，WikiExtractor[2]を使い本文テキストを取り出し，文単位で改行する[3]．
2. それぞれの記事で，Python3 difflib ライブラリ[4]を使い，ある版とその直前の版との行単位の差分をとり，一文の中に編集がある文ペアだけを取り出す．ここで，編集後の文の文字数が 10 以下または 200 以上の文は，分の長さとして不適切であると判断しそのペアを取り除いた．
3. 文ペアを MeCab[5]で形態素分割し，difflib で形態素単位の差分をとり，編集箇所を特定する．ここで，difflib で出力される差分は最小編集とは違い，複数の形態素を置換するような編集も出力される．そのため，複数の形態素を変化させる入力誤りも収集できる．編集箇所における編集距離が 5 を超えるものを，入力誤りの修正ではないと考え，そのペアを取り除いた．
4. このようにして得られた文ペアのうち，以下の条件を満たすものを取り出す[6]．

   誤字　編集箇所における文字単位編集距離が 1・編集箇所の文字数変化が 0・編集箇所の文字は前後とも，ひらがな又はカタカナ又はアルファベット

   脱字　編集箇所における文字単位編集距離が 1・編集箇所の文字数変化が +1・追加文字は，ひらがな又はカタカナ又はアルファベット

   衍字　編集箇所における文字単位編集距離 1・編集箇所の文字数変化が −1・削除文字は，ひらがな又はカタカナ又はアルファベット

   誤変換　編集前後の文の読みが一致・編集箇所の文字は前後ともに漢字が含まれる[7]

[1] undo や revert された版を取り除く，A→B・B→A のような編集が循環している文ペアを取り除く，A→B・B→C のような編集が推移している文ペアを A→C に置換するなどの操作も行ったが，紙面の都合上詳細は省略する．

[2] https://github.com/attardi/wikiextractor

[3] https://github.com/ku-nlp/textformatting を利用した．

[4] https://docs.python.org/3/library/difflib.html

[5] https://taku910.github.io/mecab/

[6] ひらがな・カタカナ・漢字の判定には，Python3 regex ライブラリ (https://pypi.org/project/regex/) を利用した．

[7] 文の読み取得は形態素解析器 Juman++(http://nlp.ist.i.kyoto-u.ac.jp/index.php?JUMAN++) と MeCab を用い，少なくともどちらか一方の読みの解析結果が一致すればよいとした．

4.2 フィルタリング

上記の操作で得られた文ペアには，入力誤り修正ではないと考えられる文ペアも含まれる．それらを取り除くため，以下の方法でフィルタリングを行なった．

品詞・形態素解析　修正前も修正後もどちらも文として自然なペアを取り除くために行なった．人名，数詞，時制に関する修正などを Juman++の解析結果をもとにこれらを取り除いた．また，「とは → は」などの Wikipedia 特有のスタイルの修正が多く見られたため，これらも取り除いた．

リダイレクトデータ　「ケニヤ」と「ケニア」のような，表記が異なるが意味は同じ単語間の修正を含むペアを取り除くために行なった．Wikipedia ではこのような単語を検索すると，どちらも同じページに遷移するリダイレクト機能がある．リダイレクトする単語対リストを作成し，修正箇所における修正前後の単語対がリストに含まれているペアを取り除いた．

言語モデル　文字レベル LSTM 言語モデルを用いて２つの方法でフィルタリングを行なった．学習データは日本語 Wikipedia の最新ページ全てを使った．

フィルタリングの方法の 1 つ目は修正による損失の減少量を用いたものである．修正が間違っているペア，および，修正前も修正後もどちらも文として自然なペアを取り除くために行なった．学習したモデルに修正前の文と修正後の文を入力し，それぞれの損失 (negative log likelihood) を計算する．修正前・後の合計損失を $loss_{pre}$・$loss_{post}$ とし，以下を満たすペアを取り除いた．

$$\frac{loss_{post} - loss_{pre}}{\text{編集のあった文字数}} > \alpha$$

しきい値 α はヒューリスティックに決定した値であり，誤字は −4，脱字は −5，衍字は −6 とした．このフィルタは，誤変換の「若干 → 弱冠」などの高頻度語から低頻度語への正しい修正も取り除いてしまったため，誤変換には適用しなかった．

フィルタリング方法の２つ目は修正後の損失を用いたものである．修正後の文が不自然なペアを取り除くために行なった．以下を満たすペアを取り除いた．

$$\frac{loss_{post}}{\text{修正後の文の文字数}} > \beta$$

しきい値はすべての入力誤りに共通でヒューリスティックに 5 と決定した．

種類	フィルタ前	フィルタ後
誤字	680,097	86,742
脱字	490,673	89,428
衍字	407,413	110,305
誤変換	296,757	240,219
合計	1,874,940	526,694

表 2: フィルタリング前後の文ペアの数

5 データセットの分析

4 章の方法を 2019 年 6 月の Wikipedia 修正履歴に適用し，データセットを構築した．得られた文ペアの数を表 2 に示す．ここで，誤変換のフィルタリングによる減少量が少ないが，これは言語モデルフィルタの 1 つを適用していないためである．データセットにおける各入力誤りの上位 10 個の修正を表 3 に示す．修正の単位は Juman++で修正前の文，修正後の文をそれぞれ形態素分割し，difflib に入力して得られた形態素単位の挿入・削除・置換の編集である．脱字では「る → いる」などの，い抜き言葉，「れる → られる」などの，ら抜き言葉の修正が多く見られた．これらは話し言葉で用いられるが，書き言葉においては一般に不適切とされるものである．誤変換では，「製作 → 制作」のように意味が似ている漢字間の修正が見られた．これらの漢字は使い分けの判断が難しく，また，入力誤りとは言えないものも多かった．このような修正を取り除くことは今後の課題である．

6 クラウドソーシングによる評価

クラウドソーシングを用いて，JWTD の評価を行った．評価対象はランダムに選んだ記事 2,996 ページ中の，誤字 1,861，脱字 2,147，衍字 2,395，誤変換 4,388 文ペアである．同じ修正が数多く存在しているページが存在したため，データの偏りを避けるために，選ばれたページ内に同一の修正が 4 つ以上含まれていた場合，そのうちの 3 つのみをランダムに採用した．

6.1 タスク内容

修正前と修正後の文を同時に提示して，それぞれ書き言葉として自然かどうかを質問した．このとき，提示する際は単に文 A，文 B として提示し，どちらが修正前か修正後かをわからないようにした．選択肢は「A は書き言葉として自然な文であるが，B は不自然な文である」「B は書き言葉として自然な文であるが，A は不自然な文である」「どちらの文も自然である」「どちらの文も不自然である」「わからない」の 5 つとした．また，それぞれの文ペアにつき 10 人のワーカーに回答してもらった．

6.2 結果

文ペアのうち，一つの選択肢が過半数の票を持つ修正を，その回答によって分類した．どの選択肢も過半数の票を持たないものは「その他」に分類した．このようにして得られた分類の割合を表 4 に示す．ここで，過半数を超えている回答が，修正前の文が不自然・修正後の文が自然という回答のペアを「正しい修正」，修正前の文が自然・修正後の文が不自然という回答のペアを「間違った修正」と分類している．

誤字は 83.0%，脱字は 77.6%，衍字は 88.8%，誤変換は 69.8%のペアが「正しい修正」と分類された．脱字の「どちらも自然」が 13.1%と他と比較して高い．このうち 41%が，書き言葉においては正しい修正とされるべき，い抜き言葉の修正であった．誤変換は「その他」の割合が他より高いことから，ワーカーの判断が分かれていることがわかる．これは誤変換の修正は他と違い漢字の知識を必要とするためだと考えられる．また，誤変換の「間違った修正」と分類された中には，「若干 → 弱冠」などの実際は正しい修正であるものも含まれていた．そのため，脱字・誤変換の実際の質は表 4 の値よりも高い可能性が考えられる．

7 データセットの利用

JWTD を利用して，入力誤り修正システムの実験を行った．

7.1 実験設定

代表的な encoder-decoder モデル機械翻訳ツールである OpenNMT [3][8]を用いて，入力誤り修正を学習した．クラウドソーシングで評価していない入力誤り修正文の，誤字 79,714，脱字 82,227，衍字 102,897，誤変換 230,490 文を学習データとし，それぞれ 5,000 文を検証データとした．テストデータはクラウドソーシングで「正しい修正」と分類されたもの，誤字 1,689，脱字 1,665，衍字 2,127，誤変換 3,061 文とした[9]．入出力を形態素レベルと文字レベルの 2 種類で実験を行った．モデルは 1 種類の入力誤りのみを与えて学習させた．OpenNMT の設定は embedding・hidden サイズを 500 次元，encoder・decoder を 2 層の RNN，train steps を 200,000，learning rate を 0.5 とした．

7.2 評価方法

評価指標は入力誤り修正の適合率・再現率・F 値，システム出力文と修正後の文の完全一致率，SARI [6] を用いた．

[8] https://github.com/OpenNMT/OpenNMT-py

[9] 誤字においては，6 章で選んだ文ペアに，ひらがなの「へ」とカタカナの「へ」に関する見分けのつかない修正が 145 ペア存在した．これらはクラウドソーシングで出題するのは不適切であると判断し，クラウドソーシングに用いず，我々が評価した．全て正しい修正であったため，誤字のテストデータに加えている．

誤字		脱字		衍字		誤変換	
の → を	4.0%	る → いる	13.3%	− の	14.2%	製作 → 制作	2.8%
の → に	3.5%	＋ に	10.9%	− を	11.3%	始めて → 初めて	1.4%
づつ → ずつ	3.1%	＋ を	10.4%	− に	10.5%	制作 → 製作	1.0%
を → の	2.7%	＋ と	5.1%	− は	6.6%	運行 → 運航	1.0%
の → が	1.9%	た → いた	4.5%	− が	6.5%	後 → 跡	0.9%
に → の	1.8%	＋ が	4.3%	− と	4.7%	作詩 → 作詞	0.9%
を → が	1.8%	＋ の	3.8%	− で	4.4%	勤めた → 務めた	0.8%
が → を	1.3%	れ → れて	2.6%	− い	1.7%	勤める → 務める	0.7%
か → が	1.1%	＋ い	2.2%	− た	1.5%	開放 → 解放	0.5%
と → を	1.0%	れる → られる	1.3%	− し	1.5%	付属 → 附属	0.5%

表 3: データセット上位 10 個の修正

種類	正しい修正	間違った修正	どちらも自然	どちらも不自然	わからない	その他
誤字	83.0	0.3	6.8	0.1	0.2	9.6
脱字	77.6	0.1	13.1	0.0	0.0	9.3
衍字	88.8	0.4	7.1	0.0	0.0	3.6
誤変換	69.8	7.5	3.1	0.0	0.1	19.5

表 4: クラウドソーシングの結果（値は%）

モデル	種類	適合率	再現率	F 値	一致	SARI
形態素	誤字	11.9	39.7	18.3	30.6	61.1
	脱字	23.2	69.9	34.8	59.6	80.2
	衍字	16.8	79.7	27.7	68.1	83.8
	誤変換	30.8	57.0	40.0	47.9	71.0
文字	誤字	4.6	37.3	8.1	22.1	54.3
	脱字	6.5	59.4	11.8	41.6	69.6
	衍字	6.2	76.6	11.4	46.9	72.5
	誤変換	10.0	43.5	16.3	30.2	60.9

表 5: 実験結果

入力誤り編集の適合率・再現率は次の定義で計算した．各文において，修正前の文と修正後の文の文字単位の最小編集 G と，修正前の文とシステム出力文の文字単位の最小編集 O を求め[10]，$|G \cap O|$ を求める．全ての文における $|G \cap O|$ の合計を N_T，$|G|$ の合計を N_G，$|O|$ の合計を N_O として，適合率=N_T/N_O，再現率=N_T/N_G とした．

SARI はテキスト編集の評価に用いられる指標である．n-gram 単位で，入力文・正解文・システム出力文間のテキストの追加・削除・維持を数え上げ，それらの F 値平均を算出する．本実験では文字レベルの 4-gram を採用した[11]．

7.3 実験結果

実験結果を表 5 に示す．全てにおいて，形態素レベルが文字レベルの結果を上回った．形態素レベルの衍字では，完全一致が 68.1%であるが，適合率は 16.8%と低い．テキスト生成がうまくいかず，同じトークンを繰り返し生成している出力文が見られたため，これらが適合率を下げていると考えられる．入力文をそのままシステム出力文として計算した場合，SARI スコアは 30 程度の値となることを考慮すると，どの結果もスコアが向上しているのを確認できた．

[10]Python3 python-Levenshtein ライブラリ (`https://pypi.org/project/python-Levenshtein/`) を用いた．

[11]実装は `https://github.com/tensorflow/tensor2tensor/blob/master/tensor2tensor/utils/sari_hook.py` [1] を利用し，β-deletion は 1 とした．

8 おわりに

本研究では，Wikipedia の修正履歴から日本語入力誤りデータセット (JWTD) を作成した．JWTD の質をクラウドソーシングで評価した．また，JWTD を利用して入力誤り修正を学習する実験を行なった．今後は，クラウドソーシングで得られた結果をもとに，さらなるフィルタリングを検討予定である．

参考文献

[1] Mor Geva, Eric Malmi, Idan Szpektor, and Jonathan Berant. Discofuse: A large-scale dataset for discourse-based sentence fusion. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 3443–3455, 2019.

[2] Masato Hagiwara and Masato Mita. Github typo corpus: A large-scale multilingual dataset of misspellings and grammatical errors. *arXiv preprint arXiv:1911.12893*, 2019.

[3] Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander M. Rush. OpenNMT: Open-source toolkit for neural machine translation. In *Proc. ACL*, 2017.

[4] Aurélien Max and Guillaume Wisniewski. Mining naturally-occurring corrections and paraphrases from Wikipedia's revision history. Proceedings of the Seventh conference on International Language Resources and Evaluation (LREC'10), 2010.

[5] Keisuke Sakaguchi, Kevin Duh, Matt Post, and Benjamin Van Durme. Robsut wrod reocginiton via semi-character recurrent neural network. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[6] Wei Xu, Courtney Napoles, Ellie Pavlick, Quanze Chen, and Chris Callison-Burch. Optimizing statistical machine translation for text simplification. *Transactions of the Association for Computational Linguistics*, Vol. 4, pp. 401–415, 2016.

[7] 金山博, 荻野紫穂. Wikipedia の編集履歴を用いた書き換えパターンの抽出. 言語処理学会第 17 回年次大会論文集, 2011.