大規模添削コーパスを用いた統計的機械翻訳手法による日本語誤り訂正

水本智也　　小町守　　松本裕治
奈良先端科学技術大学院大学
{tomoya-m, komachi, matsu}@is.naist.jp

1　はじめに

「外務省の広報文化外交（海外広報・文化交流）」[*1] によると，海外 133 の国・地域で約 365 万人が日本語を学習しており，学習者の数は 30 年で 30 倍に増加している．そのような中，現在「相互添削型 SNS Lang-8」[*2] が注目を浴びており，設立から約 3 年半で会員数 20 万人を突破した．Lang-8 は言語学習者用 SNS サイトで，第 2 言語学習者が学習言語で日記を書くと，その言語を母語とするユーザが添削をしてくれる．図 1 を見ると，Lang-8 の中でも日本語を学習している人は全体の 29% と 2 番目に多く，約 7 万人いる．また，表 1 を見ると，日本語学習者の文が 767,219 文と最も多く，日本語添削が望まれていることがわかる．

日本語学習者の日本語誤り訂正に関する研究は格助詞に焦点を当てているものが多い [3, 4, 5]．また，格助詞誤り検出のシステムとして，奈良先端科学技術大学院大学で開発されている「Chantokun」[*3] がある．

しかしながら，現実の日本語学習者の作文には，格助詞誤りだけでなく，単語の表記誤り，コロケーション誤りなども多く存在している．そこで，本研究では，格助詞誤りだけでなく，学習者の誤り全般に焦点を当てる．具体的には，添削前後の大規模なパラレルデータを用いて，統計的機械翻訳の手法による誤り訂正を行う．本研究では，文字単位の分かち書きコーパスを使用し，高い精度で訂正できることを示した．

2　統計的機械翻訳 (SMT) を使った誤り訂正

本研究では SMT を使って誤り訂正を行う．

$$\hat{e} = \arg\max_e P(e|f) = \arg\max_e P(e)P(f|e) \qquad (1)$$

式 1 はノイジーチャネルモデルを使った SMT の式である．ここで e はターゲット側（翻訳後の言語）であり，

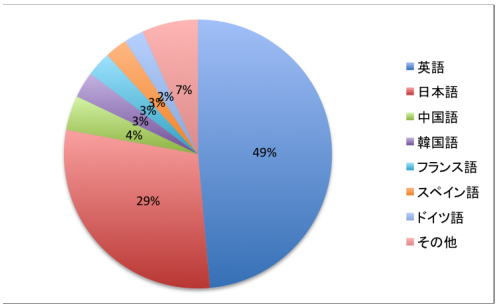

図 1　Lang-8 の学習言語によるユーザ数の分布

f がソース側（翻訳前の言語）である．また，$P(e)$ が言語モデル（LM）から得られる確率であり，$P(f|e)$ が翻訳モデル（TM）から得られる確率である．TM は文単位で 1 対 1 対応のとれた対訳コーパスから学習し，LM はターゲット側言語から学習する．

これを誤り訂正に適応した場合，f は学習者の書いた添削前の文となり，e は添削された後の文となる．また，TM は添削前後の文で 1 対 1 対応のとれた添削コーパスから学習し，LM は正しい文から学習したものになる．こうすることで SMT を使って，誤りを含む文を正しい文に変換することができる．

SMT を用いる利点は 2 つある．(1) 添削コーパスがあれば，言語教育の知識がなくてもよい．(2) SMT のツールを単純に用いることができる．(3) SMT の手法が改良されれば，その恩恵を受けることができる．

SMT を使った誤り訂正の先行研究としては Brockett ら [1] によるものがある．Brockett らは SMT を用いて，英語の加算・不加算名詞の使い方を対象とした誤り訂正を行っている．学習に用いているデータは 45,000 文で擬似的に作り出した 346,000 文からランダムに抽出してきている．本研究は，(1) 誤りの種類を 1 種類に限定しない，(2) 実データを用いている点で異なっている．

3　文字単位の分割を用いた統計的誤り訂正

一般的に機械翻訳で日本語から他の言語へ変換する場合，単語単位に分割してから行う．しかしながら，学習者の書いた文には誤りやひらがなが多く含まれており，

[*1] http://www.mofa.go.jp/mofaj/gaiko/culture/koryu/edu/index.html
[*2] http://lang-8.com/
[*3] http://cl.naist.jp/chantokun/

表 1　Lang-8 の学習言語別投稿文数

言語	日本語	英語	中国語	韓国語	フランス語	スペイン語	ドイツ語
文数	880,408	870,126	113,521	93,955	50,813	41,341	29,451

新聞記事などの日本語母語話者の書いた編集済みのコーパスから学習された形態素解析器では，単語にうまく分割できない．たとえば，

でもじょずじゃりません

といった学習者の書いた文があるが，これを MeCab [*4] を用いて分かち書きを行うと，

でも　じ　ょずじゃりません

と分割される．実際に人が添削を行った正解の文を MeCab で分かち書きすると，

でも　じょうず　じゃ　あり　ません

のようになる．この 2 つを見ると，日本語学習者の文を単語単位で分割したものを機械翻訳を用いて誤り訂正することは困難であると予想される．

そこで，単語単位よりも細かい文字単位に分割することを考える．文字単位の分割にすることにより形態素解析器の誤りの影響を受けないため，頑健な解析が可能であると考えられる．実際，上記の学習者の例と正解の例を文字単位に分割すると，

で も じ ょ ず じ ゃ り ま せ ん
で も じ ょ う ず じ ゃ あ り ま せ ん

となり，「じょず」と「じょうず」との対応を学習できればよくなるので，単語単位で分割していた場合と比べると頑健な誤り訂正が期待できる．

4　大規模添削コーパス

本研究で用いた大規模コーパスは Lang-8 の添削データから作成した．データは 2010 年 11 月より約 1 ヶ月クローリングを行い取得した．本研究では取得したデータのうち日本語のみを使用する[*5]．表 1 で示した内，添削されている文数は 865,208 文であった．Lang-8 のデータでは，一つの文に複数の添削がつくことがある．添削後の文数は 1,436,031 文になっており，1 つの文に対しておよそ 1.66 文の添削がついていることがわかる．

実際の文は大きく分けると，(1) 文字列の置換，挿入，削除による添削，(2) コメント付き添削の 2 種類がある．たとえば，

学習者の文：ビデオゲームをやました
訂正後の文：ビデオゲームをやりました
学習者の文：銭湯に行った。
訂正後の文：銭湯に行った。 いつ行ったかある方がいい

といったものがある．1 つ目の例は，学習者の文に誤りがあり，それを文字列の挿入によって添削しているものである．2 つ目の例は日本語は正しいがコメントを書いているものである[*6]．1 つ目のような添削のみを行っているものはスペル訂正などの手法で訂正できると考えられる．2 つ目のようなものはコメント部分を前処理で検出することで，1 つ目と同様に扱うことができる．

4.1　削除数・挿入数によるデータの分類

実際の添削において，訂正箇所は少なく，ほとんどの文字列は元の文と一致することが予想される．そこで，文の編集距離（削除数・挿入数）に従って文の分類をすることによって，Lang-8 のデータの定量的な分布の調査を行った．

表 2 に削除数と挿入数の分布を示す．削除数，挿入数ともに 1 の場合が一番多く，数が増えると少なくなっていくといった似た傾向が見て取れるが，削除数と挿入数で分布が異なっている点も存在している．まず，0 の場合を見ると，挿入数は削除数と比べて少ないことがわかる．また，9 以上の場合を見ると，削除数よりも挿入数の方が多い．これはコメントを書く人が多く，削除数は書いてあることを消すだけなので数も限られているが，挿入数はコメントを書くと多くなるからだと考えられる．また，コメントを英語で書くといった人もおり，挿入数が多くなる一因となっている[*7]．

5　誤り訂正の分割単位による比較実験

3 で提案した分かち書きの単位の違いが誤り訂正に与える影響を評価するために実験を行った．手法を比較するためにベースラインとして単語分かち書きしたコーパスを用いる．以下，この手法を W と表記する[*8]．提案手法の文字単位の分かち書きコーパスのものは，言語モデルを文字 3 グラムと文字 5 グラムの 2 種類を用いる．それぞれ C-3，C-5 と表記する．

コーパス規模の影響を調べるために，実験は TM を 10 万文に固定し LM の大きさを変えたものに加えて，そ

[*4] http://mecab.sourceforge.net/

[*5] 言語の判別は，学習者自身が設定しているので学習言語を元に行った．そのため，文単位では内容の補足説明など日本語以外の文が含まれていることがあるが，そのような文には添削が付かないので無視できる．

[*6] 学習者の文が間違っていないとき，添削者が元の文を削除して OK と書き変えているものもある．

[*7] ひとつ注意しておきたい点は，この分布が実際の学習者の誤りの分布と異なっているところである．また，今回集めてきたデータはユーザが実際に添削を行ったデータのみを集めてきており，学習者が正しく書いた文，すなわち添削されなかった文は含まれていない．削除 0，挿入 0 の 5,287 文は添削者が添削しようとしたが，添削されなかったものだと考えられる．

[*8] LM は単語 3 グラムである．

表 2　Lang-8 の日本語添削コーパスにおける削除数と挿入数の分布

挿入数 / 削除数	0	1	2	3	4	…	合計
0	5287	38357	13681	7035	3149	…	71121
1	63699	118807	30060	15072	8137	…	245457
2	31661	47008	44596	21200	11592	…	171248
3	19783	28867	28716	23610	14055	…	135786
4	15858	18001	20014	18949	14327	…	112401
5	10646	11731	13273	13187	11542	…	87434
6	7540	8212	8961	9502	8850	…	69565
7	6094	5779	6397	6922	6823	…	56719
8	5334	4292	4749	5037	5138	…	46695
9	4610	3351	3421	3720	3856	…	38810
⋮	⋮	⋮	⋮	⋮	⋮	⋱	⋮
合計	271191	327422	207422	153125	113043	…	1436031

れぞれ MERT[2] [*9]をかけたものと比較した．また，LM を 10 万文に固定し，TM の大きさを変えた場合，およびそれに MERT をかけたものを比較した．機械翻訳のデコーダとして，Moses 2.1[*10] を使用した．また，手法 W の単語分かち書きには MeCab 0.97 を使用し，その辞書には UniDic 1.3.12[*11]を使用した．

5.1　実験データ

実験データは Lang-8 日本語文 1,436,031 文を用いて作成した．全データをトレーニングセット，ディベロップメントセットに 1,306,031 文, テストセットに 130,000 文と分割した．トレーニングセットとディベロップメントセットは，添削文にコメントが入っている文を除くために，学習者の文と添削文の長さを比べたときに 2 倍以上になっているものは使用せず，それぞれ 1,162,248 文, 15,000 文と分割して使用した．

トレーニングデータには LM，TM ともにトレーニングセットからランダムに抽出してきたものを使用した．MERT に使用したデータはディベロップメントセットからランダムで 100 文抽出した．テストに使用したデータは，正解との編集距離が近い文がどれだけ訂正できるか調査するため，まずはテストセット 130,000 文を削除数・挿入数で分類した．その後，削除 0，挿入 0 から削除 4，挿入 4 までの部分を抽出し，それぞれ 50 文，合計 1,250 文を使用した．

5.2　評価尺度

評価尺度として最長共通部分列に基づく再現率（R）と適合率（P）[6] を用いた．正解に含まれる文字数を $N_{CORRECT}$，システムによる誤り訂正の結果に含まれる文字数を N_{SYSTEM}，これらの最長共通部分列の文字数を N_{LCS} とすると，

$$\text{再現率} = \frac{N_{LCS}}{N_{SYSTEM}}, \quad \text{適合率} = \frac{N_{LCS}}{N_{CORRECT}}$$

で定義される．また，F 値は再現率と適合率の調和平均である．

[*9] MERT は BLEU を最適化するように学習した．
[*10] http://www.statmt.org/moses/
[*11] http://www.tokuteicorpus.jp/dist/

表 3　分かち書き単位による誤り訂正精度の比較（LM）

LM 学習データ数 / 手法		W	C-3	C-5
10 万	R	0.91203	0.91750	0.91818
	P	0.90143	0.90681	0.90193
	F	0.90670	0.91212	0.90998
30 万	R	0.91399	0.91841	0.91892
	P	0.90252	0.90813	0.90293
	F	0.90822	0.91324	0.91086
50 万	R	0.91439	0.91845	0.91963
	P	0.90398	0.90899	0.90375
	F	0.90916	0.91369	0.91162
100 万	R	0.91524	0.91851	0.92007
	P	0.90539	0.91070	0.90539
	F	0.91029	0.91459	0.91267
10 万 MERT	R	0.91233	0.91899	0.91622
	P	0.90212	0.91454	0.91640
	F	0.90720	0.91676	0.91631
30 万 MERT	R	0.91632	0.91797	0.91666
	P	0.90634	0.91414	0.91216
	F	0.91130	0.91605	0.91440
50 万 MERT	R	0.91845	0.91855	0.91821
	P	0.90542	0.91595	0.91416
	F	0.91188	0.91725	0.91618
100 万 MERT	R	0.91838	0.91841	0.91493
	P	0.90968	0.91745	0.90922
	F	0.91401	**0.91793**	0.91207

5.3　実験結果

表 3 に LM 学習データ数を変化させた場合の実験結果を示す．結果を見ると，単語単位のものよりも文字単位の方が再現率・適合率ともに高いことが分かる．文字単位の中でも，C-3 の方が C-5 よりも適合率と F 値は高く，逆に再現率は低いことが分かる．また，LM 学習データ数は増えれば適合率が高くなっている．MERT をかけると基本的に適合率が向上している．

表 4 は TM 学習データ数を変化させた場合の結果である．結果から分かるように，W は MERT をかけた場合 TM 学習データ 10 万文の場合よりも適合率・再現率ともに向上している．しかしながら，C-3 と C-5 は TM 学習データ数が 15 万文になることで MERT をかけた場合，適合率が低下している．

6　議論

今回の実験結果の実例を示す．ここでは，TM 学習データ 10 万文，LM 学習データ 100 万文，削除 1，挿入 1 のテストデータの場合の W と C-3 を対象に比較を

表 4　分かち書き単位による誤り訂正精度の比較（TM）

TM 学習データ数 / 手法		W	C-3	C-5
15 万	R	0.91105	0.91716	0.91821
	P	0.90136	0.90678	0.90239
	F	0.90618	0.91194	0.91023
15 万 MERT	R	0.91443	0.91943	0.86993
	P	0.90753	0.90527	0.90089
	F	0.91096	0.91229	0.88514

行う．まず，いずれも訂正できた例を示す．学習者の文（テスト文）とその添削後の正しい文は

学習者：日本語 わ 楽しいです
添削後：日本語 は 楽しいです
学習者：こん い ちは
添削後：こん に ちは

である．次に C-3 で正しく訂正できた例を挙げる．

学習者： 私はプリクラ に とりました
添削後：私はプリクラ を とりました
W：私は ことですよねに とりました
C-3：私はプリクラ を とりました

W がこのような結果になった理由は，学習データの学習者の文で “近所の証明写真のプリクラ” となっているものが，“会場の近くのスピード写真（のことですよね？）” と添削され，フレーズテーブルで “プリクラ” が “ことですよね” と対応が付いているからであった．他にも削除 0，挿入 4 のデータでは文末に “GOOD” だけ挿入されている文があり，Lang-8 の添削コーパスを用いる際にはデータのクリーニングが必要不可欠である．逆に W で正しく訂正できた例を挙げる．

学習者： 数学 は テストでした
添削後：数学 の テストでした
W：数学 の テストでした
C-3：数学 は テストでした

このように，学習者の作文と添削後の文の両方が正しく形態素解析できるような場合は，単語 3 グラムを用いることができる手法 W の性能が高くなると考えられる．両方とも間違った例には，

学習者： メルボルンにいくつも れ です
添削後：メルボルンにいくつも り です
W：メルボルンにいくつ か も れ です
C-3：メルボルンにいくつも れていま す

がある．W が失敗したのは，フレーズテーブルに “つもれ” を “つもり” に直すようなものが出てないことが原因である．C-3 のフレーズテーブルでは，“れです” が “れしてます” と対応がついていることが原因である．いずれも TM の学習データ数が少ないことに起因すると考えられるので，TM の学習データを増やすことで対処可能である．

また，手法とは別にテストデータの問題が挙げられる．以下の例を見ると，

学習者： この 4 つ が 僕は少年 ころ時 に発売されて
添削後：この 4 つ は 僕は少年 ころ時 に発売されて
W：この 4 つ が 僕は 幼いころ時 に発売されて
C-3：この 4 つ が 僕は少年 のころ時 に発売されて

となっており，添削後の文でも “僕は少年ころ時” は訂正できてないことがわかる．これを直すとしたら “僕が少年のころ” もしくは “僕が少年の時” となる．C-3 を見ると，“僕は少年のころ時” となっており，完璧ではないが “の” を挿入できている．しかしながら，添削後のデータと異なっているため適合率が低くなる．これを解決するために，信頼度の高い添削文は間違いが全て訂正されていると仮定し，各添削をユーザが「いいね！」という投票ボタンを使って評価した信頼度を用いてフィルタリングを行うことを考えている．

もうひとつの問題として評価方法が挙げられる．今回は最長共通部分文字列に基づく再現率と適合率で評価を行っている．この他にも文字単位の BLEU や，文字単位ではなく誤り単位での再現率と適合率，そして文完全一致率などが考えられる．どの評価方法を用いるべきか検討が必要である．

7　おわりに

大規模添削コーパスを用いた機械翻訳手法による日本語誤り訂正を行った．誤りを含んだ文は通常の形態素解析器では上手く単語単位に分割することができないため，本稿では文字単位に分割する手法を提案した．その結果，文字単位に分割する手法は単語単位に分割する手法よりも高い適合率で訂正できることがわかった．今後は学習者の文は文字単位，添削後の文は単語単位で分割する手法を検討する予定である．

謝辞

喜洋洋さん，大山浩美さん，永田昌明さん，Graham Neubig さんに深く感謝いたします．

参考文献

[1] C. Brockett, W.B. Dolan, and M. Gamon, “Correcting ESL Errors Using Phrasal SMT Techniques,” In Proc. of COLING-ACL, pp.249–256, 2006.
[2] F.J. Och, “Minimum Error Rate Training in Statistical Machine Translation,” In Proc. of ACL, pp.160–167, 2003.
[3] H. Oyama and Y. Matsumoto, “Automatic Error Detection Method for Japanese Case Particles in Japanese Language Learners,” In Corpus, ICT, and Language Education, pp.235–245, 2010.
[4] 今枝恒治，河合敦夫，石川裕司，永田亮，桝井文人, “日本語学習者の作文における格助詞の誤り検出と訂正,” 情報処理学会研究報告コンピュータと教育研究会報告, pp.39–46, 2003.
[5] 南保亮太，乙武北斗，荒木健治, “文節内の特徴を用いた日本語助詞誤りの自動検出・校正,” 情報処理学会研究報告 自然言語処理研究報告, pp.107–112, 2007.
[6] 森信介，土屋雅稔，山地治，長尾真, “確率モデルによる仮名漢字変換,” 情報処理学会論文誌, vol.40, no.7, pp.2946–2953, 1999.